TOWARDS EVASIVE ATTACKS:
ANOMALY DETECTION RESISTANCE ANALYSIS ON THE INTERNET

by

Jing Jin
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____	Dr. Jeff Offutt, Dissertation Director
_____	Dr. Brian Mark, Committee Member
_____	Dr. Angelos Stavrou, Committee Member
_____	Dr. Robert Simon, Committee Member
_____	Dr. Stephen Nash, Senior Associate Dean
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: _____ Fall Semester 2013
George Mason University
Fairfax, VA

Towards Evasive Attacks: Anomaly Detection Resistance Analysis on the Internet

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Jing Jin
Master of Science
Huazhong University of Science and Technology, 2006
Bachelor of Science
Huazhong University of Science and Technology, 2003

Director: Dr. Jeff Offutt, Professor
Department of Computer Science

Fall Semester 2013
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my parents, my husband and daughter.

Acknowledgments

I would like to acknowledge many people for helping me during my doctoral work. I would especially like to thank my advisor, Prof. Jeff Offutt, for his generous time and commitment. Throughout my doctoral work he has always encouraged me to develop independent thinking, research skills and rigorous writing skills. His words of encouragement, quiet urgings and careful reading of all of my written work will never be forgotten.

I am also very grateful for having an exceptional doctoral committee and wish to thank Dr. Brian Mark, Dr. Robert Simon and Dr. Angelos Stavrou for their continual support and encouragement. They are valuable faculty members at Mason and provided timely comments about my dissertation proposal and final dissertation.

I also wish to express my sincere gratitude to a number of other people for helping me make it through graduate school. Prof. Haining Wang from College of William and Mary collaborators for important parts of this research. He gave me the opportunity to work with his team on the web bots detection project. Lisa Nolder patiently answered all the questions I had about the graduate program. I also would like to thank my company VMware, especially vSECR team that is very supportive to the last stage of my dissertation research.

I would not come to this far without support from my family and friends. Thank my parents for building a strong supportive system to make it possible. Thank my husband Feng Mao as he is a great family leader to carry all the burdens so that I can focus on finishing my dissertation. My little baby girl Angela Mao has offered her beautiful smile to comfort me along the tough journey. I am thankful to my dear friend Jennifer Espinola, Carrie Zheng, Donglin Wang and Anyi Liu for their support, friendship and trust.

I also place on record, my sense of gratitude to one or all who, directly or indirectly, have lent their helping hand in this venture.

Table of Contents

List of Tables

List of Figures

Abstract

TOWARDS EVASIVE ATTACKS: ANOMALY DETECTION RESISTANCE ANALYSIS ON THE INTERNET

Jing Jin, PhD

George Mason University, 2013

Dissertation Director: Dr. Jeff Offutt

The Internet is rapidly improving as a platform for deploying sophisticated interactive applications especially in Web 2.0. Although the shift from desktop-centric applications brings many benefits to web-based computing and cloud computing, such as efficient communication with ubiquitous access and availability, the way that Internet users share and exchange information also opens their own information to security problems. Today, attackers conduct malicious activities to routinely track the identities of internet-connected users, steal privacy data, abuse users personal information, and expose the users unwanted data or programs. Although these attackers can also accomplish these goals by other means, the Internet has made it much easier for attackers to locate victims, discover sensitive information and initiate unsolicited communication with the victims.

To detect attacks from the Internet, anomaly detection methods have been proposed to compare abnormal behavior from malicious activities with legitimate behavior. While detection techniques have been developed, evasive techniques have not been widely studied. This dissertation explores the limitation of current anomaly detection in the context of the battle between detectors and attackers by finding potential evasive attacks and measuring detection resistance of evasive techniques.

This dissertation studies detection resistance at user application and IP layer. This dissertation first explores the limitations of current Human Observational Proofs (HOP) based bot detection systems by creating a new evasive bot system that masquerades as human beings on the Web. Specifically, I characterize the existing HOP-based web bot detectors and develop an evasion framework based on human behavior patterns. Instead of subverting a specific detection system, the major goal of this study is to provide a systematic approach to evaluate and explore the limitations of current HOP-based detection systems on the web. This dissertation also explores the limitations of IP timing covert channel detection systems by analyzing the stealthiness of timing covert channels. For evasive techniques, this dissertation studies passive detection resistance and active detection resistance with various evasive methods such as mimic, mix and replay, coding scheme rotation, etc. It defines a new measurement approach to evaluate covert channel evasion capabilities. The major goal of this study is to provide a systematic approach to better understand the design of IP timing covert channels.

Both studies use similarity measurement that measures the similarity between legitimate behavior and abnormal behavior. This similarity measurement evaluates the capability of evasion against detection methods with detection independent approach.

1. Introduction

The internet is rapidly improving as a platform for deploying sophisticated interactive applications, as people start to use the internet to share information with others. The web can be viewed as a large, transparent database that its information can be retrieved and updated from time to time [1]. Although the shift from desktop-centric applications to web-based computing and cloud computing brings many benefits, such as efficient communication with ubiquitous access and availability, the way that internet users share and exchange information also opens their own information to new web-related security and privacy problems. Today, attackers routinely track the identities of internet-connected users, steal privacy data, abuse users' personal information, and expose users' unsolicited data or programs using malware. Although these attackers can also accomplish these goals by other means, the web has made it much easier for attackers to locate victims, search private information and initiate unsolicited communication with the victims. Therefore, internet users have raised concerns on attacks that can cause billions of dollars in loss [2–5].

Due to the challenges in security and privacy community [6], this dissertation first focuses on anomaly detection. While detection techniques have been developed, evasive techniques also come into play to bypass anomaly detection. This dissertation further explores the limits of current anomaly detection in the context of the battle between detectors and evaders by finding potential evasive attacks and measuring detection resistance of evasive techniques.

Anomaly detection refers to detecting patterns in data sets that do not conform to an established normal behavior. The patterns thus detected are called *anomalies* and often translate to critical and actionable information in several application domains. Anomalies are also referred to as outliers, change, deviation, surprise, aberrant, peculiarity, intrusion,

etc. [7]

Anomaly behaviors might be induced into the data for a variety of reasons in information security, usually as malicious activity. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized destination. A web bot as an anomalous user against a web application could mean sending unsolicited data or crawling sensitive data from users. A covert channel carried on an overt channel can be used by attackers to leak sensitive information on the cloud [8, 9].

Anomaly detection at a higher level was to find patterns that do not conform to expected normal behavior. Therefore, it defines a region representing normal behavior and declares any observation in the data that does not belong to this normal region as an anomaly. Existing research has proposed a wide range of detection techniques [10–12]. Particularly, supervised anomaly detection techniques have been heavily discussed. Supervised anomaly detection requires a data set that has been labeled as 'normal' and 'abnormal' and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection) [7].

On the contrary, attacks also evade detection by hiding their intentions in normal behavior [5, 13]. These attacks mimic normal behavior from human users, such as sending traffic that mimic legitimate traffic. The evasive attacks tend to be similar to the normal behavior and hence are difficult to distinguish and detect. The evasive technique declassify detection methods that use machine learning classification techniques. Therefore, I believe it is essential to understand the threat of evasion.

1.1 Problem Statement

This dissertation facilitates a better understanding of how evasive techniques can be developed to increase detection resistance, while motivating a better understanding of existing directions in which research has been done on anomaly detection, and new directions that

can improve anomaly detection. This dissertation studies evasive attacks and anomaly detection resistance and answers the following questions:

- What evasive attacks can be detection resistant?
- How to develop evasive attacks to mimic normal behavior?
- How resistant to detection can attacks be?
- How can we evaluate the capability of developed attacks?
- What properties of communication systems would affect evasion and detection?

This dissertation answers the above questions in two applications: web bot detection and timing covert channel detection, and their evasive techniques. This dissertation first explores the limits of current robot detection systems by creating a new evasive robot system that masquerades as human beings on the Web. Specifically, it characterizes the existing Human Observational Proofs (HOP) based web robot detectors and develop an evasion framework based on human behavior patterns. This evasion technique is to mimic legitimate behavior. Instead of subverting a specific detection system, the major goal of this study is to provide a systematic approach to evaluate and explore the limits of current HOP-based detection systems on the web.

This dissertation later explores the limits of current covert channel detection system by analyzing the stealthiness of covert channels. For detection, this dissertation uses *shape test* and *regularity test* as examples. For evasive techniques, this dissertation surveys various evasive methods such as mimic, mix and replay, coding scheme rotation, etc. It defines a new measurement approach to evaluate noisy covert channel evasion capabilities under the constraint of channel capacity and reliability. The major goal of this- study is to provide a systematic approach to better understand the design of covert channels.

Both of the two applications use similarity measurement that basically measures the similarity between legitimate behavior and abnormal behavior. This similarity measurement

implies the capability of evasion against detection methods, since it is more challenging to differentiate normal and abnormal behavior when they appear to have very similar patterns.

1.2 Research Contributions

This dissertation makes the following contributions:

- Characterizes web robot detection observation proofs that use web application events, with the focus on the different events and their statistics that are associated with the activities of robots and human clients, such as timing and location.
- Proposes a generative approach to build an evasive web robot based on human behaviors. The proposed web robot can mimic human behaviors to hide its activities within legitimate user events; in other words, de-classifying robotlike traffic.
- Provides a practical framework for evasive web robot design and implementation, which helps researchers and engineers explore the limitations of their robot detectors.
- Implements a prototype of the proposed evasive robots, and validates the power of evasive robots by measuring the similarity on various metrics from selected web applications.
- Defines a novel detection resistance metric that functions as a fundamental parameter for the design of evasive timing covert channel schemes.
- Analyzes covert channel evasion from detection resistance metric. Identifies the relationship between evadability and channel noise.
- Surveys evasive techniques such as *Mimic*, *Replay*, *Mix*, *Coding Scheme Rotation* etc.
- Surveys detection techniques such as *shape test* and *regularity test*.
- Conducts experiments to evaluate evadability.

1.3 Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 reviews existing anomaly detection techniques on bots and covert channels and defenses. Chapter 3 presents the design of evasive bots and the parameter estimation framework. Chapter 4 evaluates the detection resistance of timing covert channels. Lastly, chapter 5 concludes the dissertation and discusses future work.

2. Background

This dissertation addresses two challenging problems of anomaly detection and evasion in network and system security. The first one is how to best design web bots evasion techniques using statistical behavior from human beings. The second is a study of how to analyze and measure covert channels detection resistance. This chapter introduces anomaly detection fundamentals, the current state of the art of web bots and covert channel detection. This chapter serves to motivate the research problems for the following chapters.

2.1 Anomaly Detection

Anomaly detection refers to detecting patterns in data sets that do not conform to an established normal behavior. The patterns thus detected are called anomalies and often translate to critical and actionable information in several application domains. *Anomalies* are also referred to as outliers, change, deviation, surprise, aberrant, peculiarity, intrusion, etc. [7, 14] Intrusion detection is the act of detecting unauthorized access or misuse of a computer system. Intrusion detection systems (IDSs) [15–21] are used to monitor network traffic and/or computer logs, and alert the system or the user of suspicious activities.

In the context of abuse and network intrusion detection, the interesting objects are often not rare objects, but unexpected bursts in activity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular unsupervised methods) will fail on such data, unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro clusters formed by these patterns.

Three broad categories of anomaly detection techniques exist [7]. Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set under the assumption

that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set. Supervised anomaly detection techniques require a data set that has been labeled as normal and abnormal and involves training a classifier (the key difference with many other statistical classification problems is the inherent unbalanced nature of outlier detection). Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set, and then test the likelihood of a test instance to be generated by the learned model.

2.1.1 Statistical Learning

Statistical machine learning [22], as a branch of statistical learning, designs algorithms that learn patterns from data, and apply the pattern to predict application behaviors. It can be used to capture anomaly behaviors. Most network anomaly detectors are based on network traffic. As attackers usually display short term network anomalies, the short term deviation from the normal operation of the network could be intrinsically different from the pattern learned before. Therefore some anomalies are detected and the alarm is raised for malicious activities. Researchers utilize statistical learning algorithms such as Bayesian learning, neural networks, and maximum likelihood estimation, in both modeling and detecting of covert channels and bots.

2.2 Web Bots Detection

This section describes the basics of web bots. This section describes definitions of the web bots, demonstrates examples of various bots and botnets, and presents state of the art of web bots in the web applications. As my research emphasizes detection resistance, this section further focuses on detecting web bots, so the evasion research could be better presented.

2.2.1 Web Bots and Examples

A *web bot* is a program that automatically and recursively traverses web sites, fulfilling one of many possible online tasks. Although automating online activity has improved the intelligence, performance, and efficiency of online systems, web bots also have significant negative impacts on Internet service providers and users. For example, in August 2012 it was reported that 80% of the clicks that a startup company paid for came from Facebook bots [23]. These bots automatically load pages and drive up advertisement costs, leading to financial losses for the online vendors. Spam posting behaviors also differ from legitimate postings, and can be leveraged to detect those bots. Thomas et al. [24] presented Monarch, which can crawl and analyze URLs submitted to servers in real time, and determines which URLs lead to spams. Gao et al. [25] proposed an effective technique to filter out spam campaigns in online social networks (OSNs). In their approach, OSN spam messages are reconstructed into *campaigns*, and can be detected in real time with high accuracy and efficiency. Jacob et al. [26] presented PUBCRAWL to detect web crawlers based on the observation that the traffic generated by crawlers is very different from the normal user traffic.

Here I briefly review some of the most common bots on internet such as spammers and crawlers.

Benign bots such as web search engines, digital libraries, and many other web applications such as internet marketing softwares and intelligent searching agents heavily depend on crawlers to acquire documents [13, 28, 73]. Google, Yahoo and MSN crawlers traverse billions of web pages periodically to support the variety of services provided by these search engines. Shopping bots and price bots bring users discounted products and price comparisons everyday. These functions and activities include regular crawls of web pages for general-purpose indexing and public services. However, there are also different types of unethical functions and activities such as automatic extraction of email and personal identification information as well as service attacks. Even general-purpose web page crawls can

lead to unexpected problems for Web servers such as a denial of service attack in which crawlers may overload a website such that normal user access is impeded. Crawler-generated visits can also affect log statistics significantly so that real user traffic is overestimated.

Spam is an increasing problem for all forms of online messaging, including email, instant messaging, social media, blogs, and Web forums. Many past and current approaches to tackling spam rely heavily on content-based approaches, where filters use the content of spam messages to distinguish them from legitimate messages. This approach, however, aims at a moving target: spammers are free to evolve the content of their messages in a variety of ways in response to filtering rules, leaving content-based filters to play catch-up. Content-based filters also incur more overhead, because they need to accept, store, and process the content of an email before making a decision; with 90% of email and over 50 billion messages a day being spam, content-based filters are expensive both to maintain and to scale. To scale with the increasing volumes of email each year, mail server administrators either must constantly upgrade their systems or face long queuing delays even for legitimate email.

2.2.2 Detection Approaches for Web Bots

To tackle the ever increasing threats from malicious bots, several serious research projects have tried to differentiate human users from bots. This section presents selective defensive approaches such as Human Interactive Proofs (HIPs), Human Observational Proofs (HOPs), and other approaches.

Robots Exclusion Protocol

The Robots Exclusion Protocol has been proposed [37] to provide advisory regulations for crawlers to follow. A file called robots.txt, which contains crawler access policies, is deployed in the root directory of a website and is accessible to all crawlers. The Robots Exclusion Protocol allows website administrators to indicate to visiting robots which parts of their site should not be visited as well as a minimum time interval between visits. If there is no

robots.txt file on a website, robots are free to crawl all content. Ethical crawlers read this file and obey the rules during their visit to the website.

Human Interactive Proofs: CAPTCHAs

Many sites send CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [27] validation forms to users *before* accessing resources. However, this kind of validation requires human power, which degrades user experience, and also generates authentication traffic load to a web site during authentication. About 200 million CAPTCHAs are solved by humans around the world every day. In each case, roughly ten seconds of human time are being spent. Individually, that's not a lot of time, but in aggregate these little puzzles consume more than $150,000$ hours of work each day.

Traditional approaches such as CAPTCHAs differentiate humans and bots based on active user interactions (users are prompted explicitly for a test), and are generally known as human interactive proofs (HIPs). CAPTCHAs have been found to be an effective challenge response Turing test, and are deployed in many online services (including online account registration, online voting, and message posting). However, they are not suitable for continuous monitoring, as users must wait additional time ($\sim$10 seconds) to be verified [28]. Such frequent interruptions can cause an intolerable usability burden on users. But if users are not continuously monitored, human attackers can pass the one-time test and then let web bots take over. Recent studies have found that web bots often solve CAPTCHAs by using powerful optical character recognition techniques or cheap human labor [29], further lowering barriers for attackers to evade detection. In other words, CAPTCHAs make access more difficult for legitimate users, but do not interrupt illegitimate users.

Human Observational Proofs (HOPs)

Unlike HIPs, human observational proofs (HOPs) take a non-interactive approach, passively monitoring input behaviors. Thus they are uitable for continuous bot detection. Measurements have found strong evidence for complexity and irregularity of human behaviors that

are rarely found in existing web bots. Gianvecchio et al. exploited these intrinsic differences to accurately detect chat bots (in online chatting) [30], as well as game bots (in online games) [31]. Critical behavioral differences from humans are also found in twitter and blog bots [32,33], allowing detectors to use advanced machine learning techniques to detect malicious bots in Twitter and blogs.

Other Detection Approaches

Previous anomaly detectors [34–37] were proposed to detect malicious crawlers and spammers both offline and in real time. An offline detection model is trained using a significant amount of offline log data. Serval papers used this approach and proposed machine learning classifiers to differentiate bot traffic from human traffic [34–37]. In contrast, real time bot detection imposes immediate restrictions on the bots access to site resources, which means malicious bots may be detected *before* or *during* an active session [34]. In many scenarios, these approaches have been demonstrated to perform very well, with the assumption that (1) web bot behaviors are intrinsically different from human beings, and (2) bots do not intend to respond to detectors by hiding their identities in normal behaviors of human users.

Spam posting behaviors also differ from legitimate postings, and can be leveraged to detect those bots. Thomas et al. [24] presented Monarch, which can crawl and analyze URLs submitted to servers in real time, and determines which URLs lead to spams. Gao et al. [25] proposed an effective technique to filter out spam campaigns in online social networks (OSNs). In their approach, OSN spam messages are reconstructed into *campaigns*, and can be detected in real time with high accuracy and efficiency. Jacob et al. [26] presented PUBCRAWL to detect web crawlers based on the observation that the traffic generated by crawlers is very different from the normal user traffic.

Botnet Evadability

Another route of research is to evaluate the evadability of botnets [38]. However, this research demonstrates evadability evaluation in a completely different scenario, i.e., it is

not for covert channel. However, it is worth mentioning since not much research focuses on detection resistance measurement or evadability. Stinson etc, in [38] analyzed eight characteristics of botnets that detectors rely on. First, [38] surveys the following botnets at network level or host level. Later, it used Strayer, Rishi, Karasaridis, BotSwat, BotHunter and BotMiner [39–45] to demonstrate how these detection systems can be defeated by choosing a proper evasion approach. Stinson [38] emphasizes implementation complexity and utility and use two weight metrics to identify low, medium and high evadability.

Strayer [39] identified tight command and control as exhibited by IRC networks. Rishi [40] presented techniques are mainly based on passively monitoring network traffic for unusual or suspicious IRC nicknames, IRC servers, and uncommon server ports. Karasaridis [41] identified botnet controllers. BotHunter [42] presented a method for botnet detection that entails correlating alarms from different network intrusion detection system elements that reside at the egress boundary. BotMinder [43] presented a botnet detection method that clusters communication traffic. Cui [44] presented a method for identifying extrusions,which are user-unintended malicious outbound connections. Stinson [45] characterized the remote control behavior of bots by identifying when selected system call arguments contain data received over the network.

