Issa Traoré · Ahmed Awad
Isaac Woungang *Editors*

Information Security Practices

Emerging Threats and Perspectives

Springer

Information Security Practices

Issa Traoré • Ahmed Awad • Isaac Woungang
Editors

Information Security Practices

Emerging Threats and Perspectives

Springer

Editors
Issa Traoré
Department of Electrical and Computer Engineering
University of Victoria
Victoria, BC, Canada

Ahmed Awad
New York Institute of Technology
Vancouver, BC, Canada

Isaac Woungang
Department of Computer Science
Ryerson University
Toronto, ON, Canada

ISBN 978-3-319-48946-9 ISBN 978-3-319-48947-6 (eBook)
DOI 10.1007/978-3-319-48947-6

Library of Congress Control Number: 2016961242

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

With the rapid development of Internet-based technologies and the increasing reliance of society on these technologies, providing security and assurance to information systems has become a critical endeavor for practitioners and the various stakeholders impacted by information and system insecurities.

In fact, the omnipresence of threats of malicious attacks has raised the importance of devising new paradigms and solutions in addition to professional skills, knowledge, and human resources in the area of information assurance. This book is a compilation of peer-reviewed papers from the first International Workshop on Information Security, Assurance, and Trust (I-SAT 2016), which introduce novel research targeting technical aspects of protecting information security and establishing trust in the digital space.

The book consists of eight chapters outlined as follows.

Chapter 1 is a brief introduction on the context of emerging security threats and a discussion of the need for new security paradigms in tackling these threats.

Chapter 2 presents contemporary and emerging botnet architectures and discusses best practices in protecting against such threats and how these protection schemes could possibly be evaded.

Chapter 3 introduces a new approach for leveraging behavioral biometrics for online fraud detection.

Chapter 4 introduces a suite of online tools to automate the complex computations involved in analyzing hardware Trojan viruses. This represents an important step in mastering the complexity involved in locating malicious modifications in integrated circuit design and implementation.

Chapter 5 presents a multimodal biometric system that combines at the feature level mouse and eye movement biometrics for user authentication. In this system, mouse movement and eye movement data are collected simultaneously and aligned based on timestamps.

Chapter 6 takes on the pressing challenge of protecting online exam integrity by introducing a multimodal biometric framework involving three modalities, namely, mouse dynamics, keystroke dynamics, and face biometrics.

Chapter 7 tackles lingering limitations in anomaly detection in computing systems (e.g., false alerts, low detection accuracy) by presenting an enhanced CUSUM algorithm for network anomaly detection. The new algorithm enables modeling various features from different sources and reporting alerts according to some decision strategies.

Chapter 8 provides a final summary of the research presented in previous chapters and discusses future trends and challenges in tackling emerging cybersecurity threats.

Victoria, BC, Canada — Issa Traoré
Vancouver, BC, Canada — Ahmed Awad
Toronto, ON, Canada — Isaac Woungang

Contents

Chapter 1
Introduction: Emerging Threats Call for New Security Paradigms

Issa Traoré, Ahmed Awad, and Isaac Woungang

1.1 Emerging Threats Landscape

Hacking incidents have become so commonplace that no organization seems out of reach for hackers. Even the US National Security Agency (NSA) seemed to have been the victim of successful hacks, as witnessed by recent public document dumps related to sensitive cyber warfare tools and technologies used by this organization. No day passes by without news reports on new hacking incidents. While two decades ago, most hackers were script kiddies motivated primarily by simple curiosity or the need for fame, many hackers, today, are professionals seeking financial gains, or conducting political activism, or involved in state-sponsored cyber espionage.

Today's hackers are emboldened by the unprecedented level of sophistication of the current hacking utilities. There is an underground software industry which develops and licenses malicious software tools and payloads for cybercriminals. The organizations involved in this illicit market provide to their customers the same services as legitimate software companies (e.g., regular updates), except that those customers are criminals.

The pinnacle in the sophistication is the so-called Exploit Kits (EKs), which federate in automated platforms most of the emerging hacking threats vectors (Eshete et al. 2015). These kits are professionally developed hacking apparatus,

I. Traoré (✉)
Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada
e-mail: itraore@ece.uvic.ca

A. Awad
New York Institute of Technology, Vancouver, BC, Canada

I. Woungang
Ryerson University, Toronto, ON, Canada

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6_1

which include sophisticated command and control (C&C) software servers, and fed from constantly updated repositories of malware payload and exploit code. EKs are marketed in the dark web (underground cyber world) and make heavy use of automation by making it possible to install malware payload on remote machines and controlling infected machines from a remote Web site. Infection happens when potential victims visit a compromised site (under control of the criminals) or click on links (sent by spam or instant message) to a Web site with the exploit kit installed. By fingerprinting the victim's browser, the kit selects which exploit to use according to the country of origin, browser type and version, operating system type and version, etc. Successful exploitation is then followed by installing malware code and taking control of the victim's machine. The scariest aspect of this is that it all happens automatically and transparently in the background without the victim's knowledge about it. In a few clicks, your machine is infected with the latest malware and becomes part of a network of zombies controlled remotely.

EKs represent a unifying framework for the latest cyber security attack vectors and tools. Around EKs revolves a nebula of emerging cybersecurity threats, including botnets, ransomware, and banking Trojans. Since its appearance a decade ago, botnet technology has evolved in sophistication, by adopting more complex command and control architecture and communication schemes, and less-prone to disruption domain naming scheme (Zhao et al. 2013).

Early botnets used centralized architecture for transmitting C&C messages. The most prevalent communication protocol used in those earlier botnets was the Internet Relay Chat (IRC). However, this type of botnet is easy to detect and disrupt due to the single point of failure embodied by the IRC server, which manages the C&C communications. Once the server is shut down, the botmaster loses control of the network of bots.

The next generation of botnets, which started appearing a decade ago, addressed the aforementioned weakness by using peer-to-peer (P2P) protocols (e.g., eDonkey) for command and control (Zhao et al. 2013). Due to its distributed and resilient control structure, P2P botnet is harder to shut down than an IRC-controlled botnet. However, in the last few years, as more knowledge has been acquired about P2P botnets, more effective solutions have been proposed to detect them and mitigate their impact.

As a result, more recently, there have been a shift in the control of many botnets from IRC and P2P channels to Web sites, using HTTP—a common protocol. Due to the prevalence of http communications and sites, detecting botnets that use http protocols is much harder (Garasia et al. 2012; Venkatesh and Nadarajan 2012; Tyagi and Nayeem 2012). Many organizations host Web sites for regular business activities and as such enable http communications. Hence, it is easy for http-based botnets to evade detection by hiding their command and control messages in legitimate http traffic.

Based on exploitable vulnerabilities, different kinds of payloads can be installed on the victim's machines, capable of achieving specific goals. One of the most common and deadliest ones consists of taking remote control of the machine. This allows the hacker to spy on the activities of the victim and steal private information

(e.g., photos, credit information, social security numbers, and emails). Such information can be used to blackmail or embarrass the individuals. For instance, in the case of politicians and celebrities, it can be used in a more targeted ways to achieve specific outcomes, such as influencing election results or discrediting the victim.

This may also be used to install specialized Trojans and spy or interfere with the victim's online banking transactions. Furthermore, taking remote control of the victim's machine provides a pathway to enrolling it in a botnet (which is merely a network of enslaved machines), and using such botnet to conduct large-scale activities such as spreading spams or conducting distributed denial of service (DDOS) against potential targets. Instead of using directly enslaved machines, some hackers specialize in renting them to other scammers through the criminal black market. Those scammers can then use the machines to carry out directly the aforementioned scams.

Another deadliest type of payloads, which appeared in the last few years, is ransomware (Lee et al. 2016). After infecting the victim's machine, the malware collects basic machine identification information (e.g., Mac address, IP address, user account information) and sends those information to the hacker' C&C server. The C&C server generates a pair of public/private key (using algorithms such as RSA), stores locally the private key, and sends the public key to the malware client on the victim's machine. The malware uses the public key to encrypt selected files (which are in general important data files) and then displays a message for the victim. In general the message will inform the victim that his/her files have been encrypted and that he/she should pay a ransom to be able to recover those files. The message will also contain directions to pay, which most of the time consists of opening a bitcoin account and transferring the ransom payment using such currency. Quite often, the message will include a payment deadline beyond which the amount will increase (e.g., double, triple, and so on). In case, where the ransom is paid, the victim will receive the private key and can then decrypt and restore the files.

To make it harder to trace them, hackers use privacy-preserving networks such as TOR for communications. It is the same line of thought which is behind using bitcoins for payment. While electronic cash such as bitcoins has been designed originally to exhibit the same traits as paper cash (i.e., user and transaction anonymity, payment and cash untraceability, and cash transferability), those same characteristics are turned on its head by criminals to perform illicit cash transactions online. Tracing those transactions is extremely difficult due to the underlying e-coin system design.

Malware designers and writers have become better and better at evading detection by using an arsenal of sophisticated deceptive techniques. For instance, different techniques are used to identify the presence of specific brands of antivirus software and circumvent them or fight back when virus scan is triggered, for instance by launching a denial of service against the victim.

One of the lifeline of most malware is the ability to communicate with the C&C server hosted by the hacker. While this is crucial for the malware, it makes it vulnerable, as antivirus software can monitor and detect such communications. The

address of the C&C server used to be hard coded in some of the earlier malware payload. However, it became quickly clear that either through reverse-engineering of malware code or by monitoring the C&C traffic, it is easy to identify, block, and blacklist the C&C address. In the last few years, more sophisticated techniques using fast flux DNS technique and domain generation algorithms (DGA) have appeared that increase stealthiness.

Fast flux DNS consists of linking a fully qualified domain name with a large number (hundreds or thousands) of individual IP addresses and swapping these IP addresses around in extremely short time periods (e.g., a few seconds or minutes) (Zhao and Traore 2012). Fast flux networks establish a level of indirection, by having the front end nodes serving only as redirectors to backend servers which actually serve requests. When some query is made to a malicious domain, the redirectors forward effectively the request to the actual C&C server which then processes it and returns the response.

DGA may either build or not on the fast flux network infrastructure. DGA consists of a mechanism used by malware to generate on the fly new domain names that would be used to contact the C&C server (Schiavoni et al. 2014). The generation of the new domain may be based on a seed and environmental factor such as time/date, and location, known only by the C&C server and the malware payload. The malware payload will generate a bunch of these domains and try to connect to the C&C server through trial and error until one of the domains is successful. The C&C server operators executing the algorithms and knowing the correct parameters will generate, register, and activate only one or a few of these domains. Such process is repeated on a regular basis, enabling hackers to move the C&C servers around continuously, making detection extremely harder.

In the emerging threats landscape, one of the serious threat vectors is stolen identity. Stolen identities are hot commodities in the underground online black market. Often now and then, we hear such and such site has been hacked and private users information such as social security numbers, addresses, credit card information (and so on) have been compromised. Quite often, such hacks go unnoticed for a long period of time. The proceeds of these hacks typically end up being sold online in the black market. Stolen identity pieces are packaged as what is known as “fullz” and sold for pennies to cyber criminals, who can use them to create seemingly legitimate accounts and conduct illegally transactions such as online auctions and online banking.

1.2 Next Generation Cybersecurity Systems

In the emerging threat landscape outlined above, we are faced with an arms race, where hackers are turning defensive technologies on their heads by coming up with smarter and increasingly sophisticated malicious software tools and payloads.

In this context, security researchers and practitioners must develop new security paradigms by rethinking conventional protection approaches and architectures. The new paradigms should provide more reliable means of defining and enforcing

human identification. Since digital identity is central to any actions on computing devices, ensuring the integrity and genuineness of such identity is crucial. Due to the increasing role of automation in malicious activities, it is also important to define reliable signatures and patterns exposing malicious automation agents and activities. By the same token, differentiating human-driven activities from robot-driven automated actions is essential.

The Information Security, Assurance, and Trust (I-SAT) workshop series has been established with these goals in mind. Its primary objective is to bring together security practitioners and researchers from government, academia, and industry to present and discuss ongoing work and innovative solutions against emerging security threats.

A diversity of themes are covered in subsequent chapters. Specifically four different themes are tackled in the proceeding. The first theme is a discussion on botnet architecture and evasion techniques against existing botnet protection strategies. The second theme relates to the analysis of hardware Trojans. While in the security community there is greater awareness of malicious software, malicious hardware is still an esoteric topic for most researchers and practitioners. However, the threat of malicious hardware is real and represents a great concern in areas such as cyber warfare and cyber terrorism.

The third theme revisits some key limitations of existing intrusion detection systems, which have been persisting, and proposes a different take on how these could be addressed.

Finally, the fourth theme covers new approaches and applications of software-based biometrics. Software-based biometrics represent a growing field of research which seeks to answer critical challenges related to the genuineness of human identity, and by extension how human behavior can be discriminately accurately from automated robot-driven behaviors.

As an indication of the importance of this emerging field, DARPA (US Defense Advanced Research Project Agency) has launched in January 2012 a new Research and Development program for innovative software-based biometric modalities to be used by over two million US military personnel (DARPA Broad Agency Announcement 2012).

According to the DARPA announcement, the main rational behind the new program is the fact that traditional approach for "validating a user's identity for authentication on an information system requires humans to do something that is inherently difficult: create, remember, and manage long, complex passwords. Moreover, as long as the session remains active, typical systems incorporate no mechanisms to verify that the user originally authenticated is the user still in control of the keyboard. Thus, unauthorized individuals may improperly obtain extended access to information system resources if a password is compromised or if a user does not exercise adequate vigilance after initially authenticating at the console."

The main goal of the new program termed by DARPA as the "Active Authentication Program" is "to change the current focus from user proxies (e.g., passwords) when validating identity on DoD IT systems to a focus on the individual. Within this program, the intention is to focus on the unique factors that make up the individual, also

known as their biometrics, without requiring the deployment of additional hardware sensors. Research resulting from this BAA (Broad Agency Announcement) will support that overall program intent by investigating novel software-based biometric modalities that can be used to provide meaningful and continual authentication when later integrated into a cybersecurity system."

References

DARPA Broad Agency Announcement # DARPA-BAA-12-06 (2012) http://www.darpa.mil

Eshete B, Alhuzali A, Monshizadeh M, Porras P, Venkatakrishnan V, Yegneswaran V (2015) EKHunter: a counter-offensive toolkit for exploit kit infiltration. In: NDSS symposium, 8–11 February 2015, San Diego, CA, USA

Garasia SS, Rana DP, Mehta RG (2012) HTTP Botnet detection using frequent pattern set mining. Int J Eng Sci Adv Technol 2(3):619–624

Lee JK, Moon SY, Park JH (2016) CloudRPS: a cloud analysis based enhanced ransomware prevention system. J Supercomput. doi:10.1007/s11227-016-1825-5

Schiavoni S, Maggi F, Cavallaro L, Zanero S (2014) Phoenix: DGA-based Botnet tracking and intelligence. In: Dietrich S (ed) DIMVA 2014, LNCS, vol. 8550. Springer, Heidelberg, pp 192–211

Tyagi AK, Nayeem S (2012) Detecting HTTP Botnet using Artificial Immune System (AIS). Int J Appl Inf Syst 2(6):38–45. ISSN: 2249-0868, Foundation of Computer Science FCS, New York, USA. www.ijais.org

Venkatesh GK, Nadarajan RA (2012) HTTP Botnet detection using adaptive learning rate multilayer feed-forward neural network. In: Askoxylakis I, Pöhls HC, Posegga J (eds) WISTP 2012, LNCS, vol. 7322. International Federation for Information Processing (IFIP), Laxenburg, pp 38–48

Zhao D, Traore I (2012) P2P Botnet detection through malicious fast flux network identification. In: 7th International conference on P2P, parallel, grid, cloud, and internet computing-3PGCIC, 12–14 November 2012, Victoria, BC, Canada

Zhao D, Traore I, Sayed B, Lu W, Saad S, Ghorbani A, Garant D (2013) Botnet detection based on traffic behavior analysis and flow intervals. Comput Secur 39:2–16

Chapter 2
Botnets Threat Analysis and Detection

Anoop Chowdary Atluri and Vinh Tran

2.1 Introduction

A botnet is a collection of Internet computers that have been set up to execute unintended operations. The owners of these machines often are not aware of the status of their devices, which is due to a lack of protection on the computers (e.g., no antivirus or firewall). When a computer without basic protection is used to browse the Internet, the user may click on a number of different links as well as download many types of files. If the files are Trojan/malware, they can automatically create a backdoor to communicate to the command center and hide their processes from the end user.

This chapter gives a walkthrough of the botnet phenomenon by centering the discussion on some famous examples, which are also representative of some of the main bot families available.

The chapter starts with a brief historical review and a discussion of botnets architectures. This is followed by a review of famous botnets examples, a discussion of techniques used by botnet to evade detection, and finally, a review of protection techniques and strategies.

2.2 Evolution of Botnets: History and Topologies

Botnet evolution started with Sub7 (a trojan) and Pretty Park (a worm) in 1999; both introduced the concept of a victim machine connecting to an IRC channel to listen for malicious commands (Ferguson 2015a, b). Then it comes to the Global Threat

A.C. Atluri • V. Tran (✉)
New York Institute of Technology,
701 West Georgia Street, 17th Floor, Vancouver, BC, Canada, V7Y 1K8
e-mail: ranlucvinh@gmail.com; atlurianoop.4@gmail.com

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6_2

Bot (Gtbot) in 2000; this botnet is based on the mIRC client which makes it possible to run custom script depending on the IRC commands. One of the most famous Gtbot attacks is to scan for host infected with Sub7 and update it to Gtbot.

In 2002, two new botnets were introduced, called SDBot and Agobot. SDBot was a single binary file, written in C++. The corresponding code was commercialized, and as a result, many new botnets were born inspired from it. Agobot, on the other hand, was considered a more advanced botnet, which suggested the principle of modular, staged attacks as payloads. Agobot infection comprises of three stages: first stage consists of installing a backdoor, then trying to disable the host antivirus, and lastly blocking access to websites of known security vendors.

In 2003, Spybot was created, as a transformation of SDBot. This new botnet introduced some new functionality such as keylogging, data mining, and SPIM (instant messaging spam). Rbot was also surfaced in the same year. This bot introduced the SOCKS proxy and included DDOS feature and information stealing tools. Moreover, the bot was also the first one to use compression and encryption to avoid detection. The year 2004 saw the rise of Bagle and Bobax, the first spam botnets. In 2006, ZeuS or Zbot was introduced and is still now one of the world most famous botnets. The year 2007 saw the birth of Storm, Cutwail, and Srizbi botnets.

The history of botnets closely correlates with the evolution of botnets topologies and architectures. Botnets are implemented using different topologies, including the following four main architectures (Ollman 2009):

- Star: This hierarchy (see Fig. 2.1) allows the bot to communicate directly with its master. This approach helps the simplest one; it facilitates bot management and makes sure the communication between both the parties are fast and accurate. However, it suffers from single point of failure and system administrators can easily block the connection to the master.
- Multi-server: This topology (see Fig. 2.2) is a more advanced form of the Star architecture. It tackles the problem of single point of failure and also makes sure that the bots can reach its closest geographical master (assuming the C&C servers are set up in multiple countries). Nevertheless, this hierarchy requires more effort to set up and plan from the master.
- Hierarchical: This topology (see Fig. 2.3) allows a bot to act as a supervisor for a group of other bots. The supervisor bot can directly connect to the master and update instructions/code base. This approach hides the presence of the master and makes tracing back to the master more difficult. Also, botmaster can easily share/lease/sell a portion of the botnet to other botmaster. Nonetheless, this architecture adds a level of latency to the update between bots, because the lower-level bot needs to wait for instructions sent from the supervisor bot, making real-time attack harder than the previous topology.
- Random (Peer to Peer): The last design (see Fig. 2.4) is called random or peer to peer (P2P). This is by far the most advanced topology in botnet. Any bot agent can send/forward commands to the next one; these instructions are often designed

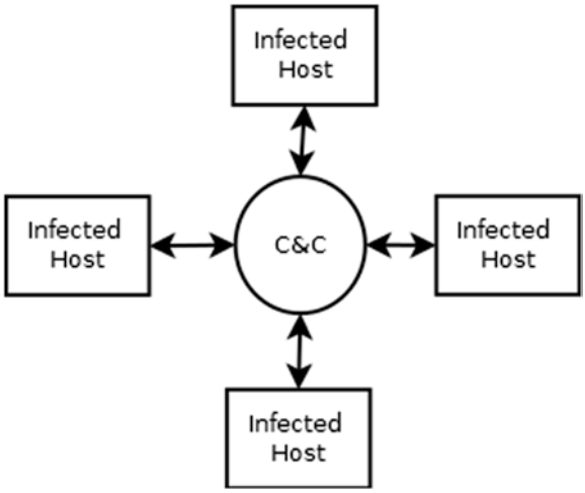


Fig. 2.1 Star formation

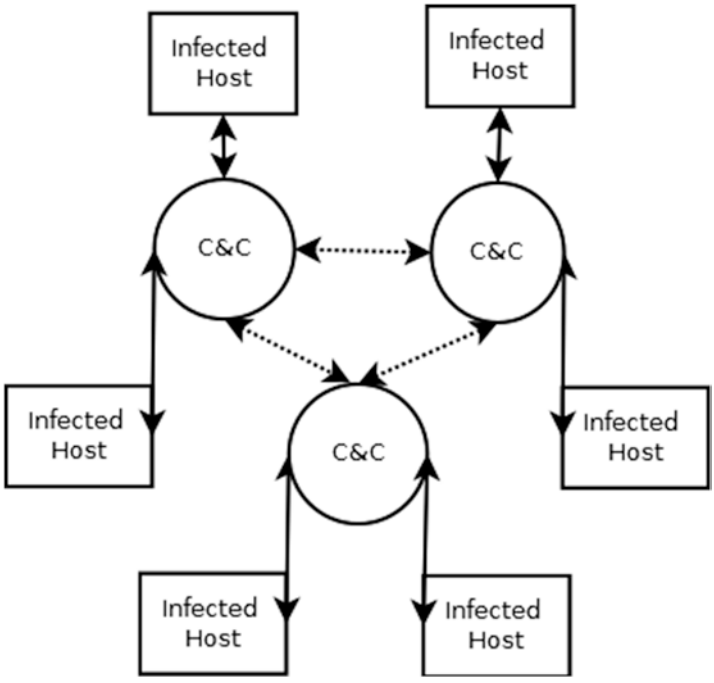


Fig. 2.2 Multi-server formation

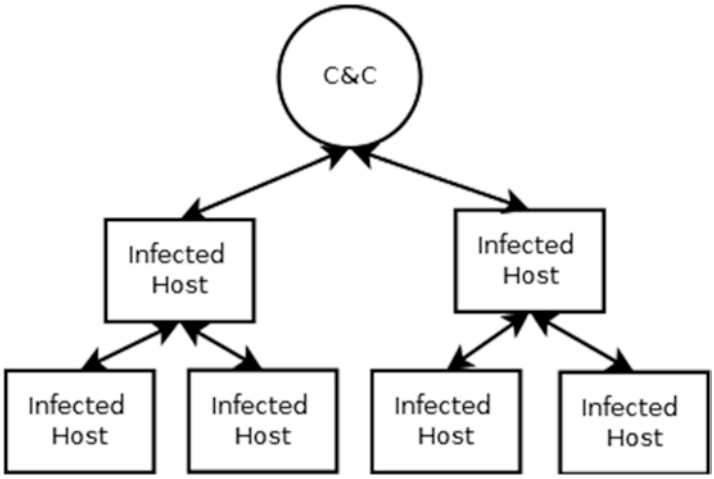
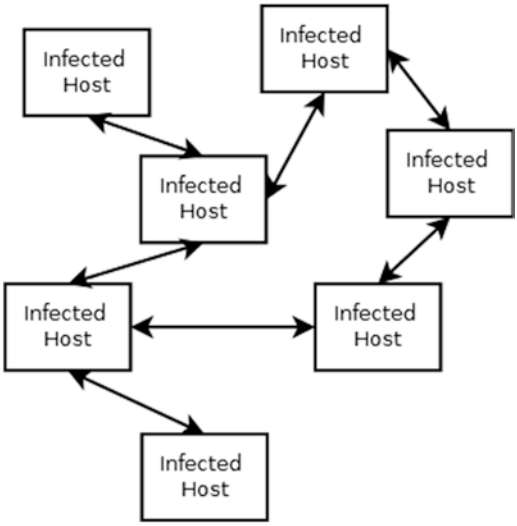


Fig. 2.3 Hierarchical formation

Fig. 2.4 Peer to peer formation



in a way that it can pass on to the next available node in the net. This method allows the botmaster to avoid detection/shutdown, as it would take a considerable amount of time to trace the communication between bots. However, the design helps researchers to track down the infected hosts easily, since monitoring one bot can reveal information about its communication with others.

2.3 Famous Botnets

There are a great number of botnets worldwide; however, most botnets have similar functionality and often are variants of some previous botnet. This chapter only focuses on ZeuS, Koobface, and Windigo as examples of popular botnets. The reason for picking these is that ZeuS is one of the early botnets that still remains famous until now, and it has gone through multiple waves of revolution. As for Koobface, this botnet represents a new form of malware that spreads through online social network, and using user's friend list as a means of propagation. Lastly, Windigo represents one of the few famous botnets targeting primarily Linux platforms.

2.3.1 ZeuS or Zbot

Overview: ZeuS is a family of credentials-stealing trojans which first surfaced around 2007 (Andriesse and Bos 2014). Since then, ZeuS has grown to be one of the world's most famous botnets. Older versions of ZeuS, which relied on IRC Command Center, have been studied by scientists and security professionals (Falliere and Chien 2009). In 2011, a more advanced version of ZeuS was introduced, called GameoverZeuS. This variant uses P2P with encryption instead of IRC channel. The modern Zeus versions with advanced features such as encryption and communication pattern not only harden detection process but also prevent the network from being infiltrated by "outsiders."

Encryption: Early versions of ZeuS use a simple mechanism for encryption, known as "visual encryption," which basically encrypts each byte by XORing with the preceding byte (Andriesse et al. 2013). Later versions introduce RC4 encryption. "Outsider" bots, which are used by researchers and security personnel, to penetrate the network, becomes counterproductive, since the fake bot needs to know under what identifier it is known to other bots in the network in order to decrypt the message.

Communication pattern: (Andriesse et al. 2013) Zeus maintains a passive and an active thread. The passive thread acts like a server, listening for incoming request. The sender's information of any successful handled request is stored in a bot's peer list. On one hand, if the receiving bot already has more than 50 peers in its list, the sender bot data will be saved in a queue for future peer list update. However, the sender bot will be automatically added if the peer list is 50 or less. On the other hand, if the sender identifier is already on the list, all information (such as IP and ports) is updated, to keep a fresh connection with its peer.

The active thread runs in a cycle and automatically repeats after a specified amount of time. In each iteration, the bot attempts to connect every peer in its list, asking for updated version of binary and configuration file. Each peer has five chances to reply to the request; if there is no response after five times, the bot will

first check if it actually made the request to the recipient by checking for Internet connection; then depending on the Internet status, it will drop the unresponsive peer and update the list. Moreover, if the bot has less than 25 peers, it will try to connect to all its neighbors asking for the neighbor's peer list. This mechanism assures the botnet network always stays fresh and long-period-disconnected bot can recover quickly even with a minimal number of peers.

