Denial of Service in Web Domains: Building Defenses against Next-Generation Attack Behavior

DUSAN STEVANOVIC

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, CANADA

MAY 2016

Abstract

The existing state-of-the-art in the field of application layer Distributed Denial of Service (DDoS) protection is generally designed, and thus effective, only for static web domains. To the best of our knowledge, our work is the first that studies the problem of application layer DDoS defense in web domains of dynamic content and organization, and for next-generation bot behaviour. In the first part of this thesis, we focus on the following research tasks: 1) we identify the main weaknesses of the existing application-layer anti-DDoS solutions as proposed in research literature and in the industry, 2) we obtain a comprehensive picture of the current-day as well as the next-generation application-layer attack behaviour and 3) we propose novel techniques, based on a multidisciplinary approach that combines offline machine learning algorithms and statistical analysis, for detection of suspicious web visitors in static web domains. Then, in the second part of the thesis, we propose and evaluate a novel anti-DDoS system that detects a broad range of application-layer DDoS attacks, both in static and dynamic web domains, through the use of advanced techniques of data mining. The key advantage of our system relative to other systems that resort to the use of challenge-response tests (such as CAPTCHAs) in combating malicious bots is that our system minimizes the number of these tests that are presented to valid human visitors while succeeding in preventing most malicious attackers from accessing the web site. The results of the experimental evaluation of the proposed system demonstrate effective detection of current and future variants of application layer DDoS attacks.

Acknowledgements

First and foremost, I would like to thank Professor Natalija Vlajic, my research supervisor, for her endless guidance, encouragement and patience over the past five and half years. Professor Vlajic was the main reason for the successful completion of my thesis. She has invested immeasurable hours in helping me co-author 12 research papers, a book chapter and 4 journal articles. It has been a great privilege and an honour to work with her.

I am grateful to Professor Matthew Kyan, Professor Suprakash Datta, Ashraf Matrawy, Regina Lee, and Professor Uyen Trang Nguyen for agreeing to serve on my examination committee and for providing very useful feedback on my thesis. I especially would like to thank Professor Suprakash Datta, Professor Bil Tzerpos and Professor Uyen Trang Nguyen for serving on my supervisory committee and for their valuable input during my studies.

I thank the exceptional members of the Signal Processing and Communications Laboratory, who have made my experience in graduate school at York University most enjoyable.

Lastly, I would like to express my deepest gratitude to my dearest mother Rada, father Mirce, brother Nikola, grandma Mira and uncle Rade for their continuous love and encouragement. I thank my wife Natalie, for her everlasting love, understanding and support.

This work is dedicated to my son Milan.

Table of Contents

List of Acronyms

Modified ART2	Modified Adaptive Resonance Theory 2 algorithm
AS	Autonomous System
BGP	Border Gateway Protocol
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
C&C	Command-and-Control
CBS	Chronological Browsing Sequence
CDF	Cumulative Distribution Function
CSE	Computer Science and Engineering Department
DDoS	Distributed Denial of Service
DNS	Domain Name Server
DoS	Denial of Service
EDoS	Economic Denial of Sustainability
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IRC	Internet Relay Chat
KNN	k-nearest Neighbor
LCS	Longest Common Subsequence
NLLCS	Normalized Length of the Longest Common Subsequence
NN	Neural Networks
PDF	Probability Density Function
PMF	Probability Mass Function
PVT	Page Viewing Time
SOM	Self-Organizing Maps
TCP	Transmission Control Protocol
TFN	Tribe Flood Network
TFN2K	Tribe Flood Network 2000
UDP	User Datagram Protocol
VoIP	Voice-over-IP

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Overview of the Problem and Motivation

In an era marked by hyper production and consumption of information, there are many web sites that experience very rapid and dynamic change in their structure – both in terms of the number of web pages and hyper-link connections among them and the contextual information that each individual page provides. Online newspapers, online trading sites and social media sites, are examples of web domains where the creation of new pages and changes to their content occur on the time scale of minutes, if not seconds.

The browsing behaviour[1] of users visiting a web site is very much influenced by the site's organization and content. Consequently, every time the site's organization and/or content changes (one example of this would be the creation of a 'breaking-news' page on a news-agency web site), the users' browsing behaviour on that site is also likely to change. Figure 1.1 illustrates the structure of a typical news agency web site, at two different instances of time - before and after a number of pages have been added/deleted. Figure 1.1 also demonstrates that the typical web session trail of regular human visitors can change from times t_1 to t_2 as the web site's content evolves (i.e., new breaking news is added to the site). Note that

[1] *Browsing behaviour* is characterized by the number and sequence of pages visited, viewing time spent on each visited page, etc..

several studies have shown that even in web sites of static structure and content, the browsing behaviour of its users may change with each subsequent re-visitation of the site ([1], [2]).

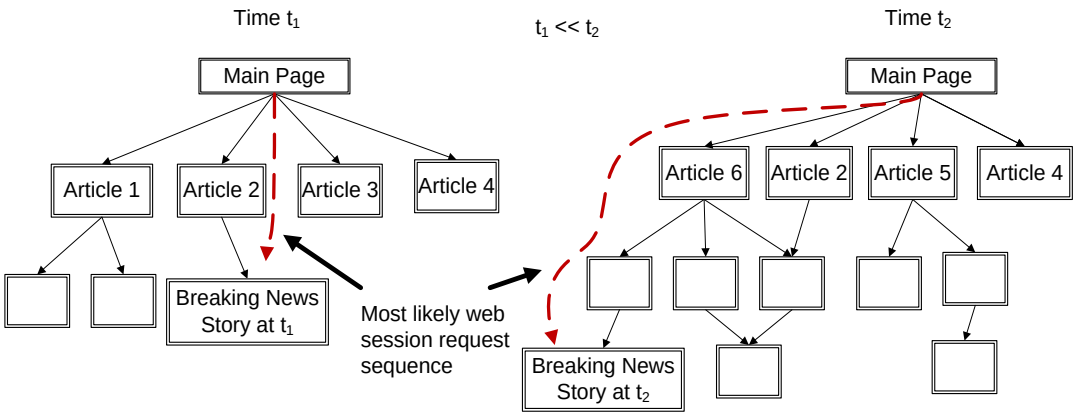


Figure 1.1 – A sample news agency web site with evolving content

Successful operation of any given web site − static or dynamic − is closely tied to its ability to provide instantaneous response to its users. Security attacks capable of severely degrading the response-rate and quality at which web-based services are offered are known as Denial of Service (DoS) attacks. Distributed Denial of Service (DDoS) is an amplified, and thus increasingly more potent, version of DoS that involves a whole distributed network of attacking machines – also known as a Botnet. Nowadays, the means to carry out sophisticated DDoS attacks that can result in immense financial losses are within easy reach of anyone with a PC and an Internet connection [3]. Due to their great affordability combined with a significant damage potential, DDoS attacks have emerged as the weapon of choice for both financially-motivated cyber criminals as well as politically motivated hacktivists.

From the technical standpoint, a DDoS attack can be conducted at the network, transport or application layer involving a range of different protocols. The leading DDoS solution providers commonly agree that

the use of application-layer DDoS attacks is growing at a much higher rate relative to network and transport layer DDoS attacks (see [4]). The main reason behind the growing popularity of application-layer DDoS is the fact that the conventional anti-DDoS tools have much looser control over application layer traffic [5]. That is, application-layer DDoS attacks have to deal with fewer layers of protection/security, and thus are more likely to be successful.

Current state-of-the-art in the field of application-layer DDoS protection, refer to Tables 3.3 and 3.4 in chapter 3, is generally designed (and thus effective) only for static web domains. The main goal of our work is to identify the main weaknesses of the existing application-layer anti-DDoS solutions, and to obtain a comprehensive picture of the current-day as well as the future application-layer attack behaviour. With this knowledge in mind, we propose a novel anti-DDoS system which, through the use of advanced techniques of data-mining, has a potential to defend against a broad range of application-layer DDoS attacks, both in static and dynamic web domains.

1.2 Thesis Contributions

Our body of work has two main contributions:

1. We examine the current state-of-the-art in automated benign and malicious web crawler's browsing behaviour. Specifically, we apply machine learning techniques in combination with traffic pattern analysis and syntactic log parsing, on several different sets of real-world web logs, to identify similarities/differences in the browsing behaviour of various web visitors. Through this novel experimentation, we identify malicious bots that exhibit human-like browsing behaviour as well as human visitors that demonstrate malicious bot browsing behaviour and could potentially be malicious themselves. Next, we identify several non-contextual web session browsing features that can be used to separate these two groups of visitors. Lastly, we demonstrate that two non-parametric statistical

tests can be employed to separate suspicious (i.e. malicious) and benign visitors based on the difference in the distribution of contextual web session features such as web page popularity rankings and web page viewing times. This work is outlined in chapters 4 and 5 and has been published in [6], [7], [8], [9], [10]and [11].

2. Building on the results of 1, we propose a novel next-generation anti-DDoS system that ensures effective detection of current and future (i.e., next-generation) application layer DDoS attacks in both static and dynamic web domains. According to our knowledge, this system is the first to tackle the problem of detection of application layer DDoS attacks in dynamic web domains. Also, this system is the first to combine the deployment of automated unsupervised outlier detection algorithms with the deployment of manual challenge-response tests in order to separate legitimate web visitors from malicious attackers. The key advantage of our system relative to other systems that resort to the use of challenge-response tests (such as CAPTCHAs) in combating malicious bots, is that our system ensures that an absolute minimum of these tests is presented to valid human visitors while completely succeeding in preventing malicious attackers from accessing the web site. This work is outlined in chapter 6, and has been published in [6], [12] and [13].

Additionally, we focus on the following research problems in the thesis:

- We present the limitations of current research and commercial anti-DDoS solutions. This work is outlined in chapter 3.
- We evaluate the application of both the supervised and the unsupervised machine learning techniques for classification of various web site visitors. We demonstrate that *supervised machine learning* can be employed to 1) classify user sessions as belonging to either automated benign web crawlers or human visitors, and 2) identify which browsing features are the most useful in identifying various visitors groups. Note that this methodology can be applied in sites where there exists a ‘natural separation’ of benign vs. malicious behaviour that is possibly known to the system administrator.

Secondly, we also show that *unsupervised machine learning* can be utilized to obtain a better insight into the types and distribution of visitors to a public web site based on their link-traversal behaviour, as well as to identify suspicious outlier malicious/unknown visitors that exhibit human browsing behaviour and suspicious outlier human visitors that mimic malicious browsing behaviour on a web site. Note that the unsupervised learning should be used in domains/sites where the differences between benign and malicious visitors are far more subtle, and thus difficult to identify or capture. This work is outlined in chapters 4 and 5.

- We identify and rank the quality of the features that are typically employed in research literature for the task of detecting web crawlers by employing supervised machine learning. We also introduce two novel classification features that are highly ranked by the information gain and gain ratio metrics in terms of their purity or value in the classification task. This work is outlined in chapter 4.
- We propose two novel sophisticated HTTP-based DDoS attacks that were employed to evaluate the attack detection performance of our next-generation anti-DDoS system. These attacks are discussed in chapter 6.

1.3 Outline of the Thesis

To provide the relevant context, the remainder of the thesis is organized as follows:

- In chapter 2, we introduce the reader to DDoS motivation, techniques and existing defenses as proposed in research literature.
- In chapter 3, we focus our discussion on the application-layer DDoS attacks and outline the limitations of previous approaches, both in theory and in industry, on detection and mitigation of such attacks.

- In chapter 4, we review the state-of-the-art in automated web-crawler browsing behaviour and utilize several supervised machine learning and statistical techniques for detecting both benign and malicious web crawlers in static web domains.
- In chapter 5, we study suspicious web visitors, identified in chapter 4, by employing unsupervised machine learning and two statistical techniques.
- In chapter 6, we present our next-generation anti-DDoS system that is intended for dynamic web domains facing sophisticated variants of application-layer DDoS attacks. We also present the experimental results concerning the detection of synthesized DDoS traffic using our proposed system.
- Lastly, in chapter 7, we conclude the thesis with a summary of our main contributions and findings, and we discuss some possible directions for future work.