2.3 Timing Covert Channels Detection

This section describes the covert channel definitions, properties and examples. As in the previous section, we also present the defense mechanisms in TCC.

2.3.1 Timing Covert Channels Definition

A covert channel [46–52] is a communication channel that violates a security policy by using shared resources in ways for which they were not initially designed. Covert channels can be divided into two broad categories, timing and storage. Covert channels that modulate timing information for a shared resource or process are known as timing channels, while

covert channels that alter the content of a resource are referred to as storage channels. Both of these types of channels can be used in either a single system or networked environment. We define a covert channel in a single- system environment as any channel that passes hidden messages between entities on the same machine, whereas a channel in a networked environment is one that transmits the message between machines, potentially across network boundaries. In general, we can distinguish between *timing covert channels* (TCC) and *storage covert channels* (SCC). In SCC, the sender transmits data by modulating payloads including unused or "random"bits in the package headers. Research has been proposed that SCC was easier to be detected compared to TCC.

I use the following terminology while discussing covert network timing channels. The sender of the channel is the subverted entity that is responsible for modulating the timing to encode information. It can be an application program, part of the operating system, or a hardware device. The receiver in the channel can either be a network connection endpoint or a passive eavesdropper that extracts information from the channel by looking at network packet timings.

To modulate the covert signals into normal traffic, the attackers want to inject covert timing signals so that they appear to be timing delays of network traffic. There are two common encoding schemes for timing delay modulation. The first encoding scheme uses *inter-packet delays* (IPD) of network traffic that it injects covert signal by adding delays to IPDs. It assumes that the packets have a corresponding relationship between incoming and outgoing traffic. A modified version of the IPD approach selects potential packets or constructs extra packets for Y if packets are added or removed. The second encoding scheme is called *interval timing delay* (ITD) of network traffic. The ITD approach divides network traffic into fixed length time intervals (*time windows*) so that each interval can be treated equally to inject covert signals. The latter approach does not assume the one-to-one relationship of packets of end-to-end traffic.

This dissertation focuses on timing channels used in a network environment. From this point forward, any reference to a timing covert channel (TCC), unless otherwise stated, will

refer to the channels that operate in a network environment.

Timing Covert Channels Properties

A well designed timing covert channel must consider three properties: *reliability*, *efficiency* and *stealthiness*. Reliability implies a covert channel scheme robustness that must be able to recover the prior agreed timing signal from various network inferences of noise (that is of high decoding rate [53]). Efficiency indicates the capacity of the amount of covert timing signal that can be reliably transmitted. Stealthiness represents the ability to hide covert signals in normal traffic [54, 55], or the ability to evade covert channel detection.

Researchers have repeatedly suggested that high reliability can be achieved using various error control strategies [53–56] with sufficient covert signals. Further, to make an attacker more evasive, stealthiness has received more attention in covert channel design [54, 55]. In particular, these stealthy covert channels [54, 57] carried significantly lower timing signals than others [53]. To maintain enough SNR for effective covert channel communications, the covert noise has to be very low as well. Moreover, these approaches measure SNR based on end-to-end *inter-packet delays* (IPD), which lead to underestimation of the covert noise in a practical environment. This is due to the fact that the fast growing and practical use of anonymous systems have increasingly imposed significant challenges on timing covert channels. An empirical analysis [58] found that the fast rate and bursty traffic of anonymous web browsing systems, circuits rotation of Tor [59], and packet drops of Anonymizer.com [60], created large timing perturbations in network flows. In other words, there are increasing amounts of channel noise in the form of signal insertion and deletion. The attacker has to impose more covert signals to overcome the timing perturbation, which makes the attacker less stealthy. In these cases, the IPD approaches did not work well to calculate SNR because the packets are not in correspondent relation in end-to-end network flows.

2.3.2 Timing Covert Channels Examples

Storage and timing IP SCC The Storage IP Simple Covert Channel (Storage IP SCC) [61] first applied a simple binary encoding scheme to encode bits transmitted in the channel, as a covert channel. The bit sent as 1 or 0 refers to either *send* or *not send* a packet in a given time interval for modifying packets length. The sender and receiver agreed on the time interval and the encoding scheme beforehand. The time interval must be an appropriate value, since a small interval may fail to achieve reliability, and a large interval lowers the channel capacity so that it may not sufficiently transmit covert signals. Similar to the timing interval of the storage IP SCC, timing IP SCC [61] also encodes bits as 1 and 0, but it encodes covert signal as time delays to IPDs of packets.

JitterBug Shah et al. [57] proposed a practical Keyboard Jitter Bug that uses a timing channel to solve the data exfiltration problem for keystroke loggers by leaking captured passwords through small variations in the precise times at which keyboard events are delivered to the host. Whenever an interactive communication application (such as SSH, Telnet, instant messaging, etc) is running, a receiver passively monitoring the host's network traffic can recover the leaked data, even when the channel is encrypted. Jitterbug covertly transmits data by perturbing the timing of input events likely to affect externally observable network traffic. Jitterbug uses a timing window w to determine the delay required for encoding each symbol. JitterBugs positioned at input devices deep within the trusted environment (e.g., hidden in cables or connectors) can leak sensitive data without compromising the host or its software.

Model-based TCC Gianvecchio et al. [62] proposed a Model-based Covert Timing Channel (MBCTC) to create a TCC that attempts to mimic the statistical properties of legitimate network flows. Their approach is a typical supervised learning approach because they create models of legitimate traffic beforehand, and then use the model to determine the properties of TCC. They built a framework to collect and analyze empirical traffic and

model it to a distribution.

The message is then split into symbols that are mapped to IPDs based on the inverse distribution function of the chosen distribution. Finally, packets are sent using the calculated IPDs. Decoding is performed using the cumulative distribution function. The distribution can be changed over time to redirect any changes in the target traffic.

Covert Channel in Anonymous Communication Systems

Chaum [63] introduced the mixed network for anonymous email. Later, low-latency anonymous networks have been proposed, developed, and deployed [59, 60]. Tor [59] uses a sequence of proxies and public key encryption technique to protect the anonymizing TCP flows. Tor clients build three-hop circuits for anonymous communication, where the three intermediate nodes are called entry, middle, and exit nodes. The entry node is always chosen from a set of candidates to prevent long session attacks. Although Tor claimed that it is vulnerable to black-box timing covert channels, Tor's circuit rotation introduced large timing perturbations as channel noise [58]. Anonymizer [60] dropped packets from network flows, so it also created timing perturbations as noise to timing covert channels. To have a robust timing covert channel, it is feasible to increase the length of the encoding messages, but that makes them more likely to be detected by defenders.

Anonymous communication systems protect users' privacy by restricting unauthorized access to their identities [53, 55, 64–71]. These systems set up one or more intermediate proxies between the sender and receiver along the path, which prevent unauthorized parties from connecting the identities of the sender and receiver through traffic analysis. However, an attacker can use a *black-box timing covert channel* to correlate both parties in anonymous communications [53–55,61,70,71]. Attackers typically use a black-box timing covert channel to correlate timing patterns of the network flows entering and exiting an anonymous network system, then use that information to detect relationships among communication parties in the system. That is, the attackers detect who is talking to whom, and decide whether the communication parties match up in an anonymous network. In a black-box timing covert

channel, a sender injects a timing signal (i.e. *encode*) to the inbound traffic, while the receiver eavesdrops on the outbound traffic and tries to recover the patterns (i.e. *decode*) the sender and receiver previously agreed upon.

2.3.3 Timing Covert Channel Defense

To defend timing covert channels, researchers [61,72] have proposed two main defense mechanisms. The first is to detect the presence of a channel. This is also a focus of my dissertation. The second is to eliminate a covert timing channel by removing all timing information from adding large timing distortion. The second mechanism has degraded network performance for overt traffic, hence detecting covert channels is a better way for timing covert channel defense.

To detect a covert channel we usually need to distinguish covert channel traffic from legitimate traffic. For example, Liu et al. [73] proposed a framework for detecting sensitive data exhilaration by an insider attack using statistical traffic classification. Legitimate traffic is used to train the detector, and the detector uses a statistical test to discover differences between legitimate and covert traffic. Further, the covert channel detection problem can be formulated as a statistical significance testing problem. This section introduces the most common statistical tests to detect the existence of covert channels. The "differences" are usually quantified as similarity scores, distances, entropy scores, etc. A passive approach in defending covert channels usually monitors the network traffic passively to detect such channels. The detection techniques include shape tests [61], regularity tests [74], and entropy tests [75]. This subsection summarizes all existing detection methods.

Shape tests

The most well-known test in covert channel detection is the Kolmogorov Smirnov test (KS test) [76]. The KS test is a nonparametric test for the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample KS test), or to compare two samples (two-sample KS

test). The KS statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples. The KS test may also be used to test whether two underlying one-dimensional probability distributions differ. Since the KS Test directly compares the empirical distribution functions, the samples do not need to be the same size. As detectors determine a threshold of certain scores for detection, the designer of the covert channel finds ways to lower the score below a threshold to evade detection.

Regularity tests

Regularity tests are used to measure the correlation of data. The regularity test by Cabuk [61] determines the differences of variance of the inter-packet delay between legitimate traffic and covert traffic. The author of [61] observed that the variance of the inter-packet delays changes over time in legitimate traffic, while the variance of the inter-packet delays remains relatively constant over time as the encode schemes are less likely to change over time.

Entropy tests

Entropy [77, 78] is a measure of randomness information content, or complexity. It is a concept central to information theory. Covert channels either create regularity, resulting in low entropy, or encode extra information, resulting in high entropy [79–81]. Gianvecchio and Wang [75] proposed an approach with entropy and conditional entropy of a network stream to detect covert timing channels. Entropy can be used to detect timing channels that causes differences in shape, whereas conditional entropy can be used to detect the regularity of a channel. They postulated that a combination of these two metrics would be effective against known timing channels and robust against future channels. They showed that entropy-based detection could be used to detect well-known timing channels such as IPCTC and the Jitterbug CTC. The technique [75] detected three different covert timing channels that were not detected by previous detection methods.

3. Web Bots Evasive Framework

This chapter escalates the battle against web bots by exploring the limits of current HOP-based bot detection systems. This chapter develops an evasive web bot system based on human behavioral patterns. Then it prototypes a general web bot framework and a set of flexible de-classifier plugins, primarily based on application-level event evasion. It further abstracts and defines a set of benchmarks for measuring our system's evasion performance on contemporary web applications, including social network sites. The results show that the proposed evasive system can effectively mimic human behaviors and evade detectors by achieving high similarities between human users and evasive bots.

3.1 Introduction

Human interactive proofs (HIPs) such as CAPTCHAs are used in many web systems to distinguish bots from humans. CAPTCHAs require a user to pass challenge-response tests by either recognizing distorted characters or selecting images. However, the interactive requirement of CAPTCHAs makes them unsuitable for continuous and passive monitoring, and its usage is mainly limited to one-time authentication or registration. To make the bot detection transparent to online users, human observational proofs (HOPs) based approaches have also been developed [30, 31]. The design rationale behind HOP-based detection is that human behavior is more complex than bot behavior, and their interactions with the web application are also very different. These HOP-based systems are able to accurately distinguish bots from human users by identifying bot behaviors that differ intrinsically from human's behaviors, such as keystrokes and mouse movements.

From the perspective of web bot creators, some researchers have proposed evasive bots [82, 83]. The goal of evasive bots is to "poison" anomaly detectors and then evade the

bot detection systems. Given the aggressive nature of web bots, I believe it is essential to understand the threat of evasion against HOP-based defense. In particular, the web bots that mimic human behavior patterns on the Web have not yet been systematically studied. This dissertation explores the limits of current HOP-based bot detection systems by creating a new evasive bot system that masquerades as human beings on the Web. Specifically, it characterizes the existing HOP-based web bot detectors and develops an evasion framework based on human behavior patterns. Instead of subverting a specific detection system, the major goal of this study is to provide a systematic approach to evaluate and explore the limits of current HOP-based detection systems, and motivate online business to develop more advanced bot detectors.

This chapter makes three contributions. First, it characterizes web bot detection observation proofs that use web application events, with the focus on the different events and their statistics associated with the activities of bots and human clients, such as timing and location. Second, it proposes a generative approach to build an evasive web bot via adaptive human behavior learning. The proposed web bot can mimic human behaviors to hide its activities within legitimate user events; in other words, de-classifying bot-like traffic. Third, it provides a practical framework for evasive web bot design and implementation, which helps researchers and engineers to explore the limitations of existing bot detectors.

This research implements a prototype of the proposed evasive bots, and validated the power of our evasive bots by measuring the *similarity* on various metrics from a set of selected web representative applications. In the experiments, this research redefines the Kullback-Leibler (KL) distance formula [84] to evaluate the effectiveness of the proposed evasive model and algorithm. The experimental results show that the proposed evasive system can effectively mimic human behaviors and achieve high similarities between human users and evasive bots.Note that the purpose of our experimental evaluation is not to show how successful our approach can bypass a specific HOP-based detector, but to demonstrate how close the proposed evasive bots are to normal human users, which will ensure our approach can evade any existing HOP-based detectors. To capture the proposed evasive

bots, future HOP-based detectors must introduce new features for bot detection. Otherwise, even if future detectors further tune up their classifiers, the detection of evasive bots will result in many false positives, due to the very close behavioral similarity between our evasive bots and human users, making these fine-tuned detectors useless.

3.2 Characterization of Behavior-Based Web Bot Detection

Web browsers capture user action behaviors by generating events on the web application such as keystrokes, mouse motions, and clicks. The application events often trigger traffic events that can be observed on the server. As Figure 3.1 shows, detectors can use both application events and traffic events to detect anomalies by analyzing user action events and classifying bot features in network traffic characteristics. Since the traffic events are driven by the application events, my goal was to study human input event patterns and generalize behavior-based web bot detection from the perspective of application events. This research also discusses traffic events but only for the purpose of filtering noise during user action collection. It starts with simple boolean events, which identify bots by capturing certain unexpected or expected human behaviors. Then, it examines a more complicated scenario where bot detection relies on various statistics of event metrics.

3.2.1 Detection Abstraction

Advanced bot detectors use machine learning to identify important features. Let R denote the set of bot actions, and H denote the set of human actions. A human input is X, traffic events are Y, and the decision maker function is F, also called the *binary classifier*. Detectors learn and build the classifier function $F : (X, Y) \rightarrow \{0, 1\}$ from a training set of (X_h, Y_h) from H, compared with (X_r, Y_r) generated from R. The classification process is formulated as $Z = F(X, Y)$, where Z represents the output of the classifier, $Z \in \{0, 1\}$.

This research assumes that X is a set of all actions of a web user. It also assumes the detector is able to collect user events in a timely manner. From the detector's perspective,

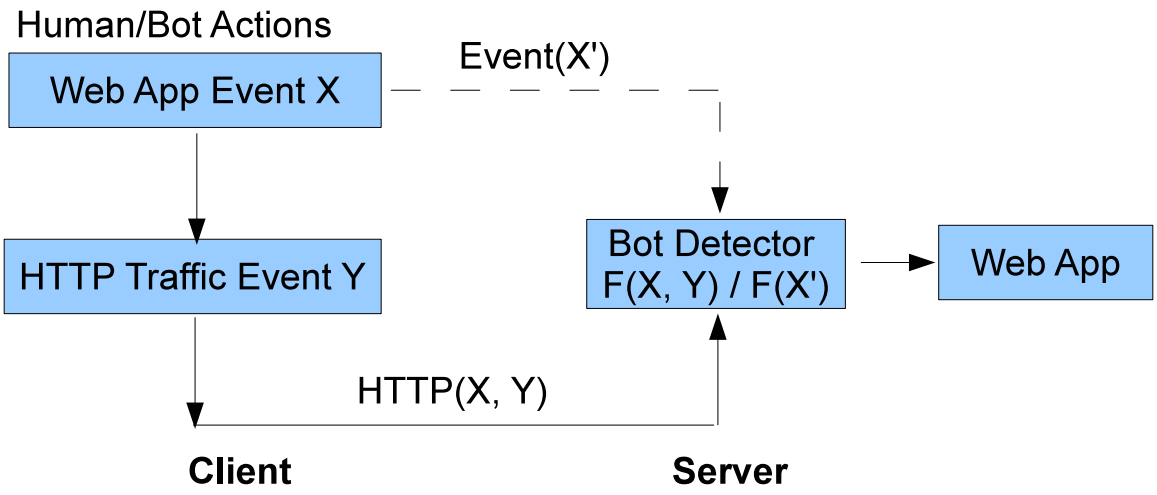

Figure 3.1: The behavior-based detection process

Table 3.1: Web application event-driven detection

	App Event Variable X				
Human Action	Trigger	Timing	Location/Obj	Detection Function F	Semantic
`mouse`click	X	-	-	$Z = X$	movement
`mouse`over	X	-	-	$Z = \neg X$	movement
`key`stroke	-	-	X hidden form	$Z = \neg X$	abnormal
`mouse`click	-	-	X hidden link	$Z = \neg X$	abnormal
`mouse`click	-	-	X random element	$Z = X$	integrity
`mouse`click	-	X	-	$Z = F_{dist}(X)$	behavior
`mouse`click	X_1	X_2	-	$Z = F_{rate}(X)$	behavior
`mouse`move	-	-	X	$Z = F_{dist}(X)$	behavior

the user input is taken in two different scenarios as shown in Figure 3.1. The first scenario is when human input events generate network traffic. In this case, X and Y are dependent, and let us define the formula to be $F : Z = F(X, Y)$. For example, a click on a web link generates traffic by downloading a web page. The second scenario is when a user action X', such as a mouse movement without a click, does not generate any network traffic. Let us simplify the formula to be $F : Z = F(X')$ in this scenario. In Figure 3.1, the user input of the first scenario is shown with a solid line, while that of the second scenario is presented as a dotted line.

3.2.2 Application Event-Driven Detection

A web browser on the client notifies web applications of changes by generating user input events, such as keystrokes, mouse movements, and clicks on buttons and links. These application events are collected and sent to the server. The detection engine on the server notifies the web application administrator when detecting malicious web bot activities.

Boolean events. This research summarizes some common event-driven detection features in Table 3.1. The simplest events are boolean events, as a special case of event statistics, which represent expected or unexpected presence. These event-driven detectors perform online detection on the server. Most detectors try to identify bots in real time. This set of detection techniques is usually combined with specific web implementation technologies such as client-side JavaScript and form tracking. This research formalizes categories of keyboard/mouse, random elements, and hidden elements as a 0 and 1 boolean pattern, because the user input X to the classification function F is also a boolean variable. The classification function F is straightforward, either $F : Z = X$ or $F : Z = \neg X$. This kind of detection takes advantage of the prior knowledge of differences between real human input actions and bot input actions to the web interface.

Event Statistics. The application event-driven detection also uses statistical features observed from human inputs on the client side. These detection approaches are based on the observation that bots' behaviors vary less than humans'. For example, a bot may periodically repeat one action or periodically trigger a sequence of actions, while human actions follow a normal or Pareto distribution. These kinds of detectors usually attempt to detect bots in real time with simple implementations. In this scenario, widely used distribution models like uniform, Pareto, or Weibull distributions [21] are assumed. More sophisticated detectors based on estimating more complex model parameters learned from human behaviors can distinguish bots from humans.