2.3.2 Koobface

Overview: Koobface is one of the first malwares to target online social networks (OSN) (Baltazar et al. 2009; Thomas and Nicol 2010; Sophos Press 2007). The botnet first appeared around early 2009 and has caused severe damage to social networks users. The koobface malware, unlike others, has its binary split into multiple modules, each of which has a separate functionality that handles different type of OSN. Additionally, instead of spreading through spam email, the malware uses OSN messaging service to propagate. This is a very effective way to escalate the infection, as people often have no doubt about their friend's messages (Fortinet White Paper 2013). Once clicked on the link in the message, user will be redirected to a fake page, created by social engineering toolkit (usually fake YouTube page). Here, users will be asked to install a fake plugin in order to view the content. The fake plugin is the koobface downloader, which will attempt to find out the OSN the user is using and then download the necessary components accordingly. As of 2009, the malware was able to identify a significant amount of various OSN such as Facebook, Twitter, MySpace, Friendster, Hi5, Netlog, Bebo, and so on.

Features: This botnet not only breaks captcha by forcing other infected machine's user to solve it but also creates fake OSN accounts in order to befriend with potential victims. Research has shown that a normal user has 41 % probability to accept a friend request from strangers on Facebook (Irani et al. 2011); this is why KoobFace has become so successful and led the way for a new form of malware that spread through OSN.

2.3.3 Windigo

Overview: The botnet has a long history (Bilodeau et al. 2015), starting from 2011; it comprises of a few different malwares which take care of different tasks. Most of the modules (e.g., Linux/Ebury, Linux/Cdorked, Linux/Onimiki), however, are specialized in compromising linux servers (e.g., web, dns servers). There are also two other malwares (Win32/Boaxxe.G and Win32/Glubteta.M) targeting Windows computers' end users. Like any other modern botnet, Windigo also carries out a number of tasks ranging from sending spams, drive-by downloads, advertisement

fraud, and credentials stealing; however, one important point to notice is the main victims are Linux servers, which mean they have more resources, bandwidth, and also have more potential to reach end users via web servers. The main Linux components are summarized in the following.

Linux/Ebury (Bilodeau et al. 2015): main functions are creating backdoor shell and credentials stealing. One of the outstanding attributes of this malware is its ability to run in a very stealthy way, because maintaining an SSH backdoor shell is a difficult task. In order to do this, the creator has applied many different techniques, and some of them are as follows:

- Utilize linux pipes as much as possible
- Leave no information in log files
- Alter OpenSSH binaries code at runtime instead of modifying the current files on disk
- Use a centralized backdoor in a library

Linux/Cdorked (Bilodeau et al. 2015): It is used to redirect traffic from infected servers to malicious sites; some of the most common web servers (apache, nginx) have been infected with variants of this malware. In order to deploy this malware, the botnet uses previously installed Linux/Ebury to download a complete source code of the web server; it also gets another patch from an infected server. Then the patch is applied on to the new source code and a new binary is compiled, after that, the original web server binary is replaced by the new malicious binary. When making a redirection, the malware tries to guess if the current user is a system admin by checking a number of url keywords and cookies; this mechanism allows the malware to act under the radar and thus avoid detection.

Linux/Onimiki: It is a domain name service component which acts together with Linux/Cdorked. Whenever a redirection is made from a Cdorked infected machine, Onimiki will try to resolve the domain name in the url. It is also noted that Onimiki uses BIND name server and this offers a number of advantages, such as the following:

- It is stateless and requires no configuration when Onimiki is installed, thus allowing the malware to act alone without any further interaction with the operators.
- It allows fast rotation of subdomains and legitimate domains.
- Its reputation of the legitimate domains helps Onimiki avoid blacklisting.

Table 2.1 summarizes the main features of the three botnets examples considered above.

2.4 Botnet Detection Evasion Techniques

Botnet uses many different methods to avoid detection; some popular techniques are as follows:

Table 2.1 Comparison of Zeus, Koobface, and Windigo

	Zeus/Zbot	Koobface	Windigo
Infection vectors	Infection vectors vary widely; some main mechanism are spam, drive-by download	Mainly through online social network	Spread through linux servers
Features	A DIY bot that is features-rich, easy to use. Underground criminals can easily purchase a copy of Zeus and build a version of this malware. Maintaining a sophisticated P2P network which makes taking down operation harder	Can detect multiple online social network and has various components that can act differently depending on the detected online social network. Break captcha by forcing users to solve it	Although this botnet mainly targets linux servers, it has the ability to take control of the windows machines which established connection to the infected linux servers
Availability and distribution	Freely available with purchasability makes Zeus the most popular botnet	Not available for purchase	Not available for purchase

- Domain generation algorithm (DGA or Domain Flux): According to Khattak et al. (2014), DGA is an approach to dynamically generate the C&C address. The botmaster builds a specific mechanism to randomly create the server address and sets up the DNS record to point the address to the C&C. An example using this technique is ZeuSGameover malware (Andriesse et al. 2013); the algorithm is triggered when all peers are unresponsive or the bot fails to update for more than a week.
- IP flux (Shin and Gu 2010): this technique is similar to DGA, but instead of associating multiple domains with one IP, it attempts to alter DNS records to have various IP addresses linked to one domain. The method is aided with the help of Dynamic DNS. IP flux has two different types:
  - Single FLUX: the idea is to have intermediaries between the bot clients and bot master, providing a layer of anonymity for the bot master. These middle layer machines are often called "proxy bots," which are also infected machines chosen by the master.
  - Double flux: is an advanced version of single flux, which abstracts the domain name and IP address of the proxy bots. When bot agents try to connect to proxy bots, they will be redirected to name servers controlled by the master. These name servers will handle the domain name matching and generating, and make sure that name and IP pairs change frequently so the connection will not be blacklisted or blocked.
- Binary obfuscation (Shin and Gu 2010): the bot client uses various techniques to defeat host-based security application, one of which is polymorphism. It is an attempt to reconstruct the bot into different forms but still maintain the same

Table 2.2 Botnet detection evasion techniques summary

Domain generation algorithm	A mechanism which allows bot agents to connect with master through a variety of different domain names. The bot master takes care of generating the domain name and sets up the servers; the agents will have a list of names to try connecting to
Single flux	Bot master chooses some infected machines to become proxy bots or fake master. This technique helps the master to become harder to track down and thus stay alive longer
Double flux	An advanced version of single flux, which takes the connection to another level by adding the complexity of domain name generation
Binary obfuscation	To defeat the host-based defense, bot agents can be built into different binary form, but still maintaining the functionality. This technique is carried out often by encryption and packing
Security suppression	Certain types of botnets have the ability to disable local security service and also block users from finding a security solution
Anti-analysis	Some early day botnets have the ability to change its behaviors based on the environment it's running on

functionality, by using encryption or packing. Some advanced packers can build a completely different binary for every packed request. Despite success in hiding its identity, the bot binary can still be detected while executing due to memory-based detection approach. To work around this problem, the bot agent uses another practice called metamorphism, which gives the bot the ability to be rebuilt into different, but semantically equivalent code.

- Security suppression: When infecting a weak machine, the malware attempts to disable all or several security services on the host. For example, (Bilodeau et al. 2015) the malware Conficker will attempt to disable some security service in Windows when infected; it also sets up a blacklist which prevents users to access certain security site.
- Anti-analysis: Certain botnets have the ability to scan the environment which they are running on, and depending on the results, they can disable/change their behaviors to appear harmless or mislead the researchers. This technique was quite popular in the early days of botnet, but after the explosion of virtual technology, this method is being forgotten as criminals also want to target virtual users.

Table 2.2 summarizes the evasion techniques outlined above.

2.5 Botnet Detection Methodologies

Botnet Detection techniques can be grouped in various categories. Figure 2.5 depicts those different categories, which include both active and passive techniques (Plohmann et al. 2015; SANS Institute InfoSec Reading Room 2015).

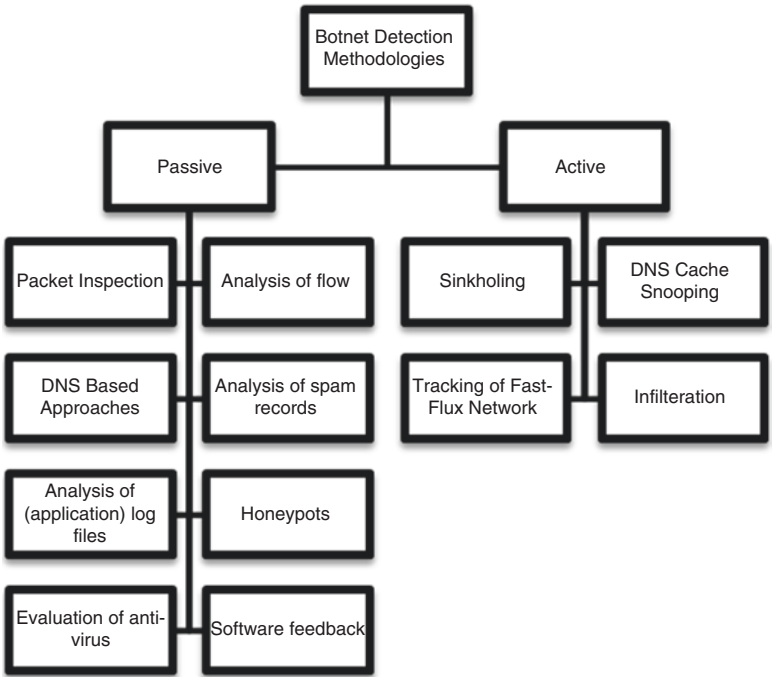


Fig. 2.5 Botnet detection methodologies tree

2.5.1 Passive Techniques

Passive measurement techniques are a group of few methodologies where data is gathered through monitoring and observation alone. Using passive measurement techniques, we can track activities without interfering the production network or making changes in any kind of evidence. There will always be a limited amount of data which can be collected from passive methods and this data can be used for analysis (Plohmann et al. 2015).

2.5.1.1 Packet Inspection

The most common methodology under passive botnet detection system is Packet Inspection of local network data. The main objective of this technique is to ensure various parameters of packets are matched like protocol field, identification, flags, and content with huge database of predefined abnormal and suspicious behavior which allows identifying bots by analysis of data only.

For instance, there might be a data packet consisting of shell script code which is being used to inject malware in network and that particular malware is communicating

with the public address which can host the data. The predefined patterns are also referred to as detection signatures.

The main characteristic of packet inspection approach is that it can be incorporated in typical Intrusion Detection Systems (IDS) where attacks are identified based on predefined signature database. The inspection service runs in two deployment modes, which include as a single appliance for entire network that is known as network-based detection system (NIDS), and Host-Based Network Detection service (HIPS) where every node of the network runs a separate instance of the detector.

Intrusion detection system is primarily used to just detect the malicious activities and they are reported to network administrator, and network administrators are responsible to take action on infected areas.

Some commonly identified drawbacks of intrusion detection or prevention techniques include the fact that when the network traffic flow is very high it is difficult to perform complete inspection of the packets. If we make use of techniques like packet sampling or packet filtering prior to analysis, chances of missing malicious packets are too high.

Furthermore, intrusion detection systems are known for their high false alarm rates, which is a serious limiting factor.

2.5.1.2 Analysis of Flow Records

Analysis of flow records can be considered as a technique for tracing network traffic at a nonrepresentational level. In the packet inspection approach, the packet is described to some level of details; each and every packet should be inspected in an aggregated form. In the flow record approach, when a data stream is considered for analysis it goes under a process where several parameters are matched. These parameters include addresses of the source and destination, port numbers and the protocol which is used in the packets, how many packets are transmitted, and size and duration of the session.

Net flow can be considered as one of prominent examples for the analysis of flow record format. Like with packet inspection, the main aim of flow record analysis is to differentiate and identify the traffic patterns by creating a scheme to detect malicious traffic.

2.5.1.3 DNS-Based Approaches

A connection should be initiated and established with infected hosts or commanding server by considering botnet infrastructure whenever hosts have been infected by a botnet. This can usually be achieved by integrating a communication protocol with the malware. This can be done in two ways as follows.

An IP address can be integrated into the bot, which will be executable upon distribution, but the IP address should be fixed. A predefined domain name should be used, which will be contacted if the host system is compromised. To avoid downtime

by providing redundancy, multiple IP addresses can be associated or mapped with a single domain name. On demand, these IP addresses can be changed to dynamic whereby they are not configured for static use (Plohmann et al. 2015).

2.5.1.4 Analysis of Spam Records

Spam emails are irrelevant messages sent to a large number of users. Spam represents a common drive of botnets. The analysis of spam records provides a method of identifying and anticipating botnet infection attempts. Unlike DNS-based approaches, which target primarily the C&C phase, spam analysis aims at detecting botnets at the infection phase, and this technique will eventually detect botnets that essentially do spamming. Spam analysis involves identifying regular emails communications and distinguishing illegitimate message contents.

Distinguished patterns of spam mails are produced by the bot eventually forming the foundation or base for botnet detection. The content of message offers a good initiation point for matching and characterization of messages related to the email protocol header and content.

The correct placement of spam traps will be helpful summation to this schema. Usually spam traps are mailing addresses with no prolific functionality other than to accept unrecognized and unwanted mails and can be distinguished as a distinct variety of honey tokens or honeypots.

2.5.1.5 Analysis of (Application) Log Files

It is common practice for devices and applications to maintain records of events related to different operational aspects in the form of log files.

Log files analysis is a secondary approach of botnet detection system. The basic analysis is done using network devices log files, which come as a basic match method from entire network devices; this analysis can be done in parallel over entire range of network devices.

2.5.1.6 Honeypots

Figure 2.6 depicts a network architecture with two honeypots (A and B). A honeypot is a dedicated machine with a purpose of exposure to outside (i.e., Internet) with a focused goal to attract attackers and learn attacking methods or even to get compromised by malicious activities. In this setup, the network is always secured in the backend and only honeypots are kept visible by exposing them to the outer world. Honeypots help administrator to understand the attackers' techniques against the network and develop and deploy adequate security policies and mechanisms for protection.

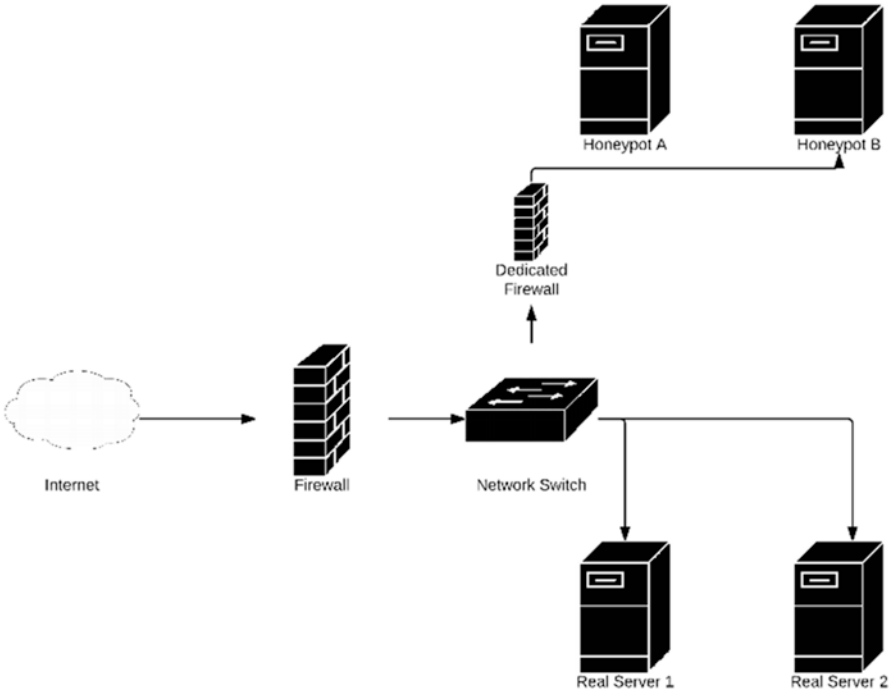


Fig. 2.6 Honeypot network

There are different types of honeypots including the following:

- *Client and server honeypots*
- *Low interaction honeypots*

The main motive for using honeypots in botnet analysis is the opportunity to collect different data about the practices and strategies used by inventors of malware and hackers. In general, two types of data can be collected by honeypots:

- Types of attack vectors in OS and software used for attacks, as well as the real exploit code which links to them.
- Actions done on an exploited workstation. These can be noted, while malware loaded on to the workstation can be conserved for further analysis.

2.5.1.7 Evaluation of Antivirus

This approach simply consists of relying on existing antivirus software capability. Different antivirus products have different signature databases, with some overlapping signature set. New generation of antivirus systems not only pushes updates

regularly to their clients, but they also learn from new instances of viruses occurring at specific endpoints, by pulling information from the clients. So it is a two-way communication stream.

2.5.1.8 Software Feedback

Software installed in the user work stations and data flow in network are analyzed and automated feedbacks of software reported to vendors. In this scenario of network each host machine acts as a sensor and the entire network is converted into a big sensor network (Plohmann et al. 2015).

2.5.2 Active Techniques

The group of active methods contains methods that involve communication with the information sources being observed. While these allow deeper probing and analysis, their application may leave traces that impact consequences or include events that can be observed by the concerned. This can cause counter-reactions, such as a DDoS attack or trigger other attempts at evading detection.

2.5.2.1 Sinkholing

This is a process of mitigating botnets by cutting off the source and breaking of communication between bots and C&C server.

As shown in Figs. 2.7 and 2.8, sinkholing consists of redirecting requests from the bot to the sinkhole (typically a server under control of the good guy) rather than letting such communications go through to the C&C server.

If one or more domains with fixed IP addresses are used by the malware, then discovering and blacklisting them will quarantine the specific malware examples that rely on them, making those useless. By using the direct IP addresses, there is no need of the DNS queries and the botnet can be terminated by deregistering the domain name (Plohmann et al. 2015).

This approach could help discover more malicious activities beyond the initial detection. For example, if a domain is identified as malicious, it is known that all incoming queries for this entry are given out by infected hosts with high probability.

2.5.2.2 DNS Cache Snooping

As shown in Fig. 2.9, DNS Cache Snooping approach leverages the caching property implemented and used by several DNS servers. If a DNS server is asked for a domain for which it has no entry defined, it will issue a query towards the responsible authoritative name server on behalf of the querying client and store the resultant data record later in a local cache. Caching is mainly used to increase the performance of a name server and reduce its traffic load.

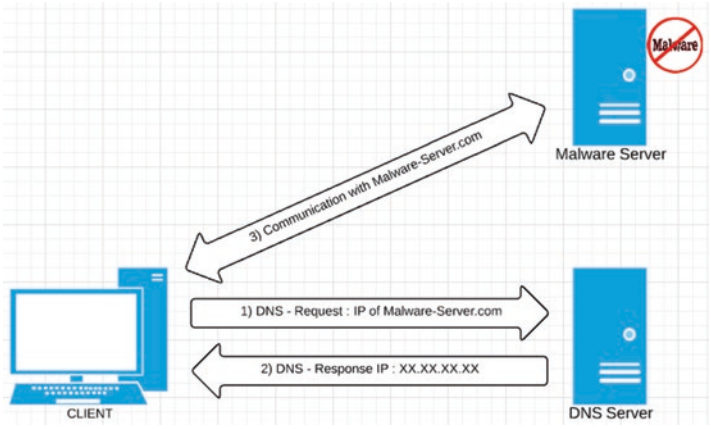


Fig. 2.7 Sinkhole attack

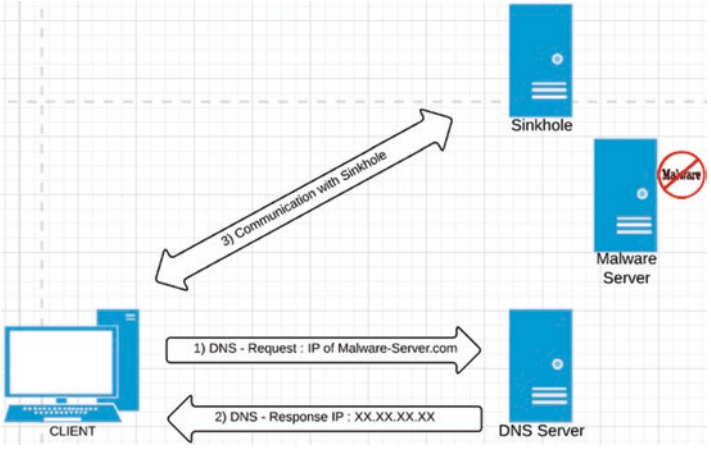


Fig. 2.8 Sinkhole redirection

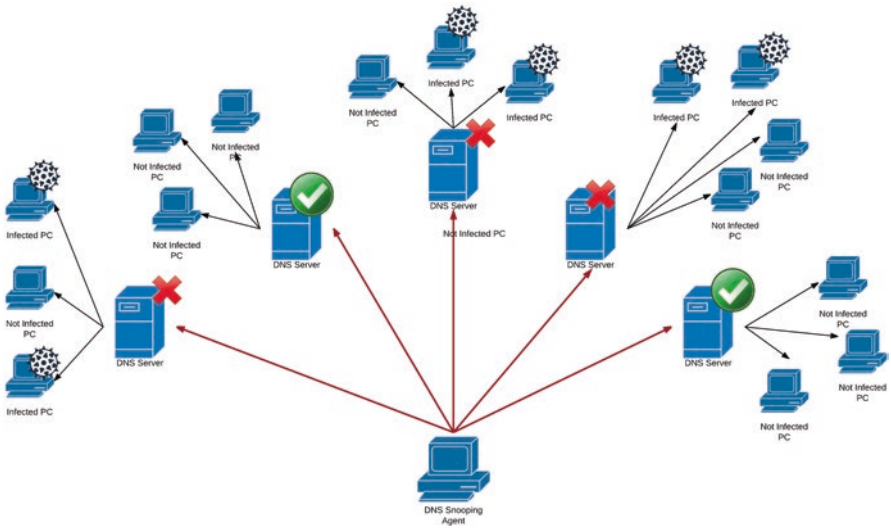


Fig. 2.9 DNS cache snooping

Cache snooping approach consists of analyzing the caches to identify illegitimate or unexpected DNS queries, which potentially could point to botnet presence.

2.5.2.3 Infiltration

Infiltration techniques can be divided into software- and hardware-based techniques. Software-based infiltration technique can be used to monitor the traffic and bots executable to achieve control of bots in network whereas hardware-based infiltration allows to access command and control server and also to wiretap the communication between the nodes.

This usually requires the reverse engineering of the botnet infrastructure. This infiltration is a precise analysis which is useful for identification of potential weakness of infrastructure. The extracted knowledge is always very useful to achieve a commanding position in fighting back against botnet infection (Plohmann et al. 2015).

2.5.2.4 Tracking of Fast-Flux Network

Fast-flux networks consist of linking a single or few domain names with a large pool of IP addresses controlled by the botmaster, as illustrated by Fig. 2.10. Botnets use fast-flux networks to introduce secrecy of their actions and grow the consistency of their network and command configuration. This increases the stealth of the botnet, making detection of the C&C server much harder. Fast-Flux networks use promptly altering DNS records, indicating at a large number of hosts, and substitute as supplementary

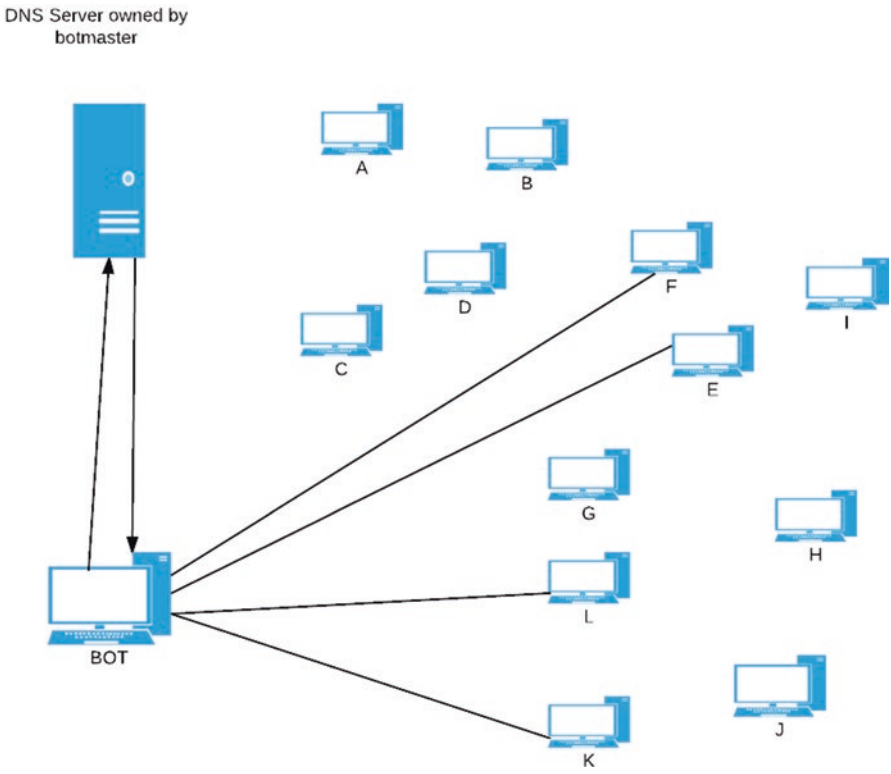


Fig. 2.10 Fast-flux network attack

proxy layer to hide the actual content delivery systems. The proxy nodes are usually compromised workstations of the botnet itself (Plohmann et al. 2015).

Typically, the IP address assigned by the botmaster DNS Server is valid for only a few minutes that is indicated by the Time to Live (TTL) value.

The detection approach used in this case consists of monitoring and identifying the DNS server with low TTL values. Correlating such information with other parameters could expose the presence of botnet activity.

2.6 Defense Against Botnet Using Network Security Devices

Traditional network security appliances and devices (i.e., IDS, firewall, antivirus) play an important role in defending against botnet. Although taken in isolation these devices may not be enough, but they are essential components in any protection strategy. However, appropriate configuration must be performed for these devices to be effective in the fight against botnets.

2.6.1 Intrusion Prevention and Detection Systems

Intrusion Detection Services are performed on three different platforms: some instances filter intrusions on each individual node of network with help of applications which are called host-based intrusion prevention systems, whereas some other scenarios consist of a central device acting as an intrusion prevention system and a single device serving the entire network needs. In very high-risk infrastructure a combination of Network and Host Intrusion prevention is used to detect Botnets including when encrypted data is involved, as Network-Based Intrusion Prevention cannot detect Botnet in Encrypted traffic (Andriesse and Bos 2014; Ollman 2009).

2.6.2 Network Firewalls

Most of the network firewalls enabled with Botnet traffic filtering provide reputation-based control in network based on ratings of IP address or domain name. This integrates with an external central repository of database of known malicious devices and domains, and dynamically stops the attacks originating from these sources. For unknown attack sources, the firewall always checks for traffic flowing to/from communication potential botnet C&C server reports/logs such occurrences.

Network firewalls filter traffic with the following components (Stawowski 2015).

2.6.2.1 Dynamic and Administrator Blacklist Data

Filtering is done using a central database of malicious domains and IP addresses from central repository. This database is maintained by different vendors like cisco, Websense, and IronPort (Cisco White Paper 2015).

2.6.2.2 Traffic Classification and Reporting

For classification of Botnet Traffic, the active filter associates the source and destination addresses of user data besides the IP addresses that have been revealed for the several lists and logs and accounts the administrator and dynamic database (Cisco White Paper 2015).