Chapter 2 Denial of Service - Motivation, Methodology and Defenses

DoS attacks have become an increasingly prevalent form of security threat in today's Internet [4]. The first section of this chapter (section 2.1) gives a general overview of DoS motivations, impacts and execution mechanisms. section 2.2 looks into the DDoS attacks, as the most common as well as the most challenging form of DoS attacks. This section also discusses different DDoS variants. In section 2.3, the DDoS malware, as utilized by attackers to launch and manage actual attacks, is described. Lastly, in section 2.4, we discuss and evaluate briefly some traditional mechanism for DDoS defences.

2.1 Denial of Service: Motivations, Impacts, Execution Mechanisms

DoS attack falls into the group of so-called 'availability attacks', i.e. attacks whose main goal is to prevent legitimate users from accessing information or services. Unlike attacks on data confidentiality and integrity, the DoS attacks do not aim to steal, corrupt or modify data directly, nor do they attempt to gain unauthorized access into a victim's machine. DoS attacks are accomplished by either crashing the victim system, or simply exhausting the system's processing and/or bandwidth resources.

2.1.1 DoS Motivations

The most common motivations for DoS attacks are: personal, economic, and political.

1) DoS Motivated by Personal Goals: The early DoS attacks were merely proofs of concept or simple pranks played by hackers. The ultimate goal was to prove that something could be done, such as taking a large, popular web site or an online chat forum offline. For instance, most of the attacks on IRC (Internet Relay Chat)[2] networks in the late 1990s were done by sophisticated hackers participating on the IRC chat forums [14].

2) DoS Motivated by Economic Incentives: In recent years, DoS attacks have increasingly become part of extortion attempts, as mentioned in [14], [15], [16], [17], [18] and [19]. In these criminal schemes, an online business is threatened with a DoS attack that would slow down the business to a crawl, and a payment is requested as means of avoiding the attack and its consequences.

3) DoS Motivated by Political Agendas: Political agendas are another common motive for DoS attacks. As described in [15], in 2007, a series of large and sustained DoS attacks were launched against several Estonian government web sites immediately after Estonian government decided to relocate a Soviet-era war memorial commemorating an unknown Russian who died fighting in the World War II. The clashes between Russia and Georgia over the region of South Ossetia in July 2008 have resulted in DoS attacks as well [20]. Another politically motivated attack from September of 2012 was believed to be orchestrated or sponsored by the Iranian government [21]. In this event, a stream of DDoS attacks were launched against several large US banks in protest of YouTube's refusal to take down a trailer of the controversial anti-Islam movie.

[2] IRC is a form of real-time Internet text messaging protocol running on top of TCP protocol.

2.1.2 DoS Impacts

Use of Internet services – both commercial and private – has become an integral part of our lives. Examples of daily Internet activities include: conducting business transactions, storing and accessing patient records in hospitals, taking online courses, socializing with friends, etc. An attack to the availability of Internet services can incur serious damages to both - the sites (i.e., companies) under the attack as well as their end-users. In some instances, as will be discussed here, it can even cost human lives. Particular examples of damages incurred by DoS attacks are summarized in Table 2.1.

Table 2.1 – DDoS Impact on Various Online Services

	Impact
E-commerce	E-commerce companies that offer services to users through online stores can only accumulate revenue if the users can access their web sites. A DoS attack on such a web site can reduce the company's revenue not only for the duration of the attack, but even well after the attack has been successfully thwarted. Namely, studies like [22] and [23] have shown that unavailability of a web site tends to drive users away and limit future investor funding.
News and Search Engines	Large news sites and search engines are paid by marketers to present their advertisement to the public, and the revenue is typically accumulated on 'per-view' or 'per-click' basis. Hence, whenever such a web site undergoes a prolonged DoS attack, the financial gains of all companies involved can be significantly reduced
web Conferencing	web-based video conferencing and web-based file sharing are commonly utilized by multi-branch companies, to enable effective communication and collaboration between remote offices. Clearly, a DDoS attack on the network infrastructure of these companies could have a detrimental impact on their operation.
Internet Backbone	A DoS attack on a DNS server – especially the upper-tier ones – can be very damaging to a broad range of companies and users. Namely, by flooding the domain name server of a zone, the DNS records of all organizations in the given zone may (will) become unavailable, thus preventing the future users from properly accessing the multitude of impacted sites
Public Infrastructure	The Internet is gaining an increasingly important role in the management of public services such as hospitals, police, water, power and sewage systems, etc. Unquestionably, a DoS attack on any of these critical services, aimed at interrupting their normal operation, could ultimately endanger human lives. Fortunately, there have not been documented real-world cases of where an actual human life was lost due to a DoS attack. However, in the future, as these services become more increasingly connected to the Internet, this scenario could be possible

2.1.3 DoS Execution Mechanisms

The ultimate goal of a DoS attack is to prevent legitimate users of an online service from using that service. This goal can be achieved in a number of ways, some of them involving the following:

1. Consumption of computational resources, such as bandwidth, disk space, memory pool or processor time, on the victim system.
2. Disruption of state information, such as unsolicited resetting of TCP or SIP sessions, on the victim system (i.e., end-host).
3. Disruption of configuration information, such as routing information, on the infrastructure components leading towards the victim system.

Of the above, group (1) are the most common type of DoS attacks. Though, in order to magnify their power, DoS attacks also frequently utilize a combination of the above enlisted mechanisms.

Now, the DoS attacks of group (1) are successful in consuming resources of the victim system only if they involve great network and computational resources. The most straightforward way of ensuring sufficient network and computational power are consumed is by employing a multitude of attack machines. Such DoS attacks - involving a large number of (often compromised third-party) attack machines - are commonly referred to as Distributed Denial of Service (DDoS) attacks.

2.2 Distributed Denial of Service

A DDoS attack is a coordinated attack on the availability of services of a given target system or network that is launched indirectly through many compromised computing systems. The services under attack are those of the "primary victim", while the compromised systems used to launch the attack are often referred to as the "secondary victim". The owners and users of the secondary victim machines typically have no knowledge that their system has been compromised and will be taking part in a DDoS attack.

While the presence of one primary and numerous secondary victims is common to all DDoS attacks, individual DDoS attack can vary in a number of features ([24], [25] and [26]), as given below and illustrated in Figure 2.1: architecture, impact, exploited vulnerability, attack rate dynamics, and packet content.

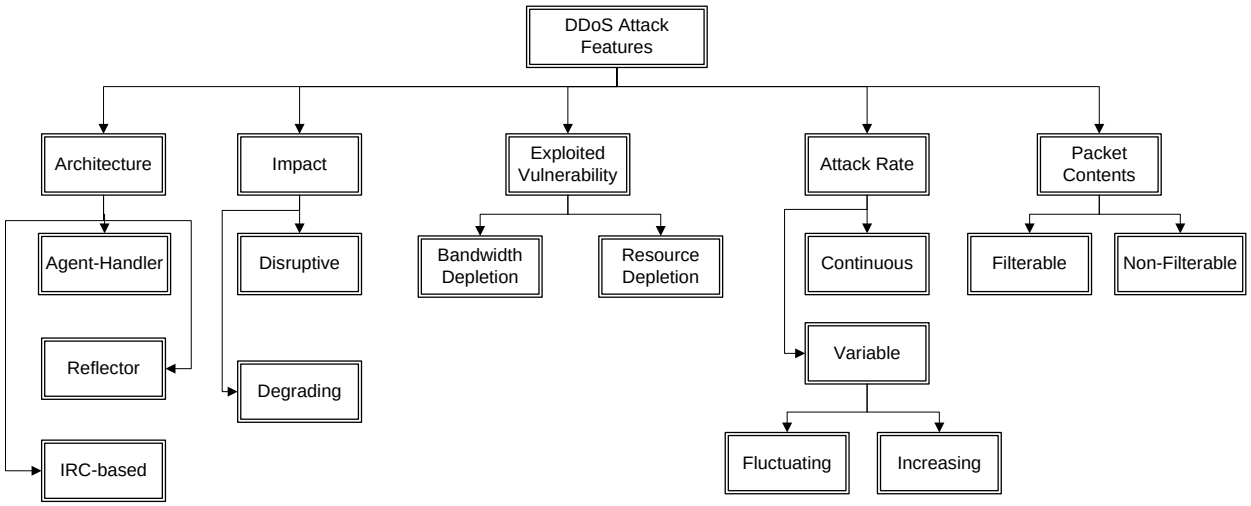


Figure 2.1 – DDoS attack features

In the remainder of this section, we will describe each of the above enlisted categories of DDoS attacks in more detail.

2.2.1 Architecture Models of DDoS Attacks

There are three architectural models of DDoS attack infrastructure: agent-handler model, reflector and/or amplifier model and IRC-based attack model.

2.2.1.1 Agent-Handler Model

This, most common model of DDoS infrastructure, consists of an attacker, handlers, agents, and the target network - commonly referred to as command and control server network (or C&C). The collection of attack machines involved in a DDoS attack is known as Botnet. Botnets are made up of vast numbers of compromised computers that have been infected with malicious code, and can be remotely-controlled through commands sent via the Internet. Figure 2.2 illustrates a sample Botnet deployment.

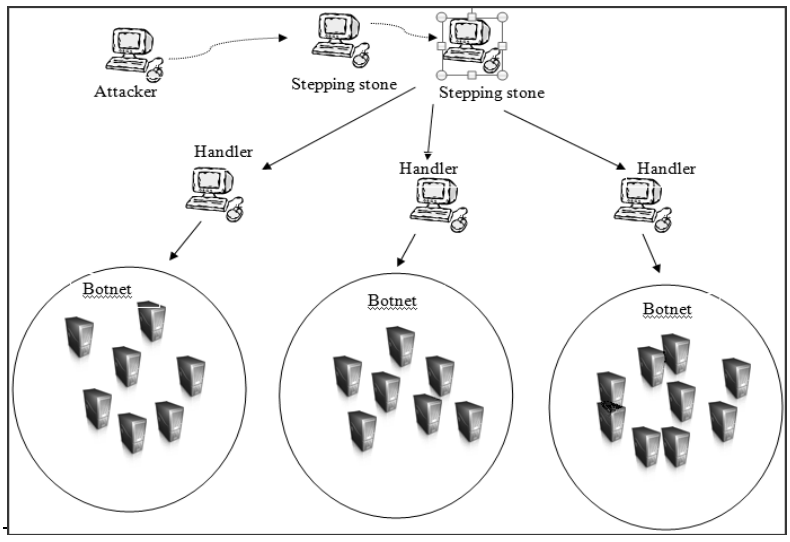


Figure 2.2 – A sample Botnet deployment

The agents are usually machines with little or no security. For instance, these machines either do not patch their operating systems and other installed software on regular basis or do not have running anti-virus packages in the background. They are usually not protected by a firewall and their users have easily guessed passwords. The software that has not been patched has well-known vulnerabilities that can be exploited by the hacker.

In order to avoid detection, an attacker attempts to hide its location by deploying several layers of indirection between itself and the agents. Specifically, the attacker utilizes machines, known as *handlers* or *masters*, to deliver (DDoS-attack) control messages to the agents. The attacker can even login in sequence through several machines, before accessing the handler. The machines accessed in sequence are known as *stepping stones*. This strategy complicates the task of tracing back the attacker from the victim's machine, especially if the stepping stones and handlers are located in different countries or continents. IP spoofing, i.e. falsification of the actual IP address, is usually employed to obscure the origin of an attack (i.e., to hide the attacker's location and identity).