Table 3.2: Metrics selected detection/evasion analysis

	Metrics		
Event	count	timing	location
`click`		x	x
`move`	x		x
`keystroke`	x	x	

3.3 Generative Evasion

This section describes how bots masquerade as human users. It selects event-based metrics from the detector's perspective, and demonstrate how bots can mimic human action events in timing and location. Further it defines similarity metrics according to a revised symmetric KL distance measurement, assuming the HOP detector would also use KL distance [36] to distinguish bots and humans. This research also builds a pool of candidate models to best fit human behaviors, followed by an algorithm that selects the right model to fit user actions to create evasive bots actions.

3.3.1 Target Behavior Metrics

This section selects a set of application independent events such as keystrokes, mouse clicks and movement counts, and measure timing and location, as listed in Table 3.2. Timing represents inter-event timing between two subsequent events, while location demonstrates the mouse click positions and movement angle information. These are used for statistical behavior analysis. This research also counts how many times the move and keystroke events appear.

Boolean Behavior

Note that boolean events can be integrated into the distribution-based detection as a special case. For example, a click on a hidden link is a single event. The distribution statistic is a simple 1/0 constant. The statistic is obtained either from prior knowledge by analyzing the code or by adversarial classification [85]. The boolean behavior events are usually

constructed as a set.

Behavior Statistics

Besides boolean events, the events that rely on statistical analysis are listed in Table 3.2. The idea behind statistical analysis is to find the distance between two distributions that belong to human and bot. This research maps each histogram of distribution into a point in a space and measures the similarity of two distributions by computing the distance in the space. While the distance can be measured in various ways, I create a new distance measurement based on a modified KL divergence, which is detailed in Section 3.3.2.

Timing-based Events

Event timing is the time interval between each pair of consecutive events. In this work, the events could be clicks or keystrokes. Suppose there is a sequence of click actions $\{x_1, x_2, ..., x_n\}$, histograms can easily be built from the time inter arrivals $\{x_2 - x_1, x_3 - x_2, ..., x_n - x_{n-1}\}$ for both humans and bots. Existing bots usually generate a large number of events almost constantly, leaving clear timing patterns for detection. The bot framework allows bots to insert idle time to mimic human behaviors.

Location-based Events

The two movement events are mouse clicks and movement, as they carry the basic information about human actions. Suppose we have a sequence of click actions on locations $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$. The movement can be characterized as a movement vector between two locations. The vectors are defined as $(x_1, y_1) \rightarrow (x_2, y_2)$, $(x_2, y_2) \rightarrow (x_3, y_3)$, ..., $(x_{n-1}, y_{n-1}) \rightarrow (x_n, y_n)$. The correlation between the movements are the angles of two contiguous vectors. The angles are defined by consecutive pairs of vectors: $\theta = \arctan((y_2 - y_1)/(x_2 - x_1))$. Let us map the angle from the range $[-\pi, \pi]$ to $[0, 360]$ to make the explanation clearer and more intuitive.

3.3.2 Similarity Measurement

Formally, this research treats a contiguous distribution as a discrete histogram calculated from the observation data, which is normalized into n buckets. It then map each histogram into a point in an n-dimensional space. The similarity of two distributions is measured by the distance between two points in an n-dimensional space. The KL formula in equation (3.1) can quantitatively compare different distributions or histograms. H_1 and H_2 stand for two normalized histograms. $H(i)$ means the normalized ith bucket value in the histogram H, where $\sum_i H(i) = 1$.

$$D_{KL}(H_1, H_2) = \sum_{i=1}^{n} H_1(i) \log \frac{H_1(i)}{H_2(i)} \tag{3.1}$$

Intuitively, KL divergence turns the difference in each bucket into a real number with two directions, negative and positive, depending on their relative values, and then weights real numbers with H_1. However, this is not a distance metric on the space of a probability distribution, because it is not symmetric; that is $D(H_1, H_2) \neq D(H_2, H_1)$. This research defines a new distance measurement based on the KL formula equation (3.2). The new definition is a distance metric, and is symmetric, that is $D(H_1, H_2) = D(H_2, H_1)$. It removes the direction on the original definition in each bucket and averages the bi-directional differences. The bigger the value of the distance is, the larger the difference there is between the behaviors. Practically, the different distributions detected on the server generate larger KL distance values for the classifier to identify the bots.

$$
\begin{aligned}
D_{SKL}(H_1, H_2) &= \frac{1}{2}(\sum_{i=1}^{n} H_1(i) \log \frac{min(H_1(i), H_2(i))}{max(H_1(i), H_2(i))} \\
&+ \sum_{i=1}^{n} H_2(i) \log \frac{min(H_1(i), H_2(i))}{max(H_1(i), H_2(i))}) \\
&= \sum_{i=1}^{n} \frac{1}{2}(H_1(i) + H_2(i)) \left| \log \frac{H_1(i)}{H_2(i)} \right| \qquad (3.2)
\end{aligned}
$$

Behavior Consistency Degree

The *behavior consistency degree* (CD) can quantitatively evaluate how a particular behavior varies across users of a web application. It indicates how hard it is to capture the user behavior for the evasive model.

Formally, assume we have a set of user behavior histograms $U = \{u_1, u_2, ..., u_n\}$ such that:

$$
CD(U) = \frac{\min D_{SKL}(c, U)}{\max D_{SKL}(c, U)} \times mean(U, c) \qquad (3.3)
$$

where $c = mean(U)$.

Intuitively, if we treat each user histogram vector as a point in the space, CD represents how the points are spread in that space. A large CD means irregular behavior that is difficult to capture. The first factor represents how points are spread in $[0, 1]$; the second part is the average distance between the center and user points in the range of $[0, \infty]$.

3.3.3 Evasion Model Fit and Selection

Model selection is the problem of choosing among different mathematical models that all try to describe the same data set. Since no single model can fit all user behaviors, this section builds a pool of candidate models and choose the best matching model dynamically. The model can be chosen in several ways, including optimizing some measure of goodness of fit, model averaging, and cross validating. My goal is to optimize D_{SKL}, the measurement of goodness distribution fit. Instead of statically selecting the best model, my generative approach selects the model dynamically by evaluating different models simultaneously. This section designs an algorithm [86] that looks for the best distribution fit for a sequence of time windows. This research smooths the model selection with a mixture of old and new data to fit the models. Algorithm 1 in figure 3.2 shows the procedure.

Although we could use a history of human events to build a prediction model for each particular input event X, I decided to reuse existing models to build a model pool for easy and efficient implementation because those models are widely used and have mature implementations. Previous studies indicate that well studied models like the Fitts' law equation [87, 88] could fit the human behavior well. My model pool has seven models.

I use the common method of maximum likelihood to find the local optimized fit. For example, given the Pareto model selected from the model pool, I use its likelihood function to estimate the parameters α and t_m. Given an n-history sample of event timing intervals $\{t_1, t_2, ...t_n\}$, the logarithmic likelihood function is:

$$\log(L(\alpha, t_m)) = n\log(\alpha) + n\alpha\log(t_m) - (\alpha+1)\sum_{i=1}^{n}\log(t_i) \tag{3.4}$$

To maximize the likelihood function, I use the estimation $t_m = min(t_1, t_2, ...t_n)$. When

$$\frac{d}{d\alpha}\log(L(\alpha, t_m)) = 0 \tag{3.5}$$

Algorithm 1: Model selection algorithm

Data: A sequence of human user data $U =< u_1, u_2, ... >$, buffer size W, retired data size α

Result: A sequence of the best model and its parameter estimation $< M, P >$

```
W = {u_1, u_2, ..., u_W} ;
while U ≠ ∅ do
    buffer the user data into set W ;
    if reach the buffer size W then
        C = (C − {c_1, c_2, ..., c_α}) ∪ W ;
        calculate the center of the user data c;
        distance = ∞ ;
        for m ∈ ModelPool do
            fit c with m ;
            generate c_m;
            if D_SKL(c_m, c) < distance then
                M = m;
                P = estimated parameters;
                distance = D_SKL(c_m, c) ;
            end
        end
        U = U − W ;
    end
end
```

Figure 3.2: Evasive Bots Model Selection Algorithm

and

$$\alpha = \frac{n}{\sum_{i=1}^{n}(\log(t_i) - \log(t_m))} \tag{3.6}$$

the maximum likelihood function can reach its maximum value.

In addition, the selection algorithm continuously re-estimates the parameters when the bot receives event timing patterns from the users. Thus, my evasive bot can gain knowledge from history and integrate it into future data. This has two advantages. First, this design can adapt to the target server automatically when there are changes on the server that could cause the timing pattern to change. Second, the adaptive algorithm can adjust parameters of a model and continue to deceive the detector.

3.4 An Evasive Framework

I implemented a prototype of the evasive bot system that supports multiple web agents. Conceptually, the framework is a continuous learning system. Human actions trigger events that train a generative engine, so the web bots can later mimic the human behaviors. The system design is summarized in Figure 3.3. The *human data flow* is shown with dotted arrows and the *web bot data flow* is shown with solid arrows.

Human data flow represents human data being collected and pre-processed. Information is retrieved from human generated events and traffic, and then saved in logs. This framework records human data with a recorder on the client and an HTTP proxy, and then the human data collected becomes history data for the bot engine to learn human-like behavior. As shown in *human data flow* in Figure 3.3, human events and traffic behaviors are first recorded in logs on the client and through HTTP proxies. As logs get pre-processed, the generative engine studies the human behavior from the log and then supports later use by the event generator. The framework requires human usage before starting the bot as the training. The human flow happened first, and before the bot flow.

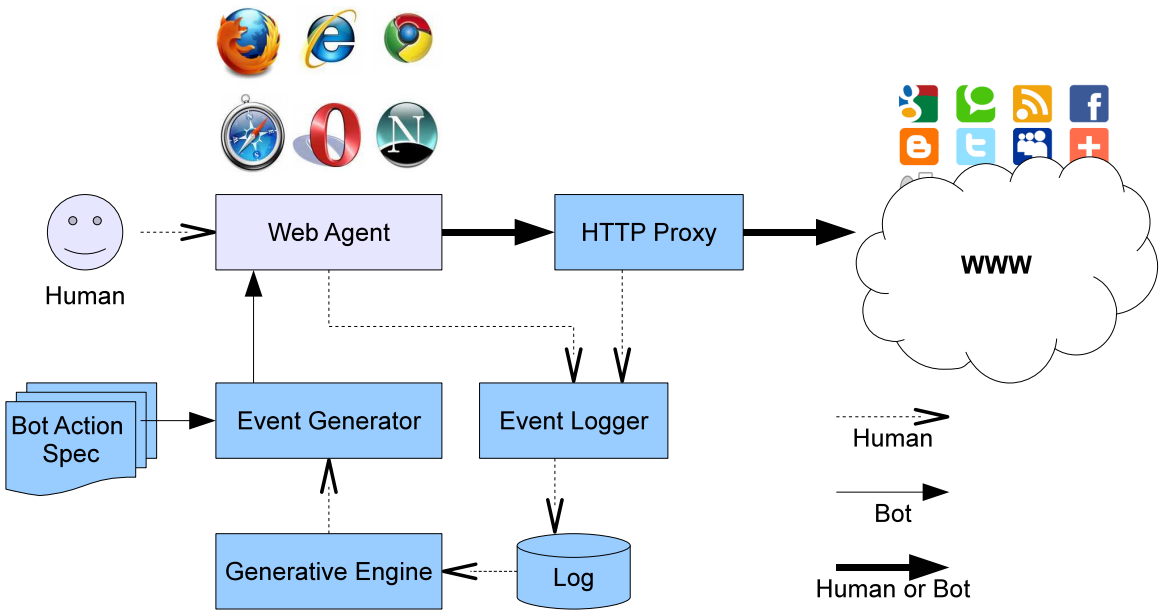

Figure 3.3: Model selection algorithm.

Bot data flow starts with a bot action specification whose most important component is the event generator. The bot action spec defines all types of actions, including mouse clicks and keystrokes. The event generator selects specifications from a set of bot actions, and then makes decisions about how to generate the bot events. The event generator applies the Selenium web driver API, which has full control of web browsers. This allows it to take advantage of any facilities offered by the native platform. Since the platform supports native events in Windows or Linux, I can easily send a mouse click event with the element's coordinates relative to the border of the screen at the OS level. Both keystrokes and mouse events use the native OS API. I implemented the event generator as a 3,000 line Java program that sends events to web applications.

Web Agent. The web agent runs the web application with JavaScript. It captures application events and passes them to the event logger. I use web browser plugins to monitor and control web application events in JavaScript.

Table 3.3: Action specs example

Input Events	Web Element XPath (or ID)	Command Calls
click	//a[text()='some url'/@href	navigate()
`mouseover`	//div[@id='dropdownlist]/	hover()
`select`	//a[text()=' second']	setSelected()
`type`	/html/body/div/table/input	sendKeys()
`click`	//button@value='submit'	submit()

HTTP proxy. The HTTP proxy captures detailed information about HTTP transactions and passes the information to the event logger. As any web event can trigger an HTTP request, it contains more detailed information such as timing information and element XPaths. I did not capture application semantics embedded in the payload, leaving them to other data capture methods.

Event Logger. This component obtains human behavior data from different data sources and performs a pre-processing step on the raw data before feeding it to the generative engine. The client-side web browser, OS, and HTTP Proxy capture different aspects of user behavior data. The JavaScript running inside the web browser can capture application specific events. This research takes advantage of tools like Record User Interaction (RUI) [89] at the OS level to help capture device events such as mouse clicks and movements. The HTTP proxy captures and records more detailed information about HTTP transactions that are associated with the human. Before the event logger stores the human behavior data into the log database, it performs a pre-processing step on the three data sources. My implementation aligns the data according to timing and filters out unnecessary data before compression.

Generative Engine. The generative engine queries the log database for the user behavior data. It analyzes the collected human behavior data and generates a suitable behavior model, then passes it to the event generator. For the boolean function, the generative engine adds a rule for each boolean fact observer from the human, such as a requirement to send a mouse movement event. For the behavior statistics, the generative engine extracts

the human behavior and outputs a generative model by selecting the best fit. For example, it passes a timing model to perform bot actions, which specifies the delay time among bot actions that trigger events.

Event Generator. The event generator generates events according the bot action specification and resolves the constraints by querying the generative engine for information such as timing and location. The event generator first builds a data structure for elements from Table 3.3 using the web driver's API. The web driver identifies the web element according to the unique XPath of the web element. Table 3.3 shows mappings between events and Java calls. In addition, the event log contains an event timing log and an event sequence log. For example, the event module determines a time sequence from the human event timing log. The action interarrival times, $t = (t_1, t_2, ..., t_i)$, by default are constant as a bot. But my goal is to create user events that follow human-like events, so I define several models using distributions such as normal and Pareto. Finally, the event generator component generates a web bot action sequence and sends commands to the web agents.

Bot Action Spec. The bot action spec defines a sequence of event commands and evasions. These actions are mapped to several Java calls in the event generator. Each event command is a triple <*action*, *element ID*, *bot command*>. Table 3.3 describes the primary elements in the specification. The timing and movement patterns are encoded in the sequence as "*nop*" actions. Those behavior events are generated during runtime according to the model from the generative engine.

3.5 Evaluation

This section presents empirical results from evaluation of the evasive capability of my evasive bot framework. This evaluation first clarifies the challenges of selecting benchmarks and creating workloads. Then it presents the results of testing evasive bots under different detection behaviors. As mentioned before, the objective of evaluation is not to demonstrate

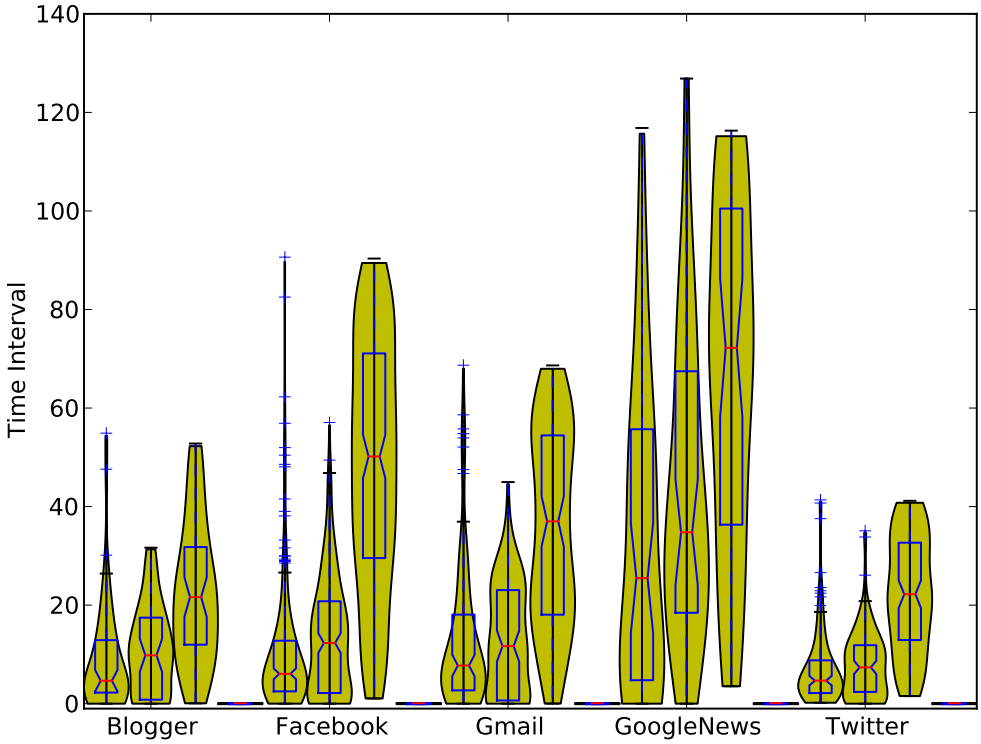

Figure 3.4: Inter-click timing in seconds for human and proposed evasive bots. From left to right in each group, they are human data, a normal distribution model, and a uniform distribution model

how successful this framework can bypass a specific HOP-based detector, but to quantify the similarities between the behaviors of my evasive bots and those of normal human users. The behavioral closeness between the proposed bots and human users will enable this approach to evade existing HOP-based detectors.

3.5.1 Benchmark Set Definition

Evading all possible behavior metrics that detection mechanisms could use is quite challenging. First, it is difficult to fully understand all the detection implementations and models deployed on commercial web servers, because the source code is usually not available and specifications are not public. Second, detectors usually rely on one or several metrics that

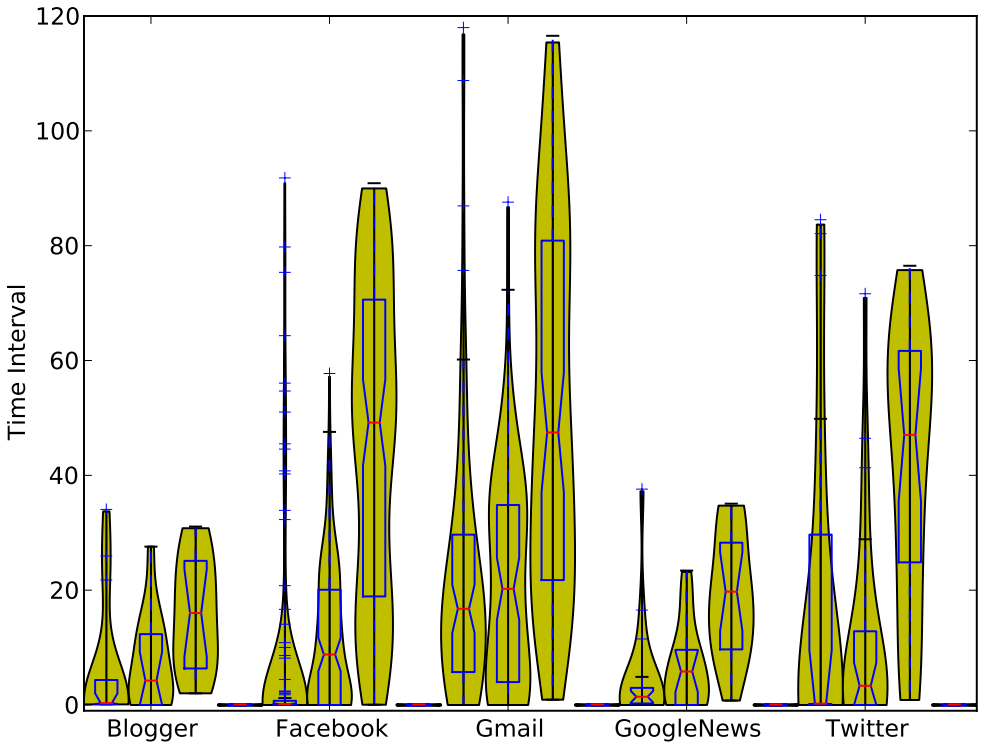

Figure 3.5: Inter-keystroke timing in seconds for human and proposed evasive bots. From left to right in each group, they are human data, a normal distribution model, and a uniform distribution model

are specific to web applications, possibly both signature-based metrics and anomaly-based metrics. It is hard to know what behavior metrics are used by the production bot detectors, and different companies might use different metrics. Finally, even if this dissertation ambitiously implements a long vector of metrics for evasion, the complexity of the detection mechanisms still make it difficult to evaluate the bots' ability to evade them.