2.6.2.3 Domain Name System Snooping

To ensure the binding of IP addresses to domains that are listed in central repository of database, the Network Firewall uses DNS Snooping in combination with DNS Inspection. The Firewall matches DNS Snooping lookup with DNS replies.

Firewall builds a reverse cache, which compares the IP address in user replies to actual known legitimate domain; if the domain matches then it is considered as clean traffic else it is flagged as bot traffic (Paquet 2015).

2.7 Security Measures Against Botnets

2.7.1 Network Design

Network sesign must be done is such a way that intruders and malware are not able to exploit existing susceptibilities. Defense in depth strategy in network against Botnet helps to mitigate even zero day attacks on network and helps to streamline security operations.

This involves making use of layered security systems on each segment to ensure security against bots (Boyles CCNA Security Study Guide) (Fig. 2.11).

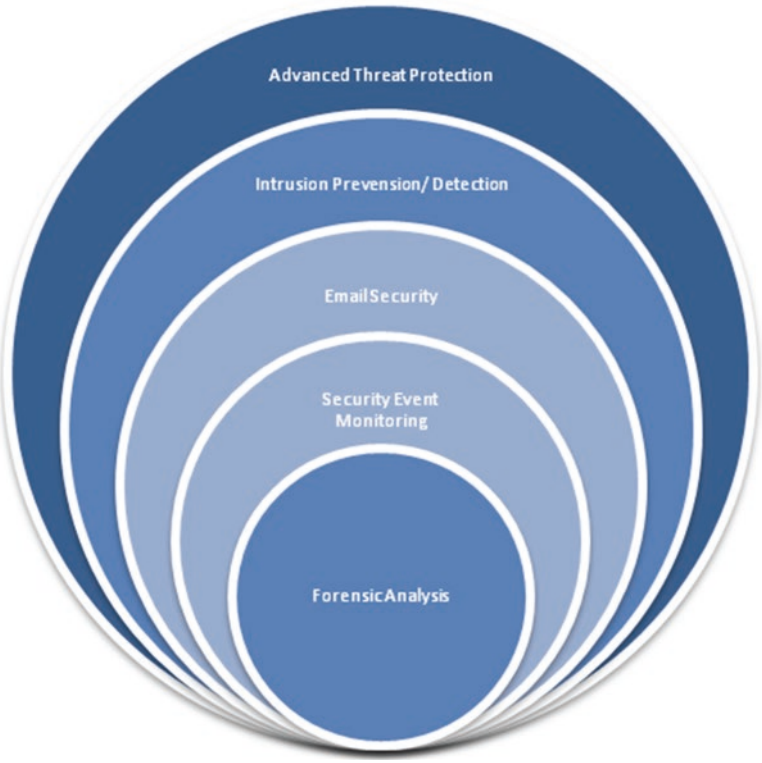


Fig. 2.11 Security measures chart

2.7.1.1 Advance Threat Protection

Advanced threat protection systems must be used to mitigate layers 3 and 4 traffic and block all unintended traffic and allow only traffic initiated for trusted network and enable data control on edge of network (Cisco White Paper 2015).

2.7.1.2 Intrusion Prevention and Detection System

It enables the capability of deep packet inspection and anomaly detection by analyzing data from layer 4 up to layer 7 of network and correlated the events to protect network against botnets. Intrusion prevention is the most important component of network filtering. Thus it is always good to deploy both host and also network-based appliances, as while network-based detection fails to mitigate encrypted attacks, it enables synchronizing with a central repository of malicious patterns of intrusions and protects network against it [25, 26].

2.7.1.3 Email Security Systems

Email being most important component of work flow it's very important to allow mails and also filter threats associated with botnet infection using email filtering engines.

2.7.1.4 Forensic Analysis

Forensic-enabled devices allow correlating the security events in network and trace the origin of attack. This allows administrators to act efficiently on bots and protect other network users against them (SANS Institute InfoSec Reading Room 2015).

2.7.1.5 Security Event Monitoring

Event monitoring allows keeping track of all events in network and gives a comprehensive report of threats against network and also enables the transparency in network monitoring (Stawowski 2015).

2.7.2 Application Usage

Botnet can be prevented by monitoring and establishing normal patterns of usage for applications on individual nodes. The following application natures can be used to mitigate botnet against host.

2.7.2.1 HIPS (Host-Based Intrusion Prevention System)

Host-based intrusion prevention systems are application model of network-based IPS services to prevent network attacks on host (Scarfone and Mell 2007).

2.7.2.2 End Point Security

End point security applications monitor and protect the host against known viruses and malware and also observe and identify malicious activities in the behavior of the host computer; once if any abnormal activity is observed in the computing process, the application itself stops the suspected processes (Scarfone and Mell 2007).

2.7.2.3 Application Firewall

Application firewall can be used to block unwanted ports especially common ports of botnet attack such as ports 25 and 21 which are generally used by bots to transfer data.

2.8 Conclusion

In this chapter, we presented Zeus, Koobface, and Windigo as some of the most impactful botnets. There are many other different ones which have their own targets (such as desktop end users or mobile devices), and also use diverse avoidance techniques. The chapter also summarizes the passive and active countermeasures against botnets as well as some defensive mechanisms which can be implemented on the network.

References

Andriesse D, Bos H (2014) An analysis of the ZeuS peer-to-peer protocol. IR-CS-74, rev

Andriesse D, Rossow C, Stone-Gross B et al (2013) Highly resilient peer-to-peer botnets are here: an analysis of Gameover Zeus. VU University of Amsterdam, Amsterdam

Baltazar J, Costoya J, Flores R (2009) The real face of KOOBFACE: the largest web 2.0 botnet explained. Trend Micro Threat Research

Bilodeau O, Bureau P, Calvet J et al (2015) Operation Windigo. http://www.welivesecurity.com/wp-content/uploads/2014/03/operation_windigo.pdf. Accessed 22 July 2015

Boyles T (2010) *CCNA Security Study Guide*. Indiana: Wiley Publishing, Inc., 2010

Cisco White Paper (2015) Combating botnets using the cisco ASA botnet traffic filter. http://www.cisco.com/c/en/us/products/collateral/security/asa-5500-series-next-generation-firewalls/white_paper_c11-532091.pdf. Accessed 26 July 2015

Falliere N, Chien E (2009) Zues: King of the bots. Symantec Corporation, Cupertino, CA

Ferguson R (2015) The history of botnet—part I. http://countermeasures.trendmicro.eu/the-history-of-the-botnet-part-i/. Updated 24 Sept 2010. Accessed 20 July 2015

Ferguson R (2015) The history of botnet—part II. http://countermeasures.trendmicro.eu/the-history-of-the-botnet-part-ii/. Updated 24 Sept 2010. Accessed 20 July 2015

Fortinet White Paper (2013) Anatomy of a botnet. Fortinet, Sunnyvale. www.fortinet.com

SANS Institute InfoSec Reading Room (2015) Defense in depth. https://www.sans.org/reading-room/whitepapers/basics/defense-in-depth-525. Accessed 30 July 2015

Irani D, Balduzzi M, Balzarotti D et al (2011) Reverse social engineering attacks in online social networks. DIMVA 2011, 8th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, Amsterdam, The Netherlands, 7–8 July 2011. Also published in "Lecture Notes in Computer Science", Vol 6739/2011, doi:10.1007/978-3-642-22424-9_4

Khattak S, Ramay NR et al (2014) A taxonomy of botnet behavior, detection, and defense. IEEE Commun Surv Tutorials 16(2):898–924

Ollman G (2009) Botnet communication topologies, understanding the intricacies of botnet command-and-control. Damballa Inc., Atlanta

Paquet C (2015) Network security concepts and policies. http://www.ciscopress.com/articles/article.asp?p=1998559. Accessed 1 Aug 2015

Plohmann D, Gerhards-Paddila E, Leder F (2015) Botnets: detection, measurement, disinfection & defense. https://www.enisa.europa.eu/publications/botnets-measurement-detection-disinfection-and-defence. Accessed 30 July 2015

Scarfone K, Mell P (2007) Guide to Intrusion Detection and Prevention Systems (IDPS). National Institute of Standards and Technology, Gaithersburg

Shin S, Gu G (2010) Conficker and beyond: a large-scale empirical study. In: Proceedings of annual computer security applications conference (ACSAC)

Sophos Press Release (2007) Sophos Facebook ID probe shows 41% of users happy to reveal all to potential identity thieves. https://www.sophos.com/en-us/press-office/press-releases/2007/08/facebook.aspx

Stawowski M (2015) Practical defense-in-depth protection against botnets. http://www.clico.pl/services/practical-defense-in-depth-protection-against-botnets. Accessed 31 July 2015

Thomas K, Nicol DM (2010) The Koobface botnet and the rise of social malware. Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software, Malware, 2010, pp 63–70

Chapter 3
Collective Framework for Fraud Detection Using Behavioral Biometrics

Ahmed Awad

3.1 Background

Fraud detection is an important topic that has been well addressed in literature before. Enhancements include building intelligent fraud prevention and detection models that are applicable to specific industries such as banking, insurance, government and law enforcement agencies, and more. Sophisticated models were built on top of analytical techniques to achieve such goal.

To the best of our knowledge, most of the biometric fraud detection researches in current literature focus on identifying the fraudulent activities by a set of predefined rules. These standards are registered during the enrollment phase, which users sign up for their biometric information.

Frank et al. proposed a set of 30 behavioral touch features extracted from raw touchscreen logs. The touch input is collected through user's normal activity on their phone, such as basic navigation maneuvers (up down, left right scrolling). Based on these data, the team introduced a classification framework, which is efficient at detecting user identity during the enrollment phase (which the system learns about the user's behaviors and gather the special features from the touch data) and is capable of accepting or rejecting the user based on his/her interactions with the device (Frank et al. 2013). This method is, however, not effective to act as a stand-alone authentication mechanism for long-term authentication, since the false positive rate is within 0–4 % which is unacceptable in certain scenarios. Nevertheless, the work proves that touch dynamic authentication is achievable and, with other complementary data such as context information, would greatly increase the effectiveness of the framework (Frank et al. 2013).

A. Awad (✉)
New York Institute of Technology, Vancouver, BC, Canada
e-mail: ahmed.awad@nyit.edu

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6_3

Bo et al. (2014) proposed SilentSense, an authentication framework which identifies users silently and transparently, by collecting user touch behavior biometric and micro-movement of the device caused by user's interaction. SilentSense faces a number of different challenges, such as user behavior modeling (the model should contain multiple features from both user's action and device's reaction), identification strategy (it is important to distinguish between guest users and owners from a limited set of behavior information), and balance among accuracy, delay, and energy (real-time observation function can quickly exhaust the device battery) (Bo et al. 2014). The researchers carried out a series of test on an Android phone, where SilentSense runs as a background service capturing the information about current app and touch events, and the test outcome shows that the application works best under two-class SVM (support vector machine) classifier with increasing amount of guest information (Bo et al. 2014).

Deshmukh and Patil (2014) came up with an iris recognition framework for credit card fraud detection, based on the natural open eyes. Their technique is to, firstly, create a preprocessed image of the iris and then detect all iris feature points by direction information, length information, width information of texture, neighboring gray information, and relativity in the effective iris area. After that, encode all the feature points and identify different patterns based on the iris code. And finally, use auto-accommodated pattern to sort the iris patterns and deliver the recognition result. The experimental result showed that the correct recognition rate is 99.687 %, false acceptance rate is 0.313051 %, and false rejection rate is 0.293945 % (Deshmukh and Patil 2014).

Gaurav et al. (2012) proposed a smart card fraud prevention scheme using a combination between fingerprint and password. The system incorporates password-based authentication with fingerprint identity, generated by fingerprint capture procedure. The suggested mechanism has three phases: registration, log-in, and authentication phases. In registration phase, user will sign up with the system his/her username, password, and fingerprint identity; the system will process fingerprint data into a digital certificate format and then transform it into a mathematical representation. In the second phase (log-in), user will send a request to the system, with all his/her registered information. And finally, in authentication phase, the server will calculate all the provided data and either accept or reject the user.

It is important to note that the above approaches fail to mention the biometric data variance between session and what necessary actions to handle them. These biometric differences or previous user activities should be taken into account for updating user's profile/history, as attacker could capture the valid past session and use it to compromise the system.

3.2 Fraud Detection Framework

The main purpose of a fraud detection system is to be able to detect fraudulent activities as soon as they occur. Report them and respond to such incidents accordingly.

A typical host-based fraud detection system consists of an agent application (could be a script) that runs on the user's machine. The agent collects all relevant information that could help in identifying the user's computing environment such as the hard drive ID, the OS version, the machine's local IP, and so on. It could also target identifying the user himself through the collection of behavioral biometric data.

After establishing a session with the business service, the agent will send the data to the server integrated within the server request. The web server will forward the fraud detection data to a dedicated fraud detection server component which will process this data and other data collected locally from the server and correlate it to previously collected data to detect frauds.

As indicated in Fig. 3.1, the data collected from the user's machine falls into one of the following categories:

- Geo-location
- Machine identifiers
- Network status identifiers
- Operating system status (includes user authentication context)
- Behavioral biometrics (keystroke dynamics, mouse dynamics, and command line lexicon)

Data collected from various factors are combined into a device-user signature token which is updated as the user uses the machine and processed and passed to the server for the purpose of fraud detection. Previous tokens are stored on the servers for future uses.

The server establishes the trust based on the provided token. It trusts that this authenticated user is whom he/she claims to be and that this user is connecting from

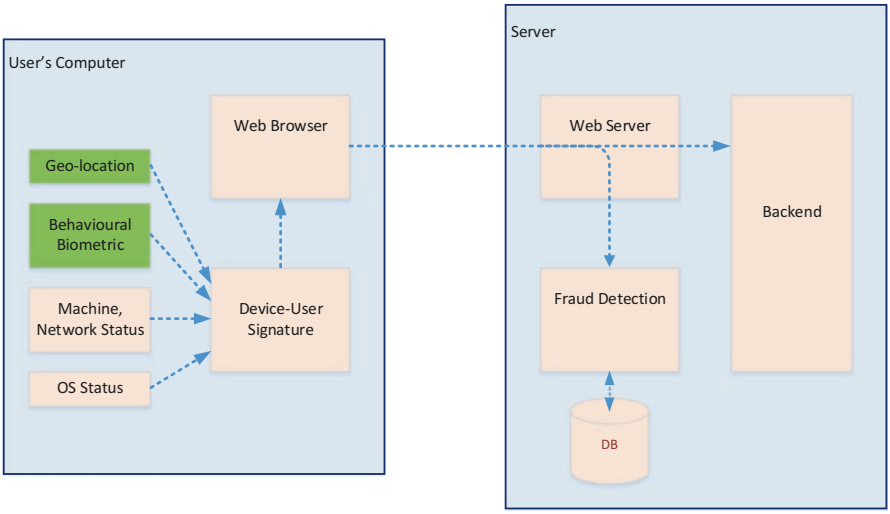


Fig. 3.1 Client/server fraud detection scenario

a known machine by comparing the different factors included in the token to the previously collected ones. One of the weaknesses of this model is that the data collected for fraud are limited only to the period of activity that is related to the user's session. Previous machine status and user activities are not sent to the server and are not included in the fraud detection analysis. Such model is vulnerable to spoofing, replaying, and man-in-the-middle attacks.

A malicious code or a rogue application installed on the user's machine can perform malicious activities before the user's session in preparation of an attack on the user's account. Such activities should be taken in consideration.

In order to overcome such weaknesses, a persistent passive agent could be installed to monitor all of the activities on the computer. The agent could pass a summary of the activities to the server when the user connects to it to establish a new session. This model faces several implementation challenges. First, it is difficult to assure that this agent is up all the time; the attacker could bypass some of the agent's monitoring functionalities forcing the agent to collect false information. Second, this model raises privacy concerns due to the fact that the agent is monitoring the activities at periods of time that are not related to the user's activities on the server. Information such as a different user with a specific biometric profile who was using the system during a specific period of time will be made available to the server. In such case, user's consent is mandatory.

The Past Activities Aware (PAA) model could be implemented using a proxy server (Fig. 3.2). In this architecture, the fraud detection component is integrated in a proxy server that is used to access various web servers through an internal network or over the Internet. In this case, the user will be made aware that his web activities will go through this server and a consent form will be displayed. The proxy server is configured to inject a script in all of the web pages that pass through it. The script runs on the user's machines and collects all machine-user signature data and passes it back to the proxy server. The proxy server intercepts these data items while processing other

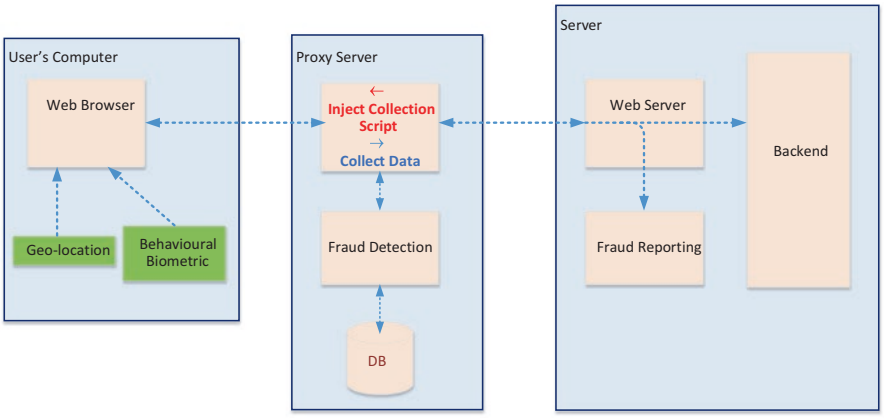


Fig. 3.2 Proxy server-based fraud detection scenario

web requests and passes them to the fraud detection module. The data will be stripped from the web request before forwarding it to the web server.

The fraud detection server will communicate fraud reporting events to the web server in cases of fraud being detected. Such events will be intercepted by a reporting system and required responses will be taken.

3.3 Behavioral Identity Verification

As shown in the above section, behavioral biometrics represent an important input to the detection system. The data can be used to passively verify the user's identity and establish the expected trust. Mouse and keystroke dynamics are considered as two good candidates for such purpose.

Mouse dynamics correspond to the actions generated by the mouse input device for a specific user while interacting with a graphical user interface. Touch dynamics is a different version of mouse dynamics when captured over a mobile device (Ahmed and Traore 2007, 2011).

Keystroke dynamics recognition systems measure the dwell time and flight time for keyboard actions (Dowland et al. 2002). The raw data collected for keystroke includes the time a key is depressed and the time the key is released. Based on the data, the duration of keystroke (i.e., length of time a key is depressed) and the latency between consecutive keystrokes are calculated and used to construct a set of monographs and digraphs producing a pattern identifying the user.

Figure 3.3 shows the architecture of the detection system. Two neural networks are involved in this design. The first one is designed to process the digraph data represented by the fly time from a specific key location to another key location. At training phase, the network is trained with the session data. This process takes place for each user, where the two neural networks are trained with the user's data. The second network is designed to process the pressure sensor data which is represented as monographs of dwell time for a specific key location. The network is also tuned with the user's session data at the enrollment phase.

The inputs to both networks are the key locations and the output is fly/dwell time. Inputs and outputs of the neural networks are normalized based on their minimums and maximums to enhance the training process.

During the testing phase, both networks are fed with the data collected from the current sessions. Outputs from both networks are compared to the actual fly/dwell time collected in the session. The output from the network represents how this output should be if the current session is actually performed by the user whose data were used to train both networks (the legitimate user).

The deviation from the expected behavior is calculated for both networks and passed to a fusion component that is used to arbitrate between both inputs to make a final decision about the user's identity. This decision is represented by the confidence ratio (CR) whose value indicates how confident the system is that the session

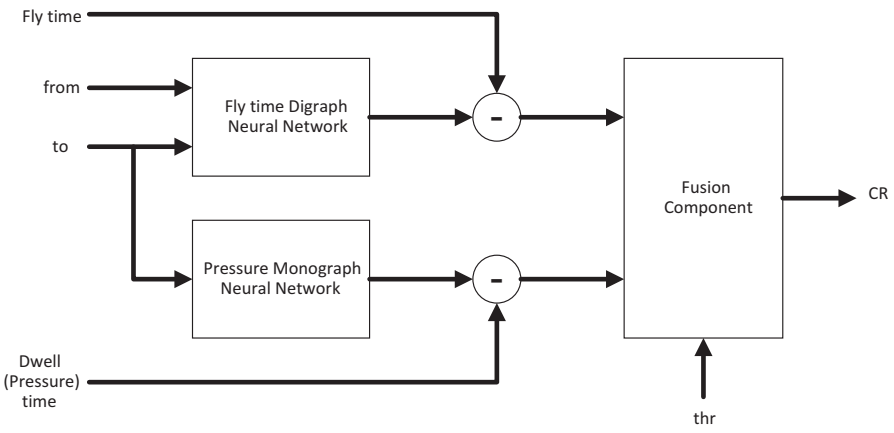


Fig. 3.3 Calculation of a trusted user signature

data belong to the legitimate system user. The higher this value the more confident the system is about such finding.

As shown in Fig. 3.3, a threshold value (thr) is used to tune the fusion component. The value is varied to achieve the best tuning results for each user in the system. The weights of the trained networks and the optimal threshold value are stored for each user as its own biometric signature calculated by this detection unit.

3.4 Experimental Evaluation

3.4.1 Evaluation Metrics and Procedures

In order to evaluate the accuracy of the PAA Fraud Analysis Model, we calculate the following:

- False acceptance rate (FAR), which measures the likelihood that an imposter may be erroneously accepted by the system
- False rejection rate (FRR), which measures the likelihood that a genuine user may be rejected by the system

Data collected in Ahmed and Traore (2014) is used in this experiment. We empirically configured the test data to simulate attacks on users' machines using other users' data. The data is engineered so that 2 % of the data are attacks. The confidence ratio (CR) is calculated for each session, and session length is also recorded. Sessions are classified into the following categories:

1. Legitimate user's session (high CR)
2. A real attack

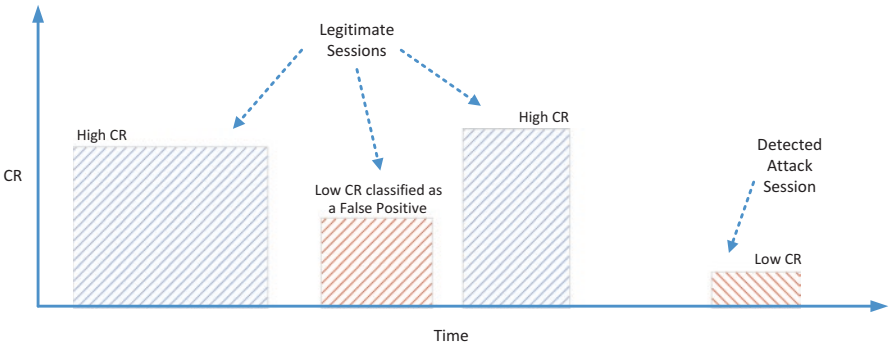


Fig. 3.4 Three categories of sessions included in the test

3. A false positive (a legitimate session incorrectly classified as an attack, low CR) (Fig. 3.4)

The test is repeated for both of the normal Past Activities Unaware (PAU) model and the new Past Activities Aware (PAA) model.

3.4.2 Results

Figure 3.5 shows the ROC curves obtained for both of the PAU and PAA models. For the PAU model, the ERR point occurs at 2.46 % for FAR and 4.64 % for FRR. The PAA model introduces a significant performance increase with the EER is at 0.1 % and 0.1 % for FAR and FRR, respectively.

3.5 Conclusion

Behavioral biometrics such as mouse and keystroke dynamics are established technologies that have several benefits over physiological biometrics. They can be used unobtrusively in both static and dynamic authentication modes without requiring special hardware sensors. Utilizing such biometrics in fraud detection helps in closing the gap associated with proving the relation between the nature of the user's activities and his/her real identity. In this chapter, we proposed a new framework for fraud detection that takes the biometric factors in consideration. The framework makes fraud detection decisions based on data collected from past periods of activities. Such technique enhances the accuracy when compared to typical past unaware detection techniques.

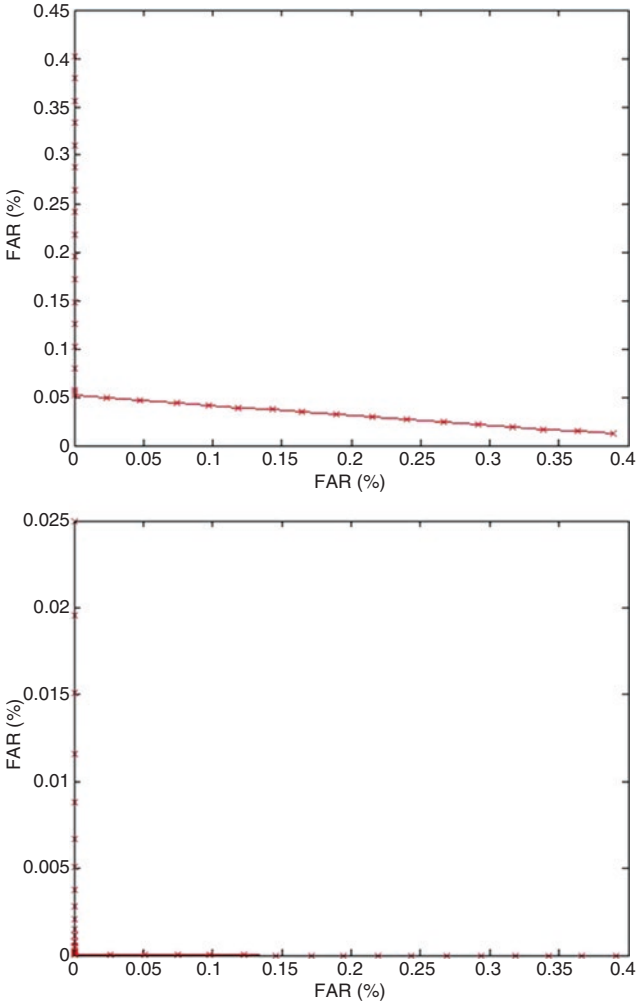


Fig. 3.5 Receiver operating characteristic (ROC) curve for (**a**) normal fraud detection model (Past Activities Unaware (PAU)) and (**b**) enhanced fraud detection model (Past Activities Aware (PAA))

References

Ahmed AAE, Traore I (2007) A new biometric technology based on mouse dynamics. IEEE Trans Dependable Secure Comput 4(3):165–179

Ahmed A, Traore I (2011) Dynamic sample size detection in continuous authentication using sequential sampling. In: Proceedings of annual computer security applications conference (ACSAC), 5–9 December 2011, Orlando, FL, USA

Ahmed AAE, Traore I (2014) Free text recognition of keystrokes. IEEE Trans Cybern 44(4):458–472

Bo C, Zhang L, Jung T, Han J, Li X, Wang Y (2014) Continuous user identification via touch and movement behavioral biometrics. In: IEEE international performance computing and communications conference (IPCCC), December 2014, pp 1–8

Deshmukh SP, Patil SH (2014) Credit card fraud detection using iris biometrics technique. Int J Sci Eng Res 5(7):131–137

Dowland P, Furnell S, Papadaki M (2002) Keystroke analysis as a method of advanced user authentication and response. In: Proceedings of the IFIP TC11 17th international conference on information security: visions and perspectives, 7–9 May 2002, pp 215–226

Frank M, Biedert R, Ma E, Martinovic I, Song D (2013) Touch analytics: on the applicability of touchscreen input as a behavioral biometric for continuous authentication. IEEE Trans Inf Forensics Secur 8(1):136–148

Gaurav J, Tyagi S, Ranjan J (2012) Smart card fraud prevention scheme using fingerprinting authentication. Int J Comput Sci Inf Technol 3:3059–3062

Chapter 4
The Hardware Trojan System: An Online Suite of Tools for Hardware Trojan Analysis

Nicholas Houghton, Samer Moein, Fayez Gebali, and T. Aaron Gulliver

4.1 Introduction

The field of hardware security is relatively new and has begun to develop a level of sophistication that requires more structure. The variety of techniques employed across the spectrum of integrated circuit (IC) designs results in a diversity of structures, behavior, and insertion points for hardware trojans. Thus, most detection methods have been designed to detect specific trojans. To date, no method capable of detecting even some of the known trojans has been developed.