2.2.1.2 Reflector & Amplifier Model

The architecture of this DDoS infrastructure model is very similar to the previous one. However, the additional parameter in this scenario, besides the attacker, agents and handlers, is the reflector machine. Namely, the difference in this type of attack is that agents are instructed by handlers to send a stream of packets with the victim's IP address as the source IP address to other uninfected machines, known as *reflectors*. (A reflector is any host that responds to requests, for example a domain name server (DNS) that responds to UDP requests or a web server that responds to TCP SYN requests with a SYN-ACK reply.) Reflectors can also be amplifiers if the attacker sends a message to a broadcast IP address which causes all systems in the subnet reached by the broadcast address to send a much larger reply back to the victim

system [27]. The benefit of this approach is that the attack traffic (and thus the DoS effect) is amplified tens or hundreds of times.

2.2.1.3 IRC-Based Attack Model

In contrast to the previous two infrastructures, which fall in the category of distributed and push-based models, the IRC-based DDoS infrastructure is centralized and pull-based. In this type of infrastructure, DDoS control messages are exchanged between the attacker and its agents by means of IRC communication channels, instead of by means of handler programs/machines.

2.2.2 Impact of DDoS Attacks

The impact of the DDoS attack can be either disruptive or degrading, as described below:

- In a *disruptive* or *high-grade* DDoS attack the entire resource, such as bandwidth, CPU, memory or connection pool, will be exhausted by the attack, thus disrupting the access to the target from its legitimate clients.
- The *degrading* or *low-and-slow* DDoS attack causes only the partial bandwidth consumption. This type of attack is sometimes difficult to detect because the traffic experienced by the target may only partially exceed the volume of traffic generated by usual/legitimate sources.

2.2.3 Vulnerabilities Exploited by DDoS Attacks

DDoS attacks can deplete either the bandwidth or the resource of the victim network/machine.

2.2.3.1 Bandwidth Depletion

In this class, attacker sends large volumes of malicious or unwanted traffic to the target network, in order to saturate the target-system's network bandwidth. Clearly, for this type of attack to succeed, the flow of

traffic towards the victim must be high enough to exceed the system's bandwidth capacity. Examples of bandwidth depletion attacks are HTTP, SSL, DNS, FTP, SMTP, SIP, UDP and ICMP flood attacks.

2.2.3.2 Resource Depletion (Protocol attacks)

The resource depletion or vulnerability attacks involve sending one or a few specifically crafted messages to the target application, that possesses a specific vulnerability, in order to shut down its service to legitimate clients [14]. The vulnerability can be either a software bug in the implementation or the exploitable logic vulnerability in the underlying protocol or application. Examples of these attacks include TCP SYN, SlowPOST [28], SlowRead [28], XML entity expansion (XEE) [29], ReDoS [30].

2.2.4 Attack Rates in DDoS Attacks

The attack rates in DDoS attacks can be either continuous or variable, as described below:

- In the *continuous* attack, the agent machines generate attack packets with full force without interruption in traffic volume. This approach is very effective in producing the DoS effect but it is easily detectable.
- The attackers that generate *variable-rate* attacks are more cautious in their engagement, i.e., they change the attack rate to avoid detection and response by the target. According to the rate change mechanism, there are two types of variable-rate DDoS attacks, *increasing* and *fluctuating*. In the increasing attack, the attacker employs a gradually increasing rate that slowly exhausts the victim's resources. In the fluctuating attack, the attacker adjusts the attack rate based on the victim's behaviour, occasionally relieving its effect to avoid detection.

2.2.5 Content of DDoS Attack Packets

The content of DDoS attack packets can be grouped into filterable and non-filterable:

- *Filterable* attacks are those that use malformed packets or packets for non-critical services of the victim's operation. These can thus be filtered by a firewall. Examples of such attacks are a UDP flood attack or an ICMP echo flood attack on a web server.
- *Non-filterable* attacks use packets that request legitimate services from the victim. Thus, filtering all packets that match the attack signature would lead to an immediate denial of the specified service to both the attacker as well as the legitimate clients. Examples are a HTTP request flood targeting a web server or a DNS request flood targeting a name server. These are also referred to as application layer DDoS attacks.

2.3 DDoS Malware

Over the last decade, attackers have developed various techniques and tools for more effective execution of DDoS, including toolkits that facilitate the process of locating/recruiting/managing agent machines as well as simplify the actual execution of a DDoS attack. The systematic overview of DDoS malware is presented in Appendix A. A more detail overview of malware pertaining to application layer DDoS attacks, the focus of this work, will be presented in the following chapter.

2.4 DDoS Defensive Strategies

A number of factors can make the defence against a DDoS attack a rather challenging task, some of them being:

1) A DDoS attack can involve hundreds of thousands of compromised hosts (i.e., attacking machines), possibly scattered all over the world. The defense against such DDoS attacks would require a widely distributed and cooperative response action by multitude of network operators. However, cooperation between different (especially multi-national) administrative domains is often hard to achieve.

2) In some cases, in addition to being located in diverse geographic locations, the compromised hosts are 'reprogrammed' to use spoofed IP addresses. The defense against such Botnets is particularly challenging, as the true origins of the attacking traffic are hard, if not impossible, to identify.
3) In the case of an application layer DDoS attack, malicious traffic tends to look very much like the traffic generated by legitimate user and, consequently, can be very difficult to identify (i.e., filter out).

In spite of the numerous (potential) challenges in dealing with DDoS attacks, a plethora of defence techniques have been proposed both commercially and in academia. As presented in [31], [25], [32] and [33] (also see Figure 2.4), DDoS defence techniques can generally be classified based on deployed activity 1) DDoS Prevention, 2) DDoS Detection, 3) DDoS Reaction/Mitigation, 4) DDoS Post-Attack Forensics. The taxonomy of the DDoS defenses is presented in Table 2.2.

In the following chapter 3, we narrow our focus onto application layer DDoS attacks, specifically HTTP-based DDoS attacks - the main research topic of this thesis.

Table 2.2 – Classification of DDoS Defences

Type	Technique	Description	Main Disadvantage
Prevention	Packet Filtering [32]	Examines the header and payload of a packet in order to detect malicious or malformed field values.	Unless the characterization is very accurate, filtering mechanisms run the risk of accidentally denying service to legitimate traffic.
	Disabling Unused Services	Drops/Filters out unnecessary network traffic, e.g., UDP packets.	Cannot prevent malicious traffic that utilizes legitimate/allowed packets.
	Anti-virus Tools (also see Section 3.4.2)	Signature detection for malware infections on a computer host.	New undiscovered vulnerabilities in the operating system (the so-called *zero-day vulnerabilities*), for which security updates are not yet available, can still be exploited by attackers to recruit new bots.
	Load Balancing	Distributes the incoming traffic across multiple concurrently deployed servers or server farms (e.g., content delivery networks such as Akamai, CloudFlare, Limelight, Amazon's Elastic Cloud and Microsoft's Azure.	Can be impracticable for small- to medium-size businesses, due to the high cost of deploying multiple servers and/or redundant bandwidth.
	Dedicated Anti-DDoS Appliances (also see Section 3.4.2)	Hardware (and software) devices specifically designed to prevent and stop DDoS attacks.	Impractical for small- to medium-size businesses, due to the high cost.
	Cloud-based Solutions (also see Section 3.4.2)	So-called "scrubbing centers" that utilize large numbers of servers and bandwidth to filter traffic.	High cost. The four well-known companies that provide this kind of service are: Akamai, CloudFlare, Cisco and Arbor Networks.
Detection	Misuse/Signature Detection [34]	Identifies well-defined patterns of known exploits and then looks out for the occurrences of such patterns. Intrusion patterns can be any packet features, conditions, arrangements and interrelationships among events that lead to a break-in or other misuse.	Relies on existing (already identified) signatures of misuse, it can easily fail to detect new forms and variants of exploits.
	Anomaly Detection [35]	Constructs a model of what is considered ‘normal behaviour’; and subsequently marks anything that does not fit this model as abnormal.	Main challenge of all anomaly-based detection techniques is minimizing both the number of false-positives and the computational overhead of attack detection.

Response/Reaction	Rate-limiting [36]	Impose a rate limit on a set of packets that have been characterized as malicious by the detection mechanism.	The disadvantage is that rate limiting will allow some attack traffic through, so extremely high-scale attacks might still be effective even if all traffic streams are rate-limited. Examples of rate-limiting mechanisms are found in [36], [24] and [37].
	Network Reconfiguration [38]	Reconfigure the topology of the victim's network or some of the intermediate networks, in order to add more resources to the victim or to isolate the attack machines.	Only a temporary measure as after the reconfiguration, the attacker can always attack the new target point.
Post-attack Forensics	Traffic and Event Log Analysis	Identify specific characteristics inside the attacking traffic. This data can, then, be used to update packet filtering, load balancing and throttling countermeasures, and thereby increase their efficiency and protection ability.	Cannot prevent/detect imminent/occurring attacks.

Chapter 3 Application Layer DDoS Attacks and Related Defenses

Application layer attacks are an increasingly prevalent type of DDoS attacks, as indicated in a recent survey [4]. These types of attacks are more stealthy than traditional network-layer and transport-layer DDoS attacks since the attacker utilizes legitimate looking application layer (typically HTTP-protocol) requests to overwhelm the victim server or degrade its performance.

In this chapter, we present an overview of application layer DDoS attacks and their defences with the specific focus on HTTP-based DDoS attacks. Firstly, in section 3.1, we categorize and describe various types of application layer DDoS attacks. In section 3.2, we overview various types of the most common application layer DDoS attacks - HTTP-based DDoS attacks. In section 3.3, we highlight the sophistication and limitations of current malware toolkits that are capable of generating HTTP DDoS attacks. In section 3.4, we review the techniques for defending against HTTP DDoS attacks as proposed in research literature and in industry and discuss their limitations. Lastly, we present the main conclusions in section 3.5.

3.1 Overview of Various Types of Application Layer Attacks

Application layer DDoS attacks can be executed by exploiting various application layer protocols such as HTTP(S), SSL, SIP, DNS, FTP, NTP or SMTP. Examples include: 1) DNS amplification attacks [27], a SSL renegotiation DDoS attack [39], a SIP (VoIP) flooding attacks [40], a FTP-based DDoS attacks against Windows Servers [41], a NTP amplification attack [42] and a SMTP-based DDoS attacks on mail servers [43].

However, as indicated in 2016 Arbor Networks' Worldwide Infrastructure Security report [4], 2014 Prolexic's Quarterly report [44] and 2015 Akamai's State of the Internet report [45], the majority of application layer attacks are executed by means of the HTTP protocol. The rates of occurrence of various attack types at the application layer in 2015 are presented in Figure 3.1.

3.2 Overview of HTTP-based DDoS Attacks

Our work focuses on the most prevalent and potent variant of application-layer DDoS attacks – attacks that involve the use of HTTP protocol. HTTP-based attacks are typically concentrated on one particular web application on one particular victim server, with the ultimate goal of tying up the server's resources. HTTP-based DDoS attacks are generally more efficient than TCP- or UDP-based attacks, primarily because the conventional anti-DDoS tools have much looser control over HTTP traffic [5].