The strategy used in this dissertation is to select some typical metrics from a wide range of commonly used web applications. It measures the event timing in each session for five widely used web applications. The web applications were chosen to cover most typical web uses. It logged and measured four types of web applications: (1) a web email application, which included logging in, and composing, sending and retrieving emails; (2) two social networking sites to update status messages, communicate with friends, etc; (3) a web news system; and (4) a blog application, which has one update and multiple read patterns.

3.5.2 Data Collection

This research selected five web applications: Gmail, Facebook, Twitter, Google News, and Blogger. It collected human data by inviting 35 people to participate in 50 experiments to capture their real-time behaviors. The users' ages ranged from 18 to 50 and they are located in different regions in US; 15 on the west coast and 20 on the east coast. This research anonymized all users' personal information to protect their privacy. The proposed framework captured both application events and traffic events including keyboard, mouse movements, and mouse clicks.

3.5.3 Experimental Setup

I built my evasive web bot framework on a mainstream desktop configured with an Intel core Dual CPU 2.40 GHz and 4GB RAM. The operating system was Windows 2008. I used Mozilla Firefox 3.5.X as my primary agent. It can be configured to represent three common agents, and can be extended to mobile web agents.

I implemented the bot event generator in Java 1.6. The event generator controls the

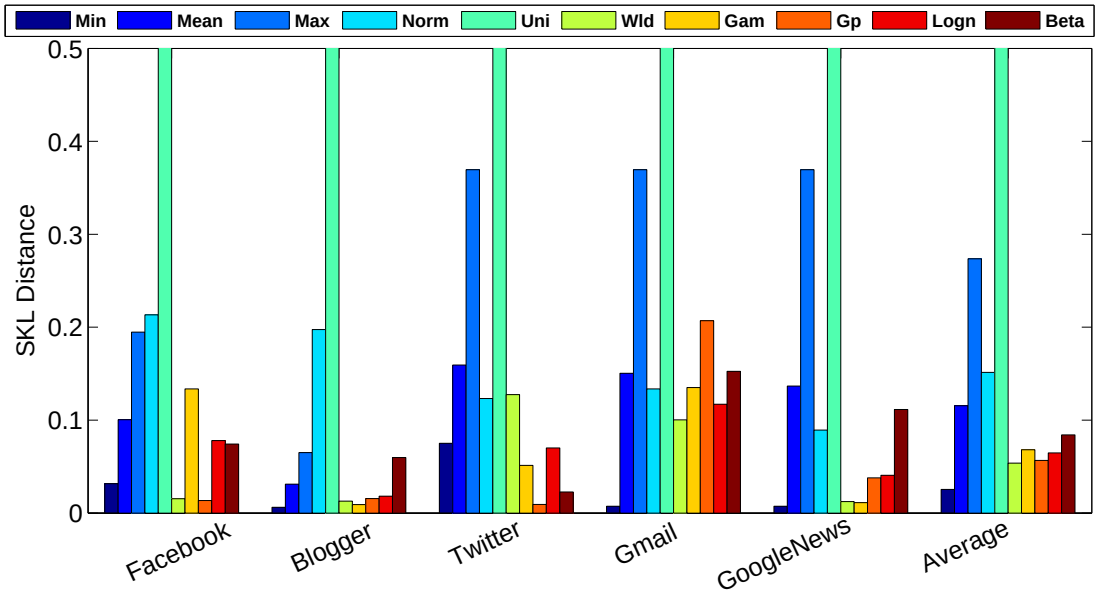

Figure 3.6: D_{SKL} for inter-events timing of click actions across different models. From left to right in each group, the first three are min, max and mean of all distances, and collectively work as a baseline. The other seven are distance fittings from normal, uniform, Weibull, gamma, Gp, Logn and beta distributions

web agent or browser through a plugin. My experiments were conducted in a high speed Internet service environment, sending traffic to a variety of web servers. There are no special network restrictions for my framework, and it can be scaled to a large number of virtual machines running on clusters or a cloud to gain extra computation power.

I also built an Ubuntu 9.10 Virtual Machine running an Apache 6.0 HTTP server to construct a synthetic web site to evaluate boolean events.

3.5.4 Evaluation of Evasion Capabilities

This section presents data from an experimental demonstration of how my framework can prevent web bots from being detected by masquerading as humans. The features used are common to web bot detectors.

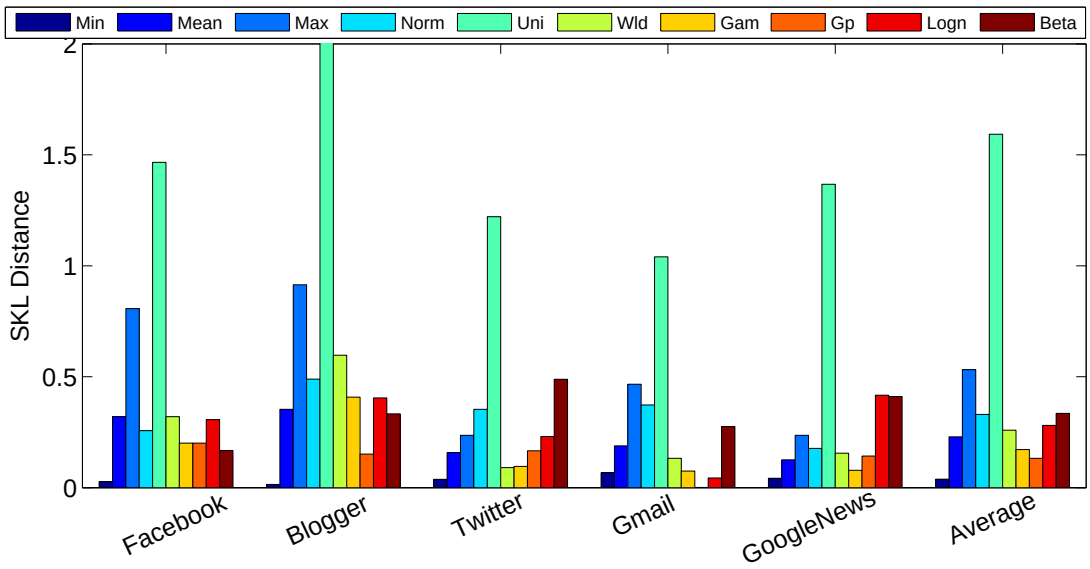

Figure 3.7: D_{SKL} for inter-events timing of keystroke actions across different models as in Figure 3.6

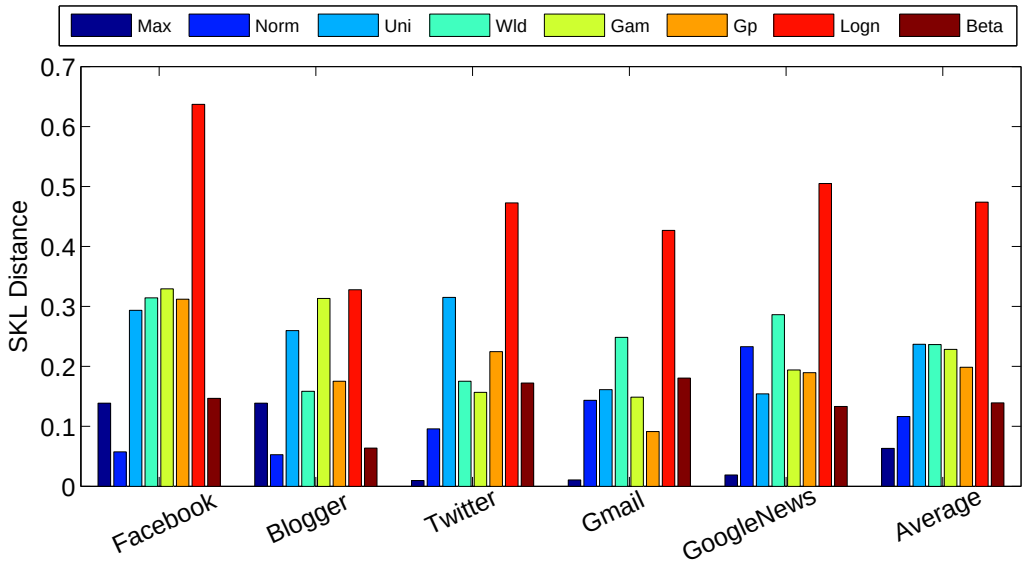

Figure 3.8: D_{SKL} for angles of mouse movement for different models. From left to right in each group, the first is the max of all distances, and collectively work as a baseline. The other seven are distance fittings from normal, uniform, Weibull, gamma, Gp, Logn and beta distributions

Evaluation Approach

One approach to measuring the evasive capability of my bots framework is to test bot detection in the presence of my evasive attacks. If I treat bot detection as a special case of an intrusion detection system, it is easier to understand the challenges of creating an appropriate workload for testing. First, most anomaly detectors require a large amount of training data to build the classification criteria. This leaves the question of how much data is enough. Second, there are no clear guidelines for how to select representative human user inputs. Web applications have become more complex over time, so historical data may not always be applicable. Third, randomness is introduced into the execution environment by the server workload, network traffic, the security protocol, asynchronous communication from Ajax, and user preferences. The Web also introduces new dynamic functionalities that increase the challenge of quantifying the workload [90]. Finally, no standard test methodology is available [91].

These challenges make it difficult to develop a general evaluation method for detectors that is repeatable and comparable across many web applications. Furthermore, the measurement metrics could be tightly coupled with the implementation of certain specific detectors that are not publicly available

I avoid the problems above by taking a different approach. In particular, I quantify and compare the similarities between human and evasive bot behaviors. The evasive bot behavior is generated according to real human behavior. It is reasonable to quantify the similarity by calculating the distance between two vectors of behavior features. The more similar to human behavior, the more evasive my bot is considered to be. For boolean events, I evaluated the evasion detection results, as I have full control and understanding of the detection systems in the synthetic web sites.

Web Application Boolean Events

The boolean events are automatically generated according to the logged human events. I mark “Pass” if the bot was not detected and “Fail” if the detection module raised an alarm.

Table 3.4 summarizes the detection and evasive results. Mouse movements and keyboard native I/O events are easy to evade. One limitation of the evasive bot is that it has limited coverage for web testing as it only follows human's footprints.

Table 3.4: Evasive capability for boolean events

User Actions	**Results**
Mouse over/movement	Pass
Key Strokes	Pass
Random Element	Pass

Web Application Event Timing

Event timing is defined as time intervals between events that are triggered by actions. Actions are those common tasks that both humans and bots need to perform, such as mouse clicks and key strokes. I am particularly interested in the inter-events time of various web applications with different usage patterns, as presented in Figure 3.4. For each timing sequence action record, I fit the distribution model via my candidate models to find the best fit. Then, my generative engine generates a timing interval sequence according to the selected model to evade the events triggered by bot actions.

I group each application into "violin plots" in Figures 3.4 and 3.5. A violin plot is a combination of a box plot and a kernel density plot. It summarizes the data distributions in five numbers: the smallest sample observation, the lower quartile, the median, the upper quartile, and the largest sample maximum. As my generative approach tries to maximize the distribution similarity, the violin plot is an intuitive way to graphically depict groups of numerical statistic data similarity. For each application in the figures, from left to right, the plots represent the human data and then the two generative data from the fitting models. The first fitting model is a normal distribution, which largely captures random human activity. The other is a uniform distribution. I map the similarity of two distributions from intuitive pictures to a number, the D_{SKL}. A summary of the inter-event timing interval D_{SKL} across all candidate models is illustrated in Figures 3.6 and 3.7.