Several hardware trojan taxonomies have been proposed (Wolff et al. 2008; Rad et al. 2008; Karri et al. 2010; Wang et al. 2008). In Wolff et al. (2008), trojans were organized based solely on their activation mechanisms. A taxonomy based on the location, activation, and action of a trojan was presented in Rad et al. (2008) and Karri et al. (2010). However, these approaches do not consider the manufacturing process. Another taxonomy was proposed in Wang et al. (2008) which employs five categories: insertion, abstraction, activation, effect, and location. While this is more extensive than previous approaches, it fails to account for the physical characteristics of a trojan. Thus, a comprehensive taxonomy was proposed in Moein et al. (2015a) which considers all attributes a hardware trojan may possess.

The trojan attributes provide information such as how it entered the host circuit, its effect, and where in the design it is located (Moein et al. 2015a). They can also be used to determine which detection methods are effective against a trojan (Moein et al. 2015b). In this chapter, two effective trojan analysis techniques are described. However, both require laborious computations which are prone to error when performed by hand. To aid in the universal acceptance of these techniques, an online suite of tools was developed. The *Hardware Trojan System* (HTS) automates the

N. Houghton (✉) • S. Moein • F. Gebali • T.A. Gulliver
University of Victoria, 2446 Sinclair Road, Victoria, BC, Canada
e-mail: nhoughto@uvic.ca

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6_4

necessary computations, provides centralized documentation and reference materials, and maintains a database to store user data. In the future, this database can be used as a resource for developers and attackers to search existing Trojans and detection methods.

The contributions of this chapter are as follows:

1. A technique for describing hardware trojans based on their attributes is presented.
2. A means of evaluating the effectiveness of both trojans and detection methods is devised.
3. An online system which automates the analysis techniques is described.
4. A database to store known hardware trojans and detection methods is developed.

The remainder of this chapter is organized as follows: Sect. 4.2 describes the analysis techniques. Section 4.3 presents the applications developed to automate these techniques, and the online system is described. Section 4.4 demonstrates the use and effectiveness of the system through a case study, and finally Sect. 4.5 provides some concluding remarks.

4.2 Hardware Trojan Analysis Techniques

The Hardware Trojan System (HTS) analyzes trojans based on the comprehensive hardware trojan taxonomy proposed in Moein et al. (2015a). This taxonomy is comprised of 33 attributes organized into eight categories as shown in Fig. 4.1. These categories can be arranged into the following four levels as indicated in Fig. 4.2:

1. The **insertion** (chip life cycle) level/category comprises the attributes pertaining to the IC production stages.
2. The **abstraction** level/category corresponds to where in the IC abstraction the trojan is introduced.
3. The **properties** level comprises the behavior and physical characteristics of the trojan.
4. The **location** level/category corresponds to the location of the trojan in the IC.

The properties level consists of the following categories:

- The **effect** describes the disruption or effect a trojan has on the system.
- The **logic type** is the circuit logic that triggers the trojan, either combinational or sequential.
- The **functionality** differentiates between trojans which are functional or parametric.
- The **activation** differentiates between trojans which are always on or triggered.
- The **layout** is based on the physical characteristics of the trojan.

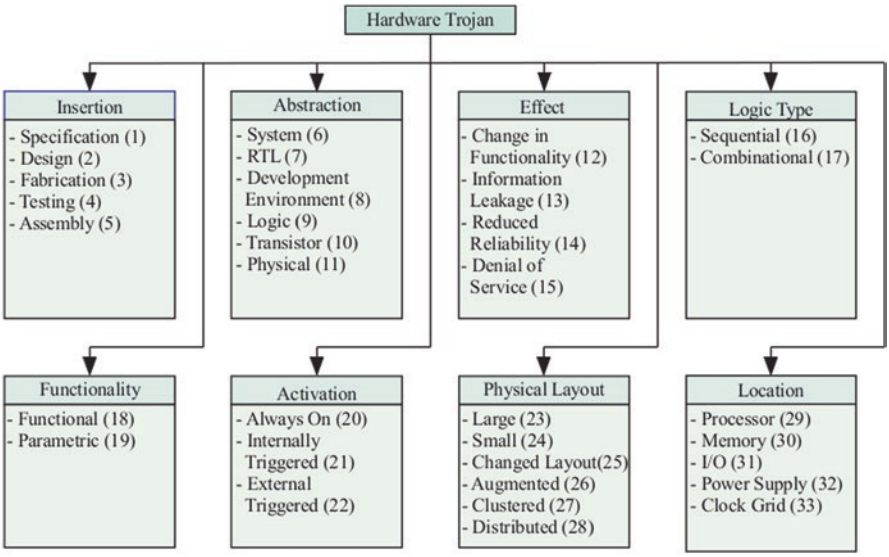


Fig. 4.1 The 33 attributes of the hardware Trojan taxonomy in Moein et al. (2015a)

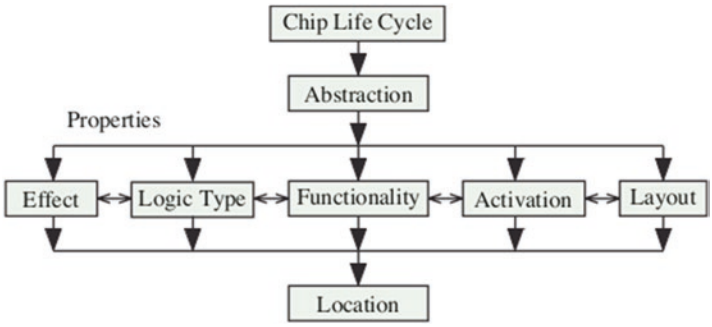


Fig. 4.2 The hardware Trojan levels (Moein et al. 2015a)

4.2.1 Trojan Classification

In order to develop a classification of hardware trojans, the relationships between the attributes were examined in Moein et al. (2015a) using a 33×33 matrix $\mathbf{R}$ which can be expressed as

$$\mathbf{R} = \begin{bmatrix} R_1 & R_{12} & 0 & 0 \\ 0 & R_2 & R_{23} & 0 \\ 0 & 0 & R_3 & R_{34} \\ 0 & 0 & 0 & R_4 \end{bmatrix}$$

The entries on the diagonal represent submatrices which describe how the attributes in each layer are interrelated. For example, submatrix $\mathbf{R_2}$ describes the relationships in the abstraction layer and is given by

$$\mathbf{R_2} = \left[\begin{array}{c|cccccc} A & 6 & 7 & 8 & 9 & 10 & 11 \\ \hline 6 & 0 & 1 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 1 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 1 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 1 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 1 \\ 11 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

The submatrices not on the diagonal describe the relationships between layers. For example, submatrix $\mathbf{R}_{23}$ describes how the abstraction attributes relate to the property attributes and is given by

$$\mathbf{R_{23}} = \left[\begin{array}{c|cccccccccccccccc} A & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ \hline 6 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 8 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 9 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 11 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

An entry $r(i, j)$ in $\mathbf{R}$ is a binary value where a 1 indicates that attribute i can lead to attribute j and is 0 otherwise. For example, $r(7,15) = 1$ indicates that a trojan in the register transfer logic (RTL) (attribute 7) can cause a denial of service attack (attribute 15).

A hardware trojan must be observed to determine its attributes and these are used to form $\mathbf{R}$. Then a systematic process of scanning the rows and columns can be used to gain insight into the characteristics of the trojan. This process is described in detail in Moein et al. (2015a). The observed attributes can be used to determine the possible locations of a trojan within the design and in which manufacturing phases it can be inserted. Conversely, the phase a trojan was inserted can be used to determine which abstraction levels are vulnerable, the trojan properties, and what locations can be compromised. To easily understand the characteristics of a trojan, a directed graph is generated from $\mathbf{R}$. Attributes are represented by nodes and their relationships by edges.

Consider a trojan that has the following attributes:

- Causes denial of service (DoD) (attribute 15)
- Composed of combinational logic (attribute 17)

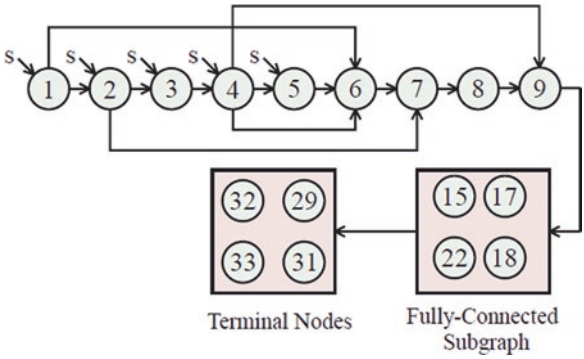


Fig. 4.3 The directed graph corresponding to a trojan

- Functional (attribute 18)
- Externally triggered (attribute 22)

A visual examination of **R** leads to the graph drawn in Fig. 4.3. If it is determined that it is not possible to insert the trojan in the design phase (attribute 2), then attribute 2 can be removed from the graph. Without a direct connection to attribute 1, attributes 3, 4, and 5 must also be removed. Further, without attribute 2 the trojan can only be inserted in the specification phase (attribute 1).

4.2.2 Trojan Evaluation

Due to the complexity of IC designs, hardware trojans are typically unique. As a consequence, detection methods developed thus far have been developed to detect specific trojans. The diversity in both trojans and detection methods makes it difficult to evaluate, compare, and organize them. A means of evaluating hardware trojans and detection methods based on the eight attribute categories was proposed in Moein et al. (2015b). A trojan or detection method will possess some combination of attributes from each of the eight categories, and each combination is assigned two numerical values. The value I is used to identify the combination, while the value C is used to denote the severity (for a trojan) or coverage (for a detection method) of the combination. Tables of I and C values for the eight categories were presented in Moein (2015). For example, the logic type category describes the circuit logic which activates the trojan. Table 4.1 shows the possible attribute combinations for this category and the corresponding values of I_L and C_L.

The I and C values from the category tables are arranged into identification and severity/coverage vectors, respectively. For a trojan, the vectors are denoted as I_T and C_T, and for a detection method, they are denoted as I_D and C_D. Thus, an identification vector is

$$I = I_R I_A I_E I_L I_F I_C I_P I_O$$

Table 4.1 Logic type category values

Attributes	I_L	C_L
Sequential (16)	2	2
Combinational (17)	1	1
Both (16 and 17)	3	3

Table 4.2 Identification and severity vectors for two hardware trojans

Technique	Parameters (I_P)								Severity (C_P)							
	I_R	I_A	I_E	I_L	I_F	I_C	I_P	I_O	C_R	C_A	C_E	C_L	C_F	C_C	C_P	C_O
Trojan A (Moein et al. 2015a)	2	6	2	1	2	1	7	7	**2**	6	4	1	2	1	5	2
Trojan B (Moein et al. 2015a)	3	3	1	2	1	2	8	1	**3**	3	2	2	1	3	6	1

Table 4.3 Identification and coverage vectors for two hardware trojan detection methods

Technique	Parameters (I_P)								Coverage (C_P)							
	I_R	I_A	I_E	I_L	I_F	I_C	I_P	I_O	C_R	C_A	C_E	C_L	C_F	C_C	C_P	C_O
Potkonjak et al. (2009)	3	3	B	1	2	4	7	V	3	3	**7**	1	2	3	5	5
Narasimhan et al. (2013)	3	3	1	2	1	4	7	V	3	3	**2**	3	1	3	5	5

where $I_R, I_A, I_E, I_L, I_F, I_C, I_P, I_O$ are the {insertion, abstraction, effect, logic type, functionality, activation, physical layout, location} category identification values, respectively, and a severity/coverage vector is

$$C = C_R C_A C_E C_L C_F C_C C_P C_O$$

where $C_R C_A C_E C_L C_F C_C C_P C_O$ are the {insertion, abstraction, effect, logic type, functionality, activation, physical layout, location} category strength values, respectively.

Table 4.2 provides a comparison of two hardware trojans. Trojan A has a lower severity than Trojan B in the insertion category, denoted by C_R. This indicates that Trojan B can be inserted in more stages of the manufacturing process than Trojan A. Table 4.3 gives a comparison between two detection methods. The method in Potkonjak et al. (2009) has a higher coverage in the effect category (C_E) than the method in Narasimhan et al. (2013), indicating that it can detect more trojans based on their effects.

4.3 The Hardware Trojan System

The Hardware Trojan System (HTS) is a dynamic website built to implement hardware security applications. In particular, applications have been developed to automate the techniques described in Sect. 4.2.

4.3.1 The Classification Tool

To investigate a trojan, users select attributes via an easy-to-use user interface (UI). Once the attributes are chosen, the tool performs the necessary analysis using matrix **R** and displays the resulting directed graph. Suppose an attacker decides to insert the trojan described in Sect. 4.2.1. The tool provides the directed graph shown in Fig. 4.4 from the selected attributes, which eliminate the need for manual analysis of the matrix. Note that Fig. 4.4 is the same as Fig. 4.3 which was constructed by hand. This verifies the results obtained using the classification tool.

If the design phase (attribute 2) takes place in a secure location, an attacker will conclude that it is not possible to insert the trojan in this phase. To determine the possible trojans that can be inserted without access to the design phase, attribute 2 should be removed from Fig. 4.4. The classification tool provides an attribute removal feature. When an attribute is removed, the directed graph is recreated based on the new matrix **R**. The result of removing attribute 2 is shown in Fig. 4.5. The new graph clearly shows that compromising the design is still possible, but it must be done from the specification phase (attribute 1). The possible locations remain the same, but the potential effects of the trojan have changed. Without access to the design phase (attribute 2), the trojan cannot be composed of combinational logic (attribute 17) or be externally triggered (attribute 22). Even though the attributes change in functionality (attribute 12) and always on (attribute 20) were not selected, the tool determined that these attributes are possible.

The classification tool automatically generates a severity/coverage vector for use with the evaluation tool described in Sect. 4.3.2. The identification and severity vectors describing the trojan in Fig. 4.4 are shown in Fig. 4.6. The trojan classification data is saved in the database along with the identification and severity/coverage vectors.

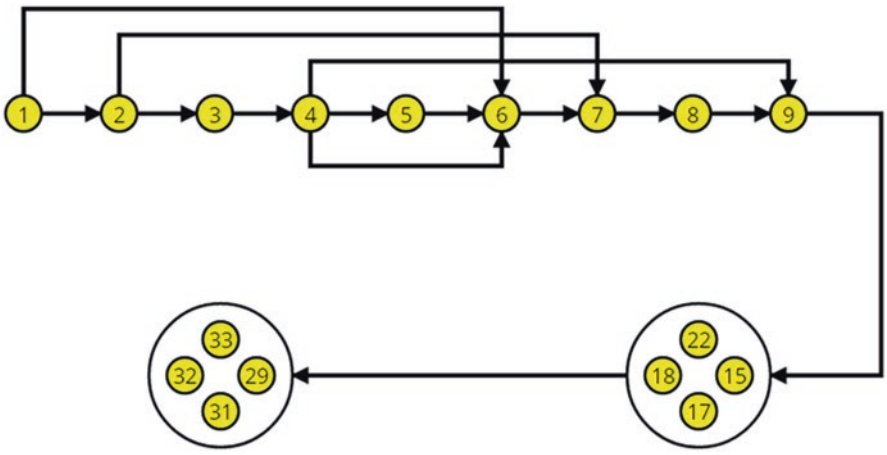


Fig. 4.4 The directed graph obtained by analyzing a hardware trojan with the classification tool

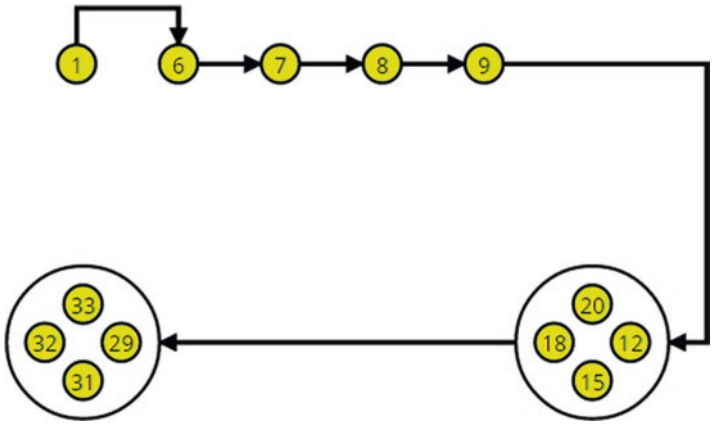


Fig. 4.5 The directed graph after attribute 2 is removed

I_R	I_A	I_E	I_L	I_F	I_C	I_P	I_O	C_R	C_A	C_E	C_L	C_F	C_C	C_P	C_O
5	6	4	1	1	3	NA	T	5	6	2	1	1	4	NA	4

Fig. 4.6 The identification and severity vectors generated by the classification tool for the trojan in Fig. 4.4

4.3.2 The Evaluation Tool

The HTS provides a series of drop-down lists to create a coverage vector for a new detection method. This vector is stored in the database along with a description of the method. The evaluation tool provides a simple means of searching the database for previously saved detection method coverage vectors and trojan severity vectors. Once a detection method and a trojan have been selected, a user can use the compare button to perform a comparison of the coverage and severity vectors. For example, the results of a comparison are shown in the bottom row of Fig. 4.7. A 1 is displayed when the detection method has a value greater than or equal to the corresponding trojan value and a 0 otherwise. The zeros in Fig. 4.7 indicate that the detection method may fail to detect the trojan based on the insertion point (I_R) and the logic type (I_F).

While the evaluation tool can be employed for individual comparisons, its greatest potential is with a centralized database. Currently the tool only provides comparisons of trojans and detection methods entered by the user. Universal adoption of the HTS will provide a centralized database of all known detection methods and trojans.

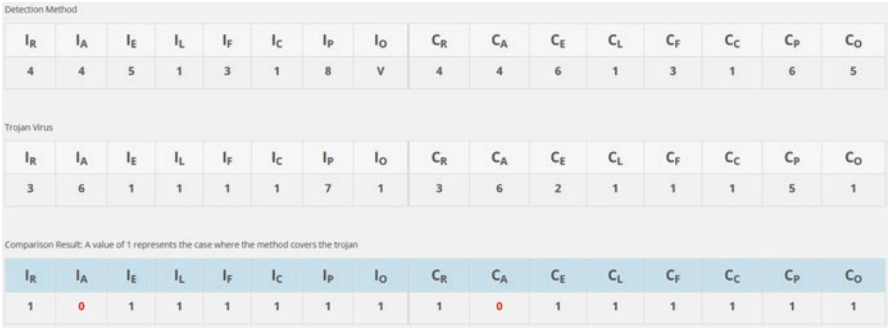


Fig. 4.7 A comparison of coverage and severity vectors

This database will allow the evaluation tool to provide extensive comparison results. Attackers can use this information to design trojans, and defenders can use it to develop security solutions.

4.3.3 The Web Environment

The HTS was designed as a web utility for portability and easy distribution. The application server performs all of the computations and generates page markup to minimize the burden on client side browsers. It communicates directly with a remote database used to store user account information and application data (attributes, categories, and matrices). Both the application server and the database are hosted on the Microsoft Azure Cloud platform (Microsoft 2010). This improves reliability, portability, and flexibility, provides on-demand resources that are automatically managed for scalability requirements, and allows for maintenance to take place anywhere. Figure 4.8 gives a block diagram of the HTS. The application server and database are both hosted on the Azure Cloud (Microsoft 2010). The entity framework provides communication via efficient and secure SQL statements, while Azure provides dynamic resource allocation. When the system is not being used, the architecture is stored in memory to reduce costs. When a client browser attempts to connect to the system, the application server and database are reactivated. Requests and responses are passed between the client side browsers and application server via JavaScript Object Notation (JSON) strings. This allows for complex object-oriented logic to be processed across the network in a simple and efficient manner.

The technologies employed are as follows:

- **Azure**: The Microsoft Cloud System (Microsoft 2010).
- **ASP.NET Web Form**: A user interface focused, event-driven model of the .NET framework. It allows powerful data binding, separation of server-client

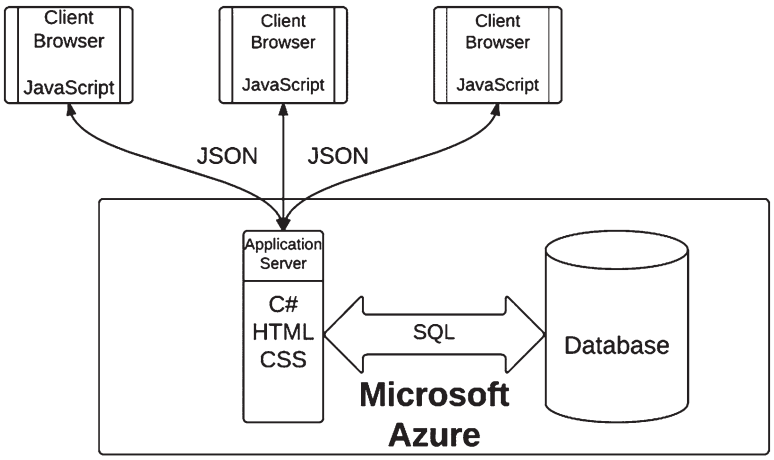


Fig. 4.8 Block diagram of the Hardware Trojan System (HTS)

side activities, a native security structure, and enhanced client performance (Microsoft 2002).

- **Entity Framework**: An object-relational database mapper designed for the .NET framework. It provides a library of high-speed SQL statements wrapped in C# commands to simplify development and ensure performance (Microsoft 2008).
- **D3.js**: A JavaScript library for visualizing data with HTML, SVG, and CSS (Microsoft 2011).

Figure 4.9 provides an overview of the structure of the HTS website. The *home, contact*, *about*, and *application information* pages are accessible to all traffic. The application information page contains three sub-pages providing information on each of the primary applications. Users are required to create an account and be logged in to access the remainder of the website. Email confirmation is used to verify user accounts.

4.4 Case Study

Consider the simple hardware trojan described in Liu et al. (2011). This involves an IC design which contains an arithmetic unit that performs a mathematical operation. The output of this operation is transmitted on the result line in Fig. 4.10.

Suppose an attacker wishes to modify this result when two functional units have been activated in a particular order. Inputs a and b receive signals from the two targeted units. If $a \neq b$, the counter is incremented, and if $a = b$, the counter is decremented. When the count reaches 127, the 8-bit counter outputs a value of 1 causing the inverse of the arithmetic result to be transmitted. The trojan in Fig. 4.10 possesses the attributes listed in Table 4.4.

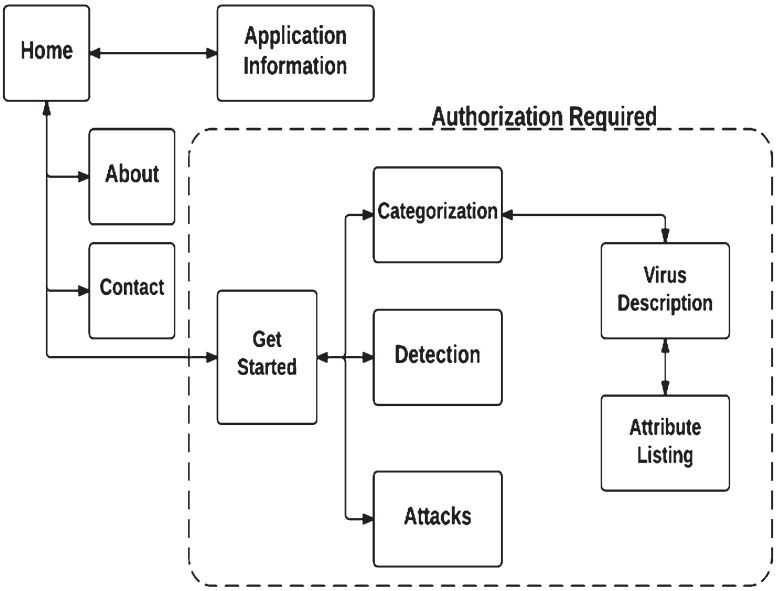


Fig. 4.9 An overview of the website architecture

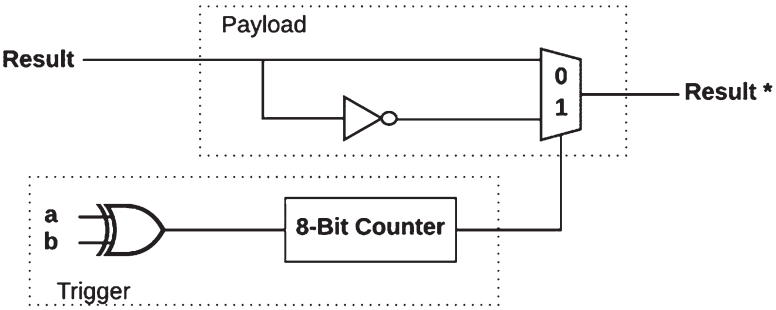


Fig. 4.10 A sequential counter hardware trojan (Liu et al. 2011)

4.4.1 Classification Tool

Suppose a test engineer discovers unusual behavior in a chip under test (CUT). After analysis it is discovered that the additional logic shown in Fig. 4.10 has been inserted into the design. To better understand the characteristics of this trojan, an examination is performed which determines that it possesses the attributes listed in Table 4.4. These attributes are selected using the classification tool, and the subsequent analysis returns the graph shown in Fig. 4.11.

Table 4.4 The observed trojan attributes

Attribute	Category
Change in functionality (12)	Effect
Sequential logic (16)	Logic type
Functional (18)	Functionality
Internally triggered (21)	Activation
Small (24)	Physical layout
Augmented (26)	Physical layout
Clustered (27)	Physical layout

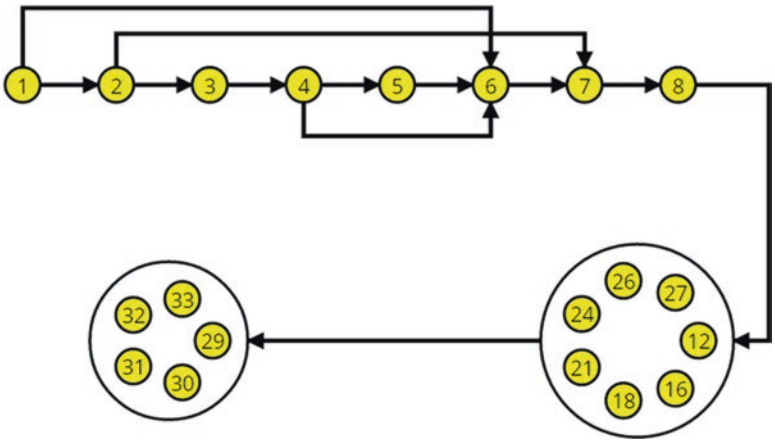


Fig. 4.11 The directed graph for the sequential counter trojan in Fig. 4.10

Figure 4.11 indicates that the trojan could have been inserted in any stage of the IC life cycle. If the test engineer believes that the facility used to fabricate the chip can be trusted, then it is very unlikely that the trojan was inserted during fabrication. With this assumption, fabrication (attribute 3) can be removed from the graph. Using the HTS classification tool attribute removal feature, attribute 3 can be deleted which results in the graph shown in Fig. 4.12.