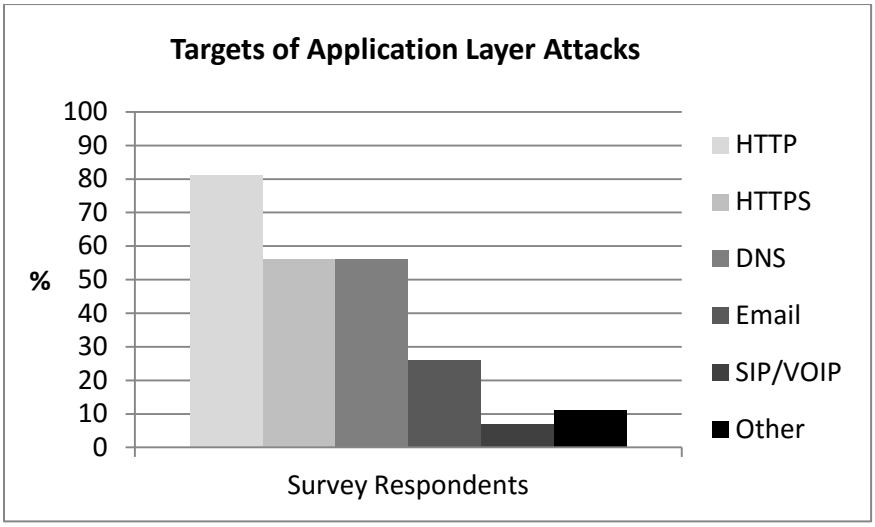


Figure 3.1 – Distribution of application layer attacks in 2015 (source: Arbor Networks Inc.)

3.2.1 Categories of Real-world HTTP-based DDoS Attacks

The categories of HTTP-based DDoS attacks, occurring in the real-world, can be categorized as following:

- *Web Server Vulnerability Attacks.* As described in [28], in the so-called SlowRead attack, the attacker exploits the vulnerability of web server's connection management logic. Specifically, the attacker exhausts the number of available HTTP sessions on a web server by sending partial time-delayed HTTP requests. The SlowPOST attack achieves the same effect as the SlowRead attack but this time with an execution of a slow HTTP POST request. Another similar attack is Apache Range Header DoS attack on the Apache web server described in [46].
- *Web Application Vulnerability Attacks*. In HashDoS attack [47], the attacker exploits vulnerable hash functions used in hash table implementations of many programming languages (e.g., PHP, ASP.NET, Python, Java) for web applications. XDoS attack [48] targets servers hosting web services. There are a number of variations of XDoS attacks such as: 1) a simple XML flooding attack on a web service, 2) XML entity expansion (XEE) [29] and 3) External entity reference

attack that exploits the XML parser functionality. ReDoS [30] attacks exploit the vulnerability specific to the most common implementations of regular expressions (regex). The vulnerability of regexes lies in the wide adoption of backtracking algorithms (rather than the traditional Deterministic Finite Automaton (DFA) contraction) in modern programming languages to implement regex matching. As a result, crafting a malicious regex will result in the matcher taking exponentially long time (in input size) to process an expression and essentially failing to terminate, causing a DoS attack [30].

- *One-shot HTTP attacks*. As described in [49], [50] and [51], an attacker, from a single machine, may repeatedly request a single HTML page with a large image file from a web server or submit a search query to increase the load on the database server, behind the web server.
- *HTTP GET flooding attack*. There are two variations of this type of an attack: a recursive and random recursive HTTP GET attack. As described in [51], and [52], in recursive HTTP GET attack, a bot is designed to continuously request a pre-defined random sequence of hardcoded list of pages from a web site. Random recursive HTTP GET attack, described in [51], is a modified version of Recursive GET but designed for forum sites or news sites where pages are indexed numerically, usually in a sequential manner. The attacking GET statements will insert a random number within a valid range of page reference numbers making each GET statement different than the previous one. The most common form of these types of attacks use GET requests but POST requests can be used as well.

3.2.2 Malware Toolkits with HTTP-based DDoS Attack Capability

There are a number of malware toolkits, introduced over the last 10 years, that provide support for generating various HTTP-based attacks. The following is the list of known HTTP DDoS toolkits with a brief summary of each:

- BlackEnergy [53] is a DDoS tool capable of launching several types of TCP, UDP and HTTP flooding attacks. The C&C operates as a PHP and MySQL server and can be accessed by its respective bots via standard HTTP protocol (i.e. through a URL of the form: http://*domain-name*/index.php). The bot's malware is known to passively propagate through spam and malicious web sites. BlackEnergy is available for purchase through Russian language forums from computer hackers at around US$40.
- Dirt Jumper [54] (also known as Russkill) is a newer version of BlackEnergy bot. It was introduced in January, 2009 with the most recent version 3.0 release in September, 2011. There are also some variants of Dirt Jumper bot such as Khan, Armageddon, Ferret, Aldi, Pandora and Di-BoTNet [52]. All of these bot variations support similar types of HTTP and TCP/UDP flooding attacks as the original BlackEnergy bot. In addition, the Dirt Jumper tool also supports a variable-rate HTTP DDoS attack.
- Low Orbit Ion Cannon (LOIC) [55] is an open source denial of service attack tool that can be easily found for download on the Internet. LOIC supports HTTP, TCP and UDP flood attacks. Variants of LOIC include the High Orbit Ion Cannon (HOIC) and HULK [55].
- R-U-Dead-Yet (RUDY), PyLoris**,** OWASP DoS HTTP POST and Tor's Hammer [55] are open source attack tools that support HTTP-based DDoS attacks such as SlowPOST and SlowRead.

3.2.3 Characteristics of Web Browsing Behavior

The successful detection of current and future HTTP-based DDoS attacks will rely on the ability to identify the differences in browsing behaviour between actual human visitors and web crawlers, both benign and malicious. Therefore, in this section, we examine the similarities and differences in browsing behaviour between human web visitors and current as well as future automated web crawlers or bots.

3.2.3.1 Browser-like and Non-Browser-like Browsing Behavior

Human visitors employ a web browser[3] (such as Chrome, Firefox, Internet Explorer, etc.) to browse web site content. When a human visitor selects a web page to view in a browser, the browser retrieves this "primary" content web page (such as an HTML/ASP/JSP page) via a HTTP request packet and then issues HTTP requests for all embedded "secondary" content (such as embedded images, videos and script files). Note that the requests for primary content pages are explicitly requested by human visitors, as they enter the URL in the address bar, click on the hyperlinks or press back/forward buttons in the browser's window menu, while the secondary content is requested inline by the browser in order to properly present the entire content in the page. The graph in Figure 3.2 illustrates the hierarchical structure of a sample web site where the directed full-line edges represent the hyperlinks between primary objects and secondary/inline content is linked to the primary content with delimited edges.

The browser's HTTP request for the primary and secondary pages includes the *User Agent String (UAS)*. UAS specifies the hardware/software (i.e., browser, search engine crawler, Smartphone, tablet, etc.) used by the client to communicate with the server. It is generally assumed that a HTTP request carrying UAS corresponding to a well-known browser (e.g., Google Chrome, Internet Explorer, etc.) belongs to a human visitor. However, note that this HTTP packet field can be easily spoofed by malicious web visitors. For instance, LOIC is one of the tools that supports spoofing of UAS.

Unlike humans, most (benign) web crawlers will typically exhibit non-browser-like browsing behaviour. Namely, web crawlers will request exclusively primary web pages (e.g., HTML pages) for indexing purposes [56] without requesting any of their embedded content. Some crawlers such as Google image bot

[3] A web browser (or browser) is a software application for retrieving, presenting and traversing information resources on the World Wide web.

will request exclusively image files (i.e. secondary/embedded content). In general, it is not expected that any benign crawler follow the request for a primary web page with requests for all its embedded content.

3.2.3.2 Characteristics of Human Browsing Behavior

Human browsing behaviour can be modeled by a number of metrics such as: 1) sequence of web pages visited on a particular web site or web domain, 2) time spent viewing/reading each web page, 3) number of pages viewed while visiting a particular web site or web domain, 4) number of images versus web pages downloaded, number of bytes downloaded from a particular web domain, etc.(these browsing behaviour features have been previously utilizing in analyzing differences between human and web crawler browsing behaviour in [56]).

A number of surveys (e.g. [57] and [58]) of real-world web sites have shown that a vast majority of human visitors request (i.e., are only interested in) a very small set of web pages from a particular web site. Namely, we can expect that the majority of human visitors will be interested in the currently most popular content on the web site. (For a typical distribution of web page popularities and page viewing times of real-world web sites we have examined in our work, see Figures 5.13 and 5.15 in chapter 5, respectively.)

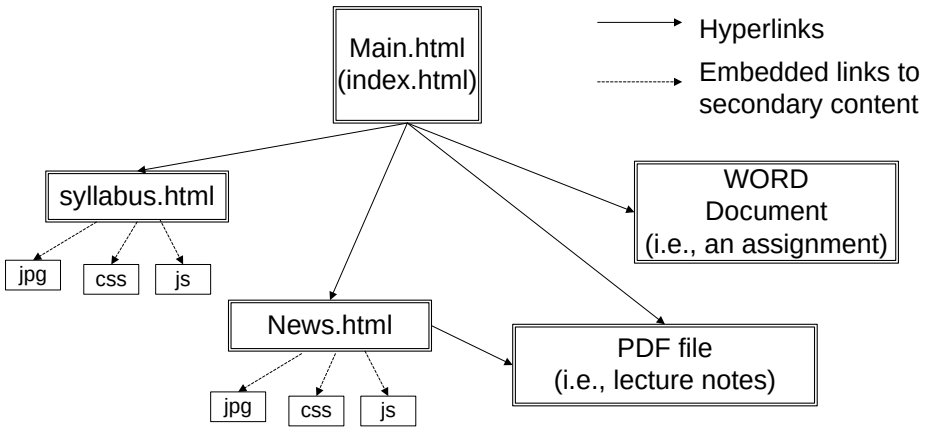


Figure 3.2 – The graph of a University course web site with primary and secondary web content

In general, on a given web site, the typical human browsing behaviour can be characterized by:

a) Typical sequential browsing behaviour.

b) Typical time-wise browsing behaviour.

a) Typical Sequential Browsing Behavior of Human Visitors

If we examine the diagram in Figure 1.1, we can expect that the "breaking news" web page would be (one of) the most accessed/viewed pages on that site. Also, a sequence of pages linking to the "breaking news" web page would be the most popular browsing path among typical human visitors. Moreover, on a University's course web page (see Figure 3.2), the newly added course material, in the form of a PDF file, will be (very likely) the most download page by students (i.e., typical human web visitors of this web site).

Therefore, the typical sequential browsing behaviour of human visitors can be defined as the collection of the most frequent sequences or paths of primary page traversals through a web site by human web visitors.

b) Typical Time-wise Browsing Behavior of Human Visitors

In general, the page viewing times will be dependent on: a) the web site's actual subject matters, b) the relevance of those subjects to the site's visitor population and c) the impact the visual organization of information on each web-page has on the amount of time a human visitor needs to absorb the given information (before moving to the next page), etc.. For instance, in Figure 1.1 we expect that the "breaking news" page would be the page with one of the longest viewing times of all web pages on the web site, while in Figure 3.2, we would expect that the PDF file would the longest viewed page on that web site. Therefore, the typical time-domain browsing behaviour of human visitors can be defined as the range of time that most humans spend viewing/reading the contents of each page.

Now, in the context of application-layer DDoS attacks, it is reasonable to expect that it will be quite challenging for attackers to engineer their bots to correctly guess the typical sequential and time-wise browsing behaviour of human visitors. This is especially true in dynamic web domains where the web content is continuously added, updated or deleted and, consequently, where the typical sequential and time-wise browsing behaviour of human web visitors continuously changes.

3.2.4 Categories of Current and Future HTTP-based DDoS Attacks

Depending on the level of their sophistication and the likelihood of their successful detection using current-day intrusion detection systems, HTTP-based DDoS attacks can be grouped into the following three categories: trivial attacks, as currently supported by real-world malware samples, and future intermediate and advanced types of attacks expected to be available and dominant in the future. The taxonomy of HTTP-based DDoS attacks is outlined in Table 3.1.