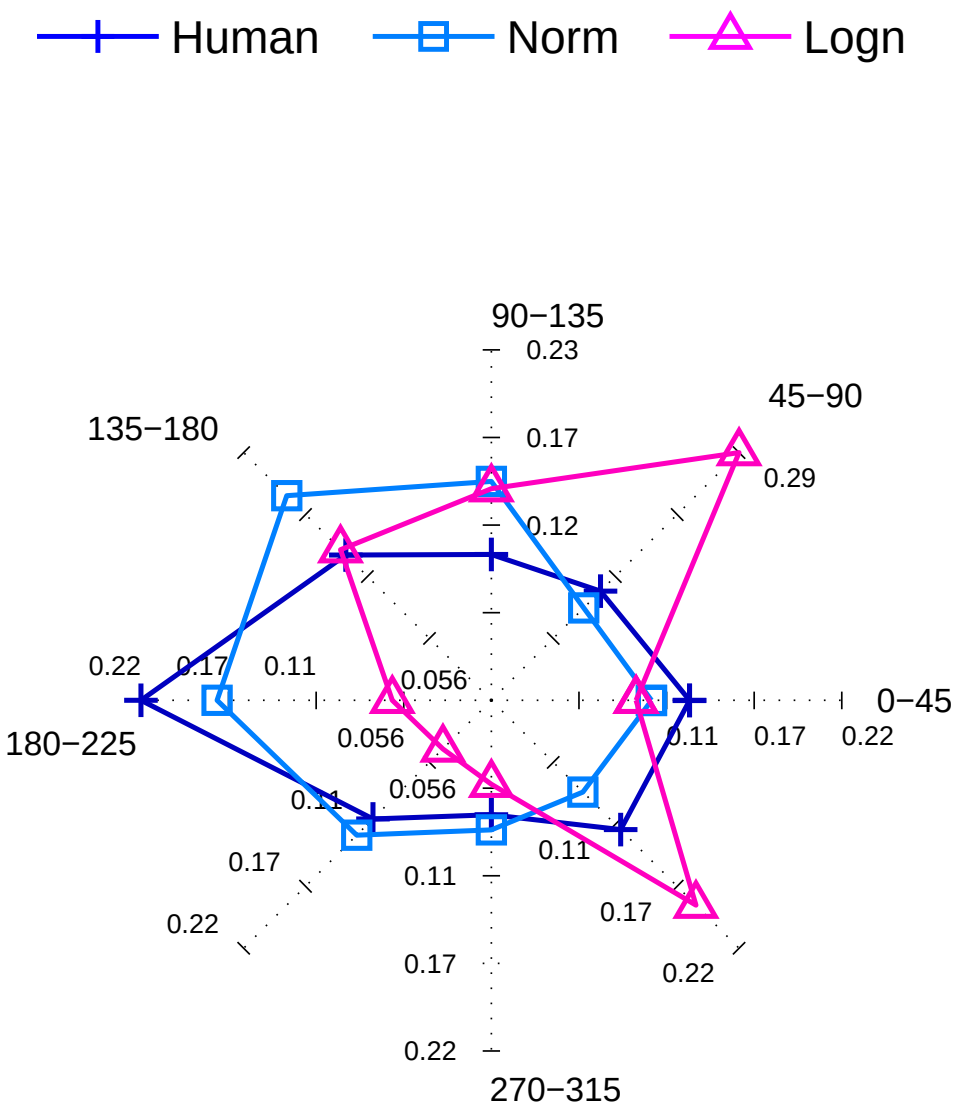

Figure 3.9: Facebook mouse clicks and movement angles

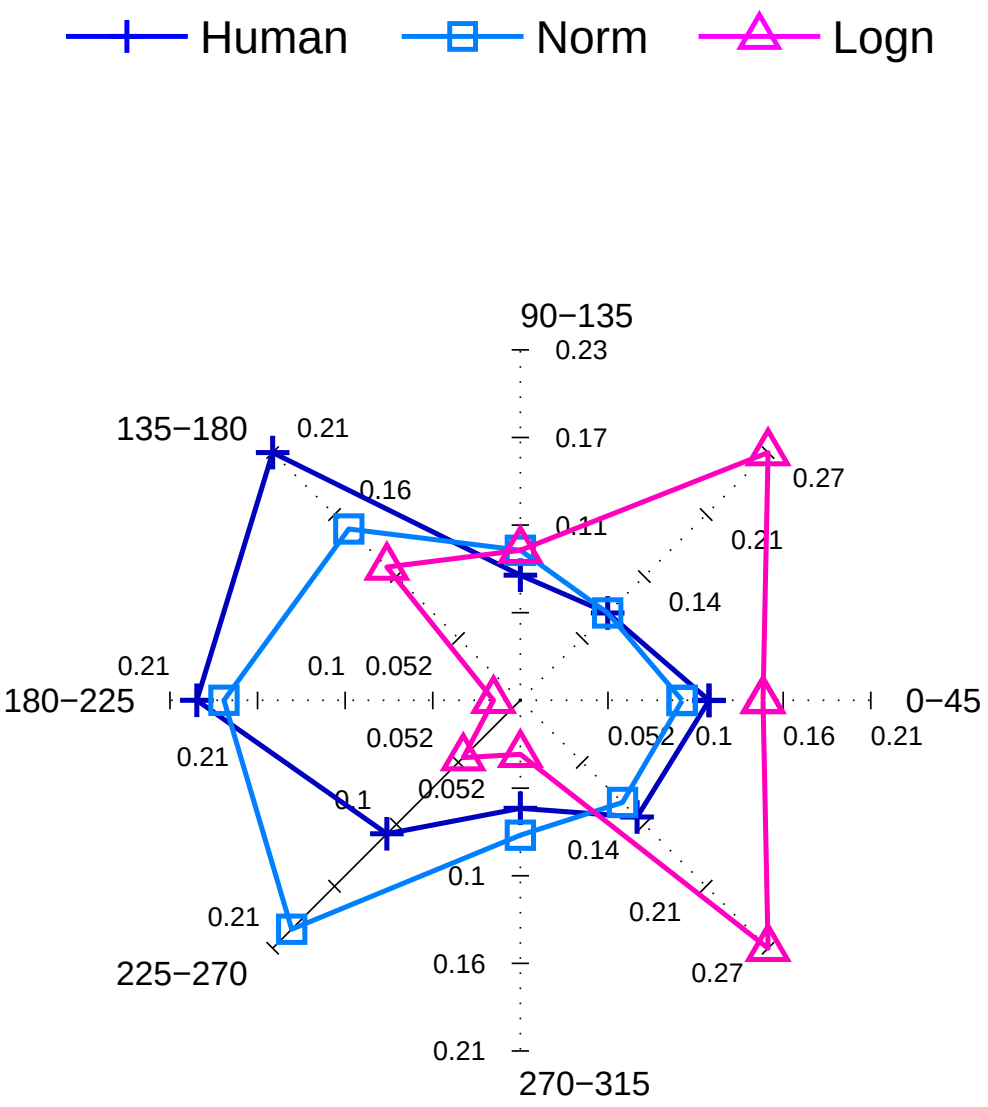

Figure 3.10: Blogger mouse clicks and movement angles

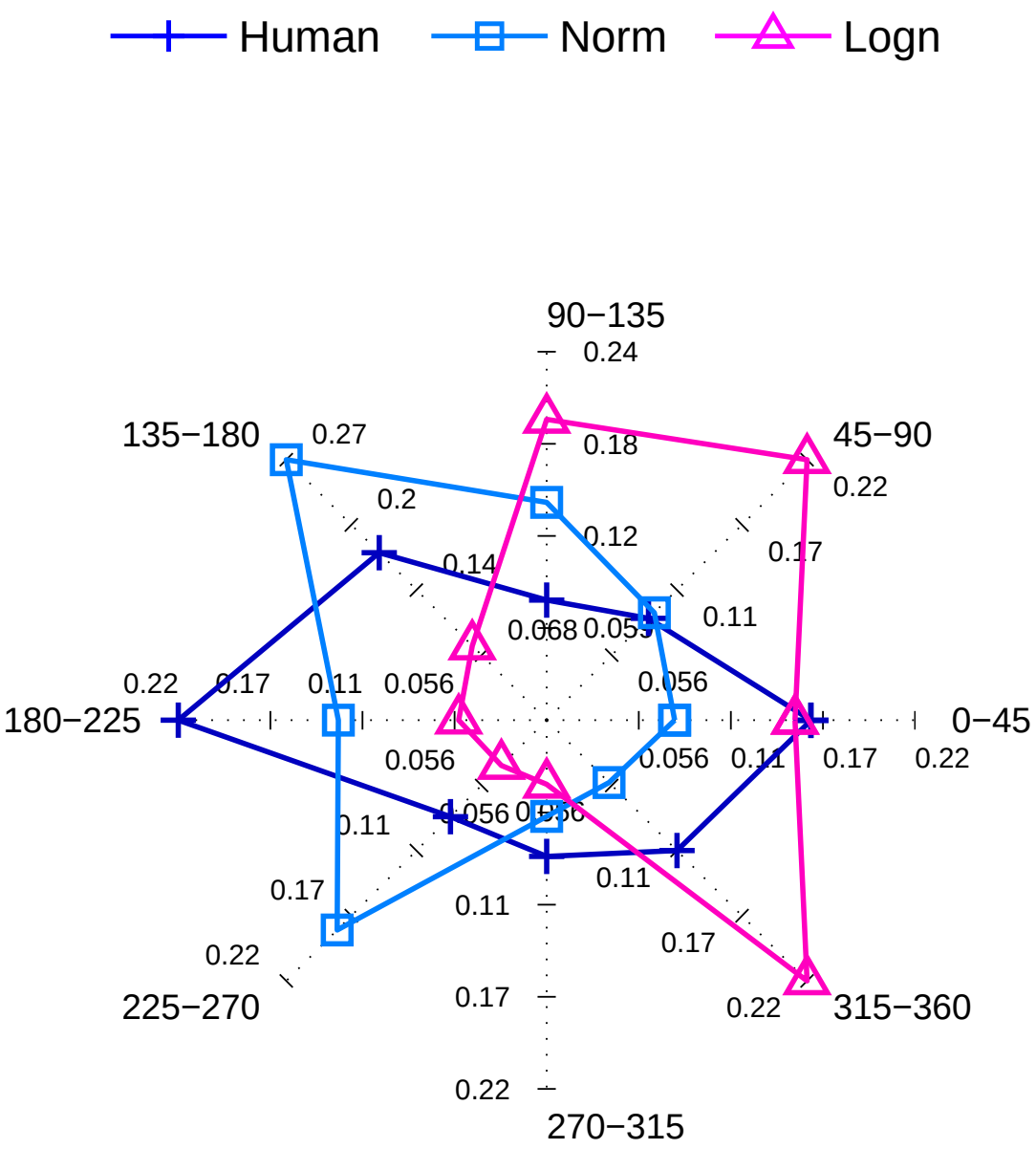

Figure 3.11: Twitter mouse clicks and movement angles

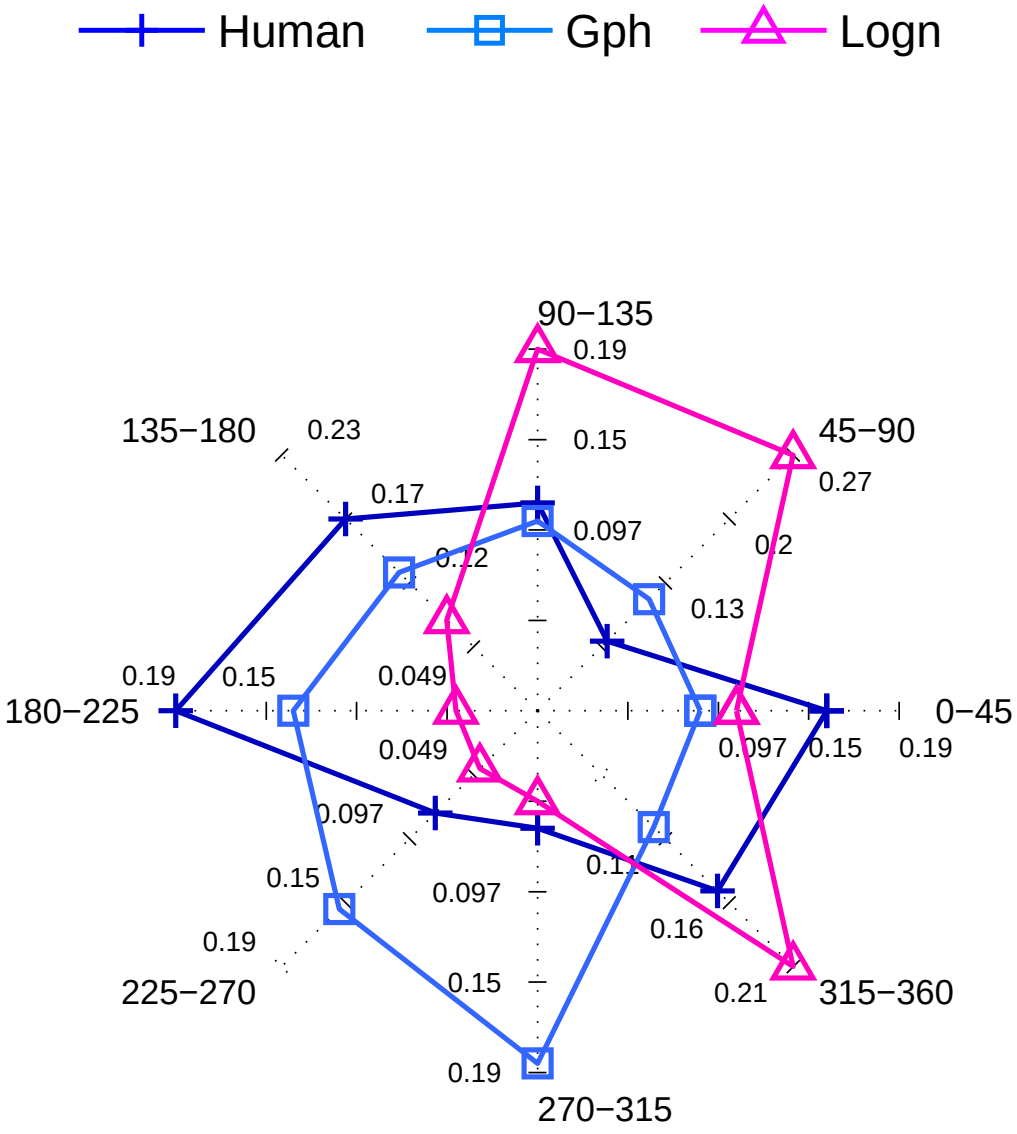

Figure 3.12: Gmail mouse clicks and movement angles

To estimate the consistent degree of human behavior, I use the statistics represented by the first three bars of min, max, and mean from Figures 3.6 and 3.7 for each application. I first locate the center point from the distribution of all the human data across all users, and then locate the central point from each human user data. Finally, I calculate the distance between the overall center point and the central point of each individual user data. These three distances of min, max, and mean will be also useful for further detectors to define their own detection thresholds.

Then I measure distance fittings from normal, uniform, Weibull, gamma, Gp, Logn and beta distributions. I observe that the different distributions perform significantly differently, no single model dominates, and each model displays different performances across different applications. I confirm that my generative engine needs to select the models dynamically. Current web bots usually use the best effort model or simple uniform behavior statistics.

Web Movement Events

I further compare the evasive fitting models to human movement models. The *event movement* is defined as the angle and distance between two continuous actions. Because distance correlates with the timing behaviors, here I only focus on the angles.

I measure the mouse movement in each session for the five web application benchmarks used in Section 3.5.4. Because the distribution is in a closed area, I map the value from $[-\pi, \pi]$ to $[0, 360]$ to display the distribution similarity intuitively. I evaluate the similarity between human and different generated movement behaviors in spider graphs, as shown in Figures 3.9 through 3.14. Each graph has one line for the human behavior and contains up to seven lines for each generative model. To display the similarity clearly, I only draw the two models with the largest and smallest distances.

The next step is to compare the bots' behaviors with the humans'. First I convert the closed paths in the spider graphs to a number defined by the D_{SKL}. For each web application, I calculate the D_{SKL} between human data and bot generated data via different fitting models. Because the user distance is relatively small compared to the fitting model

distances, I only show the maximum user distance to the human behavior distribution.

A summary of the inter-event movement's D_{SKL} across all candidate models is displayed in Figure 3.8. Unlike the timing, the user movement behaviors are very consistent and the threshold can be very strict. From these data, it appears that no bot generative fitting model can consistently avoid triggering an alarm. But in some cases, such as the Facebook and Blogger applications, the generative model can hide the bot behavior quite well.

Fitting and Model Selection for Individual Users

Recall that my framework has a generative engine that models human behaviors from a group of users and then evade those behavior-based bot detectors. In practice, my evasive model fitting and selection procedures are also capable of mimicking individual human users to seek the best model for the shortest distance. I use the Gmail web application as an example to demonstrate my dynamic model selection procedure to identify the best model to mimic individual humans. Figure 3.15 shows data from the procedure to mimic the human timing. With the human data, the framework takes the behavior data and fits it with the seven models in my model pool. Based on the distance measurement, the framework selects the best model for each individual user. The upper subgraph tracks the best models on mouse inter-click timing and the lower subgraph shows the minimum distance across the fitting results using the seven models. I plot a similar graph for demonstrating model selection on mouse movement angle and the corresponding shortest distances in Figure 3.16. Interestingly, I can see that the best model does change among different users, primarily because individual human behaviors sometimes are quite different. Different web application's functionalities can also cause subtle differences. In general, my framework is able to capture and characterize human behaviors, and it is also able to adjust its model to mimic diverse individual human behavioral patterns to evade bot detectors.

3.5.5 Defense Against Evasive Bots

Although I have proposed an evasive bot and provided a systematic approach to exploit the limitations of HOP-based detectors, my ultimate goal is to improve the security of online business by developing more advanced bot detectors. To defend against evasive bots, future detectors could take application context and user intent into account for bot detection. In the context of online applications, human users can easily perceive the application feedback and interact with the applications in a logical and sensible manner. But for bots, it will be challenging and computationally expensive to continually analyze the application context and then adjust their behaviors with the changing context in a timely manner. Moreover, the intent of a human user is very different from that of a bot. For example, in a web browsing scenario, a human user clicks on several links to read multiple web pages, while bots may click on the same link several times to abuse pay per click. The evasive bot can inject mouse movement actions and idle time between clicks to mimic a human, but it cannot fully hide the final results and its intent. Finally, the user behavior consistency degree I defined before could also be used to detect evasive bots, because evasive bots usually have a higher consistency degree than human users.

3.6 Related Work in Web Bots Detections

Several evasive approaches have been developed against web bot detectors. Spam filters can be evaded by poisoning training data with spam messages [82]. Bots can also use anonymous networks [58] to hide their identity so that it is less likely for the detectors to trace the source of web bots. In contrast, my attack focuses on web bots that mimic human behavior in a variety of ways, and applies to many more web applications. Additionally, in my attacks, compromised routers are not needed, as in Google Clickbot.A [83]. Evasion attacks also occur in evading biometrics-based authentication systems. Ballard et al. [92] presented a generative attack model by synthesizing handwriting automatically, which has been effective in evading existing handwriting based user authentication.

The technique of automatically mimicking human interactions with Ajax-based applications has been proposed in the AjaxTracker tool [93]. However, the goal of AjaxTracker is to generate workload and monitor network behaviors, instead of launching an evasive attack as in my work. Moreover, I provide a more flexible and extensible bot framework that directly controls a web browser.

3.7 Conclusions

Web bots have been widely deployed to perform tasks on the Web in an automatic fashion. However, the behaviors of existing web bots are intrinsically different from those of human beings, which allow them to be detected by HOP-based bot detectors. This paper proposes a generative approach to build an evasive web bot. Based on the generative evasion, I have prototyped a generic web bot framework that mimics human behaviors on the Web. The framework enables bots to generate events at the application level for evading bot detectors. I have abstracted and defined a set of benchmarks and metrics to measure my system's evasion performance. My experimental results demonstrate that my evasive bots can achieve high human likelihood against bot detection. This evasive bot framework can now be used to evaluate existing bot detection algorithms and help develop more advanced bot detectors.