This indicates that if the fabrication stage (attribute 3) is trusted, then the design, testing, and assembly stages (attributes 2, 4, and 5) can also be trusted. Thus, if the attacker can only access the specification stage (attribute 1) as suggested by Fig. 4.12, then only the following properties could have been observed: change in functionality (attribute 12), denial of service (attribute 15), functional (attribute 18), and always on (attribute 20). It then becomes apparent that the assumption that the fabrication phase could be trusted must have been wrong. In order for the trojan to possess the observed attributes, the attacker must have had access to the fabrication stage (attribute 3).

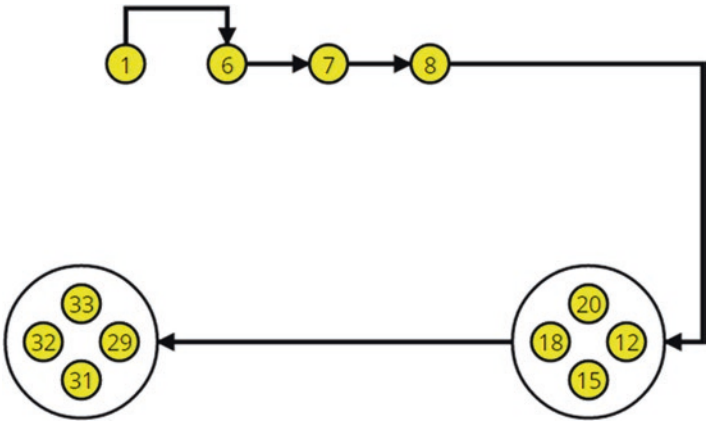


Fig. 4.12 The sequential counter trojan graph after removal of attribute 3

I_R	I_A	I_E	I_L	I_F	I_C	I_P	I_O	C_R	C_A	C_E	C_L	C_F	C_C	C_P	C_O
5	6	1	2	1	2	7	V	5	6	2	2	1	2	5	5

Fig. 4.13 The identification and severity vectors for the sequential counter trojan

4.4.2 Evaluation Tool

Consider an attacker who wishes to attack a system with the trojan shown in Fig. 4.10. The attacker evaluates the trojan and extracts the attributes listed in Table 4.4.

These attributes are then input to the HTS classification tool which returns the trojan graph shown in Fig. 4.11. The identification and severity vectors are also generated as outlined above and given in Fig. 4.13. The classification tool provides a save function which stores the graph, identification, and severity vector of the trojan. Suppose the attacker is a disgruntled employee at the company designing the IC. Being familiar with the manufacturing process, the attacker knows that the test engineers use a trojan detection method based on the path delay (Kumar and Srinivasan 2014). The tool creates the coverage vector given in Table 4.5.

To decide whether or not the attack is viable, the attacker uses the HTS evaluation tool to perform a comparison between the desired trojan and the detection method employed. The evaluation tool creates a new method feature and produces the identification and coverage vectors shown in Fig. 4.14. The trojan severity is selected from the database and the comparison performed as shown in Fig. 4.15.

Table 4.5 Identification and coverage vectors for the detection method in Kumar and Srinivasan (2014)

Techniques	Parameters (I_P)								Coverage (C_P)							
	I_R	I_A	I_E	I_L	I_F	I_C	I_P	I_O	C_R	C_A	C_E	C_L	C_F	C_C	C_P	C_O
Kumar and Srinivasan (2014)	4	4	5	1	3	1	8	V	4	4	6	1	3	1	6	5

Fig. 4.14 The identification and coverage vectors for the chosen detection method

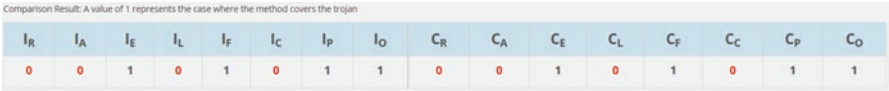


Fig. 4.15 A comparison of the sequential counter trojan and the detection method

As described in Sect. 4.3.2, the HTS evaluation tool displays a value of 1 in categories where the method is capable of detecting the trojan and a 0 otherwise. Figure 4.15 indicates that the detection method may fail to detect the trojan with regard to its insertion point, abstraction level, logic type, and activation.

4.5 Conclusion

A recently developed comprehensive hardware trojan taxonomy can be used to analyze both hardware trojans and detection methods. The complexity of the corresponding techniques motivated the development of online software tools to automate them. The resulting Hardware Trojan System (HTS) is a powerful and reliable means of evaluating and comparing trojans and detection methods. The design of the system was discussed, and the use and effectiveness of the tools were demonstrated with a case study.

The development of data mining algorithms and an appropriate user interface for statistical analysis of the database is left for future work. The implementation of these features will provide a quick and efficient means for designers to evaluate the state of hardware security. Defenders concerned about a particular vulnerability will be able to quickly browse available detection methods for an appropriate solution. An attacker who has found a weakness in a system can browse for existing trojans that can be used.

References

Karri R, Rajendran J, Rosenfeld K, Tehranipoor M (2010) Trustworthy hardware: identifying and classifying hardware trojans. IEEE Comput 43(10):39–46

Kumar P, Srinivasan R (2014) Detection of hardware Trojan in SEA using path delay. In: Proceedings of IEEE students' conference on electrical, electronics and computer science, Bhopal, India, March 2014, pp 1–6

Liu H, Luo H, Wang L (2011) Design of hardware trojan horse based on counter. In: Proceedings of international conference on quality, reliability, risk, maintenance, and safety engineering, Xi'an, China, June 2011, pp 1007–1009

Microsoft (2002) ASP.NET 4.5 [Computer Software]. http://www.asp.net/. Accessed 7 May 2015

Microsoft (2008) Entity Framework v6.0 [Computer Software]. https://msdn.microsoft.com/en-ca/data/ef.aspx. Accessed 7 May 2015

Microsoft (2010) Azure Cloud System [Computer Software]. azure.microsoft.com. Accessed 7 May 2015

Microsoft (2011) D3.js v3.5.6 [Computer Software]. https://d3js.org/. Accessed 7 May 2015

Moein S (2015) Systematic analysis and methodologies for hardware security. PhD Dissertation, University of Victoria, Victoria, BC, Canada

Moein S, Khan S, Gulliver TA, Gebali F, El-Kharashi MW (2015) An attribute based classification of hardware trojans. In: Proceedings of International Conference on Computer Engineering and Systems, Cairo, Egypt, December 2015, pp 351–356

Moein S, Subramnian J, Gulliver TA, Gebali F, El-Kharashi MW (2015) Classification of hardware trojan detection techniques. In: Proceedings of International Conference on Computer Engineering and Systems, Cairo, Egypt, December 2015, pp 357–362

Narasimhan S et al (2013) Hardware trojan detection by multiple-parameter side-channel analysis. IEEE Trans Comput 62(11):2183–2195

Potkonjak M, Nahapetian A, Nelson M, Massey T (2009) Hardware trojan horse detection using gate-level characterization. In: Proceedings of ACM/IEEE design automation conference, San Francisco, CA, July 2009, pp 688–693

Rad RM, Wang X, Tehranipoor M, Plusquellic J (2008) Power supply signal calibration techniques for improving detection resolution to hardware trojans. In: Proceedings of IEEE/ACM international conference on computer-aided design, San Jose, CA, pp 632–639

Wang X, Tehranipoor M, Plusquellic J (2008) Detecting malicious inclusions in secure hardware: challenges and solutions. In: Proceedings of IEEE international workshop on hardware-oriented security and trust, Anaheim, CA, June 2008, pp 15–19

Wolff F, Papachristou C, Bhunia S, Chakraborty RS (2008) Towards trojan-free trusted ICs: problem analysis and detection scheme. In: Proceedings of design, automation and test in Europe, Munich, Germany, March 2008, pp 1362–1365

Chapter 5
Combining Mouse and Eye Movement Biometrics for User Authentication

Hongwei Lu, Jamison Rose, Yudong Liu, Ahmed Awad, and Leon Hou

5.1 Introduction

Biometric authentication verifies a user based on its inherent, unique characteristics—who you are. In addition to physiological biometrics, such as the fingerprint, face, and iris, behavioral biometrics has proven useful in authenticating a user. As an emerging behavioral biometric, mouse dynamics, with their unique patterns of mouse movements, aims to address the authentication problem by verifying users on the basis of their mouse operating style. There are studies such as a mouse dynamics analysis framework that uses mouse gesture dynamics for static authentication (Sayed et al. 2013), a new form of behavioral biometrics based on mouse dynamics using artificial neural networks (Ahmed and Traore 2007), and a mouse movement behavioral biometric that involves image feature extraction using genetic and evolutionary computations (GECs) (Shelton et al. 2013). Besides these, there are studies about using eye movement tracking as behavioral biometrics, for instance, a score-level fusion method for eye movement biometrics (George and Routray 2015), a decision tree-based personal authentication using eye movement tracking (Dhingra et al. 2013), and an information fusion-based biometric identification via eye movement scan paths in reading (Holland and Komogortsev 2011). All these studies show a promising use of mouse movement tracking (MMT) and eye movement tracking (EMT) in behavioral biometrics.

Despite of the promising future, there is still a long way to go for using MMT and EMT for behavioral biometrics in practice. Based on a research by Chen et al. (2001),

H. Lu • J. Rose • Y. Liu (✉) • L. Hou
Computer Science Department, Western Washington University,
516 High Street, MS9165, Bellingham, WA 98225, USA
e-mail: Yudong.Liu@wwu.edu

A. Awad
New York Institute of Technology, Vancouver, BC, Canada

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6_5

there is an 84–88 % correlation between eye and mouse movement. Furthermore, a more recent research shows this correlation depends on the length of experiment time, personal browsing habits, and user's cursor behavior, such as inactive, examining, reading, or moving to perform a click (Huang et al. 2012). Based on the understanding for these studies, we make an assumption that using both MMT and EMT for behavioral biometrics will have some advantages compared to only using MMT or EMT individually. However, to our knowledge no such research has been published.

In this research, we present a behavioral biometrics model for personal authentication using combined features of MMT and EMT. The interface we used is designed to perform moving and click actions in eight directions for participants. Therefore, MMT and EMT will be able to correlate to each other well. The drawback of this system is the requirement of the support of special hardware, an eye-tracking device. The general idea of the system is to exploit the features from both the mouse movement data and the eye movement data and see how well a user verification system can be trained with those features. With the supervised learning setting, we first collect both types of data from a group of participants and process the raw data to remove noise and incomplete data points. Features are then extracted from the cleaned data, and a set of learning methods are trained with those features including a multi-class classification system, a binary classification system, and a neural network-based regression model. Experiments show that the neural network-based model works the best on the large classification task. In the following, we will report the related work and our system in more details.

5.2 Related Work

This research is on the basis of previous researches. Some behavioral features used in the work are inspired by some related work.

5.2.1 Previous Research on Mouse Movements

As a pioneer in this field, Ahmed and Traore (2007) presented an experiment using mouse movement as behavioral biometrics to identify a user. It achieved a false acceptance rate (FAR) value of 2.4649 % and a false rejection rate (FRR) value of 2.4614 %. Some new features such as movement speed and movement direction are applied to model behavior. This research brings idea on modeling behavior and choosing useful features for experiment related to mouse movement. In Jorgensen and Yu (2011), the authors reviewed existing approaches for mouse movement behavioral biometrics and researched the impact of environmental variables on these approaches. The researchers believe result of existing approaches is unlikely to be accurate under controlled environmental variables and certain common remote access scenarios. Therefore finding approaches to lessen impact of environmental variables such as combining mouse dynamic with other types of behavioral biometrics will be a likely

direction of a behavioral biometric research. In Zheng et al. (2011), the researchers are trying to build a system that is robust and has quick response by using point-to-point angle-based metrics of mouse movement to build a user identification system. They achieved FAR and FRR values of 1.3 %. It shows that angle is stable and helpful metrics in behavior biometrics. In a more recent work, Shelton et al. (2013) introduced a new approach that is called genetic and evolutionary feature extraction (GEFE) for feature extraction from image of mouse movement, which provides a good idea in relating feature extraction to image processing.

5.2.2 Previous Research on Eye Movements

Research by Holland and Komogortsev (2011) described how to utilize a fusion algorithm on standard features of eye movement. The standard features include fixation count, average fixation duration, and average vertical saccade amplitude. This fusion algorithm assigns each feature a weight, and then a similarity measure is calculated for user identification. Other features in eye tracking are also helpful for behavioral biometrics. For instance, Rigas et al. (2012) used saccadic velocity and acceleration as eye movement features and applied a nonparametric statistical test to compare the distributions of these features. After that, a k-nearest neighbors classification algorithm is applied for identification. It shows how to utilize the velocity and acceleration of eye movement as behavioral biometrics. In Holland and Komogortsev (2013), researchers investigate the effect of eye-tracking device setting on eye movement biometrics. The result from their experiment suggests that an eye-tracking system with spatial accuracy less than 0.5° and at least 250 Hz temporal resolution is recommended for biometric purposes. That is, in order to conduct a behavioral biometrics experiment using eye-tracking technique, an appropriate hardware and software environment is helpful. In a recent research (George and Routray 2015) about eye movement behavioral biometric, features are extracted from fixation and saccade separately, and then calculated scores for features from each of them using two radial basis function (RBF) neural networks. Then a fusion method is applied to compute a final score based on the two scores. This research describes how to use neural network and fusion method in building a behavioral biometrics system. In our experiment, we also used a neural network-based fusion model.

5.3 Experiment Setting and Design

5.3.1 Experiment Setting

The computer we used is a desktop PC with Windows 8 operating system. There are two monitors where one is extended on the right of the other one. The resolution of both monitors is 1680 pixels by 1050 pixels. The eye-tracking device is SensoMotoric Instruments (SMI) iView RED-m with a sample rate of 60 Hz. The mouse is wired

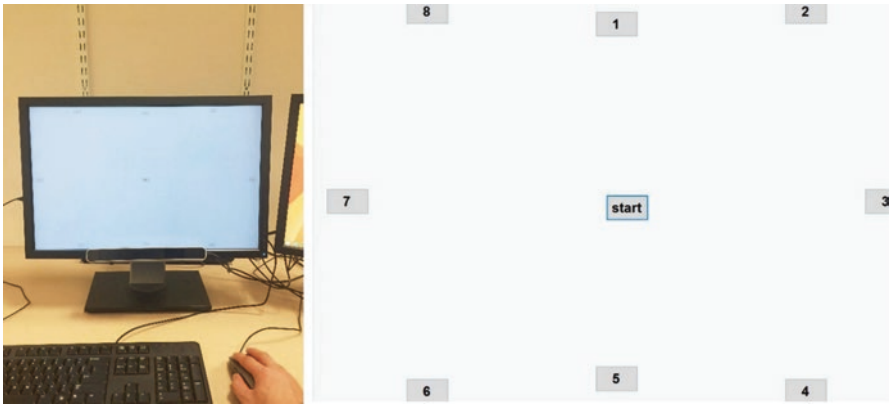


Fig. 5.1 Experiment setup and user interface that's used to collect the data

USB laser one with a dpi of 1200. The eye tracker is mounted at the bottom of a monitor screen horizontally, and the experiment tool will run on this monitor. Figure 5.1 shows the setup.

5.3.2 Participants

Experiment data is collected from 40 participants. Sixteen of them are female and 24 of them are male, and their age ranges from 18 to 58, with an average of 25. Their average number of years of using computer is 16. There are seven of them wearing glasses or contact lenses when performing the experiment.

5.3.3 Experiment Design

Figure 5.1 shows the user interface that is designed for user data collection. It is full screen and has nine buttons on it of which one is located in the center and the other eight are evenly distributed around the screen clockwise starting on the center top. The eight surrounding buttons indicate eight directions. This design is similar with the one used in Ahmed and Traore (2007). The center button is the start button. When a participant clicks on it, the program will start recording mouse and eye movement. The eight surrounding buttons are ending buttons. Clicking on them will notify the program to stop recording. The process that the mouse cursor and eye move from the center button to an ending button of a direction is an action. It means the participant moves his/her eyes toward one direction.

Every time an action is performed, a mouse data file and an eye data file will be generated at the same time. Each line of data in these files will be a time stamp followed by x and y coordinates relative to the upper left corner of the screen. All

these lines together in a file will be a path of the mouse or eye movement toward a direction. Below is an example that shows three lines of a data file where the first column is a time stamp, and the second and third columns correspond to the x and y coordinates at that time stamp, respectively:

24125824134 844.00 470.00
24125824345 846.00 456.00
24125840298 848.00 441.00

5.3.4 Experiment Procedure

When the eye-tracking device is connected to the computer and the experiment program is ready, the participant is asked to sit down in front of the monitor and eye-tracking device. They may be asked to adjust the sitting position to make sure the eye-tracking device can detect his/her eyes properly. Then a five-point calibration step is performed. Calibration is followed by a validation step to validate the eye positions. A valid position is measured by the spatial accuracy that is less than 0.5 for both x and y coordinates. The next step is to run the interface to collect data (see Fig. 5.1). Participants are asked to click buttons in order of 1 to 8, e.g., [start button] → [button 1] → [start button] → [button 2] → [start button], and so on. Participants are asked to repeat this process ten times. Once a participant finishes this part, he/she will be asked to perform the second part of the experiment, which is clicking buttons in random order, e.g., [start button] → [2 button] → [start button] → [8 button] → [start button] → [2 button], and so on. One hundred actions are performed in the order from direction 1 to 8. In both parts, the eye movement data and mouse movement data are collected simultaneously.

5.4 Data Processing and Feature Extraction

5.4.1 Data Alignment

Since the eye movement data and mouse movement data are collected via different programs, they are generated in different frequency. The eye data is collected in a higher frequency. To extract the combined features, these two sets of data have to be aligned. There will be eight pairs of data files that need to be aligned given the data is collected in eight directions. The alignment algorithm takes a mouse data file and the corresponding eye data file as input each time. The alignment is based on each time stamp of the eye movement, because in one time period, the eye movement data has a higher density. The alignment algorithm uses a threshold, which is a time range, to find a corresponding mouse movement data that falls in that time range and then appending the mouse movement data to the end of the eye movement data as one line in the new aligned data file. This process is done on each pair of the eight file pairs and generates files that only contain aligned data.

5.4.2 Data Cleaning

Due to reasons such as the inaccuracy of eye-tracking device, or in the case of blinks, or an accidental wrong operation from a participant when doing experiment, incorrect, incomplete, or improperly formatted data would be generated. An example of incorrect data is that some coordinate values are negative. An example of incomplete data is that a group of data used to indicate an eye or mouse movement path only contains few coordinates. The goal of data cleaning is to remove such incorrect, incomplete, or improperly formatted data. Our data cleaning process is done in three stages.

- Stage 1: Cleaning on raw data

  After data collection, remove data files that only contain few coordinates or are empty. If a mouse data file is deleted, the corresponding eye data file will also be deleted and vice versa.
- Stage 2: Cleaning during data alignment

  In the process of data alignment, coordinates that have incorrect values, such as negative or empty values, or improperly formatted values are removed. After removing these data, some data file may become empty or incomplete, so check and remove these incomplete data files.
- Stage 3: Cleaning before feature extraction

  For features extraction, some data are redundant and need to be removed. When the eye and mouse move from the start point to the end point during the experiment, if either the eye or mouse has arrived at the end point but its coordinates are still being recorded after this arrival, these data are redundant and should be ignored or removed.

  An algorithm is applied to this step. It finds the first point that enters the ending range by following the path from the start point. The ending range is a rectangle area that is centered by the ending button, whose coordinates are constants. All the coordinates recorded after that first entering point will be removed.

  After removing redundant data, some aligned data file may become empty or incomplete, so check and remove these incomplete data files again.

  In some cases, even though a data file contains lots of coordinates, the path based on these coordinates is very short. This indicates that these data points cluster in a small area, and the movement path is incomplete, so this data file needs to be removed.

  Another algorithm applied to this step is to first check the moving distance of eye and mouse. If any one of them is less than a threshold, remove the data file indicating this path. The distance is calculated using the Cartesian distance formula.

5.4.3 Data Visualization

Data visualization is useful for observing what the data looks like, especially in the process of data cleaning. It visualizes aligned eye and mouse movement data. The tool used for visualization simply connects all coordinates toward one direction and displays the paths for eye movement and mouse movement in different color.

Figure 5.2 shows all the movements of eye and mouse from one participant where (a) and (b) show the visualization result before and after the data cleaning is done.

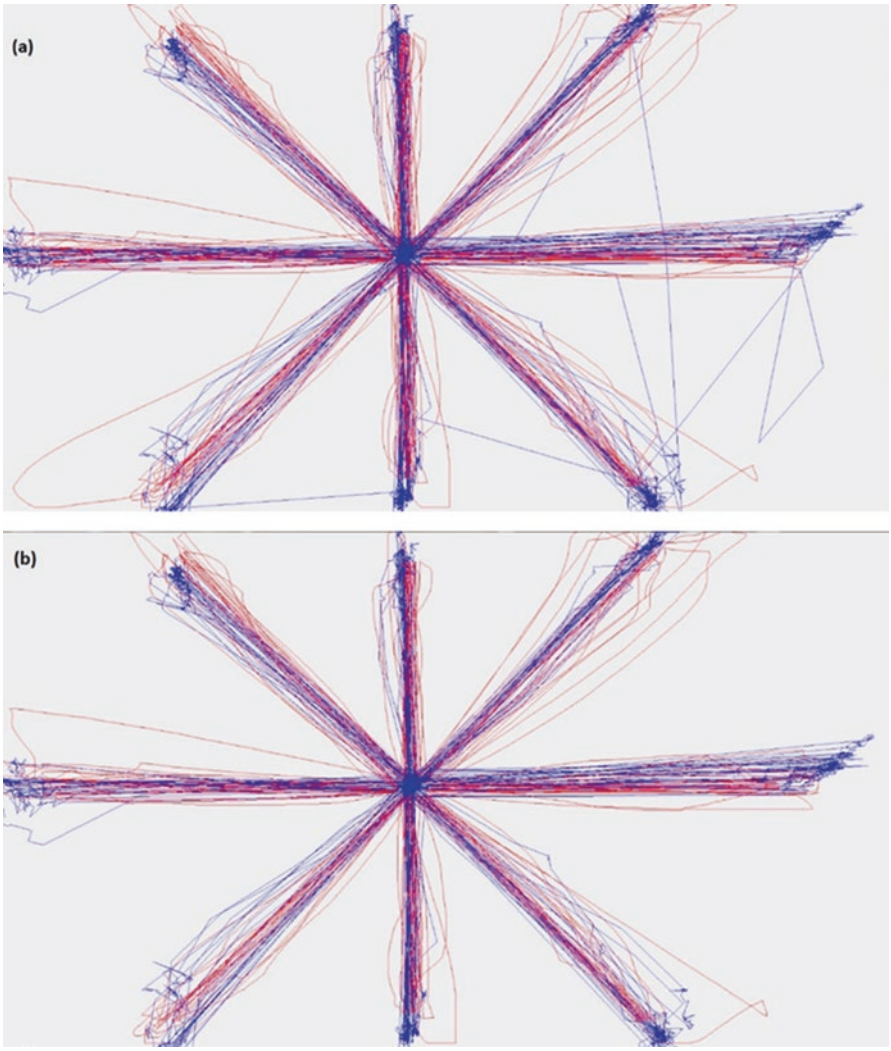


Fig. 5.2 The visualization of eye and mouse movement from one participant (**a**) before data cleaning and (**b**) after data cleaning

The blue line indicates eye movement and red line indicates mouse movement. Data visualization provides us an easy way to observe the difference of the eye and mouse movement paths among different participants. When moving toward one direction, the curve of mouse or eye movement path from one participant is different with the curve from other participants. For example, when comparing data from two participants moving the mouse to same direction, one's curve has a bell shape toward one side, another's curve has a bell shape toward the opposite side. Another difference is the combination of eye movement path and mouse movement path. For instance, some participant's eye and mouse movement paths are very close or have some overlap, but some of those paths from other participants are obviously apart. In addition, the delay of eye and mouse paths is different among participants. The start points of the eye and mouse movement paths are different. So there is a delay between them. It happens because when the start button is clicked, eyes may move faster than mouse cursor or vice versa.

5.4.4 Feature Extraction

Features are selected to characterize eye and mouse movement. In our experiment, eight features are defined. Each data point can be represented in these eight features. Considering the eye gaze and mouse cursor move from the start point to an end point as one action toward a direction, and one data point is generated from one action, these eight features are eye speed, mouse speed, ratio of eye speed and mouse speed, eye angle, mouse angle, deviation of eye angle and mouse angle, delay time, and direction.

Eye speed. It is used to measure how fast eyes move when eye gaze moves from start point to end point. It is an average speed that is calculated by dividing the distance eye moves by the time this movement takes. In each action, the distance and the time begin from the start point and stop when the eye moves into the rectangle area of end point as described previously. The distance and the time period are calculated by getting the difference between the end position and start position of the eye gaze. Getting start position of eye gaze is straightforward. It is where the eye is when the start button is pressed. Finding end position is more complicated. It is the first eye gaze point that enters the area of end point, as described before. And eye movement data recorded after this point will be ignored. The reason why the end position of eye gaze is calculated this way is that starting and ending each action is controlled by the mouse. Since mouse cursor usually moves slower than eye gaze, at the time the eye gaze arrives at an end point the action hasn't been stopped. When the mouse cursor arrives at the end point, eye gaze has been waiting in the ending area for some time. Therefore, this time period shouldn't be taken into account for calculating eye movement speed.

Mouse speed. Mouse speed is calculated in the similar way as eye speed. That is to divide the distance that the mouse moves by the time this movement takes. The start

position is obtained by getting the mouse cursor position when a start button is clicked. The end position is obtained by getting the mouse cursor position when an end button is clicked. The speed will be calculated by the following equation:

$$\text{speed} = \frac{\text{Distance}}{\text{endTime} - \text{startTime}} \tag{5.1}$$

where end time is time stamp of end point and start time is time stamp of start point.

Ratio of eye speed and mouse speed. It is obtained by dividing the eye speed by the mouse speed.

Eye angle. The angle is defined as the acute angle between the vertical line and the line from start position to end position of eye movement. In order to get this angle, first thing we need to do is to find the start position and end position of eye movement, which have been found for the feature of eye speed. After that, arctangent function is used to get the angle. Arctangent function needs an input which is the division of the horizontal distance and the vertical distance between end position and start position of eye movement. The equation is shown below:

$$\theta = \left(\tan^{-1} \left| \frac{(y_1 - y_0)}{(x_1 - x_0)} \right| \right) * \frac{180}{\pi} \tag{5.2}$$

where θ is the angle, $\tan^{-1}$ is the arctangent function, (x_0, y_0) is the start point, (x_1, y_1) is the end point, and π is approximately as 3.14159.

Mouse angle. Mouse angle is calculated in the similar way as eye angle as shown in Eq. 5.2, except here the start position and the end position of mouse movement are used.