Table 3.1 – Taxonomy of HTTP-based DDoS attacks

Timeline	Detection Level / Type	Attack Characteristics
Current (supported by existing Malware)	**Trivial**	• Each bot issues one or a few unrelated HTTP requests labeled with unknown or known-blacklisted UASs • Each bot issues one or a few unrelated HTTP requests labeled with spoofed UASs of well-known benign crawler (e.g., Googlebot and MSNbot) • Each bot issues one or a few unrelated sequence of HTTP requests labeled with spoofed UASs of legitimate web-browsers (e.g. Mozilla Firefox) • Web Server Vulnerability, Web Application Vulnerability, One-shot and HTTP GET/POST attacks
Future (**not** supported by existing Malware)	**Intermediate**	• Each bot issues a (semi-random) sequence of HTTP requests that appear to be generated via a legitimate web-browser (i.e., by a legitimate human visitor)
	Advanced	• Each bot issues a sequence of HTTP requests that appear to be generated via a legitimate web-browser and match a sequence of web-pages (possibly) browsed by a legitimate human visitor

3.2.4.1 Trivial Attacks

In trivial attacks, each bot is instructed to send one or a limited number of unrelated HTTP attacks towards the target site. In terms of their HTTP-packet implementation, trivial attacks can be grouped into the following four sub-categories:

1. *Attacks that employ atypical/unknown or known-blacklisted UAS.* In this type of attack, each malicious crawler issues HTTP requests containing a UAS that is not known or that has been previously identified as malicious by intrusion detection systems.[4]
2. *Attacks that employ spoofed UASs of legitimate web crawlers.* In this type of attack, each malicious crawler issues HTTP requests labeled with a spoofed UAS of a known benign crawler such as Googlebot, MSNbot, etc.. By doing so, the attacker aims to evade an easy detection by the victim's intrusion detection system.
3. *Attacks that employ spoofed UAS of legitimate web-browsers*. In this type of attack, each malicious crawler issues HTTP requests labeled with a spoofed UAS of a legitimate web-browsers (e.g., the UAS of Google Chrome or Internet Explorer). By doing so, the goal is to make the attack packets appear indistinguishable from the packets generated by regular human visitors.
4. *Web Server Vulnerability Attack, Web Application Vulnerability Attack, One-shot HTTP Attack* and *HTTP GET/POST Flood Attack.* In a Web Server Vulnerability Attack, the attacker breaks HTTP GET/POST requests into segments and then sends them to the target with a delay. In the Web Application Vulnerability Attack, One-shot HTTP Attack and GET/POST Flood Attack, the attacker sends a specially crafted valid but unusual HTTP request to the target. For instance, a specially crafted HTTP request could contain a computationally-expensive database query that triggers the web server to severely exhaust its computational resources. Any of these attacks may use either a known valid

[4] [64] is a good public repository of all known UAS, which are grouped into benign UASs (e.g., UASs corresponding to legitimate crawlers and browsers) and malicious UASs (e.g., UASs corresponding to spam bots such as bwh3_user_agent). In one of our earlier studies using real-world web traces, (see [10] and chapter 5), we have identified 1000s of distinct web visitors whose web requests are labeled with malicious or unknown UASs.

browser-like UAS, a known benign crawler-like UAS, a known malicious crawler-like UAS or an unknown UAS.

3.2.4.2 Intermediate Attacks

The *intermediate attacks* are more sophisticated than trivial attacks. In these attacks, bots are designed to continuously request a pre-defined random sequence of web pages with all of their embedded content. (Note, at the HTTP-packet level, the attacker can still choose to use any of the above mentioned strategies of UAS spoofing; though, the strategy of using UASs of legitimate browsers is most likely.) By producing longer sequences of requests that appear as being generated through a legitimate web-browser, the goal is to make the attack traffic 'blend in' with the regular human traffic much better than in the case of trivial attacks in group 3).

3.2.4.3 Advanced Attacks

It is believed that in the next few years the intermediate attacks will undergo further evolution – instead of a random attack sequence, the sequence of HTTP requests will be carefully chosen so as to better mimic the browsing behaviour of regular human visitors [59]. Namely, in the case of regular human browsing, a request for one web-page is typically followed by a request for another page that is directly linked to it. Moreover, the actual choice of the 'next page' is often determined by its contextual relevance[5] with the first page requested. Clearly, to design an attack of this level of sophistication, the attacker will have to put a considerable effort into studying the victim's site (i.e., its particular content and structure), and then make some educated guesses about the most likely human behaviour on a site with such characteristics. As shown in Figure 3.1, we expect that in an advanced attack, bots will be 'more-likely' to request web pages in a popular/typical sequence (unlike in the intermediate attack).

[5] For instance, let us suppose a web page contains topics on science fiction books, where each topic is marked as a link to another page where that topic is covered in more detail. If a user is interested in science fiction books about time travel, he/she will (most likely) follow the link on "science-fiction and time travel" web page.

Note that the goal of the attacker responsible for executing the advanced, intermediate and also trivial attacks such as Web Application/Server Vulnerability and One-Shot HTTP attacks might not be to inundate the server resources, but stealthily bring them down over a prolonged period of time. This is especially true in the case of the Economic Denial of Sustainability (EDoS) attacks. (For more on EDoS see [60], [61], [62] and [63].) However, attacker is free to choose to program their bots to generate volumetric/flooding versions of the above attacks as well. (Note that in some literature volumetric HTTP-based attacks are referred to as Flash-crowd[6] attacks.)

The similarities and differences in the browsing behaviour of various attack bots (i.e., malicious crawlers), benign search engine crawlers (such as Googlebot, Yahoo, etc.) and human visitors are illustrated in Figure 3.3.

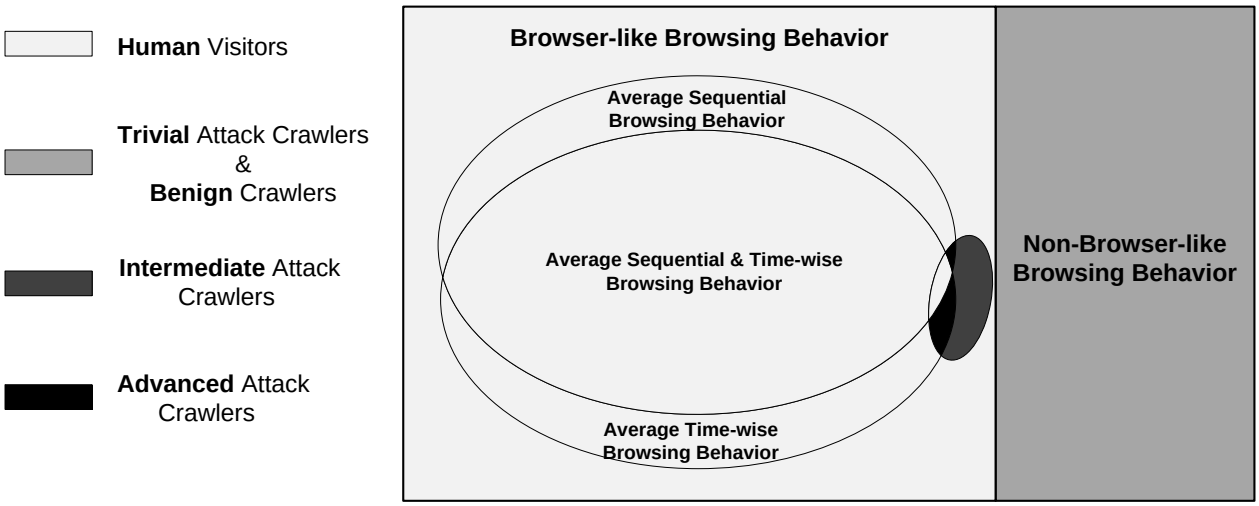


Figure 3.3 – Browsing behaviour of human web visitors and various types of crawlers

[6] A legitimate Flash Crowd event is a situation where some popular information emerges on a web site (such as a news story or a sports event), and many browsers (i.e., human visitors) attempt to access that information, thus creating a large demand/load on the server. An attacker can easily achieve a "Flash Crowd"-looking effect by performing an excessive number of seemingly legitimate actions on the target web application

3.2.5 Defences Against Trivial, Intermediate and Advanced Attacks

In this section, we outline the strategy for defending against trivial, intermediate and advanced attacks.

3.2.5.1 Trivial Attack Detection

The detection of subcategories of trivial attacks 1) to 3) can be accomplished as follows. As shown in [10], trivial attacks of group 1) can be detected by matching the UASs of received HTTP requests to UASs enlisted on [64] or other publically available UAS-whitelists. A possible way of detecting trivial attacks of type 2) is by verifying whether the bot's IP address really belongs to the domain corresponding to the bot's UAS (e.g., through reverse DNS lookup). Trivial attacks of type 3) can be either detected by verifying whether the true browser-like behaviour is exhibited (i.e., a request for a web-page is followed by requests for all of its embedded content). Note that, as shown in Figure 3.3, we expect that in a trivial attack scenario, bots will exhibit a non-browser-like browsing behaviour.

As pointed earlier, the HTTP-based DDoS attacks from the fourth category of trivial attacks (see Table 3.1) are only special cases of the first three types of trivial attacks, and consequently can be dealt with using the same defensive techniques.

3.2.5.2 Intermediate Attack Detection

As shown in Figure 3.3, we expect that in an intermediate attack scenario, bots will mimic a browser-like browsing behaviour but still request web pages in an atypical or uncommon web page sequence. We believe that the only way to defend against this type of attack will be through the use of sophisticated data mining techniques – techniques that can facilitate accurate modeling of sessions that are truly human-generated (i.e., truly legitimate) and then effectively separate them from sessions that are 'made up' to look legitimate. Namely, the detection of intermediate attacks could be achieved by comparing the

sequences of requests in suspicious/attack sessions (i.e., the sequences of actually requested web-pages) with sequences that are typically generated by actual human visitors.

3.2.5.3 Advanced Attack Detection

Similarly to the case of intermediate attack detection, the detection of advanced attacks will require the use of sophisticated data mining techniques. In the advanced attack scenario, the amount of time that humans spend on a page will be very important in separating human visitors from human-mimicking bots as it is dependent on the amount of 'contextual' information found on that page, which cannot be easily faked/forged by automated bots. As shown in Figure 3.3, we expect that in an advanced attack scenario, bots will mimic a browser-like browsing behaviour, may or may not request web pages in a typical or popular web page sequence, but most like will not view web pages with an average web page viewing time.

The detection and defence against the intermediate and advanced attacks will also be particularly challenging in dynamic web domains, as it will require the use of intelligent data-mining techniques that also operate in a real-time manner while performing adaptive tracking of ever-evolving user behaviour.

3.3 State-of-the-Art and Limitations of HTTP-based DDoS Toolkits

In Table 3.2, we summarize the current-day sophistication of various DDoS malware introduced in section 3.2.2 (in the context of attack taxonomy outlined in Table 3.1), and outline their limitations.

We can infer the following two major conclusions:

1. None of the existing tools are capable of generating either an intermediate or an advanced DDoS attack. Namely, the existing tools do not implement the functionality of a browser (i.e., to the best of our

knowledge, these tools cannot be instructed to follow each request for a primary page with requests for all of page's embedded content). Therefore, they are only capable of executing trivial attacks such as Web Application/Server Vulnerability, One-Shot and HTTP GET/POST flood attacks) - see 3.2.4.1 above.

2. Also, in general, there is little variation in the capability of the existing tools to generate the HTTP-based DDoS attacks. Some of these tools are very specialized (e.g., designed to exclusively generate SlowRead/SlowPOST attacks).