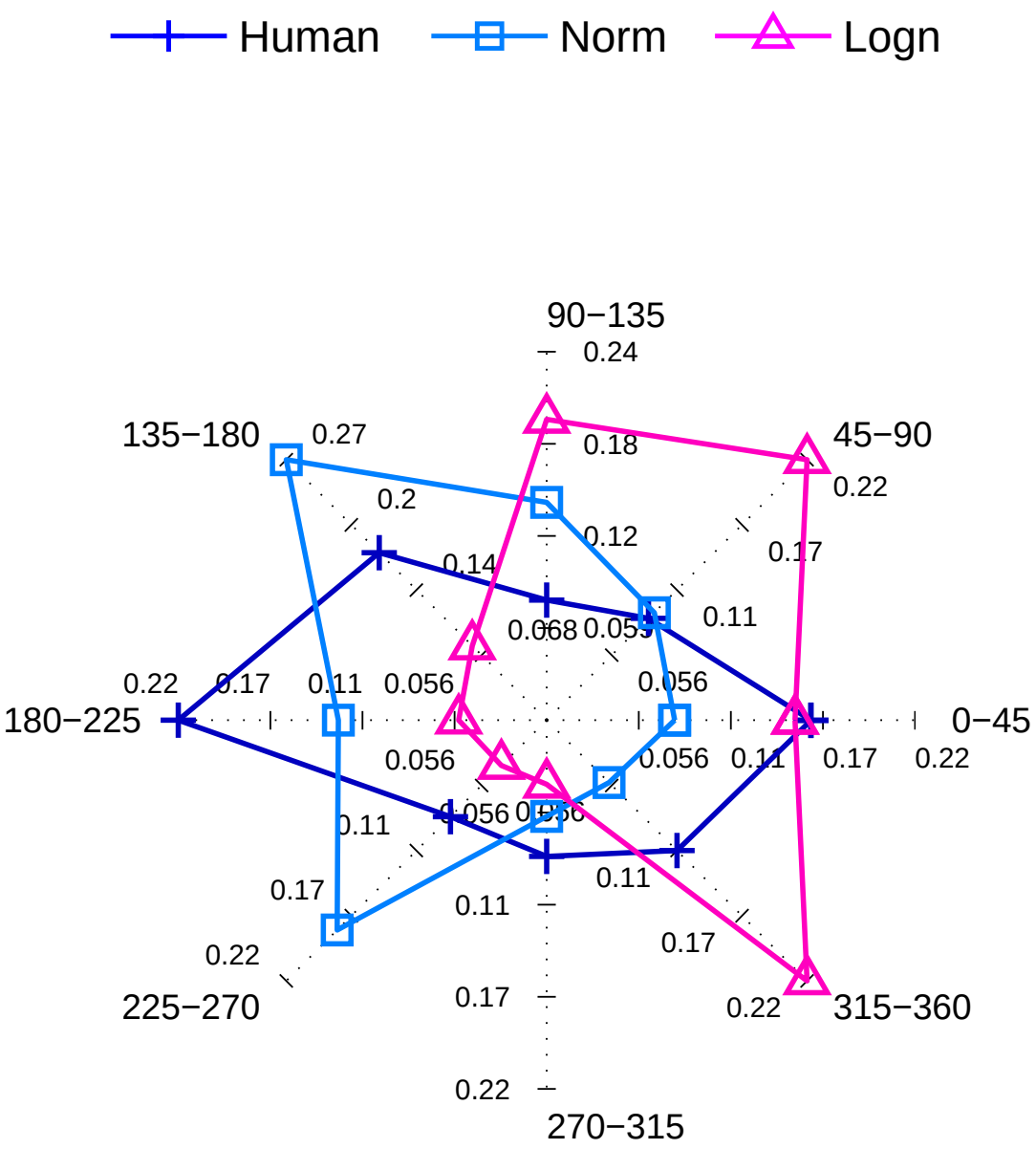

Figure 3.13: Twitter mouse clicks and movement angles

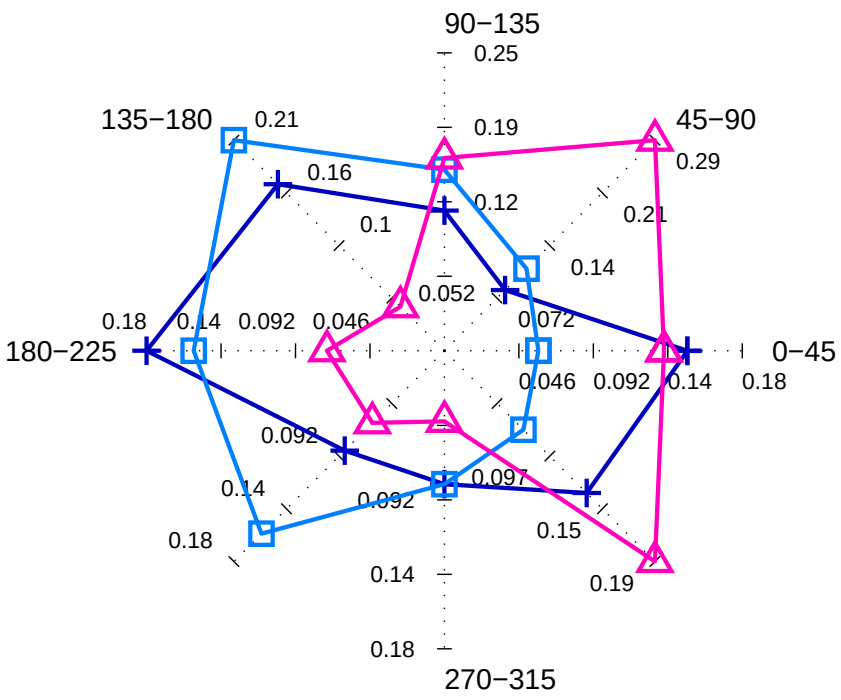

Figure 3.14: GoogleNews mouse clicks and movement angles

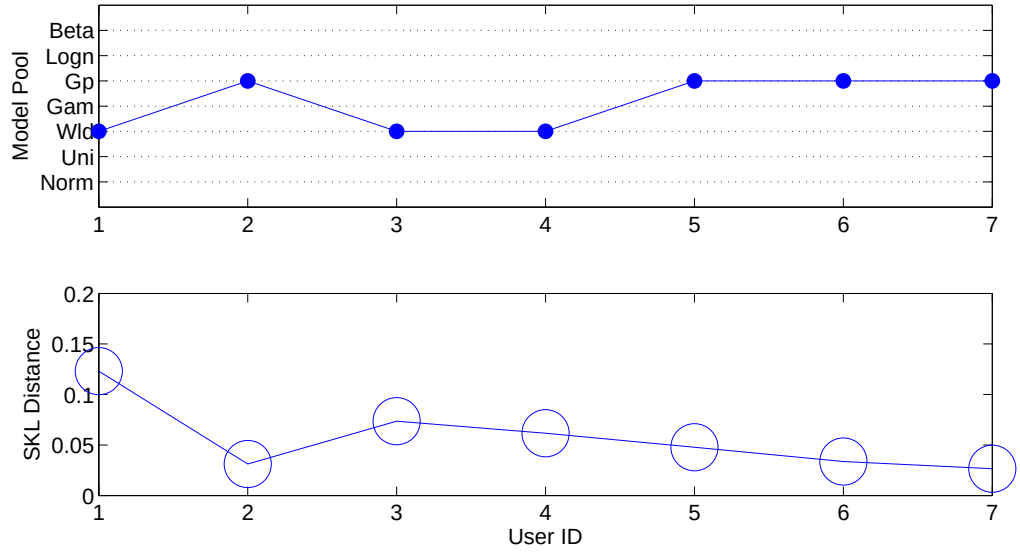

Figure 3.15: Model selection for a set of Gmail mouse click inter-events timing

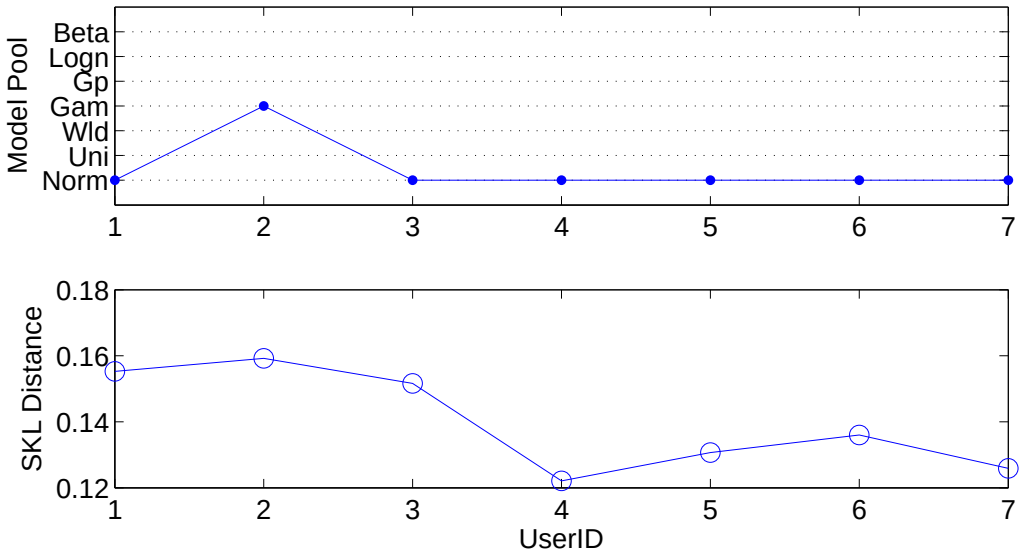

Figure 3.16: Model selection for a set of Gmail mouse angle

4. Timing Covert Channel Detection Resistance Analysis

Timing covert channel detection is a challenging research problem that is well-known in the security community. This chapter first studies existing detection methods. Then, it systematically studies stealthiness of a noisy covert channel by evaluating the detection resistances. Detection resistance is based on passively or actively modifying or removing traffic timing statistical patterns in the presence of timing covert channels. The source of the passive detection resistance of timing covert channel is detection noise introduced by network dynamic conditions, while the source of active detection resistance is the detection noise that attackers introduced by evading and modifying statistics of traffic pattern over time. Passive approaches take advantage of network natural characteristics of dynamic delay and bandwidth. Active approaches include mimicking legitimate traffic, mixing traffic, and changing encoding schemes.

This chapter proposes the first detection resistance measurement for IP timing covert channels. It first compares several potential evasive techniques by defining a detection resistance score. Then it provides approaches to measure detection resistance of covert channels that use both passive and active detection resistance techniques. The goal is to introduce the first step to the research community to show that it is necessary to consider detection resistance metrics in practical scenarios.

This research makes the following contributions. First, it performs an advanced study that experimentally measures detection resistance. Second, this research empirically studies stealthiness of existing covert channels.

4.1 Timing Covert Channels

As described in chapter 2, a covert channel [46–52] is a communication channel that violates a security policy by using shared resources in ways for which they were not initially designed. Covert channels that modulate timing information for a shared resource or process are known as *timing channels*. The chapter discusses covert network timing channels at IP layer. The sender of the channel is the subverted entity that is responsible for modulating the timing to encode information. It can be an application program, part of the operating system, or a hardware device. The receiver in the channel can either be a network connection endpoint or a passive eavesdropper that extracts information from the channel by looking at network IP packet timings.

To defend timing covert channels, researchers [61, 72] have proposed two main defense mechanisms. The first is to detect the presence of a channel. This is also a focus of my dissertation. The second is to eliminate a covert timing channel by removing all timing information from adding large timing distortion. The second mechanism has degraded network performance for overt traffic, hence detecting covert channels is a better timing covert channel defense method.

4.2 A Review of Detection Approaches

In general, the detector compares the pattern of the samples of legitimate traffic with the pattern of the passively monitored traffic, , in an attempt to detect covert channels. The Kolmogorove-Smirnov test compares the distributions of legitimate traffic with covert channels traffic. The regularity test emphasizes the traffic correlations.

4.2.1 Kolmogorove-Smirnov Test

One of the most frequently discussed approaches is the Kolmogorov-Smirnov test (KS test) [76], because it applies to different types of traffic with different distributions. Research [75, 94] has successfully applied the KS test to detect invisible inter-packet delays in

covert channels. In statistics, the KS test is a nonparametric test for the equality of continuous, one-dimensional, probability distributions that can be used to compare a sample with a reference probability distribution (one-sample KS test), or to compare two samples (two-sample KS test). The KS statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples.

Let $S(x)$ be the distribution function derived from empirically monitored IPD samples. These samples may include covert channel IPD traffic. Let L(X) be the cumulative distribution function (CDF) from the legitimate traffic IPD samples. In this case, the KS statistic H_s is:

$$H_s = sup_x|L(x) - S(x)|$$

The detectors choose to find a threshold according to a specific level of H_s. Since the KS test directly compares the empirical distribution functions, the samples do not need to be the same size. As detectors determine a threshold of certain scores of detection H_s, the designer of the covert channel finds ways to lower the score below the threshold to evade detection.

4.2.2 Regularity Test

Regularity test measures the correlation of data. The regularity test determines the differences of variance of the inter-packet delays between legitimate traffic and covert traffic [61]. Cabuk et al. observed that the variance of the inter-packet delays changes over time in legitimate traffic, while the variance of the inter-packet delays remains relatively constant over time as the encoding schemes are less likely to change over time [61]. Here I briefly summarize how the regularity test is applied.

Take two sample sets of data, divide the data sample into multiple sets with equal size window W. Select a set i with IPDs, compute the standard deviation of each set i as

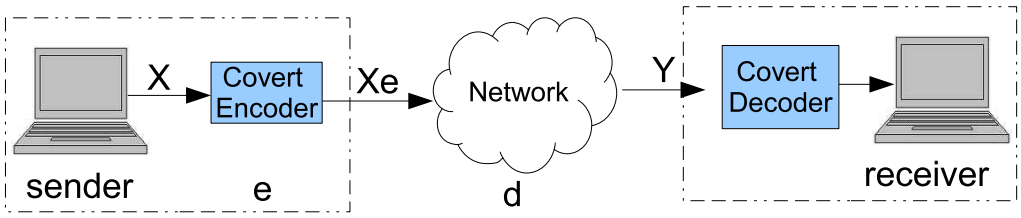

Figure 4.1: Model of a noisy timing covert channel

denoted σ_i. For any set j that is greater than i, calculate the standard deviation σ_j. The regularity score is the standard deviation of the absolute difference of any pair of σ_i and σ_j with a weight σ_i.

$$H(i,j) = std\left(\frac{|\sigma_i - \sigma_j|}{\sigma_i}\right), \forall i,j,i<j$$

As detectors determine a threshold of certain scores of detection $H(i,j)$, the designers of the covert channel find ways to lower the score below the threshold to evade detection.

4.3 Detection Resistance Technique

The detector observes IP traffic coming out of the network to compare covert traffic with previously trained legitimate data. Later the detector discovers the pattern of the timing covert channel as compared with the legitimate pattern. Ideally the detector should be physically located very close to the encoder. However, in practice, the detector can only monitor some network intermediaries nodes or even at the receiver. Therefore I assume that traffic Y in figure 4.1 is also observed by the detectors, which is traffic coming out of network. Either the network itself can introduce noise, or the sender can intend to hide cover channel from detection.

4.3.1 The Source of Passive Detection Resistance

Figure 4.1 illustrates the abstract model of a timing covert channel and its related entities. Let X be the incoming IP traffic to a target network, and Y be the corresponding outgoing traffic. A timing covert channel hidden within the network consists of a sender who embeds timing covert messages in a receiver through the network channel. Specifically, a sender embeds timing covert signals into the legitimate traffic X. Let the legitimate traffic X be *encoded* to be X_e. The encoded traffic X_e enters a network, so traffic X_e becomes Y due to covert channel noise n. Traditionally, noise is defined as any unwanted signals or effects in addition to the desired signal, which means a signal from the sender may be perturbed by noise. In timing covert channels, traffic manipulation may add unwanted timing signals or remove desired timing signals. Consider *natural* noise to be the noise generated by network perturbation naturall without intentionally delaying traffic by the sender or intermediaries in the network. Most of the natural noise comes from network delays at intermediate nodes, all of which could perturb timing signals of the channel, in other words, the sender or intermediaries can intentionally add perturbations to network traffic, causing noise insertion to cover desired signals. Here e denotes the embedded timing delay, and d denotes the timing changes from n. Hence, the relationship among X, X_e and Y can be specified as

$$X_e = X + e \tag{4.1}$$

$$Y = X_e + d = X + e + d \tag{4.2}$$

.

4.3.2 Active Evasion

Chapter 2 surveyed several timing covert channels and discussed how they work. This section discusses the stealthiness of these channels from the perspective of the capability

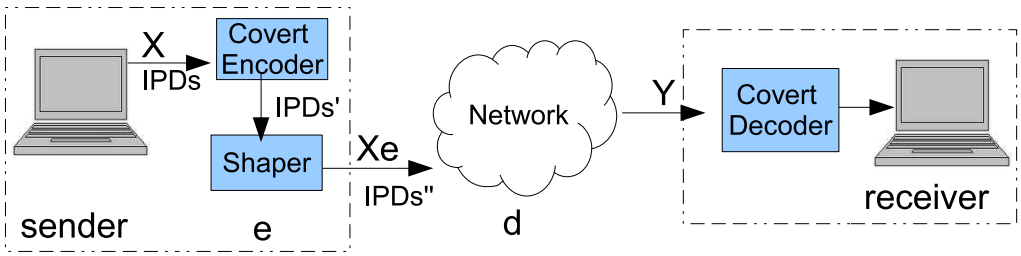

Figure 4.2: Deployment of timing covert channel evasion

of evading detection methods and presents the properties of these channels for potential detection resistance approaches.

Timing covert channel designers implement evasion either at the encoder or at the shaper, or both the encoder and the shaper. As shown in figure 4.2, the encoder could randomize the way IPDs are added to the traffic, and the shaper could also remove statistics in the traffic. The encoder is mandatory in covert channel, while the shaper is optional.

Evasion at the Encoder

Encoding with Random IPDs In the simple case of a binary scheme in IP SCC [61], the covert channel encoder assigns a distinct IPD value to represent bit 0 and bit 1. Since only two different values are used to encode the bits, the timing IP SCC has low detection resistance. To evade detection on static IPDs, it is good to use random IPDs to encode a 0 and a 1 bit. The random IPDs could be a 0 for a 0 bit, and a positive value for a 1 bit. However, to maintain the capacity of the covert channel, the positive IPD must have an upper bound.

Rotating Time Window Similar to timing IP SCC, the timing window is set to balance the number of errors that are caused by network jitter and channel capacity. Jitterbug [57] uses a pseudo-random sequence, s, so that the IPDs do not cluster around multiples of w. Each value in s is in the range $[0, w]$.

Before adding the random s, Jitterbug [57] encodes the symbols by delaying a key stroke such that the resulting encoding has the following equation:

$$IPD_i \ mod \ W = \begin{cases} 0 & if \ b_i = 0 \\ W/2 & if \ b_i = 1 \end{cases}$$

After adding the random sequence s as agreed by encoder and decoder, the equation becomes:

$$(IPD_i - s_i) \ mod \ W = \begin{cases} 0 & if \ b_i = 0 \\ W/2 & if \ b_i = 1 \end{cases}$$

Changing Coding Schemes Another thread of research was to encode covert messages using *error control coding (ECC)* [95]. For example, Houmansadr et al. [96] proposed CoCo, a general coding scheme using various error control coding schemes to maximize covert channel capacity. CoCo encodes messages based on IPD, and compares lots of error control codes. Yali et al. [97] designs a robust and high capacity covert timing channel by manipulating the delay between successive packets. At the same time, the covert channel should be undetectable by common statistical tests reported in the literature. Although these ideas were not designed to evade detection, various coding schemes are used for covert channel design. I could rotate using different coding schemes to make the traffic less prone to certain patterns from one coding scheme. One of the challenges of rotating different coding schemes is synchronization. The attacker needs to find the start and end point of each round of encoding so that the decoding can be successfully performed with very low bit error rate.

Evasion at the Traffic Shaper

The traffic shaper is not mandatory, but it is a good supplement to the encoder for evasion. As in figure 4.2, the shaper only modulates traffic coming out of the encoder. Similar to evasion on encoder, the goal of shaper is also to increase the randomness of IPDs while

maintaining low bandwidth.

Mimic and replay Gianvecchio et al. [62] proposed a Model-based Covert Timing Channel (MBCTC) to create a TCC that attempts to mimic the statistical properties of legitimate network streams. This research created models of legitimate traffic, which helps to build MBCTC. The message is then split into symbols that are mapped to IPDs based on the inverse distribution function of the chosen distribution. Packets are sent using the calculated IPDs. The cumulative distribution function determines the decoding process. This research is the first attempt to mimic legitimate IPDs to construct a timing covert channel. The covert traffic can be considered as a reply of legitimate traffic with covert symbols.

Shaping Walls et al. [98] proposed a passive CTC that uses a portion of the compromised stream to smooth out the distortion detected by shape detection tests (e.g., entropy based detection). This technique, called Liquid, uses a combination of transmitting and shaping IPDs. IPDs used to encode the hidden message are called transmitting IPDs and IPDs used to maintain the shape of distribution are called shaping IPDs. Liquid uses half of the IPDs to encode the hidden message using JitterBug encoding and the other half of the IPDs to increase the probability of the symbols, which are not used by transmitting IPDs in the entropy test. In the entropy test, each IPD is mapped to one of the M symbols, which are histograms of IPDs. Liquid keeps track of the mapped symbols during message encoding and tries to generate other symbols that have not yet been generated or have been generated only a few times. This ensures that the probability of any symbol appearing in the entropy test is almost equal, thus increasing the entropy value. Furthermore, Walls et al. provided an estimation framework to determine the trade off between the amount of shaping required and the desirable detection resistance. To achieve high detection resistance, the attacker must reduce the throughput of the CTC.

4.4 Detection Resistance Measurement

Previous research did not address detection noise for detection resistance, which is the network noise or client evasion effect on IPDs, when the detector can only monitor covert traffic in the network or traffic going out of the network. As a result, the detection methods only give the best case scenario to detectors. For example, random IPDs in JitterBug was used to increase the probability of the symbols not being used by transmitting IPDs in the shape test. However, the network or client introduces random delays when transmitting IPDs, and these random delays also increase the probability of the symbols not used by transmitting IPDs. This leads to significantly higher false positive rate to detectors and thus increases detection resistance. Given the real world network scenario, this section depicts a detection resistance score that quantifies the detection resistance from detection noise and evasion.

4.4.1 Normalized Channel Delay Noise

Let σ_0 be the standard deviation of the channel noise with index i calculated by the IPD approach, where

$$\sigma_{ipd} = std(D_{ipd}) = std\{d_1, d_2, \ldots, d_n\} \tag{4.3}$$

$$where \quad d_i = (t(p_i^y) - t(p_i^x)) - (t(p_{i-1}^y) - t(p_{i-1}^x)) \tag{4.4}$$

d_i is the ith channel noise, where $t(p_i^x)$ and $t(p_i^y)$ denote the time stamps of the i-th packet for traffic X and Y. Similarly, $t(p_{i-1}^x)$ and $t(p_{i-1}^y)$ denote the time stamp of the packets of the i-th packet of traffic X and Y. Let n be the number of packets of both traffic X and traffic Y, then $i = (2, \ldots, n)$.

To improve equation (4.3), I have developed an interval approach (ITD) that divides a traffic duration into time intervals of equal length as fixed size buckets. Once I have fixed

size buckets, I assume that there is only one virtual packet in each bucket. This packet could be a *compound* of multiple packets if this bucket holds more than one packet. This packet could also be the *original* packet if there is only one original packet. Moreover, this packet could be a *fake* packet when there is no packet originally in this bucket. The following section explains the interval-based approach, including how to divide traffic into fixed interval lengths, and how to construct the virtual packets in intervals.

Assume m_x and m_y are the number of packets in traffic X and Y, $m_x \neq m_y$, $m_x > 0$ and $m_y > 0$. The sequence of packets in X is $p_1^x, \ldots, p_{m_x}^x$ and the sequence of packets in Y is $p_1^y, \ldots, p_{m_y}^y$. Since the covert signal carrier is a low latency network, I can assume that the time durations of incoming traffic X and outgoing traffic Y are almost identical. T_f denotes the duration, so $|T_f^x - T_f^y| = \epsilon$, $\epsilon \leq T$ is very small. This makes adjusting the time duration straightforward:

$$T_f = T + max(T_f^x, T_f^y) - (max(T_f^x, T_f^y) \mod T) \tag{4.5}$$

Further, I divide the traffic duration T_f into $N = \frac{T_f}{T}$ time intervals of equal length $T > 0$, and use $S(j)$ to represent the start time point of interval $j = (1, \ldots, N)$. Packet p_i falls into the interval $j = \lfloor \frac{t(p_i) - t(p_1)}{T} \rfloor$. I use $n(f, j)$ to denote the number of packets in interval j of traffic f, then $\sum_{j=1}^{N} n(x, j) = m_x$, and $\sum_{j=1}^{N} n(y, j) = m_y$. Now the i-th packet can be directly converted to the k-th packet in its own interval j, and the time offset of p_i (or $p_{j,k}$) from the start point of the j-th interval is

$$\delta_{t(p_i)} = t(p_i) - S(j) \tag{4.6}$$

By replacing the notation i with (j, k), I get

$$\delta_{t(p_{j,k})} = t(p_{j,k}) - S(j) \tag{4.7}$$

$k = (1, \ldots, n(f, j))$ of interval j. Further, I calculate the mean of the offsets in the j-th interval

$$\bar{\delta_j} = \frac{\sum_{k=0}^{n(f,j)} (t(p_k) - S(j))}{n(f, j)} \tag{4.8}$$

Then the revised channel noise d_j for normalization is constructed as

$$d_j = (\bar{\delta}_j^y - \bar{\delta}_j^x) - (\bar{\delta}_{j-1}^y - \bar{\delta}_{j-1}^y) \tag{4.9}$$

and the new standard deviation with index j becomes

$$\sigma = std(D) = std\{d_1, d_2, \ldots, d_N\} \tag{4.10}$$

Note that if no packet is located in this interval, that is, $k = 0$, I construct a fake packet at the center of the interval with $d_j = \frac{T}{2}$. That is, I assume the missing corresponding packet is sitting in the middle of the interval.