Deviation of eye angle and mouse angle. After getting eye angle and mouse angle, this feature is simply calculated by getting the difference of eye angle and mouse angle, which is given by:

$$\text{Deviation} = \theta_1 - \theta_2 \tag{5.3}$$

where θ_1 is the eye angle and θ_2 is the mouse angle.

Delay time. The delay time describes after the start button is clicked, how much later the mouse starts to move compared to the eye. It's defined as the difference of eye movement start time and mouse movement start time in milliseconds.

Direction. Direction is a property of an eye or mouse action. It is indicated by a number from 1 to 8. Direction 1 is up, direction 3 is right, direction 5 is down, direction 7 is left, direction 2 is between direction 1 and direction 3, direction 4 is between direction 3 and direction 5, direction 6 is between direction 5 and direction 7, and direction 8 is between direction 7 and direction 1.

5.5 Proposed Approaches

The features obtained from feature extraction are divided into two sets: one is used for training the model, and the other is used for evaluating the model. Three models including simple multi-class classification model, binary classification model, and regression model using fusion method are applied to implement the personal authentication system. Learning algorithms used in these models are from Accord.NET Framework (http://accord-framework.net/). The latter two models are based on neural networks. In these two models, each user has an individual neural network that would answer whether data belongs to them or not. Each neural network uses a bipolar sigmoid function for its activation function, the Nguyen-Widrow algorithm Nguyen, D., & Widrow, B. (1990) for network initialization and the Levenberg-Marquardt algorithm for training. The training data is normalized by centering it on 0 and scaling it to fit in between −1 and 1 for each of the data values in the input vectors. The testing data is then normalized by using the same way of centering and scaling.

5.5.1 Simple Multi-class Classification Model

In this model, the training data from different participants are labeled with different numbers to model the task as a multi-class classification task. Decision tree, Naive Bayes, and resilient backpropagation are applied for training this model. Each learning algorithm takes as input the feature vectors of eight features as described in Sect. 5.4.4. The output is a number that identifies a user. Besides, the model is tuned by adjusting parameters for those learning algorithms using a small development set.

5.5.2 Binary Classification Model

This model is the first generation of assigning neural networks to each user. In this model, the training algorithm is Levenberg-Marquardt neural network. Each user/class is assigned a neural network and the network is trained for binary classification. Each Levenberg-Marquardt neural network uses bipolar sigmoid function as the activation function, which means the output values are between −1 and 1. The model is trained by feeding it with unlabeled feature data and target data and adjusting parameters such as number of neurons, number of hidden layers, and threshold for bipolar sigmoid function to improve the performance.

This binary classification model uses the same feature vectors as the simple multi-class classification model. It is trained using an input layer of eight nodes which are features from feature vectors, a single hidden layer using a variable number of nodes and a single output layer that outputs a value close to 1 when the user should

be valid and a value close to −1 when the user should be invalid. To distinguish between users, a threshold between −1 and 1 is chosen to determine what should and should not be accepted. Each user's network is trained using the entire training data set allowing for positive and negative examples to be provided.

5.5.3 Regression Model Using Fusion

This model is an improvement of the previous binary classification model. It is the second generation of assigning neural network to each user. It uses Levenberg-Marquardt neural network as learning algorithm and trains the neural network with three features: eye speed, mouse speed, and direction. It focuses on only positive training examples. To do this, each user was assigned two neural networks as shown in Fig. 5.3a. The first one takes the user's mouse speed and direction of movement and outputs the user's eye speed. The second one takes their eye speed and direction of movement and outputs the speed of the user's mouse. There are two types of input for each network: 9-dimensional and 12-dimensional vectors. For the nine-dimensional vector, it has the first value being the speed, the *n*th value was a one where n is the direction the action is in, and all other values are zero. The 12-dimensional has three more input features, mouse angle, eye angle, and delay time, than nine-dimensional vector. These networks are tested by running an input through the network and then checking the amount of error the network has. If this error is below a threshold then it will accept a user. The threshold is first chosen globally for all networks, but this approach did not provide acceptable results, so we create a threshold for each user which provides much better results. The results of the two networks are combined in two ways. We test whether having both networks agree or only requiring one network for acceptance is better. These two ways are two of fusion methods. The other two methods focus on only testing one network for a user.

Another feature implemented in this model is batching the input vectors into sessions. A session is initially composed of a sample action from each of the eight directions and has a size of eight, but sessions containing two samples and four samples from each direction are also tested. To evaluate the sessions, the errors of each sample in the session are averaged and then the average is checked against the threshold. In our experiment, we collect data of ten sessions from each participant, and each session has eight actions (one from each direction), so the total session size should be 80 for each class. However, some actions are incorrect or incomplete (as mentioned in Sect. 5.4.2), so this causes some sessions become incomplete due to missing one or more actions. For instance, for session size 32, there are only 15 classes with four samples from each direction.

Our data set contains data collected from 40 participants. This means there are 40 classes totally. In simple multi-class classification model and binary classification model, we train and test it with all the 40 classes. In regression model using fusion, for session size 8 and 16, all 40 classes are tested, but for session size 32, since the complete sessions are insufficient, it is only able to test 15 classes.

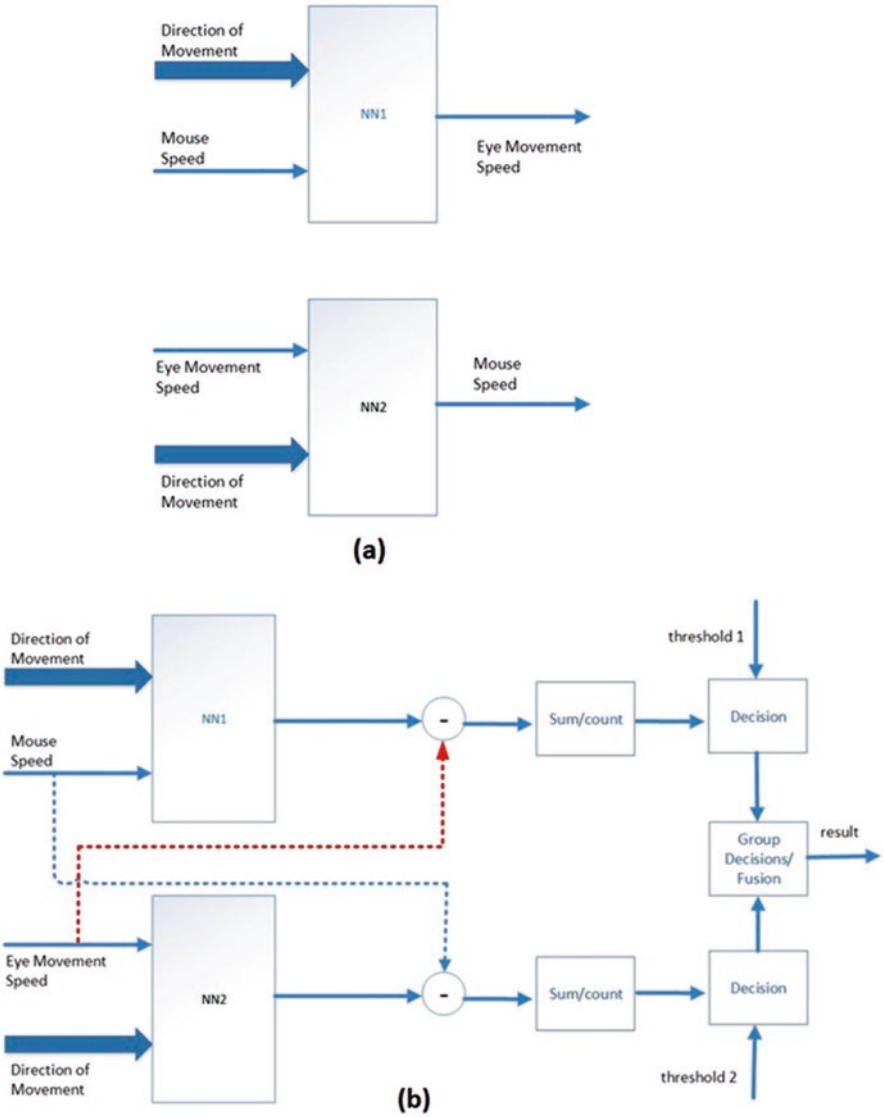


Fig. 5.3 Regression model with fusion. (**a**) Training: two neural nets per user. (**b**) Testing: fusion model

5.6 Result and Discussion

The performance from the simple multi-class classification model is getting poor as there are more classes. The best performance is from a three-class classification model (see Table 5.1). Table 5.1 shows a test with three different learning algorithms using

Table 5.1 Result of three-class classification

Learning method	Class	Accuracy	F-Score	Precision	Recall
Rprop	0	0.88	0.67	0.71	0.63
	1	0.92	0.71	0.67	0.75
	2	1	1	1	1
	AVG	0.94	0.83	0.84	0.83
Naïve Bayes	0	1	1	1	1
	1	1	0.94	0.89	1
	2	0.96	0.93	1	0.88
	AVG	0.99	0.96	0.96	0.96
Decision tree	0	1	1	1	1
	1	1	0.94	0.89	1
	2	0.96	0.93	1	0.88
	AVG	0.99	0.96	0.96	0.96

Rprop resilient backpropagation

80 % data for training and 20 % data for testing from a data set containing about 3200 data points. When there are more classes, such as 40, the precision and recall drop down to less than 1 %. Because of this, we start to explore other approaches such as binary classification.

However, the binary classification model does not perform well either. For 40-class classification, we train it with 40 % of data and test it with 60 % of data from data set. This model achieves very high false acceptance rate (FAR) and false rejection rate (FRR) values. We believe that this is because the amount of negative examples is so much higher than positive examples that they overpower the positive ones.

The regression model using fusion is an improved model on the basis of binary classification model and achieves a much better performance. The result of evaluation for this model is displayed in Table 5.2. In this model, the fusion strategy refers to four types of input and output combination.

- Eye → mouse: only tests output from the network that maps eye to mouse speed.
- Mouse → eye: only tests output from the network that maps mouse to eye speed.
- Both &&: uses both networks and only accepts an input if both networks are accepted.
- Both ||: uses both networks and accepts an input if either network is accepted.

Table 5.2 shows the result of a test for 15 classes using 40 % of data for training and 60 % of data for testing from the data set, and the leftmost column labeled "layers structure" shows the structure of the neural network. For example, 9-10-1 is a network that takes in nine values and has a hidden layer of ten nodes and outputs a single value. The networks that have nine inputs use direction and speed solely. The networks with 12 inputs use the angle and delay time as well. There are four tests for each configuration. The criteria of choosing thresholds for these tests is choosing ones that minimize

Table 5.2 Performance of regression model with session size 32

Layers structure	Fusion	Threshold to minimize EER					Threshold to maximize F-Score				
		F-Score	Precision	Recall	FAR	FRR	F-Score	Precision	Recall	FAR	FRR
9-10-1	E ⇒ M	0.579	0.519	0.8	0.22	0.2	0.629	0.594	0.9	0.157	0.1
	M ⇒ E	0.483	0.39	0.833	0.239	0.167	0.551	0.512	0.9	0.18	0.1
	Both &&	0.689	0.628	0.867	0.156	0.133	0.745	0.707	0.933	0.086	0.067
	Both ||	0.604	0.539	0.833	0.202	0.167	0.685	0.663	0.933	0.12	0.067
9-20-20-1	E ⇒ M	0.408	0.322	0.733	0.325	0.267	0.478	0.401	0.967	0.305	0.033
	M ⇒ E	0.488	0.393	0.867	0.237	0.133	0.515	0.445	0.933	0.21	0.067
	Both &&	0.6	0.528	0.8	0.23	0.2	0.667	0.607	0.967	0.139	0.033
	Both ||	0.558	0.475	0.8	0.245	0.2	0.602	0.533	0.967	0.178	0.033
12-50-1	E ⇒ M	0.594	0.509	0.9	0.176	0.1	0.613	0.522	0.967	0.172	0.033
	M ⇒ E	0.372	0.289	0.767	0.369	0.233	0.424	0.392	0.933	0.358	0.067
	Both &&	0.637	0.571	0.833	0.205	0.167	0.687	0.606	0.967	0.13	0.033
	Both ||	0.566	0.515	0.733	0.277	0.267	0.66	0.612	0.933	0.145	0.067
12-50-50-1	E ⇒ M	0.511	0.416	0.833	0.21	0.167	0.538	0.467	0.967	0.23	0.033
	M ⇒ E	0.442	0.364	0.767	0.279	0.233	0.489	0.39	1	0.29	0
	Both &&	0.613	0.55	0.8	0.203	0.2	0.687	0.606	1	0.15	0
	Both ||	0.541	0.459	0.8	0.23	0.2	0.605	0.536	0.967	0.172	0.033
12-15-15-15-1	E ⇒ M	0.496	0.398	0.867	0.203	0.133	0.531	0.43	0.967	0.192	0.033
	M ⇒ E	0.445	0.381	0.767	0.289	0.233	0.494	0.412	0.967	0.273	0.033
	Both &&	0.643	0.563	0.833	0.18	0.167	0.697	0.634	0.933	0.082	0.067
	Both ||	0.559	0.488	0.8	0.241	0.2	0.632	0.539	0.967	0.135	0.033

Layer structure: structure of layers in neural network
E ⇒ M (eye speed ⇒ mouse speed)
M ⇒ E: mouse speed ⇒ eye speed
Both &&: E ⇒ M && M ⇒ E
Both ||: E ⇒ M || M ⇒ E

Table 5.3 Comparison of best performance of regression model in different session size

Session size	Layers structure	F-Score	Precision	Recall	FAR	FRR
8	12-100-1	0.388	0.464	0.494	0.057	0.506
16	12-50-1	0.637	0.713	0.698	0.042	0.302
32	9-10-1	0.745	0.707	0.933	0.086	0.067

Layer structure: structure of layers in neural network
It shows a best performance for each of the specified session size

equal error rate (EER) or maximize F-Score. The result indicates that the performance is best when we use neural network with four layers, 12 inputs, 24 neurons in second and third layers, and single output. The FAR achieves 8.2 % and the FRR achieves 6.7 %. There are ten different layer structures for neural network that are tested in total, and the result shows that the layer structure of neural network does not have too much impact on performance. Comparing to only use of eye → mouse or mouse → eye, using both && and both || greatly reduce the value of FAR and have higher F-Score. In addition, session size has major impact on the performance.

Table 5.3 shows a best performance for each of the specified session size. Sessions with three different sizes are tested. As the results show, training the model with a session size 32 doubled the F-Score when training it with session size 8. The greatest session size in our tests is 32, and it produces best performance. Therefore, it makes sense to test a session size greater than 32. Due to the insufficiency of complete sessions, it is not able to perform a test with session size greater than 32. We expect to collect more data with complete sessions and test greater session size in future work.

Figure 5.4 shows the ROC curves for one of the 15 users. There are two curves. The mouse threshold curve shows the ROC when the eye threshold is held constant at its best value. Similarly the eye threshold ROC shows the curve when the mouse threshold is held constant at its best value. The curve appears jagged because the number of sessions tested at 32 samples per session was low causing the ROC curve to change in steps rather than smoothly.

5.7 Conclusion and Future Research Direction

The idea of applying combined MMT and EMT in behavioral biometrics is a new exploration in this area. Our experiment shows promising results of using the regression model using fusion on the combined data of MMT and EMT. We conclude that the eye-tracking biometrics, by combining with the mouse biometrics, are a viable method of authenticating a small population of user ($n \leq 15$).

In the future, we plan to continue the study in three directions. Firstly, we would like to conduct a research where the goal is to compare the performance of combined MMT and EMT in behavioral biometrics with only using MMT or EMT. The current fusion model uses the features from both MMT and EMT, which makes it difficult to adapt it to the MMT-only or the EMT-only setting directly.

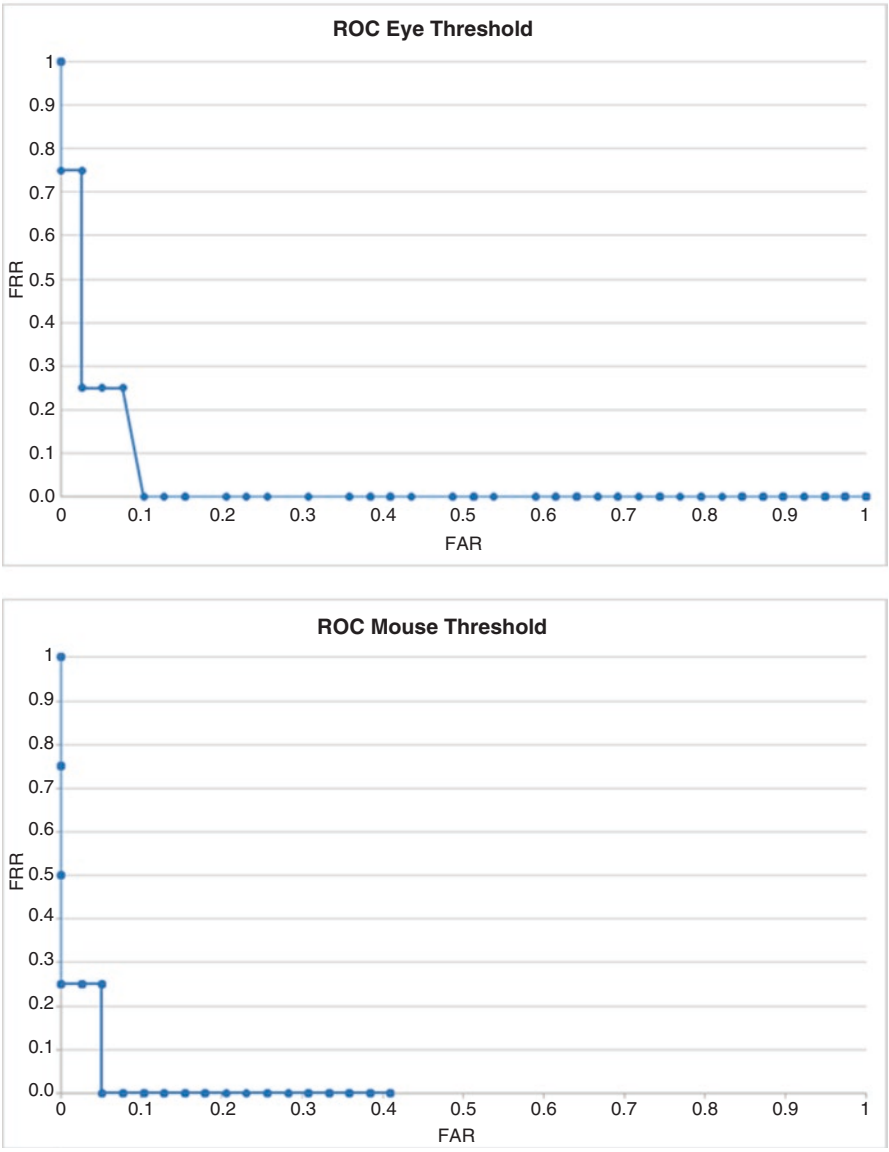


Fig. 5.4 ROC curve of eye and mouse thresholds for one of the 15 users

Secondly, we plan to collect more data of higher quality through a better designed data collecting user interface where a quality checking layer is added. This way we believe a higher quality set of data can be collected, and therefore the data cleaning component will have less impact on the amount of data we end up having. In addition, some new features need to be developed. For example, a feature that captures the curve of the path of eye and mouse movement would be a choice.

Finally, we would like to refine the current model in order to achieve a better performance. Splitting the threshold to be unique for each user makes much better results in the regression model using fusion, so it makes sense to build a new model that assigns a threshold for each direction a user has. In this new model, a neural network is trained for each of the eight directions a user does an action in. Once these neural networks are trained, a new fusion network will be created that takes as input the values that represented errors. The errors will be created by running all of the training data through the trained directional networks. The input for the fusion network is then an error in each direction the user does an action in for both mapping eye speed to mouse speed and mouse speed to eye speed. The output of the fusion network will be then a number between -1 and 1 representing the confidence that a user should be accepted. For testing, more data will be collected in order to test with a greater session size, and the evaluation will be based on the average confidence value rather than the average error.

References

Ahmed AAE, Traore I (2007) A new biometric technology based on mouse dynamics. IEEE Trans Dependable Secur Comput 4(3):165–179

Chen MC, Anderson JR, Sohn MH (2001) What can a mouse cursor tell us more? Correlation of eye/mouse movements on web browsing. In: CHI '01 extended abstracts on human factors in computing systems—CHI '01

Dhingra A, Kumar A, Hanmandlu M, Panigrahi BK (2013) Biometric based personal authentication using eye movement tracking. In: Swarm, evolutionary, and memetic computing lecture notes in computer science, pp 248–256

George A, Routray A (2015) A score level fusion method for eye movement biometrics. Pattern Recogn Lett 82(2):207–215

Holland C, Komogortsev OV (2011) Biometric identification via eye movement scanpaths in reading. In: International joint conference on biometrics (IJCB)

Holland C, Komogortsev O (2013) Complex eye movement pattern biometrics: the effects of environment and stimulus. IEEE Trans Inf Forensic Secur 8(12):2115–2126

Huang J, White R, Buscher G (2012) User see, user point: gaze and cursor alignment in web search. In: Proceedings of the 2012 ACM annual conference on human factors in computing systems—CHI '12

Jorgensen Z, Yu T (2011) On mouse dynamics as a behavioral biometric for authentication. In: Proceedings of the 6th ACM symposium on information, computer and communications security—ASIACCS '11

Nguyen D, Widrow B (1990) Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In 1990 IJCNN international joint conference on neural networks. IEEE, pp. 21–26.

Rigas I, Economou G, Fotopoulos S (2012) Human eye movements as a trait for biometrical identification. In: 2012 IEEE fifth international conference on biometrics: theory, applications and systems (BTAS)

Sayed B, Traore I, Woungang I, Obaidat M (2013) Biometric authentication using mouse gesture dynamics. IEEE Syst J 7(2):262–274

Shelton J, Adams J, Leflore D, Dozier G (2013) Mouse tracking, behavioral biometrics, and GEFE. In: 2013 Proceedings of IEEE Southeastcon

Zheng N, Paloski A, Wang H (2011) An efficient user verification system via mouse movements. In: Proceedings of the 18th ACM conference on computer and communications security—CCS '11

Chapter 6
Ensuring Online Exam Integrity Through Continuous Biometric Authentication

Issa Traoré, Youssef Nakkabi, Sherif Saad, Bassam Sayed, Julibio D. Ardigo, and Paulo Magella de Faria Quinan

6.1 Introduction

The last decade has witnessed a growing interest in the area of continuous authentication, with several publications being produced by the research community and diverse products being released by the industry. Continuous authentication consists of verifying repeatedly the identity of a user throughout computing or online session, with the purpose of preventing identity fraud (Traore and Ahmed 2012).

Identity fraud can broadly be categorized in three classes: identity theft, identity sharing, and identity denial. Identity theft occurs when the identity of an unsuspected user is hijacked by a fraudster and used to conduct malicious activity pretending to be the legitimate user. Vehicles for conducting such attacks include phishing, social engineering, and password cracking.

Denial of identity occurs when an authorized individual conducts illegal actions and repudiates such actions when caught. Typically, this would consist of a malicious insider who repudiates malicious actions associated with their identity.

Identity sharing, also referred to as identity gift, occurs when an authorized individual willingly share their credentials with other users, in violation of established policies and regulations. Illegal password sharing can happen, for instance, in the financial industry to circumvent two-man rules, or for paid subscription services such as Netflix.

I. Traoré (✉)
Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada
e-mail: itraore@ece.uvic.ca

Y. Nakkabi • S. Saad • B. Sayed • P.M. de Faria Quinan
Plurilock Security Solutions, Inc., Victoria, BC, Canada

J.D. Ardigo
Santa Catarina State University, Florianopolis, SC, Brazil

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6_6

A prominent area where illegal credential sharing occurs is online education. With current learning management systems (LMS), students can easily cheat in tests by giving their passwords to others who can take the tests on their behalf. While some Exam Management Systems (EMS) support strong authentication technologies using biometrics, such authentication occurs only statically at login time, this still opens up the door to the possibility for impersonation to occur after the initial login phase.

We propose to address this threat using continuous authentication using a multimodal biometric framework. The proposed multimodal framework combines three complementary biometric technologies: face, mouse dynamics, and keystroke dynamics. All three modalities are collected and processed transparently during the exam without requiring any predefined actions from the test taker.

The proposed framework has been implemented as one the core modules of a new comprehensive exam monitoring platform called ExamShield that has been released recently by Plurilock Security Solutions Inc.

The rest of the chapter is structured as follows. In Sect. 6.2, we discuss and summarize related work. In Sect. 6.3, we present the general architecture of the multimodal biometric framework and its integration in the ExamShield platform. In Sect. 6.4, we discuss the challenges involved in developing our continuous face biometric authentication scheme and give an overview of the approach taken to overcome these challenges. In Sect. 6.5, we make some concluding remarks.

6.2 Related Works

The protection of the integrity of online exams through continuous using biometric technologies is an emerging area of research with relatively few papers (Ahmed and Traore 2011). Furthermore most of the publications, actually, use static biometric authentication.

An example of such line of work has been authored by Ramu and Arivoli by proposing a two-layered approach to address the problem of online exam takers' authentication (Ramu and Arivoli 2013). The two-layered approach combines keystroke biometric authentication and knowledge-based authentication. Although a biometric technology is used, exam participants are authenticated only statically at login time. As mentioned before, this is not enough to prevent cheating from occurring during the course of the exam.

A departure from the above line of work is the approach proposed by Flior and Kowalski who introduced a proof-of-concept implementation of an online exam security system based on continuous keystroke biometric authentication (Flior and Kowalski 2010). In the proposed system, enrolment requires 500 characters collected in a restricted setting (e.g., no backspace or delete is allowed). Furthermore enrolment is performed using fixed text (i.e., predefined text). Similarly, during the exam, verification occurs when 50 keystrokes with no deletion or significant pauses are generated. While the relatively small amount of samples required for enrolment and verification can be considered as a benefit of the system, the restricted nature of these processes will be a significant limitation in real-world deployment. It is not

very realistic to expect an exam to be performed without typos occurring on a regular basis. Furthermore no evaluation of the proposed work has been provided.

Monaco et al. investigated the use of keystroke dynamics and stylometry for continuous authentication of students during online exams (Monaco et al. 2013). Stylometric analysis consists of determining the authorship of a piece of text or document based on the writing style. Like keystroke dynamics, stylometry can be captured transparently using standard keyboard devices. In the proposed work, different studies were conducted using keystroke dynamic and stylometry separately and then by combining both modalities. The combination of both modalities happens at the feature extraction level by concatenating the separate feature vectors into a combined keystroke-stylometry feature vector, which is then submitted to a common classification system. An advantage of this approach over the abovementioned approaches (from the literature) is the use of free text detection, which is crucial to effectively carry continuous authentication. An important limitation, however, is the reliance on a closed-world assumption for authentication. The system relies on a closed population of students serving as basis to train all authorized users. Such assumption is flawed as students cheating in online exams do not necessarily do so with the involvement of other fellow students known to the system. Online cheating may involve sharing credentials with outside individuals totally unknown to the local authentication system.