Table 3.2 – Current attack capabilities of DDoS malware

Malware	HTTP-based Attack Sophistication	Main Disadvantages
Dirt Jumper [52] (some variants of Dirt Jumper bot include Khan, Pandora and Di-BoTNet **[52]**	• One-shot • Recursive HTTP GET floods • Recursive Random HTTP GET floods • Spoofing UASs of legitimate web crawlers • Spoofing UAS of legitimate web-browsers	• Do not parse web pages and do not retrieve embedded content • No capability to execute Web Server Vulnerability Attacks, intermediate or advanced attacks
BlackEnergy [53] **Armageddon [65]** **Ferret [66]** **Aldi [67]**		
Low Orbit Ion Cannon (LOIC) **XOIC** **HULK [55]**		
R-U-Dead-Yet (RUDY) / PyLoris **OWASP DoS HTTP POST** **Tor's Hammer [55]**	• Web Server Vulnerability Attacks • Spoofing UASs of legitimate web crawlers • Spoofing UAS of legitimate web-browsers	• No capability to execute intermediate or advanced attacks

3.4 Overview of the State-of-the-Art in HTTP-based DDoS Defense

In this section, we overview the techniques as proposed in research literature and in the industry for defending against the HTTP-based DDoS attacks and highlight their limitations in handling these types of attacks.

3.4.1 Overview of Countermeasures against HTTP-based DDoS Attacks Proposed in Research Literature

Table 3.3 presents the state-of-the-art in HTTP-based DDoS defense techniques as proposed in research literature. These techniques can be grouped into the following two main categories:

1. Techniques that attempt to build the model(s) of regular human-user browsing behaviour from user-traffic logs, and then match the behaviour of new users against these pre-built models. This approach is also known as anomaly detection (recall section 2.4). In these proposals researchers have employed a variety of methods to model regular or typical human-user browsing behaviour such as statistical modeling (i.e., wavelet analysis, auto-correlation analysis, Zipf's Law analysis, entropy analysis, Spearman correlation test), machine learning, Markov chains and traffic pattern modeling (such as packet rate flow analysis, browsing sequences and page viewing times analysis).
2. Techniques that attempt to differentiate between human- and machine-generated HTTP sessions on the fly by employing active or passive CAPTCHA tests.

Table 3.3 – State-of-the-art of HTTP-based DDoS defenses in research literature

Strategy	Research Solutions	HTTP-based Attacks Protection	Main Disadvantages
Statistical Modeling	[68], [60], [61], [62], [69], [70], [71], [72], [73], [74]	• Recursive HTTP GET floods • Recursive Random HTTP GET floods	• No protection from intermediate attacks in dynamic web domains • No protection from advanced attacks • Mostly employ features that can be easily emulated by attacking malware/bots
Machine Learning	[75], [76], [77], [78] [79], [80], [81], [82]		
Markov Modeling	[83], [84]		
Deterministic Finite Automaton	[85]		
Network Layer or Application Layer Metrics	[86], [87], [88], [89], [90], [91], [92] [93], [94], [95], [96], [97], [98]		
	[99]	• Protection from intermediate and advanced attacks in static web domains	• No protection from intermediate/advanced attacks in dynamic web domains
Active CAPTCHAs	[63], [100], [101], [102]	• Provide protection from all HTTP-based DDoS attacks	• Presents CAPTCHA to every web visitor
Passive CAPTCHAs	[103], [104], [105]		• Attacker can pre-process the source file of the page to defeat the detection

3.4.1.1 Anomaly Detection

The techniques in group 1) are almost exclusively optimized to deal with static web domains and assuming relatively stable patterns of user browsing behaviour - they are <u>not</u> designed to deal with *intermediate or advanced attacks* against domains with evolving web content. For instance, in all works from group 1), except [90], the model(s) are constructed once, from the sample dataset and then evaluated on another (test) sample dataset. Note that such a pre-built model will become outdated as the typical browsing behaviour (page viewing times and sequences of web page accessed) of web visitors changes over time as the web content is updated, added or removed.

Also, some of the previous body of research on HTTP-based attacks considers distinguishing packet arrival rate differences between regular human visitors and attack crawlers. Note that these application-layer DDoS mitigation solutions activate once the victim web server is under extreme duress. However, the attack traffic in intermediate or advanced attacks may not be volumetric but instead degrading (e.g., EDoS attacks discussed in section 3.2.4.3). As such, the packet rate between regular human visitors and 'human-mimicking' malicious bots is expected to be similar and hence will be ineffective feature in capturing these types of bots.

3.4.1.2 "On-the-fly" Detection (CAPTCHA, decoys and Turing Tests)

On the other end of the spectrum, in group 2), authors propose employing active or passive CAPTCHA tests to differentiate between human and machine generated HTTP sessions on the fly. An active CAPTCHA test requires web visitors to solve graphical puzzle that can be easily solved by a human but not by a bot, e.g., recognizing numbers and letters on a distorted image. Only users who solve the puzzle are granted access to the service. However, although generally effective in accomplishing their task, the graphical puzzle tests are, often, irritating to human visitors, and thus must be served infrequently. (In a recent Scientific American article “Time to Kill-Off CAPTCHAs” [106], the author eloquently summarizes the commonly felt negative sentiment about CAPTCHA technology.) Also, they treat all automated crawlers equally – both the benign and malicious ones by completely blocking their access to a web site. In addition, the use of CAPTCHA farms - companies that employ thousands of people around the world, usually in developing countries, to solve CAPTCHAs and help spread spam [107] - has also been reported as an effective way of combating CAPTCHA-based defences.

Passive CAPTCHAs such as scripts that monitor ‘user biometrics’ (mouse movement, page scrolling, etc.) [103] or decoy hyperlinks [104] and [105] are less invasive techniques to humans. However, the method in [103] has a higher likelihood of false positives — e.g., when users disable JavaScript — and false

negatives — because bots can modify their behaviour easily to fit the required patterns. The main disadvantage of the techniques described in [104] and [105] is that an attacker can circumvent the detection by pre-processing the source file of the web page and locating the decoys.

3.4.2 Overview of Commercial Countermeasures against HTTP-based DDoS Attacks

Given their prevalence and potentially detrimental consequences of application-layer DDoS attacks , there are many commercial anti-DDoS solutions designed to prevent and respond to these types of attacks. The sophistication and limitations in defending against the trivial, intermediate and advanced HTTP-based DDoS attacks of market-leading anti-DDoS solutions is presented in Table 3.4.

Although majority of existing commercial solutions provide protection from trivial attacks, they are too generic and unsuitable for dealing with intermediate and advanced attacks with evolving web site content. Namely, they can be categorized into solutions that employ one or more of the following: 1) hard thresholds such as connection and packet rate limiting per IP address ([108], [109], [110], [111], [112], [79], [113], [114]), 2) HTTP packet header content filtering such as UAS/referrer spoofing ([115] [112]) and 3) aggregate traffic statistics ([116], [117], [118], [119], [120], [113]). The drawbacks of these approaches can be summarized as follows:

- *Techniques that employ hard thresholds and/or well-known and publicized attacks signatures.* In practice, the traffic characteristics can frequently change due to the evolving nature of the respective web domains. By relying on such static and non-adaptable parameters, these systems are clearly inadequate when it comes to web domains in which both the legitimate and attack traffic is likely to undergo continual and generally unpredictable change. It is, therefore, unreasonable to expect that hard

system-thresholds and attack signatures could provide adequate protection for such dynamic web domains.

- *Techniques that employ application layer packet header content filtering.* The major drawback of simple packet content filtering is that attackers can easily circumvent detection by ensuring that their attack parameters follow the protocol specifications.
- *Techniques that rely on aggregate traffic statistics*. Note that an algorithm that builds its detection strategy on 'aggregate traffic statistics' (e.g., total number of connections/packets) will likely not be able to distinguish (e.g.) between a true Flash Crowd incident (sudden surge of traffic generated by humans) and a sudden surge in traffic generated by a web crawler spoofing the UAS of a browser. Also, it is unreasonable to expect that these traffic statistics will stay constant in dynamic web domains.

Table 3.4 – State-of-the-art of commercial DDoS defenses

Deployment Scenario	Brand name	HTTP-based Attacks Protection	Main Disadvantages
Network Security Appliance (Firewalls and IPS)	Dell SonicWall [121] WatchGuard [122] Palo Alto Networks [108] Juniper's Junos DDoS Secure [115] [112] Barracuda [109] HP [116] McAfee (Stonesoft) [110] Sophos [111] F5 [114]	• Recursive HTTP GET floods	• Relies on hard thresholds and/or well-known and publicized attacks signatures and/or application packet header content filtering and/or aggregated traffic statistics • No protection from intermediate attacks in dynamic web domains • No protection from advanced attacks
Dedicated Anti-DDoS Appliance	Check Point Protector [117]	• Recursive HTTP GET floods • Recursive Random HTTP GET floods • Web Server Vulnerability Attacks (trivial attacks)	
	RioRey's DDoS appliances [118]. Huawei [119]		
	Fortinet's FortiDDoS [123]	• Recursive HTTP GET floods • Recursive Random HTTP GET floods • Web Server Vulnerability Attacks (trivial attacks) • Application Layer Filtering of HTTP Request packet to detect suspicious content such as User Agent strings, Cookies, Referrer fields	
Cloud-based (Scrubbing Solutions)	Prolexic [124] [120]	• Recursive HTTP GET floods • Recursive Random HTTP GET floods • Web Server Vulnerability Attacks (trivial attacks)	
	Arbor Networks and Cisco Clean Pipes 2.0 [113]		
	Akamai SiteShield [125]		

3.5 Summary

In the light of the attack taxonomy presented in Table 3.1, the detection of trivial HTTP-based attacks can be achieved through a simple packet-by-packet (i.e., request-by-request) inspection, and as such could be

easily integrated into many of the existing intrusion detection systems. (Some of the current-day commercial intrusion detection systems have such detection mechanisms already in place - see Table 3.4) The detection of intermediate and advanced attacks is considerably more complex, and generally requires the use of advanced techniques of statistical modeling. While (only) a few of the earlier research works have broached the issue of these attacks in the context of static web domains, to the best of our knowledge, there has been no previous research study on the detection and defence against the next-generation HTTP-based application-layer attacks (i.e., intermediate and advanced attacks) in dynamic web domains. Also, it is very much evident that once these attacks start emerging, none of the current-day commercial anti-DDoS solutions will be able to provide adequate defence against them.

In the following chapter 4, we investigate the state-of-the-art in automated web crawler browsing behaviour and, in more depth, identify differences between the browsing behaviour of web crawlers and humans. Additionally, in chapter 4, we propose novel techniques for detection of various suspicious web visitors in stable/static web domains by utilizing various data mining and statistical techniques. Then, in chapters 5 and 6, we move onto explaining our methodology for building an anti-DDoS system for both static and dynamic web domains.

Chapter 4 Detection of Web Crawlers and Suspicious Web Visitors in Static Web Domains

There have been a number of research works ([56], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135], [136], [137], [138], [139], [140], [141], [142]), [143], [144], [145], [146], [147] and [148]) published over the last decade and a half on the topic of detecting browsing behaviour differences between human visitors and benign crawlers. However, very few studies have examined the detection of browsing behaviour differences between human visitors and malicious crawlers.

In this chapter, we present our work on detecting malicious/suspicious web visitors with an assumption of an underlying static web domain. In section 4.1, we present an overview of the previous works dealing with the detection of mostly benign crawlers. Next, in section 4.2, we present our study on utilizing various supervised machine learning algorithms to detect malicious crawlers. To the best of our knowledge, this work is unique since it represents the first known attempt to separate malicious from benign visitors by employing novel features and multiple supervised machine learning algorithms. Lastly, in section 4.3, we present our concluding remarks.

4.1 Review of Existing Work on the Topic of Web Crawler Detection

Defences that attempt to differentiate between human and machine-generated web/HTTP sessions can be grouped into the following three categories:

1. Techniques that attempt to differentiate between humans and web crawlers by applying 'single-parameter rule-based analysis' to the contents of the web server access logs (the so-called syntactic log analysis).
2. Techniques that attempt to differentiate between humans and web crawlers by examining the 'multiple-parameter rule-based analysis' in robot traffic (the so-called traffic pattern analysis).
3. Techniques that attempt to differentiate between humans and web crawlers by employing supervised or unsupervised machine learning.