4.4.2 Detection Resistance Score DRS

At a high level, the normalized noise described in 4.4.1 reduces the actual timing perturbation information between the incoming and outgoing traffic. This is because (1) the length of the intervals N is mostly less than the length of packets m_x in the practical scenarios, and (2) I only assume one packet in each interval. To compensate for the information loss during normalization, it is necessary to explore the impact of packet number differences between incoming and outgoing traffic.

A straight forward approach is to find the packet number difference by comparing the packet number distributions of incoming traffic X and outgoing traffic Y. I can build a histogram of packet number distributions of $j = (1, 2, \ldots, N)$-th intervals. Each bar in a histogram corresponds to one interval and the bar value is the corresponding percentage of

packet numbers in this interval. As traffic Y is derived from traffic X in the network, X is considered to be the history of Y. Intuitively, the packet number distribution of traffic X must be very similar to traffic Y if the network has little timing perturbation. To make it more concrete, let Q^X and Q^Y stand for two normalized histograms of traffic X and traffic Y. q_j is the normalized jth interval value in the histogram Q. Therefore, the two distributions can be written as $Q_j^X = \{q_1^x, q_2^x, \ldots, q_N^x\}$, and $Q_j^Y = \{q_1^y, q_2^y, \ldots, q_N^y\}$, where $\sum_j q_j^x = 1$ and $\sum_j q_j^y = 1$.

Let us consider a discrete distribution having probability function Q^X, and let a second discrete distribution have probability function Q^Y. Then the relative entropy of Q^X with respect to Q^Y, also called the Kullback−Leibler divergence [84], is defined by

$$D_{KL}(Q^X||Q^Y) = \sum_{j=1}^{N} q^x(j) \log \frac{q^x(j)}{q^y(j)} \tag{4.11}$$

For each interval j, the ratio $\frac{q^x(j)}{q^y(j)}$ measures the packet number change on the interval j. The log of this ratio is then determined by the probability $q_x(j)$, and the Kullback−Leibler divergence in equation (4.11) is the sum of the weighted log ratios over all intervals. Although the Kullback−Leibler divergence is not a true metric, it satisfies many important mathematical properties. For example, it is a convex function of $q(j)$, is always nonnegative, and equals zero only if $q^x(j) = q^y(j)$.

Several things about the calculation of D_{KL} are worth further discussion. KL divergence is not symmetric, $D_{KL}(Q^X||Q^Y) \neq D_{KL}(Q^Y||Q^X)$. Second, $q^x(j)$ and $q^y(j)$ could be zero. The consequence is that $logq^x(j)$ and $logq^y(j)$ could approach infinity, which is clearly not desirable.

Next I discuss how to handle these problems.

1. $D_{KL}(Q^X||Q^Y)$ is not symmetric. To make it symmetric, I can instead use a mean of $D_{KL}(Q^X, Q^Y)$ and $D_{KL}(Q^Y, Q^X)$. Therefore, I revise the D_{KL} to make it symmetric

as in equation (4.12).

$$D_{SKL}(Q^X, Q^Y) = \frac{1}{2}(D_{KL}(Q^X||Q^Y) + D_{KL}(Q^Y||Q^X)) \tag{4.12}$$

2. The probability $q^x(j)$ and $q^y(j)$ could both be zero. Two important cases need to be addressed. First, since I know that $\lim_{q\to 0} q \log q = 0$, if both $q^x(j)$ and $q^y(j)$ are zero for the same interval j, $D_{SKL}(j) = q^x(j)\left|\log \frac{q^x(j)}{q^y(j)}\right| + q^y(j)\left|\log \frac{q^y(j)}{q^x(j)}\right| = 0$. The second case is more complicated. If $q^y(j) \neq 0$ when $q^x(j) = 0$ in $\log \frac{q^y(j)}{q^x(j)}$, or $q^x(j) \neq 0$ when $q^y(j) = 0$ in $\log \frac{q^x(j)}{q^y(j)}$, the divergence is infinite. However, in our context, Q^X and Q^Y are derived from observations and sample counting, that is, Q^X and Q^Y are probability distributions derived from frequency distributions. In this case, I should never predict that a traffic interval with at least one packet is completely impossible. When I derive the probability distribution, I must take into account the possibility of unseen packets. Then I try to perform "*smoothing*," that is, I scan through all intervals in X and Y to add one packet to empty buckets so that there are no zero probabilities for q.

3. For $D_{KL}(Q^X||Q^Y)$, the $\log \frac{q^x(j)}{q^y(j)}$ could be either positive ($q^x(j) > q^y(j)$), or negative ($q^x(j) < q^y(j)$). However $q^x(j)$ are always positive. So the $q^x(j) \log \frac{q^x(j)}{q^y(j)}$ may cancel each other out for different js. In our case, for example, when I look into the distance between $Q^X = \{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$ and $Q^Y = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$, the first addition of a logarithm operation reduces the distance. Therefore I further revise equation (4.12) to add an absolute operation as in equation (4.13):

$$D_{ASKL}(Q^X, Q^Y) = \sum_{j=1}^{N} q^x(j)\left|\log \frac{q^x(j)}{q^y(j)}\right| + \sum_{j=1}^{N} q^y(j)\left|\log \frac{q^y(j)}{q^x(j)}\right| \tag{4.13}$$

The distance indicates how difficult it is to distinguish the encoded traffic from the legitimate traffic. Formally, the detection resistance score is defined as

$$DRS_0 = \frac{1}{D_{ASKL}(Q^X, Q^Y)} \tag{4.14}$$

The higher the DRS, the better to prevent detection, because an higher score means a small distance for the detector to distinguish encoded traffic from the legitimate traffic.

4.5 Evaluation

This section analyzes the detection resistance score with passive and active evasion traffic distributions. Later I demonstrate how I select parameters of detection resistance score using offline synthetic traffic. Then I evaluate the effectiveness of this metric by analyzing SSH traffic with and without an anonymous network Tor [59]. I analyzed detection resistance score results to demonstrate how the detection resistance score can affect the stealthiness of TCC. As a baseline to start with, I used the same decoding result for evaluating detection resistance score. All data sample of Jitterbug used in this research has less than 20% of bit error rate.

4.5.1 Passive Detection Resistance

The passive detection resistance is introduced by the network condition. It is one of the natural characteristics of the traffic, so the attackers do not need to do anything to add noise. The introduced noise is not always stable due to the dynamic network status. It varies from time to time. Therefore, it is impossible for me to cover all possible situations. Instead, my research tries to explore the baselines.

Table 4.1: Network statistics from trace route between east coast and west coast, with and without Tor (212.83.131.52)

Scenarios	Label	Hops	RTT	Geographic Region
S1: client to local server	W-W	4	2.5ms	US West to US West
S2: client to Mason	W-E	13	138ms	US West to US East
S3: client to Tor to Mason	W-T-E	35 (total)	325ms (total)	US to France to US
client to Tor	E-T	18	179ms	US to France
Tor to Mason	W-T	17	156ms	France to US

Experiment Setup

I used SSH traffic to conduct timing encoding. One the one hand, SSH is used to study IP time patterns. On the other hand, it is relatively easier to control the IPDs for encoding via SSH traffic. The experiment used introduced two fold delays as the passive noise to affect the detection. First, I setup the SSH server at mason.gmu.edu on the US east coast and a client on the California, US west coast. Thus traffic went across the whole continent. Second, I selected the one node Tor 212.83.131.52 network in between for extra network routing and delay. Table 4.1 demonstrates the result from the linux command *traceroute* to show number of hops and round trip time (RTT). W-W means US west coast to west coast, W-E means US west coast to east coast, and W-T-E means US west to Tor node in France and to US west coast.

On the SSH server, I ran a remote bash script that echoed to the client randomly. Between each echo, the script slept for a random number of seconds between 0 and 1 using the internal random number generator $RANDOM. I randomly injected maximum of 1 seconds as the encoding. On the SSH client side, I ran Wireshark to capture the packets and observe IPDs. Those packets were sent from the SSH server to the client via echo commands.

Results

Figure 4.3 contains four diagrams. They capture the IPDs distributions for four scenarios: direct connection (W-W) SSH legitimate traffic, direct connection (W-W) SSH encoded

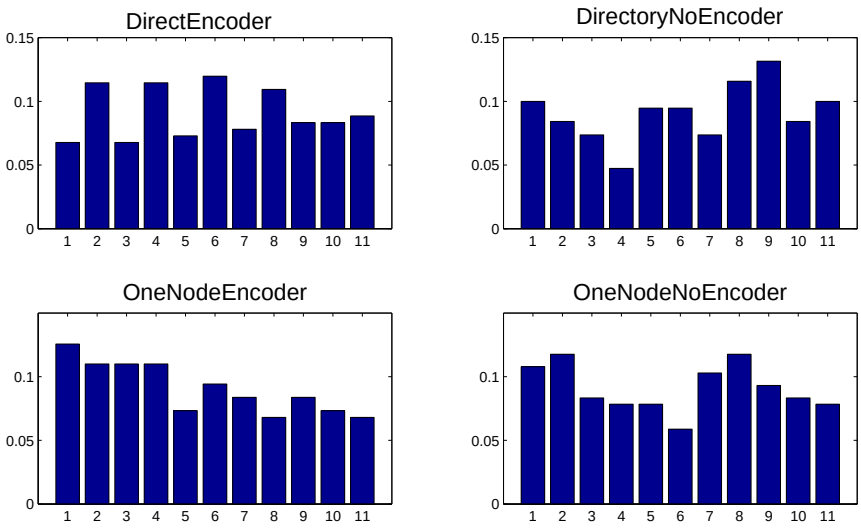

Figure 4.3: Natural passive IPDs histogram at detection point

traffic, one node Tor connection SSH legitimate traffic and one node Tor connection SSH encoded traffic.

Compared with the direct connections SSH legitimate traffic, the direct connection SSH encoded traffic has a clear pattern for its encoding. After the one node Tor, the covert channel encoding timing pattern become less obvious. In figure 4.4, the two direction resistance scores show consistent results. The one node Tor connections SSH encoded traffic has a higher DRS when compared with legitimate traffic via Tor with only one intermediate node.

4.5.2 Active Evasion

Adding noise proactively on the sender side is more reliable for evasion, because the network is a shared resource and out of the senders's control in most cases. The experiment tried different configurations for different traffic patterns. In this chapter, I chose the traffic shaper at the sender side for our experimental analysis. For other evasion approaches, I will have them as future work.

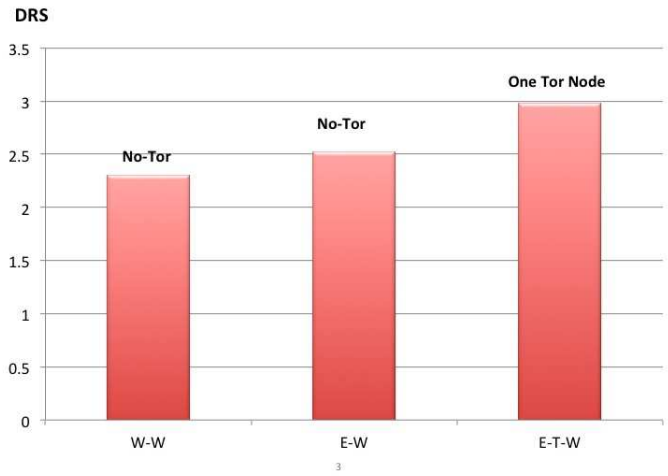

Figure 4.4: The results of passive DRS

Experiment Setup

I generated traffic of 3000 seconds long with IPD distributions of *Uniform*, *Normal*, *Poisson* and *Pareto* using random number generators as legitimate traffic. Table 4.2 and 4.3 illustrate the detailed parameters for each distribution and the number of packets in both traffic X and Y. (a, b) are the low and upper bound of packet inter-arrivals in the uniform distribution traffic, μ is the average rate, σ is the standard deviation of the normal distribution, and λ is the average rate of Poisson distribution. k is a shape parameter, α is a scale parameter, whose threshold θ is zero. m_x and m_y are packet numbers in X or traffic Y.

I assumed that there is no significant packet drop, re-packet or traffic mixing, so there is a similar number of packets in end-to-end traffic.

I set the injected delay $A = 0.5s$ as the mean of the covert signal following the same distribution of legitimate traffic, and inserted covert signals by randomly selecting packets in traffic X and added timing delays to those selected packets to evade the legitimate traffic.

Results

Figure 4.5 demonstrated DRS of different distributions with the same packet average rate $4R = 10$ pkt per second across all distributions, where R is the base rate for the synthetic

Table 4.2: The baseline of Uniform, Normal, Poisson and Pareto distributed 3000 seconds long traffic X and Y with various packet rates. Rate stands for the packet rate Pkt/Second.

	Uniform	Normal		Poisson	Pareto	
Rate	(a, b)	μ	σ	λ	k	α
R≈2.5	$(0.3, 0.5)$	0.4	0.08	2.5	0.01	0.4
2R≈5.0	$(0.15, 0.35)$	0.2	0.04	5.0	0.01	0.2
4R≈10	$(0.05, 0.15)$	0.1	0.02	10	0.01	0.1
6R≈15	$(0.05, 0.08)$	0.067	0.0134	15	0.01	0.066

Table 4.3: Number of packets in Table 4.2

	Uniform	Normal	Poisson	Pareto
Rate	m_x/m_y	m_x/m_y	m_x/m_y	m_x/m_y
R≈2.5	7589/7601	7454/7483	7632/7464	7474/7443
2R≈5.0	15003/15002	14999/15012	15020/15024	14978/14345
4R≈10	29987/29954	29968/29950	30054/29801	29045/29041
6R≈15	45078/450983	44785/44795	44989/45068	44132/44803

traffic.

I compared the detection resistance score with different packet rates under different distributions in figure 4.5. The value of the detection resistance score decreases when the packet rate increases linearly. The result indicates when the legitimate traffic has a simple traffic timing pattern like Normal, Uniform and Poisson distribution, the TCC is easier to be detected and presented low DRS. The mean of delayed time interval has less impact on it. In contrast, for legitimate traffic with Pareto distribution, the DRS is significant higher than other distributions. It may due to the fact that Pareto distribution is close to legitimate traffic such as HTTP traffic timing patterns. Also, the mean of delayed time interval has a big impact on the effect of evasion. The mean of delayed time interval must reach a certain level to really change traffic shape.

4.5.3 Discussion

Table 4.4 describes the qualities of an evasive timing covert channel. It illustrates quality metric descriptions. Time duration means the time used to evade a detection method. The longer the duration is, the more likely traffic can be shaped by inserting randomness

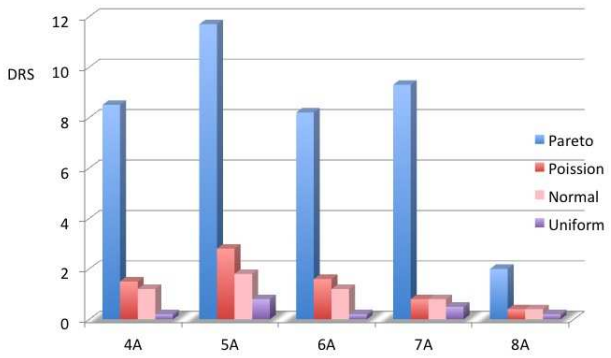

Figure 4.5: The results of active evasion DRS

Table 4.4: Quality Metric Descriptions

Quality Metric	Description
Time Duration	Time used to evade a detection methods
Time Variation	Variance in time affected by the type of inputs
Efficiency (Capacity)	Number of bits successfully transferred through the channel
Reliability (BER)	Bit error rate

to the channel. Time variation means variance in time affected by the type of inputs. The goal is related to time duration. The evasion of the channel would inject more timing variation. Efficiency and reliability are two important properties of covert channels discussed in chapter 2.

4.6 Related Work

Early timing covert channels [99] conveyed information by sending packets in an ON/OFF fashion, so that the receiver can observe timing data and infer restricted information. This method has limited covert capacity. It is not very robust to timing perturbation and easy to detect by statistical adversaries [61]. In interactive network applications, the timing information of keystrokes could potentially be used to affect the timing information of the actual network stream.

JitterBug [57] was designed to exploit this property to embed CTCs into these streams. One such network application is SSH. In the interactive mode of SSH, every keystroke in a terminal causes a packet to be sent immediately. The Keyboard JitterBug timing channel can suffer from three kinds of transmission errors: insertions, deletions and inversions.

Later researchers [53–55, 71, 100] paid attention to reliability, efficiency and stealthiness (RES) of timing covert channels. Reliability was considered first. One type of covert timing channels is that an attacker can insert the covert message directly in the inter-packet delays (IPD). For example, Liu et al. [100] modulated covert messages in the IPDs of the traffic by using spreading codes to increase the robustness of the covert channel to the network perturbations.

For storage-based channels, Smith and Knight [101, 102] proposed three parameters with regard to reliability, efficiency and stealthiness. They provided several Toroid coding schemes to handle channel noise such as covert symbol insertion and deletion. Further, researchers [53, 71] proposed to encode *watermarking* messages into interval-delay (ITD) to maximize the robustness of covert communication. Similarly, Yu et al. [55] developed a flow watermarking scheme-based Direct-Sequence Spread Spectrum (DSSS) technique due to the fact that the low-latency anonymous network such as Tor [59] have imposed large timing perturbations through circuit rotation [58, 59].

As advanced timing covert channels were proposed, some researchers emphasized stealthiness with the assumption that reliability and efficiency can be achieved. Shah et al. proposed the keyboard JitterBug [57], a low capacity channel for leaking typed information from a network connection. Rainbow [54] proposed a different watermarking scheme from Wang et al. [53] that addressed invisibility to evade watermarking detection. These covert timing channels schemes try to make covert messages harder to detect.

4.7 Conclusion

This chapter mainly focused on the stealthiness of the stealthiness of covert channel. This chapter first studied existing detection methods. It also systematically studied stealthiness of a noisy covert channel by evaluating the detection resistances. Due to the fact that detection resistance is based on passively or actively modifying or removing traffic timing statistical patterns in the presence of timing covert channels. It researched passive detection resistance by analyzing traffic that go through US west coast to east coast and the traffic that go through Tor network. It also studied active evasion approach by mimicking legitimate traffic and shaping existing traffic to Normal, Uniform, Poisson and Pareto distributions. I hope my early findings can call for more attention and future research to better understand timing covert channel detection resistance.

5. Conclusion Remarks

This chapter highlights the significance of the study and research work. It summarizes the research contributions and lists the lessons learned. This chapter also includes a short discussion of possible future research.

5.1 The Need for Studying Evasion Attacks and Detection Resistance Measurement

To detect attacks from the internet, anomaly detection methods were proposed to compare abnormal behavior from malicious activities with legitimate behavior. While detection techniques are developed, evasive techniques have not been well discussed. This dissertation explores the limits of current anomaly detection by discussing the battle between detectors and evaders by finding potential evasive attacks and measuring detection resistance of evasive techniques.

5.2 Summary of Contributions

This dissertation makes the following contributions:

- Characterizes web bot detection observation proofs that use web application events, with the focus on the different events and their statistics that are associated with the activities of bots and human clients, such as timing and location.
- Presents a generative approach to build an evasive web bot based on human behaviors. The proposed web bot can mimic human behaviors to hide its activities within legitimate user events; in other words, de-classifying bot-like traffic.