Fayyoumi and Zarrad developed a prototype for an authentication engine for online exam using continuous face biometric recognition (Fayyoumi and Zarrad 2014). The proposed approach was evaluated by obtaining experts' feedback. Specifically feedbacks were obtained from eight e-learning instructors and 32 students, through a survey using a five-point Likert scale. The proposed examination system includes a question bank which assists instructors in generating randomly different tests for the test takers. Enrolment is performed by capturing and storing images of the user. During the exam, the system tracks the face movement and compares them to the original samples captured during enrolment. A warning is generated in case of suspicion of cheating. A key limitation with the proposed approach is how cheating is characterized. The system relies on facial movement to decide wherever there is cheating or not, which potentially can be a source of large number of false alarms. Furthermore, no evaluation of the performance of the biometric system was conducted. The evaluation was limited as mentioned to the perception of the survey participants mostly on qualitative aspects of the system.

Our proposed framework combines keystroke, mouse, and face biometrics for continuous authentication and does not rely on a closed-world assumption for identity verification. This is made possible by relying only on positive training during enrolment for each of the modalities.

6.3 Online Exam Security: The ExamShield Platform

In this section, we present the ExamShield platform and introduce the general architecture of the underlying multimodal biometric framework.

6.3.1 The ExamShield Platform

ExamShield is a virtual exam center that integrates seamlessly multiple heterogeneous services (multi-biometric authentication, video streaming and recording, exam creation, storage, delivery, and marking). The exam center has been developed as a web portal that can be deployed on the cloud or on premise at the academic institution.

The high-level architecture of ExamShield, depicted in Fig. 6.1, includes the following major services:

1. Exam Management Service provides essential exam management features such as question randomization for different test takers, management of navigation between exam sections, and exam policy enforcement (e.g., exam duration, number of attempt, break time, etc.).
2. Exam Environment Monitoring Service conducts video/audio monitoring of the test taker and surrounding environment using ordinary camera and microphone.
3. Continuous Authentication Service continuously validates the identity of the test taker throughout the exam using a multimodal biometric platform. The platform provides for the first time in an integrated way the following three complementary biometric modalities: mouse dynamics, keystroke dynamics, and facial scans.

Additionally, there is an administrative module which supports technical system administration tasks (e.g., account setup) as well as institutional exam management tasks (e.g., exam scheduling, creating and managing class list and instructors, etc.).

Initially, students are registered to the system by their institutions. Students access their accounts and enroll biometrically once, prior to taking any exam.

Instructors use the system to create and schedule exams. During the exams, instructors access the proctoring panel, where video feeds of the exam participants are displayed. Students are continuously authenticated in the background, and alarms are generated and notified to the instructor through the proctoring panel in real time. Follow-up actions can be taken accordingly by the instructor.

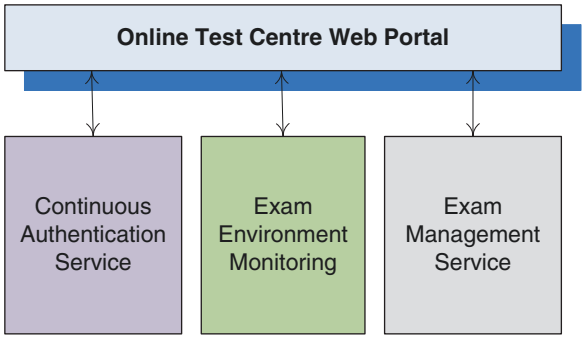


Fig. 6.1 ExamShield high-level architecture

6.3.2 Multimodal Biometric Framework

ExamShield relies on a multimodal biometric framework, through which test takers are continuously authenticated the throughout the exam session from the beginning to the end. The framework involves a combination of several biometric technologies which can be collected and processed transparently in the background without active participation or cooperation of the user.

The multimodal framework integrates three different biometric technologies: mouse dynamic biometric, keystroke dynamic biometric, and face biometric. Mouse and keystroke biometrics are already to be appropriate for continuous as because samples can be collected passively using standard computing devices (e.g., mouse and keyboard) throughout a session without any knowledge of the user. The proposed scheme uses and implements free text analysis and free mouse action analysis models whose theoretical and experimental underpinnings are described in details in Ahmed and Traore (2007, 2014). Interested readers are referred to these publications for details.

Face biometric scans can be collected using standard video cameras, which are currently being shipped with a growing number of computing systems. Facial scans are necessary complement for mouse and keystroke dynamics in order to cover the different monitoring scenarios underlying online exam process. More specifically, while mouse and keyboard may play an active role in written exams, they may be of limited use in exam situations where limited keyboard or mouse interactions are involved, in which cases, facial data could be used to authenticate the user.

However, face biometric has been studied extensively for static biometric authentication; its uses in continuous authentication raise some new challenges since the authentication system has limited control over what the user is doing, which means that there is limited control over the types of samples the application will receive. Hence, it is difficult to capture and analyze effectively biometric samples unobtrusively in a noncooperative environment. Therefore with face biometrics, the recognition must be performed accurately even if images are shifted, or involve different lighting or background, or if the person tilts their head slightly left or right or up or down or angled. This kind of variance between the conditions at enrollment and those at verification times impacts accuracy. Hence, new algorithms must be developed to address the above challenges and ensure effective biometric recognition during online exams. We revisit these issues later in the next section and give an overview of our proposed approach.

6.4 Continuous Face Biometric Authentication

6.4.1 Approach Overview

We designed and developed our continuous face biometric authentication algorithm using local binary pattern and chi-square distance. The model uses only positive training to learn the user's facial features and store the extracted patterns in XML files.

We designed a set of heuristics to improve the accuracy of the system and minimize the false rejection rate.

A major technical challenge in implementing our continuous real-time face recognition over the web was related to the capture and sending of the webcam frames from the user browser to the face recognition server. Existing approaches consist of using communication schemes such as WebSocket to send the captured frames to the face recognition server. However, these do not work for continuous face recognition in a production environment. This is because capturing and sending frames over WebSocket in a continuous setting consume browser resources (e.g., CPU, memory) and result in terminating the WebSocket connection (as in Chrome), or slow down the connection and lose the advantages of real-time authentication (as in Firefox), or even crash the browser and require the user to restart his browser.

The above problem was reported by different developers who were trying to record video stream or send large images or files using WebSocket. Most of the suggested solutions focus on decreasing the video frame rate and connection time. However, this is not possible in our application because the recorded frames are used both for face recognition and user authentication. In addition, an online exam can take up to 4 h and in some cases more than that. To solve this problem and get beyond the current limitations, we took the following steps:

- Use WebP image encoding and avoid using PNG and JPEG image encoding (some browsers do not support WebP).
- Adjust the frame rate and image resolution based on the browser support for a particular image encoding.
- Send binary image not base64.
- Implement a fault-tolerant technique to detect WebSocket connection drop by the browser.

While the matching performances of the above scheme are excellent, its success depends on facial feature being tracked effectively. The OpenCV library is the reference framework for computer vision and face implementations. However, tracking a face in video stream is not an out-of-the-box feature in the OpenCV library. This is because there is no single face-tracking algorithm that can serve different applications. For example, a face-tracking algorithm in video game console is not appropriate for other applications. To address the specific challenges of continuous face biometric authentication, we implemented initially two new techniques to support face tracking. The first technique consists of a motion detection algorithm that calculates the difference between two consecutive video frames and, based on a predefined threshold, decides if a motion exists in the video stream or not. The second technique relies on using existing OpenCV face detection algorithms, and then after the initial detection, it performs a template matching to detect the face template in the new video frames.

These two techniques yielded acceptable performance in laboratory and offline testing environments. However, they did not yield the same performance in online testing with live subjects performing real-world tasks. This is mainly because of the restrictions we have on the video stream. These include the video resolution,

which can be extremely low due to the diversity of exam participants (e.g., low-end Internet connectivity in some countries, heterogeneous platforms); the frame rate per second; and the fact that the system is running in uncontrolled environment, which is a typical characteristic of continuous authentication.

Likewise the existing tracking algorithms publicized through OpenCV do not scale/perform well in real-world environment confronted with the need for the flexibility inherent to continuous authentication. Through thorough search, we could not find in the literature any tracking algorithm that addresses our exam environment constraints (e.g., 5 fps and 320×240 resolution, webcam, and uncontrolled lighting). To address these limitations, we had to make changes to the way the recognition algorithm works and to work with available face frame and do not require a specific number of frame to take the decision. With our new algorithm, even a single frame with one face can be used for recognition, while previously at least 300 frames were required.

6.4.2 Evaluation and Observation

To evaluate the performance of our continuous face recognition system, we divided the evaluation process into three main phases. The first and the second phases focused on evaluating the recognition accuracy, while the third phase focused on evaluating the system in the production environment. In the first phase, we evaluated the detection accuracy of our face recognition algorithm with respect to positive training and novelty detection. To evaluate the detection accuracy when using only positive training, we used existing benchmark facial recognition datasets that are commonly used for evaluating static face recognition algorithms. We used the following three datasets: the AT&T Face Database, the Yale Face Database, and the extended Yale Face Database B. Our face recognition system yielded an accuracy between 91.32 and 94.71 %. These results are very encouraging considering that the algorithm uses only positive training. Most existing face biometric depends on both negative and positive training.

In the second phase of our evaluation, we recorded video streams from 11 subjects. Each subject has to visit our face recognition web application. The video streams were captured using WebRTC and transmitted to our server using web sockets. The server is implemented in python; we used Twisted and Autobahn as our network framework. All the video and image processing are handled by the OpenCV library. About five or six video streams for each subject were recorded. Each video stream is 10–15 min length. These video streams were recorded using a webcam. We used the first 3 min of video data for training. So only 3 min of the 50 min of each subject was used for building the subject face signature. Finally, we merged all the recorded video streams into one big video file and used this file to evaluate our continuous face recognition algorithm. The accuracy of the system in this experiment was 100 %. The system was able to always distinguish the legitimate subject from the imposter subject. While in static authentication our best result was 94.71 %,

in continuous authentication our result was 100 %. Such difference in accuracy is mainly due to the fact that our algorithm was designed for continuous authentication. So it was able to take advantages of the huge amount of data (300 face samples per minute) it has for training and verification in comparison to the limited number of face samples (e.g., 20 face samples) used in static authentication.

The last phase of our evaluation focused on evaluating the system in the production environment. The face recognition server was deployed on the cloud. One instance was deployed on amazon cloud on the west coast, and another instance was deployed on a private cloud hosted by Plurilock Security Solution Inc., in Victoria, BC, Canada. In collaboration with different institutions, students from Canada and Brazil connected to the ExamShield server to perform live exams over several sessions. The students were invited instructed to create their facial signatures prior to taking online exams. The face recognition system was able to record the exam sessions for all the students, perform face recognition and verification in real time, and generate alarms in real time to notify exam proctors. Alarms were generated when a student was taking an exam on behalf of another student, when the student leaves his chair during the exam, or when several students were working on the same exam together. During the production evaluation, most of the reported problems were related to technical problems such as memory leak, connection drop, etc. All these pure technical issues were handled and fixed. The most interesting issues that were reported during the production testing and affected the face recognition functionalities were related to the environmental/external conditions that appear in the exam session. For instance, a major change in the lighting conditions, such as turning the light off during the exam or changing the location of the desk lamp, can badly affect the recognition accuracy. These observations show the need for a real-time adaptation technique to mitigate the effects of the extreme external factors in the exam environment. This will be one of the main focuses of our future work.

6.5 Conclusion

Continuous authentication is an emerging technology which is proving to be appropriate in handling a variety of security threats. Concrete applications range from forensic analysis, detection of insider threat and session hijacking, and various forms of illegal identity sharing. Cheating in online exams falls in the latter category.

This chapter introduces a multimodal biometric framework combining for continuous authentication of online test takers. The framework represents a core module of the ExamShield platform, which is a new online exam monitoring system. In addition to continuous authentication, the ExamShield platform provides live video streaming and recording of exam environments and essential exam management services. The different biometric modalities have been evaluated separately using offline datasets. The biometric framework is currently being used in production in the ExamShield platform yielding very encouraging results.

It is important to highlight that while the biometric framework involves multiple biometric technologies which are complementary, each of the modality is processed separately, and the outputs are presented through separate authentication events displayed using a common dashboard. Likewise, the framework may not technically be considered as full multimodal scheme, as there is no fusion of the outcome of the separate modalities.

In our future work, we plan to address such gap by developing a fusion scheme that will combine the three biometric modalities involved in the framework (i.e., mouse, keystroke, and face) and generate a combined and unique score for overall decision-making.

The effectiveness of a multimodal scheme depends on the appropriateness of the underlying fusion technique used to combine the outcome of the separate modalities. Traditional fusion techniques rely on the availability at the time of the fusion of the separate information being fused. More specifically, the outputs of the separate biometric modalities must be synchronized.

However, synchronizing such a process is not appropriate for continuous authentication as this will delay some of the modalities, which leads to longer verification time.

Our goal is to develop an asynchronous fusion model based on the sequential sampling theory that will allow making a trade-off between accuracy and authentication delay, which is needed in continuous authentication (Ahmed and Traore 2011).

References

Ahmed AAE, Traore I (2007) A new biometrics technology based on mouse dynamics. IEEE Trans Dependable Secur Comput 4(3):165–179

Ahmed A, Traore I (2011) Dynamic sample size detection in continuous authentication using sequential sampling. In: Proceedings of annual computer security applications conference (ACSAC), 5–9 December 2011, Orlando, FL, USA

Ahmed AAE, Traore I (2014) Free text recognition of keystrokes. IEEE Trans Cybern 44(4):458–472

Fayyoumi A, Zarrad A (2014) Novel solution based on face recognition to address identity theft and cheating in online examination systems. Adv Internet Things 4(3):5–12. http://www.scirp.org/journal/ait, http://dx.doi.org/10.4236/ait.2014.42002

Flior E, Kowalski K (2010) Continuous biometric user authentication in online examinations. In: 2010 Seventh international conference on information technology, pp 488–92

Monaco JV, Stewart JC, Cha SH, Tappert CC (2013) Behavioral biometric verification of student identity in online course assessment and authentication of authors in literary works. In: IEEE 6th international conference on biometrics, BTAS

Ramu T, Arivoli T (2013) A framework of secure biometric based online exam authentication: an alternative to traditional exam. Int J Sci Eng Res 4(11):52–60

Traore I, Ahmed AAE (eds) (2012) Continuous authentication based on biometrics: data, models, and metrics. IGI Global. ISBN: 978-1-61350-129

Chapter 7
An Enhanced CUSUM Algorithm for Anomaly Detection

Wei Lu and Ling Xue

7.1 Introduction

Intrusion detection has been studied for decades, and traditionally, intrusion detection techniques include two categories: misuse (signature-based) detection and anomaly detection. Misuse detection is based on the assumption that most attacks leave a set of signatures in the stream of network traffic, and thus attacks are detectable if these signatures can be identified by analysing the network traffic behaviours. However, the biggest limitation of misuse detection is its inability to detect new attacks.

To address the weakness of misuse detection, the concept of anomaly detection was formalized, and most anomaly detection techniques attempt to establish normal activity profiles by computing various metrics, and an intrusion is detected when the actual system behaviour deviates from the normal profiles. The early network anomaly detection systems are self-learning, that is, they automatically form an opinion of what the subject's normal behaviour is. Self-learning techniques combine the early statistical model-based anomaly detection approaches (Hochberg et al. 1993; Lunt et al. 1988; Smaha 1988), the AI-based approaches (Frank 1994) and the biological model-based approaches (Forrest et al. 1996). In this chapter, we proposed an enhanced cumulative sum (CUSUM) algorithm considering its ability in point change detection. As illustrated in Fig. 7.1, the general architecture of our detection scheme consists of two major components, namely, feature analysis and

W. Lu (✉)
Department of Computer Science, Keene State College,
229 Main Street, Keene, NH 03431, USA
e-mail: wlu@keene.edu

L. Xue
Department of Computer Science, Keene State College,
229 Main Street, Keene, NH 03431, USA

City of Keene, Keene, NH, USA

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6_7

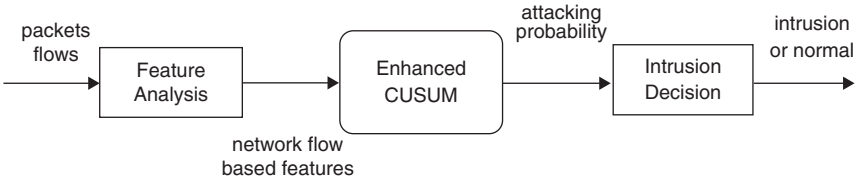


Fig. 7.1 General architecture of the detection framework

enhanced CUSUM-based anomaly detection and decision. During feature analysis, we define and generate six features to characterize the network traffic behaviours, in which we expect the more the number of features, the more accurately the entire network will be characterized. This is very different from the features used by current network anomaly detection systems because most of them use limited number of packet-based features (i.e., number of packets over a time interval) or existing features from public intrusion detection dataset (i.e., 41 features from KDD 1999 CUP intrusion detection dataset) as the information sources. These proposed features are then input to the enhanced CUSUM-based anomaly detection and decision box, in which the final intrusion decision is given through a fuzzy attack probability output by the detection system.

Although the anomaly detection algorithm in this work is not new, the idea of using detection performance for weighting each feature in the anomaly detection in order to achieve higher detection performance is original. The other contribution of this chapter is we propose six network flow-based features which can characterize the network behaviours as completely as possible. The rest of the chapter is organized as follows. Section 7.2 presents the new flow-based features and explains the reasons to select them. In Sect. 7.3, we present the enhanced CUSUM algorithm and describe our probabilistic decision engine for anomalies and intrusions. Section 7.4 presents the network anomalies analysis for the 1999 DARPA intrusion detection evaluation dataset by using our detection system. Section 7.5 makes some concluding remarks.

7.2 Feature Analysis

The major goal of feature analysis is to select and extract significant network features that have potentials to discriminate anomalous behaviours from normal network activities. In order to define our feature vector space, we select three basic metrics to measure the entire network behaviours. In the following, we describe each metric in detail and explain the motivation to choose them.

AverageFlowPacketCount is the first metric we chose, and it is the average number of packets in a flow over a time interval. The rationale behind is most attacks happen with an increased packet count. For example, DoS attacks often generate a large number of packets in a short time in order to consume the available resources quickly.

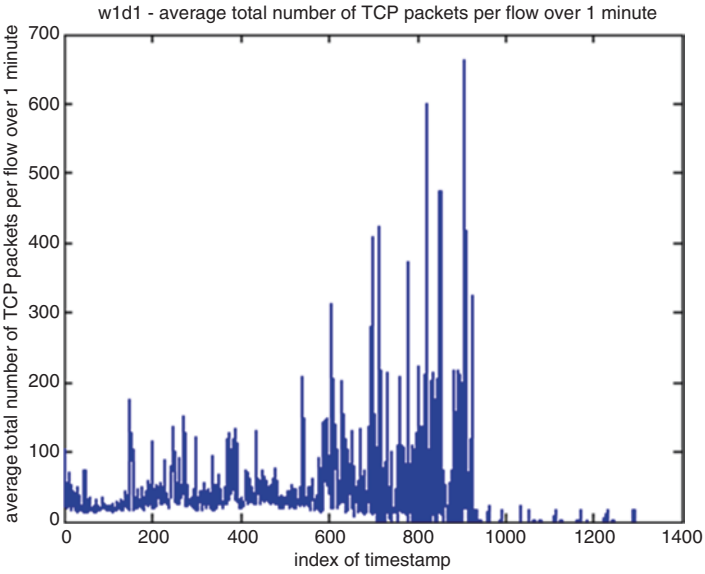


Fig. 7.2 Number of TCP packets per flow per minute over 1 day with normal traffic only

AverageFlowByteCount is the second metric we use, and it is the average number of bytes in a flow over a time interval. By using this metric, we can identify whether the network traffic consists of large size of packet or not. The rationale behind is because some DoS attacks tend to use maximum packet size to consume resources or to congest data paths, e.g., the *ping of death* (pod) attack (Figs. 7.2, 7.3 and 7.4).

Based on the above two metrics, we define a set of features to describe entire traffic behaviours on networks. Let us denote by F the feature space of network flows, a six-dimensional feature vector $f \in F$ can be represented as $\{f_i\}_{i=1,2,\ldots,6}$, where the meaning of each feature is explained in Table 7.1. As illustrated in Figs. 7.2 to 7.3, observations with the 1999 DARPA network traffic data using the features showed that network traffic can be characterized and discriminated through these features. Figures 7.2, 7.3 and 7.4 illustrate the normal network behaviours characterized by the first metric over one day. Similarly, Figs. 7.5, 7.6 and 7.7 illustrate the network behaviours including attacking activities over 1 day. Refer to Figs. 7.5, 7.6, 7.7, 7.8, 7.9, 7.10, 7.11, 7.12, and 7.13 in the Appendix.

Comparing Figs. 7.2, 7.3 and 7.4 and 7.5, 7.6 and 7.7 show that the feature "average total number of packets per flow over 1 min" has the potential to identify *neptune* (*SYN flood*) and *carshiis* attacks. Also as illustrated in Figs. 7.8, 7.9 and 7.10 and Figs. 7.11, 7.12 and 7.13, we see that feature "average total number of bytes per flow over 1 min" has the potential to identify attacks *smurf* and *pod*, to name a few. Overall, the empirical observations with the 1999 DARPA network traffic show that all the six features have the potential to distinguish anomalous behaviours from normal network behaviours.

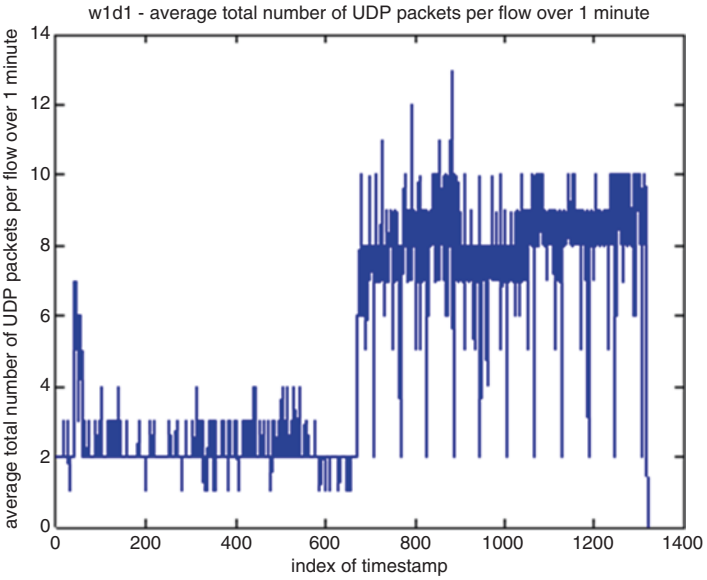


Fig. 7.3 Number of UDP packets per flow per minute over 1 day with normal traffic only

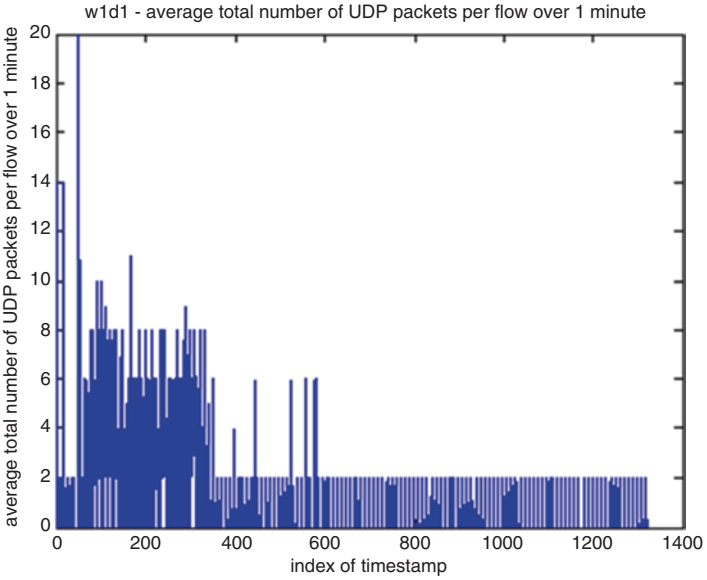


Fig. 7.4 Number of ICMP packets per flow per minute over 1 day with normal traffic only

Table 7.1 List of the six flow-based features

Notation of features	Description
f_1	Average number of TCP packets per flow over 1 min
f_2	Average number of UDP packets per flow over 1 min
f_3	Average number of ICMP packets per flow over 1 min
f_4	Average number of bytes per TCP flow over 1 min
f_5	Average number of bytes per UDP flow over 1 min
f_6	Average number of bytes per ICMP flow over 1 min

7.3 Enhanced CUSUM Metrics

The CUSUM algorithm is an approach to detect a change of the mean value of a stochastic process, and it is based on the fact that if a change occurs, the probability distribution of the random sequence will also change. Basically, CUSUM works on a parametric model of the stochastic process to be analysed. However, obtaining a stochastic model for the Internet traffic is difficult, and also the model usually depends on specific network conditions. Therefore, we apply a non-parametric version of the CUSUM algorithm as an alternative approach in this research work.

Before presenting the non-parametric CUSUM algorithm, we define some notations here. Suppose we want to analyse a random sequence consisting of the number of packets over a time interval Δ, and for simplicity, we define the random sequence $\{X_n\}$ representing the number of packets over Δ. As illustrated in Fig. 7.14, the pattern for sequence $\{X_n\}$ will be observed when there is a flooding DoS attack on networks. The dashed dot line refers to the mean value of sequence $\{X_n\}$, and during a flooding DoS attack, there is a step change of the mean value of $\{X_n\}$ from a to $a+h$ at time point m. The parameter h is defined as the minimum increase of the mean value of $\{X_n\}$ during an attack.

A basic assumption for the non-parametric CUSUM algorithm is that the mean value of the random sequence is negative during normal conditions and becomes positive when a change occurs. Consequently, a transformation of $\{X_n\}$ into a new sequence $\{Z_n\}$ is necessary, which is given by $Z_n = X_n - \beta$, where β is a constant. As illustrated in Fig. 7.15, the parameter β is set according to network normal conditions, and it guarantees that the major part of values of the sequence Z_n is negative during normal conditions and becomes positive when a change occurs.

In practice, a recursive non-parametric CUSUM algorithm is used to detect anomalies online by using a new sequence $\{Y_n\}$:

$$\begin{cases} Y_n = \left(Y_{n-1} + X_n - \beta\right)^+ \\ Y_0 = 0 \end{cases} \quad \text{where } x^+ = \begin{cases} x, & x > 0 \\ 0, & \text{otherwise} \end{cases}$$

where β is set in a fashion that the values of $X_n - \beta$ keep slightly negative during normal operations. As a result, increases in the metric are expected to be detected, once the values are bigger than β. A long time period of values larger than β will lead further increasing of the CUSUM function until a possible alarm level is reached.

A large value of Y_n is a strong indication of an attack. Based on this, we define an attacking probability p_{f_i} generated by feature f_i. It measures the anomalous degree of current networks by feature f_i, where $i = 1, 2, \ldots, 6$. The higher the value of p_{f_i}, the more anomalous the current network. Notation p is the attacking probability and we have

$$p = \sum_{i=1}^{m} p_{f_i} \times W_{f_i} \quad i = 1, 2, \ldots, m$$

where we have p_{f_i} to measure the anomalous degree of initial sequence X_n:

$$p_{f_i} = \begin{cases} \dfrac{Y_n}{\alpha \times \beta}, & Y_n < \alpha \times \beta \\ 1.0, & \text{otherwise} \end{cases}$$

where α is an adjusting parameter, which is used to amplify the value of β and is set as such constant 1, 2, etc. and Y_n is the CUSUM value of sequence X_n.