4.1.1 Syntactical Log Analysis

Syntactic log processing techniques classify web visitors by analyzing the contents of the web server access logs. A web server access log file is a time-ordered sequence of HTTP requests. Each entry (HTTP request) in the log includes the information such as the IP address/host name of the site visitor, the page requested, the date and time of the request, the size of the data requested and the HTTP method of request, UAS and the referrer field which specifies the web page by which the client reached the current requested page.

4.1.1.1 Detection by IP Address

The previous works that propose to identify web crawlers based on their IP addresses are [126], [127], [128]. In these studies, authors match the IP address of the visitor against an IP address database of known

web crawlers. The main disadvantage of this technique is the reliability and complexity of managing a database of IP addresses for each web crawler as they change over time.

4.1.1.2 Detection by User Agent String

The previous works that propose parsing a UAS to identify web visitors are [129] and [130]. The table of known user agent strings of browsers and crawlers can be found on web sites such as [149], [150] and [151]. As discussed in chapter 3, the main disadvantage of this technique is that malicious crawlers can easily spoof the user agent string in order to hide their true identity (e.g. LOIC tool can be used to spoof user agent string by modifying the UAS field inside the HTTP request packet).

4.1.1.3 Detection by Robots.txt file

Some works, such as [56], [126], [127] and [128], classify web sessions by checking whether the visitor has requested *robots.txt* file to classify web sessions. Namely, web administrators, through the Robots Exclusion Protocol, use a special-format file called *robots.txt* to indicate to visiting robots which parts or pages of their sites should not be visited by the robot. Thus, when visiting a web site, say *http://www.yorku.ca*, a robot should first check for *http://www.yorku.ca/robots.txt* in order to learn about possible access limitations. It is unlikely that any human would check this file, since there is no external or internal hyperlink linking to this file, nor are (most) users aware of its existence. Therefore, any visitor that retrieves robots.txt file is likely a crawler. However, this approach of separating automated crawlers from human visitors is not 100% effective in practice since it is known that some well-behaved web crawlers, as well as a large number of malicious bots, omit retrieving this file during a visit to a web site [152].

4.1.2 Traffic Pattern Analysis

Web visitor detection techniques which analyze patterns in robot traffic use a deeper interpretation of the entries in the web server access log. More specifically, these techniques consider aggregate attributes of

multiple requests such as: the overall amount of traffic they carry, percentage of error requests, etc. The idea behind looking at aggregate traffic characteristics is that values of web session features can identify whether the visitor is human or an automated agent. For instance, one of the features that is shown to be useful in this type of analysis is the percentage of HTTP GET/POST requests. In particular, most web crawlers, in order to reduce the amount of data requested from a site, employ the HEAD method when requesting a web page. Yet, requests coming from a human user browsing a web site via browsers are, by default, of type GET or POST. Another feature that is used to distinguish between human and crawlers visitors is the percentage of error requests. Namely, web crawlers typically have higher rate of erroneous requests than human visitors since they have higher chance of requesting outdated or deleted pages. Web-page popularity index can also help distinguish between the visitors because, typically, human visitors tend to request more popular pages in a single session and therefore end up with a higher page popularity index score. On the other hand, web robots generally request both popular and unpopular pages which results in a lower page popularity index score for their respective sessions. Previous studies that have employed one or more of these attributes to classify web visitors include [56], [126], [127], [131], [132], [133], [134] and [135]. In [56], authors identify 25 classification features that can be used to classify web site visitors.

Note that all of the features mentioned above are modelled to detect differences between the browsing behaviour of humans and benign web crawlers. However, as in the case of techniques from section 4.1.1, these features are not very effective in capturing of malicious crawlers, as malicious crawlers can be engineered to mimic all or most of the traffic characteristics of either humans or regular/benign web crawlers and thus evade detection.

4.1.3 Detection by Machine Learning

Many of the early systems for classification of web site visitors, such as the ones described in previous two sections, were based on simple rule-based logic. Clearly, the performance of such systems was greatly dependant on the accuracy of the established classification rules and visitor groups.

The classification accuracy can be improved with either supervised or unsupervised machine learning. The supervised machine learning refers to a group of algorithms that attempt to create a classification model from a pre-labelled (training) data samples, and then subsequently allow the use of this model for classification of new previously unseen data. On the other hand, in the unsupervised machine learning, sessions are classified without previous *a priori* knowledge (i.e. without a pre-labelled training dataset). In this method, data samples are clustered together based on the similarity of their corresponding feature values. Therefore, the advantage of employing unsupervised over supervised machine learning is the ability to obtain an unbiased look and understanding of the underlying data set and to generate results without human intervention.

Additionally, we can argue that machine learning models are more powerful than traffic/syntactic pattern analysis as they are able to consider the relationships among the implicit features that are considered separately in traffic/syntactic pattern analysis (see more in [152]). As a result, it may be more difficult for robot authors to circumvent detection by analytical techniques.

The main drawback of supervised learning is that it relies on accurately labelled training data to accurately classify new data. Note that the unsupervised learning does not employ labelled training data in its

learning process. However, it does require accurate post-learning data labelling to facilitate our understanding and validation of the results that are obtained by the actual learning process.

So far, several research studies have looked at the use of supervised machine learning for the purposes of data-mining and/or clustering of web sessions. Examples of supervised machine learning algorithms include decision tree mining [153], [148], [154], Bayesian classification [155], [136], and support vector machines [155], [79].

Several studies have looked at the use of unsupervised machine learning for the purpose of more general web log analysis. Examples of unsupervised machine learning algorithms include neural network algorithms such as Self-Organizing Map (SOM) ([137], [138], [139], [140], [141]), Growing Neural Gas (GNG) ([142], [156]), and Adaptive Resonance Theory (ART) ([143], [144], [145]).

In the following section, we outline the novelty aspects of our work.

4.1.4 Novel Aspects of Our Work

Table 4.1 presents taxonomy of the proposed detection methodologies for detecting browsing behaviour differences between human (benign) and machine web visitors - as found in research literature. The table also lists the main disadvantages (i.e., challenges) of each strategy in terms of its ability to detect malicious web visitors.

Table 4.1 – Main challenges of web visitor detection techniques

Strategy	Main Challenges
Syntactical Log Analysis	• Unreliable as malicious bots can spoof/emulate behaviour of human visitors
Traffic Pattern Analysis	• Unreliable as malicious bots can emulate syntactic/traffic patterns of human visitors
Supervised Machine Learning	• Biased learning – requires accurate labeling of a training dataset
Unsupervised Machine Learning	• Requires accurate labeling of a dataset for post-learning analysis
Turing Test (Section 3.4.1.2) (Active and Passive CAPTCHAs)	• Generally effective but of limited practical use

From the overview of the previous works above we can conclude the following. Firstly, previous researchers have never attempted to pursue detection of malicious web visitors to a web site by employing a combination of detection techniques that have shown promise in the past, e.g. machine learning techniques in combination with traffic pattern analysis and syntactic log parsing. We believe that such a 'holistic' (multi-technique) approach is the first and necessary step in the design of effective systems that would guard against HTTP-based DDoS attacks. Secondly, we believe that among various machine learning approaches, unsupervised learning is best suited for detection of malicious web visitors due to its ability to provide an unbiased look at the browsing behaviour of various web site visitors. In other words, the performance of unsupervised learning is least likely to suffer due to incomplete a-priori information and sporadic attempts of malicious visitors to disguise their behaviour.

In order to address the shortcomings of previous research on differentiation between malicious and benign web crawlers, we have conducted two studies: one employing supervised machine learning and another employing unsupervised machine learning. Both of these experiments are novel in the sense that we have combined well-known machine learning algorithms with traffic pattern analysis and syntactic log parsing to detect malicious/unknown crawlers.

4.2 Detecting Differences in Browsing Behavior of Web Visitors with Supervised Data Mining Classifiers

In the first stage of our study, two separate sets of experiments for detection of malicious and non-malicious web site visitors using supervised learning are undertaken. (The results of this study were initially published in [7] and [8]). The goal of the first set of experiments is to: 1) examine the effectiveness of seven supervised algorithms in distinguishing between two basic and most common groups of visitors to a web site: known well-behaved web crawlers and human visitors, and 2) evaluate the potential of our two newly proposed web-session features to improve the classification accuracy of the examined algorithms. The goal of the second experiment is to: 1) examine the effectiveness of seven supervised algorithms in distinguishing between four different and potential visitor groups to a web site: malicious web crawlers, well-behaved web crawlers, human visitors and unknown visitors, and 2) evaluate the potential of two newly proposed web-session features to improve the classification accuracy of the examined algorithms in this particular case.

4.2.1 Dataset Preparation

In this section we give a brief overview of our web access log analyser that has been used to generate a workable dataset - comprising both training and testing data samples - from any given web-log file. The operation of the log analyser is carried out in three stages: 1) session identification, 2) feature extraction for each identified session, and 3) session labelling (see Figure 4.1).

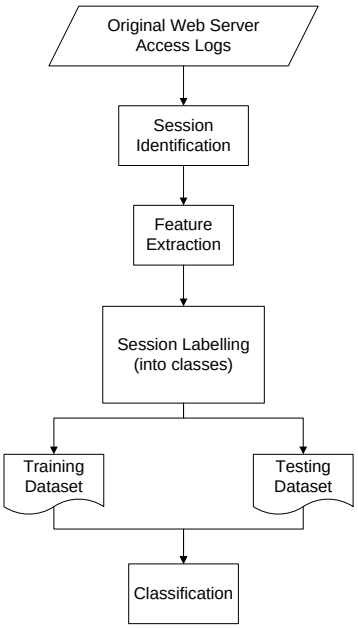


Figure 4.1 – Web server access log pre-processing

4.2.1.1 Session Identification

Session identification is the task of dividing a server access log into individual web sessions. According to [146], a web session is a group of activities performed by one individual user from the moment he enters a web site to the moment he leaves it. Session identification is typically performed first by grouping all HTTP requests that originate from the same IP address and the same user-agent, and second by applying a timeout approach to break this grouping into different sub-groups, so that the time-lapse between two consecutive sub-groups is longer than a pre-defined threshold. The key challenge of this method is to determine proper threshold-value, as different web users exhibit different navigation behaviours. In the majority of web-related literature, 30-min period has been used as the most appropriate maximum session length (see [56] and [136]). Typically, the web session definition also includes the condition that a user *does not* revisit the same page in the session. This is required in order to limit the length of a session as we

can imagine a web visitor (a malicious bot) to continuously request web pages and therefore never end a web session.

4.2.1.2 Feature Extraction

From previous web crawler classification studies, namely [56], [154], [136] and [147] we have adopted eight common features that are shown to be useful in distinguishing between browsing patterns of web robots and humans. These features are:

1) Click number – a *numerical* attribute calculated as the number of HTTP requests sent by a user in a single session. The click number metric appears to be useful in detecting the presence of the web crawlers because higher click number can only be achieved by an automated script (such as a web robot) and is usually very low for a human visitor.
2) HTML-to-Image Ratio – a *numerical* attribute calculated as the number of HTML page requests over the number of image file (JPEG and PNG) requests sent in a single session. web crawlers generally request mostly HTML pages and have the ability to ignore (i.e., not request) images on the site, which implies that HTML-to-Image ratio would be higher for web crawlers than for human users.
3) Percentage of PDF/PS file requests – a *numerical* attribute calculated as the percentage of PDF/PS file requests sent in a single session. In contrast to image requests, some crawlers, tend to have a higher percentage of PDF/PS requests than human visitors. E.g., a crawler traversing through a site would typically attempt to retrieve all encountered PDF/PS files, while a human visitor would be much more selective about what he chooses to retrieve.
4) Percentage of 4XX error responses – a *numerical* attribute calculated as the percentage of erroneous HTTP requests sent in a single session. Crawlers typically would have higher rate of erroneous request since they have higher chance of requesting outdated or deleted pages.
5) Percentage of HTTP requests of type HEAD – a *numerical* attribute calculated as percentage of requests of HTTP type HEAD sent in a single session. (In the case of an HTTP HEAD request, the

server returns the response header only, and not the actual source, i.e. file.) Many web crawlers, in order to reduce the amount of data requested from a site, employ the HEAD method when requesting a web page. On the other hand, requests coming from a human user browsing a web site via browsers are, by default, of type GET.