- Provides a practical framework for evasive web bot design and implementation, which helps researchers and engineers explore the limitations of their bot detectors.
- Implements a prototype of the proposed evasive bot framework, and validates the power of evasive bots by measuring the similarity on various metrics from selected web applications.
- Defines a novel detection resistance metric that functions as a fundamental parameter for the design of evasive timing covert channel schemes.
- Analyzes covert channel evasion with the detection resistance metric. Identifies and builds the relationship between evadability and channel noise.
- Studies evasive techniques of *Mimic*, *Replay*, *Mix*, and *Coding Scheme Rotation*.
- Studies detection techniques of *shape test* and *regularity test*.
- Presents experimental evidence that validates evadability.

5.3 Lessons Learned about the Research

This dissertation presents new attacks and answers several questions about behavior, but the performance of this research has also provided the author with several non-trivial revelations and takeaways that we hope will be useful for future

Challenges in finding human data. One of the chief problems we faced in our research was the lack of good-quality labels on data that can be used as ground truth in developing evasion algorithms. It is extremely hard to obtain a large amount of human data. The human data collected for web application is from people from both coasts, public libraries and schools. The data needs to be collected in a cheaper and more efficient way.

Challenges in developing the bots framework. Related to the observation that ground truths are not easy to obtain, I also found that implementing the evasive bots

framework was very time consuming. The selenium API used for developing bots framework is helpful, but it highly relies on web UI implementation. For example, the Gmail UI changed three times in a couple of months. It forced me to change the bots several times according to UI changes. Therefore, selecting an effective API to develop bots is essential. From my observation, if the detector could make small changes to UI, it could possibly increase the complexity of building bots in large scale. Assume the bots defender changes the UI every day, then the bots scripts would and to be updated according to the changes every day.

Challenges in developing detection measurement metric It is difficult to find good metrics to measure evadability (stealthiness) of evasive techniques for all timing covert channels. The properties of covert channels include many things such as capacity, reliability and stealthiness. Besides stealthiness, the capacity and reliability of a timing covert channel are also part of the design criteria. There is always a tradeoff among these three properties. Increasing the stealthiness would decrease capacity or increase transmission error.

5.4 Future Work

Although I have developed an evasive bot and provided a systematic approach to exploit the limitations of HOP-based detectors, my ultimate goal is to improve the security of online as business by developing more advanced bot detectors. To defend against evasive bots, future detectors could take the application context and user intent into account for bot detection. In the context of online applications, human users can easily perceive the application feedback and interact to the applications in a logical and sensible manner. But for bots, it will be challenging and computationally expensive to continually analyze the application context and then adjust their behaviors with the changing context in a timely manner. Moreover, the intent of a human user is very different from that of a bot. For example, in a web browsing scenario, a human user clicks on several links to read multiple web pages, while bots may click on the same link several times to abuse pay per click. The

evasive bot can inject mouse movement actions and idle time between clicks to mimic a human, but it cannot fully hide the final results and its intent. Finally, the user behavior consistency degree I defined before could also be used to detect evasive bots, because evasive bots usually have a higher consistency degree than human users.

Bibliography

[1] J. F. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, 6th ed. Addison-Wesley Longman Publishing Co., Inc., 2012.

[2] J. Scambray, S. McClure, and G. Kurtz, *Hacking Exposed*, 6th ed. McGraw-Hill Professional, 2009.

[3] S. Garfinkel, G. Spafford, and A. Schwartz, *Practical Unix & Internet Security, 3rd Edition*. O'Reilly Media, Inc., 2003.

[4] E. Skoudis and L. Zeltser, *Malware: Fighting Malicious Code*. Prentice Hall PTR, 2003.

[5] D. Pollino, T. B. Bill Pennington, and H. Dwiyedi, *Hacker's Challenge*. McGraw Hill Professional, 2006.

[6] M. A. Bishop, *Computer Security: Art and Science*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[7] V. K. Varun Chandola, Arindam Banerjee, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[8] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*. ACM Press, 2008, pp. 199–212.

[9] "VMware ESXi and ESX info center," http://www.vmware.com/products/vsphere/esxi-and-esx/index.html.

[10] N. Boggs, S. Hiremagalore, A. Stavrou, and S. J. Stolfo, "Cross-domain collaborative anomaly detection: so far yet so close," in *Proceedings of the 14th international conference on Recent Advances in Intrusion Detection*, ser. RAID'11, 2011, pp. 142–160.

[11] C. Smutz and A. Stavrou, "Malicious pdf detection using metadata and structural features," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. ACM, 2012, pp. 239–248.

[12] E. Stinson and J. C. Mitchell, "Characterizing bots' remote control behavior," in *DIMVA'07*, 2007, pp. 89–108.

[13] K. Borders, X. Zhao, and A. Prakash, "Siren: Catching evasive malware (short paper)," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, ser. SP '06, 2006, pp. 78–85.

[14] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," in *Insider Attack and Cyber Security: Beyond the Hacker*. Springer, 2008, pp. 1–19.

[15] T. F. Lunt, "Detecting intruders in computer systems," in *Proceedings of the Conference on Auditing and Computer Technology*, 1993.

[16] J. Allen, W. F. Alan Christie, J. P. John McHugh, and E. Stoner, "State of the practice of intrusion detection technologies," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-99-TR-028, 1999.

[17] J. Frank, "Artificial intelligence and intrusion detection: Current and future directions," in *Proceedings of the 17' National Computer Security Conference*, August 1994, pp. 1–12.

[18] S. Kumar, "Classification and detection of computer intrusions," Purdue University, Tech. Rep., 1995.

[19] A. Sundaram, "Anomaly detection: A survey," *Crossroads - Special issue on computer security*, vol. 2, pp. 3–7, 1996.

[20] H. Debar and A. W. Marc Dacier, "A revised taxonomy for intrusion-detection systems," *Annals of Telecommunications*, vol. 55, pp. 361–378, 2000.

[21] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proceedings of ACM International Conference on Measurement and Modeling of Computer Science (SIGMETRICS)*, 1998, pp. 151–160.

[22] M. Hollander and D. A. Wolfe, *Nonparametric statistical methods*, 2nd ed. Wiley-Interscience, 1999.

[23] rt.com, "Startup dumps facebook, alleging 80% of clicks came from bots," http://rt.com/news/facebook-social-media-startup-scam-ads-554/.

[24] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time URL spam filtering service," in *Proceedings of the 32nd IEEE Symposium on Security and Privacy*, 2011, pp. 447–462.

[25] H. Gao, Y. Chen, K. Lee, D. Palsetia, and A. Choudhary, "Towards online spam filtering in social networks," in *Proceedings of 19th Network and Distributed System Security Symposium (NDSS)*, 2012.

[26] G. Jacob, E. Kirda, C. Kruegel, and G. Vigna, "Pubcrawl: Protecting users and businesses from crawlers," in *Proceedings of the 21st USENIX Symposium on Security*, 2012.

[27] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," in *International Conference on the Theory and Applications of Cryptographic Techniques*, 2003, pp. 294–311.

[28] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "recaptcha: Human-based character recognition via web security measures," *Science*, vol. 321, no. 12, pp. 1465–1468, 2008.

[29] M. Motoyama, K. Levchenko, C. Kanich, D. Mccoy, G. M. Voelker, and S. Savage, "Re: Captchas – understanding captcha-solving services in an economic context," in *Proceedings of the 19th USENIX Conference on Security*, 2010.

[30] S. Gianvecchio, M. Xie, Z. Wu, and H. Wang, "Measurement and classification of humans and bots in internet chat," in *Proceedings of the 17th USENIX Symposium on Security*, 2008.

[31] S. Gianvecchio, Z. Wu, M. Xie, and H. Wang, "Battle of botcraft: Fighting bots in online games with human observational proofs," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, 2009.

[32] Z. Chu, S. Gianvecchio, and H. Wang, "Who is tweeting on twitter: Human, bot, or cyborg?" in *Proceedings of the 26th Annual Computer Security Applications Conference*, 2010.

[33] Z. Chu, S. Gianvecchio, A. Koehl, H. Wang, and S. Jajodia, "Blog or block: Detecting blog bots through behavioral biometrics," *Computer Networks*, vol. 57, no. 3, pp. 634–646, February 2013.

[34] D. Doran and S. S. Gokhale, "Web robot detection techniques: overview and limitations," in *Data Mining and Knowledge Discovery, Volume 22 Issue 1-2*, 2011.

[35] K. Park, V. S. Pai, K.-W. Lee, and S. Calo, "Securing web services by automatic robot detection," in *Proceedings of the Annual conference on USENIX Annual Technical Conference*, 2006.

[36] F. Yu, Y. Xie, and Q. Ke, "Sbotminer: Large scale search bot detection," in *Proceedings of ACM International Conference on Web Search and Data Mining*, 2010.

[37] A. Ramach and N. Feamster, "Understanding the network level behavior of spammers," in *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2006.

[38] E. Stinson and J. C. Mitchell, "Towards systematic evaluation of the evadability of bot or botnet detection methods," in *Proceedings of the 2nd conference on USENIX Workshop on offensive technologies (WOOT'08)*, 2008.

[39] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, "Detecting botnets with tight command and control," in *Proceedings of 31st IEEE conference on Local Commuter Networks*, 2006.

[40] J. Goebel and T. Holz, "Rishi: identify bot contaminated hosts by irc nickname evaluation," in *HotBots: Proceedings of 1st Workshop on Hot Topics in Understanding Botnets*, 2007.

[41] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, ser. HotBots'07. Berkeley, CA, USA: USENIX Association, 2007.

[42] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: detecting malware infection through IDS-driven dialog correlation," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, ser. SS'07. USENIX Association, 2007.

[43] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection," in *Proceedings of the 17th conference on Security symposium*, ser. SS'08. USENIX Association, 2008, pp. 139–154.

[44] W. Cui, R. H. Katz, and W. Tan, "Design and implementation of an extrusion-based break-in detector for personal computers," in *Proceedings of 21st Annual Computer Security Applications Conference*, 2005.

[45] E. Stinson and J. C. Mitchell, "Characterizing bots' remote control behavior," in *Lecture Notes in Computer Science*. Springer, 2007.

[46] T. Jaeger, R. Sailer, and Y. Sreenivasan, "Managing the risk of covert information flows in virtual machine systems," in *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM Press, 2007, pp. 81–90.

[47] M. H. Kang, I. S. Moskowitz, and S. Chincheck, "The pump: A decade of covert fun," in *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC)*, 2005, pp. 5–9.

[48] Z. Wang and R. B. Lee, "Covert and side channels due to processor architecture," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, FL, 2006, pp. 473–482.

[49] H. Wang, S. Jha, and V. Ganapathy, "NetSpy: Automatic generation of spyware signatures for NIDS," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, FL, 2006, pp. 99–108.

[50] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts, "Covert messaging through TCP timestamps," in *Proceedings of the Workshop on Privacy Enhancing Technologies*, 2002, pp. 194–208.

[51] D. X. Song and X. T. David Wagner, "Timing analysis of keystrokes and timing attacks on ssh," in *Proceedings of the 10th conference on USENIX Security Symposium*, vol. 10, 2001.

[52] M. H. Kang and I. S. Moskowitz, "A pump for rapid, reliable, secure communication," in *ACM Conference on Computer and Communications Security*. ACM, 1993, pp. 119–129.

[53] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *Proceedings of the 2007 IEEE Symposium on Security & Privacy (S&P 2007)*, Oakland, CA, May 2007, pp. 116–130.

[54] A. Houmansadr, N. Kiyavash, and N. Borisov, "Rainbow: A robust and invisible non-blind watermark for network flows," in *Proceedings of the The 16th Annual Network and Distributed System Security Conference (NDSS'09)*, San Diego, CA, February 2009.

[55] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "DSSS-Based flow marking technique for invisible traceback," in *Proceedings of 2007 IEEE Symposium on Security and Privacy (S&P 2007)*, Oakland, California, USA, May 2007.

[56] J. Jung, A. Sheth, B. Greenstein, D. Wetherall, G. Maganis, and T. Kohno, "Privacy Oracle: A system for finding application leaks with black box differential testing," in *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*. ACM Press, New York, NY, USA, 2008, pp. 279–288.

[57] G. Shah, A. Molina, and M. Blaze, "Keyboards and covert channels," in *Proceedings of the 2006 USENIX Security Symposium*, 2006, pp. 59–75.

[58] J. Jin and X. Wang, "On the effectiveness of low latency anonymous network in the presence of timing attack," in *The 41st Annual IEEE/IFIP International Conference on Dependable Systems and Network*, Lisbon, 2009, pp. 429–438.

[59] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th Usenix Security Symposium*, 2004.

[60] "The anonymizer," http://anonymizer.com.

[61] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: Design and detection," in *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*, Washington, DC, USA, October 2004.

[62] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia, "Model-based covert timing channels: Automated modeling and evasion," in *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*. Springer-Verlag Berlin, Heidelberg, 2008.

[63] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–88, 1981.

[64] E. Felton and M. Schneider, "Timing attacks on web privacy," in *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS 2000)*, Athens, Greece, November 2000.

[65] Q. Sun, D. R. Simon, Y. M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P 2002)*, Berkeley, California, USA, May 2002.

[66] S. Chakravarty, A. Stavrou, and A. D. Keromytis, "Traffic analysis against low-latency anonymity networks using available bandwidth estimation," in *Proceedings of the 15th European conference on Research in computer security*, ser. ESORICS'10, 2010, pp. 249–267.

[67] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Timing attacks in low-latency mix-based systems," in *Proceedings of Financial Cryptography (FC '04)*, February 2004, pp. 251–265.

[68] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P 05)*, May 2005, pp. 183–195.

[69] M. Gogolewski, M. Klonowski, and M. Kutylowski, "Local view attack on anonymous communication," in *Proceedings of the 10th European Symposium On Research In Computer Security (ESORICS 2005)*, September 2005.

[70] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency mix networks: Attacks and defenses," in *Proceedings of the 11th European Symposium On Research In Computer Security (ESORICS 2006)*, Hamburg, Germany, September 2006.

[71] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *Proceedings of the 26th Annual IEEE Conference on Computer Communications (Infocom 2007)*. Anchorage, Alaska, USA: IEEE, May 2007.

[72] R. Kemmerer, "Shared resource matrix methodology: an approach to identifying storage and timing channels," in *Journal of ACM Transactions on Computer Systems*, vol. 1, 1983, pp. 256–277.

[73] Y. Liu, C. L. Corbett, K. Chiang, R. Archibald, B. Mukherjee, and D. Ghosal, "SIDD: A framework for detecting sensitive data exfiltration by an insider attack," in *Proceedings of the 42nd Hawaii International Conference on System Sciences (HICSS)*, 2009, pp. 1–10.

[74] V. Berk, A. Giani, and G. Cybenko, "Detection of covert channel encoding in network packet delays," in *Tech. Rep. TR2005-536*, Dartmouth College, Computer Science, Hanover, NH, August 2005.

[75] S. Gianvecchio and H. Wang, "Detecting covert timing channels: An entropy-based approach," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07)*, Washington DC, October 2007.

[76] F. J. Massey, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American Statistical Association*, 1951.

[77] C. E. Shannon, "A mathematical theory of communication," *The Bell Systems Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[78] N. Martin and J. England, *Mathematical Theory of Entropy*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011.

[79] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proceedings of ACM Conference on Computer and Communications Security (CCS)*, 2007, pp. 286–296.

[80] S. Kullback, *Information theory and statistics.* John Wiley and Sons, NY, 1951, pp. 22 (1): 79–86.

[81] R. E. Blahut, *IEEE Transactions on Information Theory.* IEEE Information Theory Society, July 1972, vol. 18, pp. 460–473.

[82] B. Rubinstein, B. Nelson, L. Huang, and A. Joseph, "ANTIDOTE: Understanding and defending against poisoning of anomaly detectors," in *Proceedings of Internet Measurement Conference*, 2009.

[83] N. Daswani and M. Stoppelman, "The anatomy of clickbot.A," in *Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets*, 2007.

[84] S. Kullback, "The Kullback-Leibler distance," in *The America Statistician*, 1987, pp. 41:340–341.

[85] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma, "Adversarial classification," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '04. ACM, 2004, pp. 99–108.

[86] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[87] P. M. Fitts and J. R. Peterson, "Information capacity of discrete motor responses," *Journal of Experimental Psychology*, vol. 67, no. 2, pp. 103–112, 1964.

[88] I. S. MacKenzie, "Fitts' law as a research and design tool in human-computer interaction," *Human-Computer Interaction*, vol. 7, pp. 91–139, 1992.

[89] U. Kukreja, W. E. Stevenson, and F. E. Ritter, "Rui: Recording user input from interfaces under Windows and Mac OS," *Behavior Research Methods*, vol. 38, no. 4, pp. 656–659, 2006.

[90] J. Offutt and Y. Wu, "Modeling presentation layers of web applications for testing," *Software and Systems Modeling*, vol. 9, no. 2, pp. 257–280, April 2010.

[91] P. Ammann and J. Offutt, *Introduction to Software Testing.* Cambridge, UK: Cambridge University Press, 2008, ISBN 0-52188-038-1.

[92] L. Ballard, F. Monrose, and D. Lopresti, "Biometric authentication revisited: Understanding the impact of wolves in sheep's clothing," in *Proceedings of the 15th USENIX Symposium on Security*, 2006.

[93] M. Lee, R. R. Kompella, and S. Singh, "Ajaxtracker: Active measurement system for high-fidelity characterization of Ajax applications," in *Proceedings of the 2010 USENIX conference on Web application development (WebApps'10)*, 2010.

[94] P. Peng, P. Ning, and D. S. Reeves, "On the secrecy of timing-based active watermarking trace-back techniques," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P 2006)*, Oakland, California, USA, May 2006, pp. 334–349.

[95] S. Lin and D. Costello, *Error Control Coding, Second Edition.* Prentice-Hall Inc, 2004.

[96] A. Houmansadr and N. Borisov, "Coco: Coding-based covert timing channels for network flows," in *Proceedings of the International Conference of Information Hiding*, May 2011.

[97] Y. Liu, D. Ghosal, F. Armknecht, A. Sadeghi, S. Schulz, and S. Katzenbeisser, "Hide and seek in time: Robust covert timing channels," in *Proceedings of the European Symposium on Research in Computer Security*, ser. Lecture Notes in Computer Science. Springer, 2009.

[98] M. W. Robert Wallsa, Kush Kotharib, "Liquid: A detection resistant covert ting channel based on IPD shaping," *Computer Networks*, vol. 34 (1-2), pp. 1217–1228, 2011.

[99] M. A. Padlipsky, D. W. Snow, and P. A. Karger, "Limitations of end-to-end encryption in secure computer networks," in *Computer Networks. Tech. Rep. ESD TR-78-158.* Mitre Corporation, 1978.

[100] Y. Liu, D. Ghosal, F. Armknecht, A.-R. Sadeghi, S. Schulz, and S. Katzenbeisser, "Robust and undetectable steganographic timing channels for i.i.d. traffic," in *Information Hiding - 12th International Conference, IH 2010, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 6387. Springer, 2010, pp. 193–207.

[101] R. W. Smith and S. G. Knight, "Predictable three-parameter design of network covert communication systems," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 1, pp. 1–13, March 2011.

[102] R. W. Smith and G. S. Knight, "Predictable design of network-based covert communication systems," in *Proceedings of the 29th IEEE Symposium on Research in Security and Privacy*, Oakland, CA, 2008, pp. 311–321.

Biography

Jing Jin received her Bachelor Degree of Electrical Engineering and Master of Science from Huazhong University of Science and Technology in 2003 and 2006 respectively. She has been employed as a Security Engineer at VMware since 2012. Before she joined VMware, she was also an intern at Cigital, Symantac and Adobe.