Since the output of our detection system is a set of attacking probabilities, which are associated with the current network flow data through features. The attacking probability measures the anomalous degree of network flow data. The higher the value of the attacking probability, the more anomalous the corresponding network flow. The network administrator can set two thresholds for the attacking probability in order to discriminate network attacking behaviours from suspicious behaviours or distinct suspicious behaviours from normal network behaviours. In this research, we set the threshold for suspicious behaviour as 0.3 and the threshold value for attacking behaviours as 0.9. Thus, the intrusion decision strategy in our detection model is illustrated as follows:

- If the attacking probability is in the range of [0.0, 0.3], then network behaviour is normal.
- If the attacking probability is in the range of (0.3, 0.9), then network behaviour is suspicious.
- If the attacking probability is in the range of [0.9, 1.0], then network behaviour is intrusive.

This detection strategy is not fixed and in practice; the network administrator can adjust the threshold levels in order to gain an adaptive detection capability.

7.4 Performance Evaluation

We evaluate our system with the six features and 9-day DARPA testing data on week 4 and week 5, in which 201 attacks belonging to 58 attack types (40 new) are used for evaluation (DARPA 1999). During week 4, the inside traffic for day 2 (Tuesday) is missed. During week 5, the total 22 h traffic data is available, and there is no downtime of the network. For the detector using the enhanced CUSUM algorithm, Tables 7.2, 7.3, 7.4, 7.5, 7.6, and 7.7 illustrate its detection results for features F1 to F6 over 9 days of evaluation.

Table 7.2 Detection performance of feature F1 over a 9-day evaluation

Features, days	Total instances	Attacking instances	Normal instances	Total alarms	Correctly detected alarms	False	DR (%)	FPR (%)
F1-W4D1	1320	178	1142	0	0	0	0.0	0.0
F1-W4D3	1320	104	1216	0	0	0	0.0	0.0
F1-W4D4	1320	84	1236	0	0	0	0.0	0.0
F1-W4D5	1320	143	1177	45	7	38	4.9	84.44
F1-W5D1	1320	150	1170	0	0	0	0.0	0.0
F1-W5D2	1320	199	1121	0	0	0	0.0	0.0
F1-W5D3	1320	152	1168	0	0	0	0.0	0.0
F1-W5D4	1320	119	1201	0	0	0	0.0	0.0
F4-W5D5	1320	285	1035	0	0	0	0.0	0.0

Table 7.3 Detection performance of feature F2 over a 9-day evaluation

Features, days	Total instances	Attacking instances	Normal instances	Total alarms	Correctly detected alarms	False	DR (%)	FPR (%)
F2-W4D1	1320	178	1142	0	0	0	0.0	0.0
F2-W4D3	1320	104	1216	0	0	0	0.0	0.0
F2-W4D4	1320	84	1236	0	0	0	0.0	0.0
F2-W4D5	1320	143	1177	145	51	94	35.67	64.83
F2-W5D1	1320	150	1170	26	0	26	0.0	100.0
F2-W5D2	1320	199	1121	0	0	0	0.0	0.0
F2-W5D3	1320	152	1168	0	0	0	0.0	0.0
F2-W5D4	1320	119	1201	0	0	0	0.0	0.0
F2-W5D5	1320	285	1035	0	0	0	0.0	0.0

Table 7.4 Detection performance of feature F3 over a 9-day evaluation

Features, days	Total instances	Attacking instances	Normal instances	Total alarms	Correctly detected alarms	False	DR (%)	FPR (%)
F3-W4D1	1320	178	1142	13	0	13	0.0	100.0
F3-W4D3	1320	104	1216	0	0	0	0.0	0.0
F3-W4D4	1320	84	1236	0	0	0	0.0	0.0
F3-W4D5	1320	143	1177	230	31	199	21.68	86.52
F3-W5D1	1320	150	1170	0	0	0	0.0	100.0
F3-W5D2	1320	199	1121	0	0	0	0.0	0.0
F3-W5D3	1320	152	1168	0	0	0	0.0	0.0
F3-W5D4	1320	119	1201	26	0	26	0.0	100.0
F3-W5D5	1320	285	1035	0	0	0	0.0	0.0

Table 7.5 Detection performance of feature F4 over a 9-day evaluation

Features, days	Total instances	Attacking instances	Normal instances	Total alarms	Correctly detected alarms	False	DR (%)	FPR (%)
F4-W4D1	1320	178	1142	31	9	22	5.06	70.97
F4-W4D3	1320	104	1216	0	0	0	0.0	0.0
F4-W4D4	1320	84	1236	0	0	0	0.0	0.0
F4-W4D5	1320	143	1177	22	3	19	2.1	86.36
F4-W5D1	1320	150	1170	0	0	0	0.0	0.0
F4-W5D2	1320	199	1121	1	0	1	0.0	100.0
F4-W5D3	1320	152	1168	18	7	11	4.6	61.11
F4-W5D4	1320	119	1201	0	0	0	0.0	0.0
F4-W5D5	1320	285	1035	0	0	0	0.0	0.0

Table 7.6 Detection performance of feature F5 over a 9-day evaluation

Features, days	Total instances	Attacking instances	Normal instances	Total alarms	Correctly detected alarms	False	DR (%)	FPR (%)
F5-W4D1	1320	178	1142	0	0	0	0.0	0.0
F5-W4D3	1320	104	1216	0	0	0	0.0	0.0
F5-W4D4	1320	84	1236	0	0	0	0.0	0.0
F5-W4D5	1320	143	1177	220	48	172	33.57	78.18
F5-W5D1	1320	150	1170	0	0	0	0.0	0.0
F5-W5D2	1320	199	1121	0	0	0	0.0	0.0
F5-W5D3	1320	152	1168	0	0	0	0.0	0.0
F5-W5D4	1320	119	1201	0	0	0	0.0	0.0
F5-W5D5	1320	285	1035	0	0	0	0.0	0.0

Table 7.7 Detection performance of feature F6 over a 9-day evaluation

Features, days	Total instances	Attacking instances	Normal instances	Total alarms	Correctly detected alarms	False	DR (%)	FPR (%)
F6-W4D1	1320	178	1142	52	0	52	0.0	100.0
F6-W4D3	1320	104	1216	0	0	0	0.0	0.0
F6-W4D4	1320	84	1236	59	1	58	1.19	98.3
F6-W4D5	1320	143	1177	258	35	223	24.48	86.43
F6-W5D1	1320	150	1170	215	50	165	33.33	76.74
F6-W5D2	1320	199	1121	81	12	69	6.03	85.19
F6-W5D3	1320	152	1168	32	18	14	11.84	43.75
F6-W5D4	1320	119	1201	109	8	101	6.72	92.66
F6-W5D5	1320	285	1035	70	24	46	8.42	65.71

7.5 Conclusions

We propose in this chapter an enhanced CUSUM-based network anomaly detection system. In order to characterize the behaviour of the network flows, we present a six-dimensional feature vector, and the empirical observation results with the 1999 DARPA intrusion detection dataset show that the proposed features have the potential to distinguish the anomalous activities from normal network behaviours. A traffic analysis for the 1999 DARPA intrusion detection dataset is conducted using the proposed network anomaly detection system. Based on the achieved evaluation results, we conclude that even though the number of correct alerts reported by the detection system is not very large, the detection system has the potential to reduce the number of false alerts largely.

7.6 Appendix

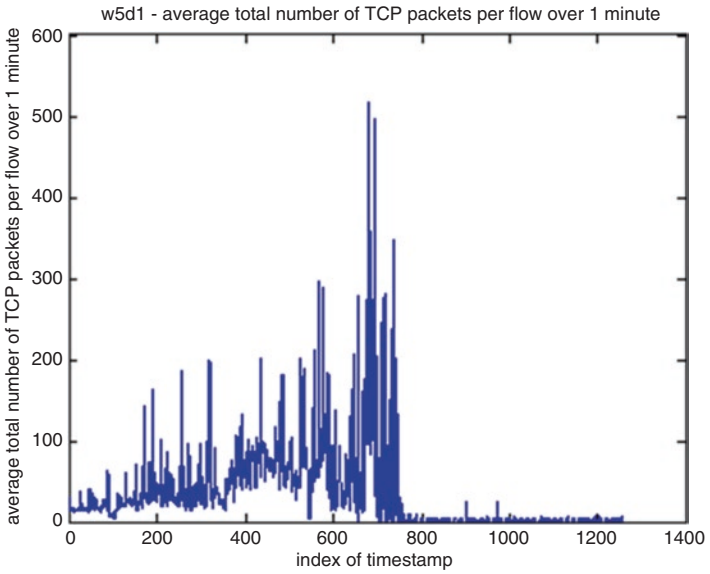


Fig. 7.5 Number of TCP packets per flow per minute 1 day with normal and attacking traffic

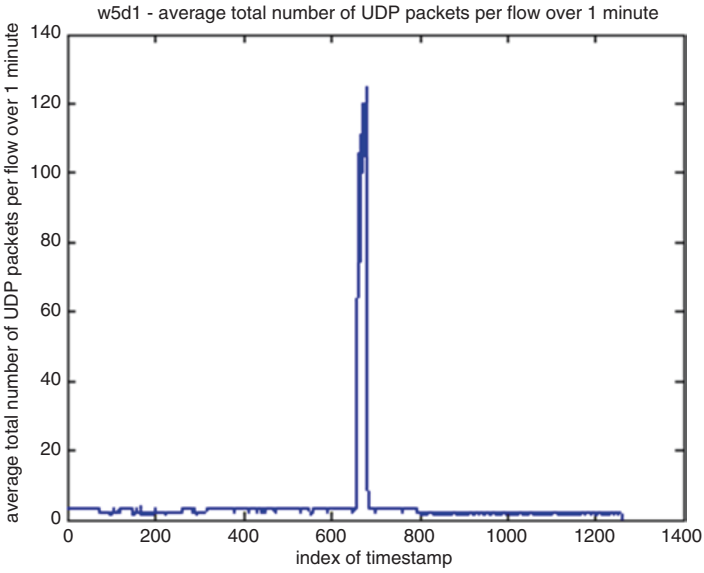


Fig. 7.6 Number of UDP packets per flow per minute 1 day with normal and attacking traffic

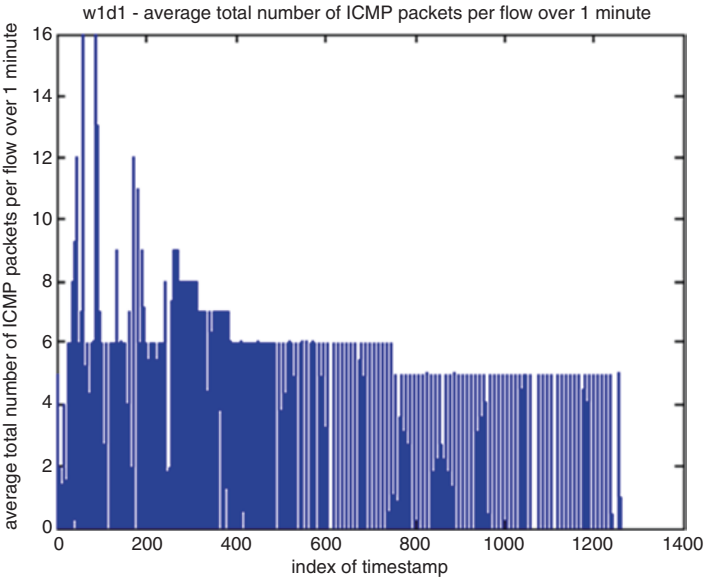


Fig. 7.7 Number of ICMP packets per flow per minute 1 day with normal and attacking traffic

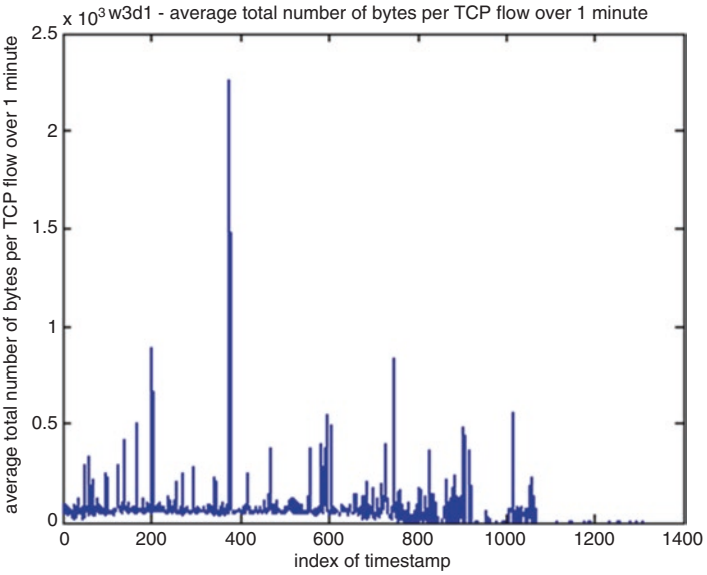


Fig. 7.8 Number of bytes per TCP flow per minute over 1 day with normal traffic only

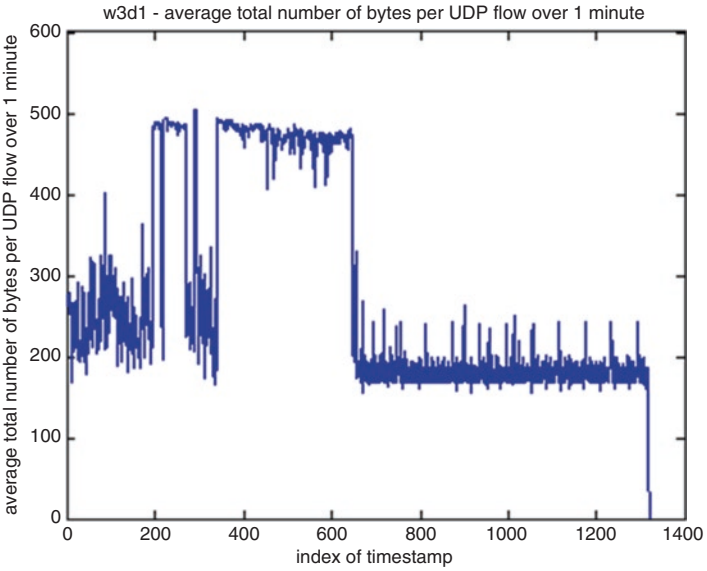


Fig. 7.9 Number of bytes per UDP flow per minute over 1 day with normal traffic only

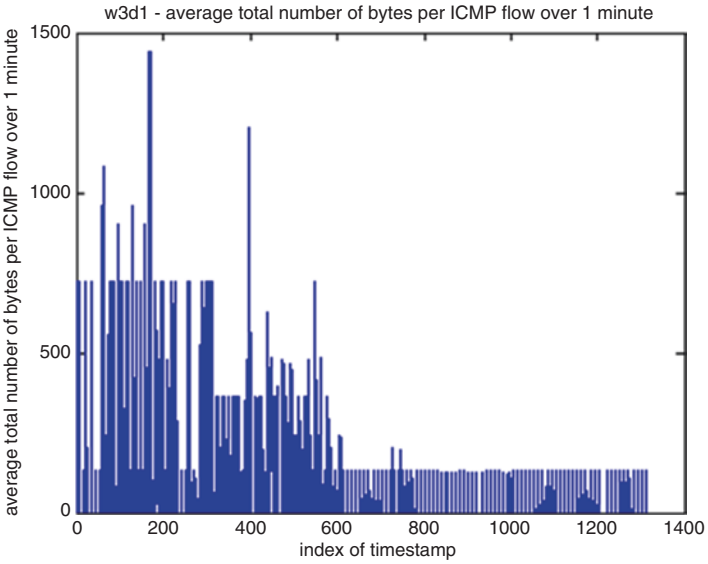


Fig. 7.10 Number of bytes per ICMP flow per minute over 1 day with normal traffic only

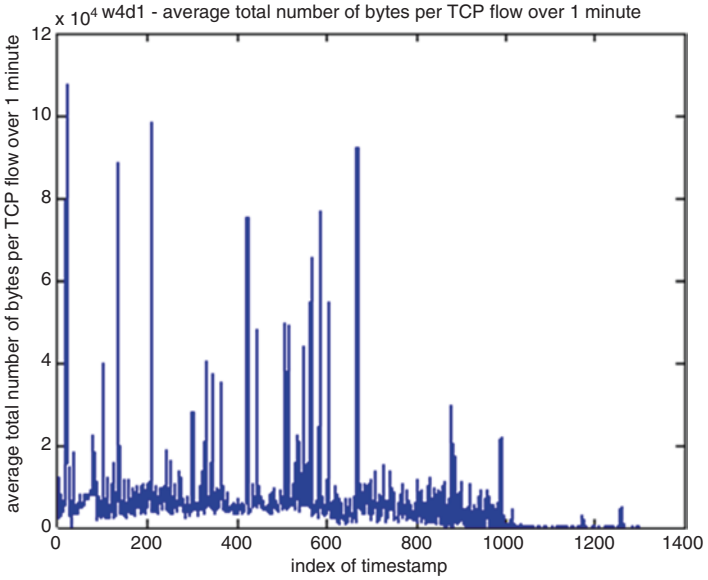


Fig. 7.11 Number of bytes per TCP flow per minute 1 day with normal and attacking traffic

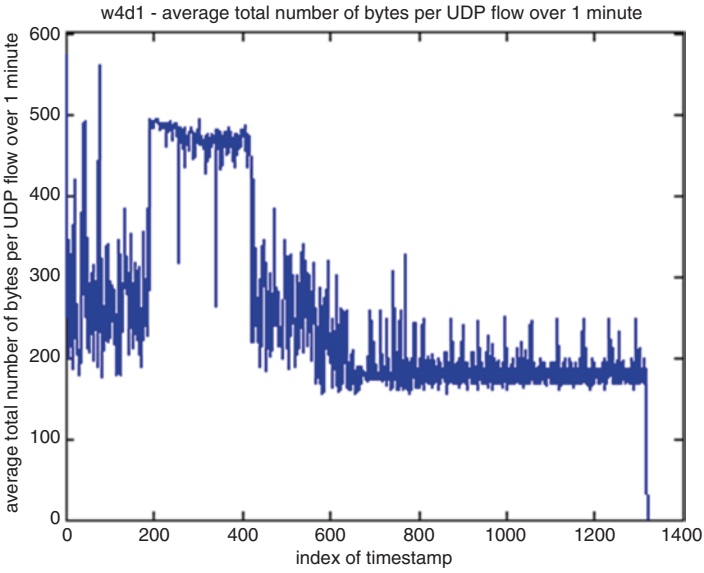


Fig. 7.12 Number of bytes per UDP flow per minute 1 day with normal and attacking traffic

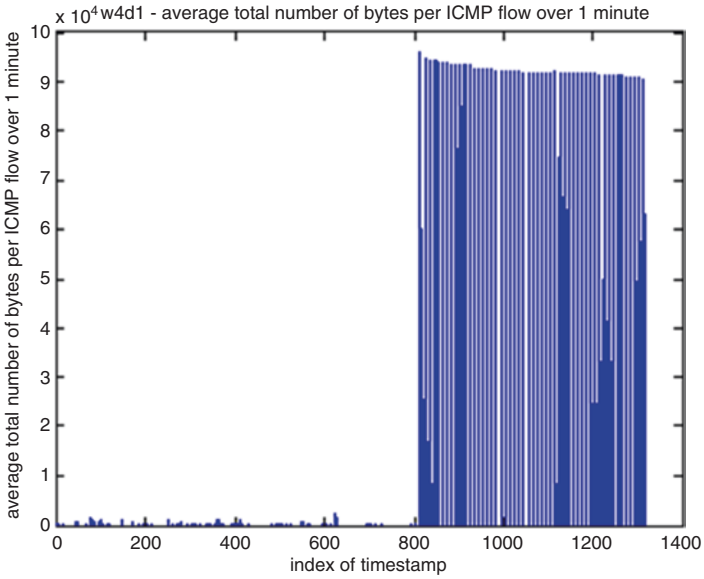


Fig. 7.13 Number of bytes per ICMP flow per minute 1 day with normal and attacking traffic

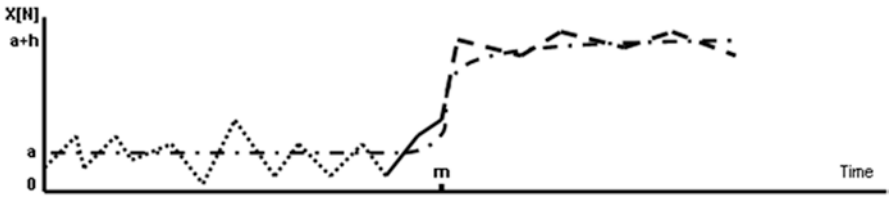


Fig. 7.14 Behaviour of number of packets in a time interval Δ during an attack

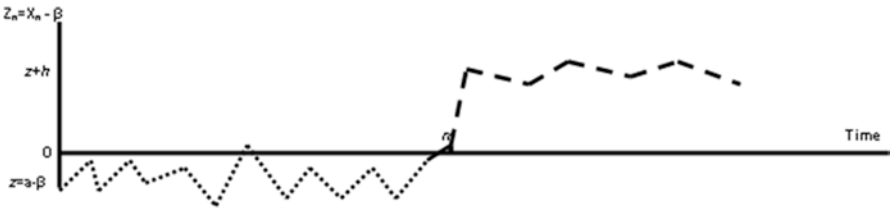


Fig. 7.15 Behaviour of sequence Zn during an attack

References

DARPA (1999) http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html

Forrest S, Hofmeyr S, Longsta A (1996) A sense of self for unix processes. In: Proceedings of 1996 IEEE symposium on security and privacy, pp 120–128

Frank J (1994) Artificial intelligence and intrusion detection: current and future directions. In: Proceedings of the 17th national computer security conference, pp 11–21

Hochberg J, Jackson K, Stallings C, McClary JF, DuBois D, Ford J (1993) NADIR: an automated system for detecting network intrusion and misuse. Comput Secur 12(3):235–248

Lunt T, Jagannathan R, Lee R, Listgarten S, Edwards D, Neumann P, Javitz H, Valdes A (1988) Ides: the enhanced prototype-a real-time intrusion-detection expert system. Technical report, Computer Science Laboratory

Smaha SE (1988) Haystack: an intrusion detection system. In: Proceedings of the IEEE 4th aerospace computer security applications conference, IEEE, Orlando, Florida, December 1988, pp 37–44

Chapter 8
Conclusion: Future Trends and Challenges

Issa Traoré, Ahmed Awad, and Isaac Woungang

One of the trends observed in the emerging threat landscape is the spread of the threats from conventional networks to specialized platforms, including cloud, mobile, Internet of things (IoT), and critical infrastructure networks such as the electrical and utility grids, power and nuclear plants.

Today's workforce is highly mobile, and business activities are no longer limited to the confines of the office or the company-issued desktop. Employees are generating and storing important corporate or institutional data on personal devices, which increases dramatically the level of vulnerability of organizations. Although the increase in worker mobility is good for morale and productivity, it can potentially have a negative impact on the organization systems and data security. In this context mobile devices such as smartphones and tablets are even more vulnerable because of their relatively open environment compared to traditional computing devices (Clarke et al. 2002; Damopoulos et al. 2013). While numerous protection schemes are available on these devices, many users view these protections as hindrances and tend to disable or bypass them (Furnell et al. 2008). In this context, the main challenges for researchers lie in devising new approaches to balance adequately security requirements with the expectations from users to be able to perform primary mobile device functions (e.g., communication) in an unrestricted way.

I. Traoré (✉)
Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada
e-mail: itraore@ece.uvic.ca

A. Awad
New York Institute of Technology, Victoria, BC, Canada
e-mail: Ahmed.Awad@nyit.edu

I. Woungang
Department of Computer Science, Ryerson University, Toronto, ON, Canada
e-mail: iwoungan@scs.ryerson.ca

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6_8

Efforts are underway to migrate the traditional electric power grid using the smart of information and communication technologies (ICT), resulting in the so-called smart grid. While this improves effectiveness of service delivery and cost efficiency, it exposes the smart grid network to several security concerns, some reminiscent of issues already known for conventional computer network (e.g., DDOS attacks), but others are very specific to smart grid environments, technologies, and protocols (Wang and Lu 2013).

Recently, we have noticed a growing interest in the IoT, which is a new computing and design paradigm addressing the proliferation of devices directly connected to the Internet. The focus so far has been on addressing challenges arising from the heterogeneity and ubiquity of these paradigms. However, the provision, operation, and usage of IoT involves serious privacy and security concerns which will increase in complexity as the user base increases and hackers start having better grasp of the underlying technologies (Heer et al. 2011).

Simply reusing and adapting existing protection technologies and strategies for these specialized platforms is not enough to alleviate the underlying security concerns and vulnerabilities. New defensive approaches and models must be developed which take into account the specific attributes and characteristics of these platforms.

Many of these specialized platforms rely on relatively closed networks. Hence, most of them are closely held and controlled by the providers. While this limits the amount of information available publicly and that can be leveraged to launch an attack, it relies on the false assumption of security by obscurity. The lack of information is compounded in the difficulty for researchers to access or create realistic datasets for security study related to these platforms.

The consequence of such reliance on security by obscurity is that determined and clever hackers can devise and execute quietly sophisticated attack methods against these platforms for an extended period of time without being caught.

For instance, for some time it was believed that cloud computing networks were immune to the threat of botnet, since these networks are tightly controlled by cloud hosting companies.

However, it has been shown in the last few years that the potential for botnet spreading over cloud networks is even much greater than in conventional networks (Graham et al. 2015). For instance, it was reported in 2009 that hackers compromised a site on Amazon EC2 and use it to deploy and operate the C&C server for the Zeus banking botnet. In 2014, researchers have shown how easy it is to establish and operate a cloud botnet using a collection of machines from free trials and freemium accounts offered by cloud hosting companies to incentivize new customers.

In the threat context outlined above and throughout this book, while future security challenges lie in specialized platforms, cooperation with the providers to gain more access and generate realistic datasets will be crucial to obtain any successful results in fighting against and anticipating emerging and future cybersecurity threats.

The goal of the I-SAT workshop series is to create and foster a space for researchers and practitioners to present and confront ideas that will represent a leap forward and proactive perspective of emerging and new cybersecurity threats.

References

Clarke NL, Furnell SM, Reynolds PL (2002) Biometric authentication for mobile devices. In: Proceedings of the 3rd Australian Information warfare and security conference, 28–29 November 2002, pp 61–69

Damopoulos D, Kambourakis G, Gritzalis S (2013) From keyloggers to touchloggers: take the rough with the smooth. Comput Secur 32:102–114

Furnell S, Clarke N, Karatzouni S (2008) Beyond the PIN: enhancing user authentication for mobile devices. Comput Fraud Secur 2008(8):12–17

Graham M, Winckles A, Sanchez E (2015) Botnet detection within cloud service provider networks using flow protocols. In: 13th IEEE international conference on industrial informatics. Cambridge University

Heer T, Garcia-Morchon O, Hummen R, Keoh SL, Kumar SS, Wehrle K (2011) Security challenges in the IP-based internet of things. Int J Wireless Pers Commun 61(3):527–542

Wang W, Lu Z (2013) Cyber security in the smart grid: survey and challenges. Comput Netw 57:1344–1371

Index

I. Traoré et al. (eds.), *Information Security Practices*,
DOI 10.1007/978-3-319-48947-6