6) Percentage of requests with unassigned referrers – a *numerical* attribute calculated as the percentage of blank or unassigned referrer fields set by a user in a single session. Most web crawlers initiate HTTP requests with unassigned referrer field, while most browsers provide referrer information by default.
7) Number of bytes requested from the server – a *numerical* attribute calculated as the amount of data, in bytes, that was requested from the server in a single session. Typically, sessions belonging to web robots should request greater amounts of data from the server in a single session than sessions initiated by human visitors.
8) Page Popularity Index – a *numerical* attribute, calculated as the average value of Page Popularity Index (PPI) for all N pages (Page(i), i=1,..,N) retrieved during one observed session j – PPI_Session(j). The expression for PPI_Session(j) is given in (4.1):

$$\text{PPI\_Session(j)} = \frac{\sum_{i=1}^{N}\{(\max(\text{PPI}) - \text{PPI(i)})\}}{\text{totalNumberOfPageRequestsDuringSession\_j}} \tag{4.1}$$

The expression for the Page Popularity Index for a page i - PPI(i) in the above expression - is shown in (4.2).

$$\text{PPI(i)} = -\log_{10}\left(\frac{\text{numberOfRequestsForPage\_i}}{\text{totalNumberOfRequestsInLog}}\right) \tag{4.2}$$

Note that in (4.1) only requests for one particular session are considered. In (4.2), the total number of requests is considered – cumulatively, from all sessions (i.e., all users). It should be obvious from (4.1) and (4.2) that in order to calculate PPI_Session(j), first PPI(i) for all pages appearing in the log file needs to be calculated. Also, note that the max(PPI) is the maximum PPI out of all pages. This metric considers all primary files requested from the server (i.e., the requests for the secondary files such as images and scripts are excluded from the calculations above). The more detail discussion of the page popularity index and its calculation can be found in [157].

In the remainder of this chapter we will refer to these features based on their numeric ID shown above. In addition to these eight features and based on the recommendations of a recent study [152], we have derived and incorporated two additional novel features in our web robot classification:

9) Standard deviation of requested page's depth – a *numerical* attribute calculated as the standard deviation of page depth across all requests sent in a single session. For instance, we assign a depth of three to a web page '/cshome/courses/index.html' and a depth of two to a web page '/cshome/calendar.html'.
10) Percentage of consecutive sequential HTTP requests – a *numerical* attribute calculated as the percentage of sequential requests for pages belonging to the same web directory and generated during a single user session. For instance, a series of requests for web pages matching pattern '/cshome/course/*.* will be marked as consecutive sequential HTTP requests. However, a request to web page '/cshome/index.html' followed by a request to a web page 'cshome/courses/index.html' will not be marked as consecutive sequential requests.

In [152], the authors argue that analytical robot detection techniques must be based on fundamental distinctions between the robot and human traffic across server domains and in the face of evolving robot traffic. We argue that the features 9 and 10 - which to the best of our knowledge have not been used in

any previous research on web robot detection - have an excellent chance in separating human and robotic users in server access log sessions. Namely, the importance of features 9 and 10 can be explained as follows. In a typical web-browsing session, humans are set to find information of interest by following a series of thematically correlated and progressively more specific links. In contrast, robots are neither expected to have such complex navigational patterns, nor would they be restricted by the link structure of the web site. For instance, web crawlers like Google browse systematically through an entire web domain, i.e., they access files at various depths in the file hierarchy. Therefore, the cumulative standard deviation of their sessions should be large. On the other hand, humans tend to concentrate on one particular type of information, typically stored over a few files in a single directory. For the above reasons, the standard deviation of requested pages' depths, i.e., feature 9, should be high for web robot sessions and low for sessions belonging to human users. Note that feature 9 will be effective at distinguishing crawlers from human visitors only when applied on log files generated from web sites with large number of distinct web pages such as a university department web site.

Also the number of resources requested in a single session is another distinction between robot and human traffic that is not expected to change over time. This distinction arises because human users retrieve information from the web via a web browser. This interface forces the user's session to request additional resources automatically. Recall from section 3.2.3 in chapter 3 that web browsers, after retrieving the HTML page, parse through it, and then send a barrage of requests to the server for embedded resources on the page such as images, videos, and client side scripts to execute. Thus, the temporal resource request patterns of human visitors are best represented as short bursts of a large volume of requests followed by a period of little activity. In contrast, web robots tend to generate their request in a more systematic and time-wise uniform (i.e., better paced) manner. For the above reasons, the number of consecutive sequential HTTP requests should be high in human user sessions and low in web robot sessions.

4.2.1.3 Why Only These 10 Features?

Note that the set of the 10 features we employed in our study is not an exhaustive set of all features that could possibly be used. For instance, in previous research such as [56], [154], [136] and [147], authors have employed additional features such as:

1) Percentage of binary execution files download from the server (e.g., .cgi/.exe/.class),
2) Percentage of ASCII files download from the server,
3) Percentage of zip files download from the server, percentage of Multimedia files download from the server (e.g., .wav/.mpeg),
4) Percentage of HTTP POST request types,
5) Percentage of other HTTP method request types,
6) The Boolean feature that indicates whether the page requests were made during the night time (e.g., between 12am and 7am) and
7) The Boolean feature that indicates whether a web visitor employed multiple user agent strings while browsing the web site.

We did not employ these additional features for the following reasons:

- Some of the features were not applicable since the content of the web sites we have examined either did not contain any or we saw very few of such files/request types. These features are: percentage of binary Execution Files download from the server (e.g., .cgi/.exe/.class), percentage of ASCII files download from the server, percentage of Zip files download from the server, percentage of Multimedia files download from the server (e.g., .wav/.mpeg), percentage of HTTP POST request types and percentage of other HTTP method request types.
- The night time feature is nowadays rather outdated. Specifically, in our experimentation we have noticed that many legitimate-looking crawlers initiated requests during the daytime (after 7am), while

many browser-based requests were made during the night time thus contributing very little to effective differentiation between human and machine-generated sessions.

- The Boolean feature that indicates whether a web visitor employed multiple user agent strings while browsing the web site may appear as a useful labeling feature to differentiate between legitimate and suspicious web visitors. However, in practice, human visitors could potentially switch from one type of browser to another when visiting sites (i.e., web-pages) that require a unique set of plugins. Also, two different versions of well-behaved web crawlers can share the same IP address (note that we identify unique web visitors based on their IP address) and therefore that visitor's requests would have different user agent strings. Thus, for the above reasons, this feature was not employed in our analysis.
- When justifying our feature selection choices, it is important to emphasize that our primary concern was to employ features that are general and independent of the web site or domain content and structure. For instance, percentage of HEAD requests, percentage of erroneous requests, click number and consecutive repeated request ratio are features that should have similar values across various web domains.
- Our objective was also to select features that cannot be easily emulated by crawler designers. For instance, features such as popularity index, server-to-client bytes and standard deviation of page request depth are very difficult to model or guess by malicious crawler designers, as they do not have a full insight into the content, structure and statistics of the victim site. Therefore, we believe that the ten features employed in our work are a reasonable choice, and are applicable in differentiating between human and machine-generate web sessions on a wide range of sites.

4.2.1.4 Dataset Labelling

Supervised data-mining algorithms require pre-labelled training samples in order to learn (i.e., build) a classification model for a particular dataset. Therefore, after the log analyzer parses the log file and

extracts the individual visitor sessions, each session (i.e. the respective feature vector) is labelled as belonging to a particular class.

In this study, we perform two types of classifications/experiments:

1) In the first experiment we examine whether human users and well-behaved crawlers can be separated by the classification algorithms. This experiment is somewhat similar to what has been previously done in [56], [136] and [147]. However, the novelty of our work is that we perform the classification with two additional (novel) features and by applying seven different well-established supervised data mining classifiers.
2) The goal of the second experiment is to investigate whether the browsing characteristics of malicious crawlers and unknown visitors are sufficiently different from browsing characteristics of human visitors and well-behaved crawlers to enable automatic classification.

The specifics of dataset labelling for two experiments are described in the following two sections.

a) Dataset Labelling in Experiment 1

The log analyzer maintains a table of user agent fields of all known (malicious or well-behaved) web crawlers. This table is built using the data found on web sites [150] and [151]. Given the information contained in this table, the dataset labelling employed in the first experiment is performed as follows:

1) If the user agent field of a session matches the entry in the table of a known well-behaved crawler, the respective feature vector is labeled as such (the vector's class label is set to 1).
2) If the user agent string matches the user agent string of a browser, the respective feature vector is labeled as belonging to a human visitor (the vector's class label is set to 0).

Otherwise, if neither of the above is satisfied, we ignore the session since it belongs either to the malicious crawler or unknown visitor.

Note that in this experiment we also ignore all sessions that carry a user agent string of a known browser (indicating a human visitor) but access the robots.txt (operation performed only by crawlers), as they are likely to be malicious. The dataset labelling process is shown in Figure 4.2.

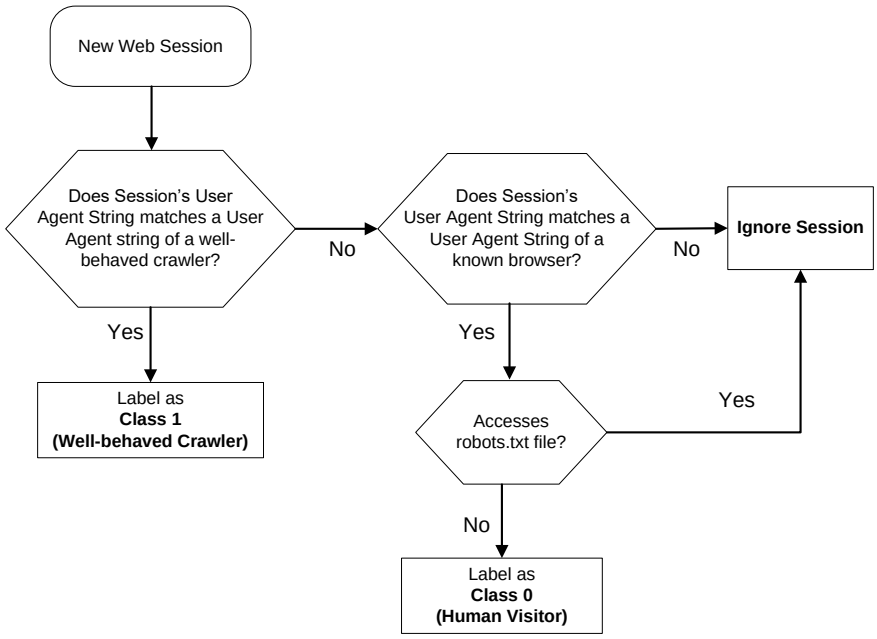


Figure 4.2 – Dataset labelling flow chart for Experiment 1

b) Dataset Labelling in Experiment 2

In the second experiment, we again utilize the table of known user agent fields in order to perform the dataset labelling. The labelling is performed as follows:

1) If the user agent string of a session matches the user agent string of a known browser or of a known well-behaved crawler, the respective feature vector is labeled with class label 0.
2) If the user agent string matches the user agent string of a known malicious crawler, or is not enlisted in the table of known user agents, the respective feature vector is labeled with class label 1.

As in the previous experiment, we assume that any session that carries a user agent string of a known browser but accesses the robots.txt is malicious, and consequently gets assigned class label 1. The dataset labelling process is shown in Figure 4.3.

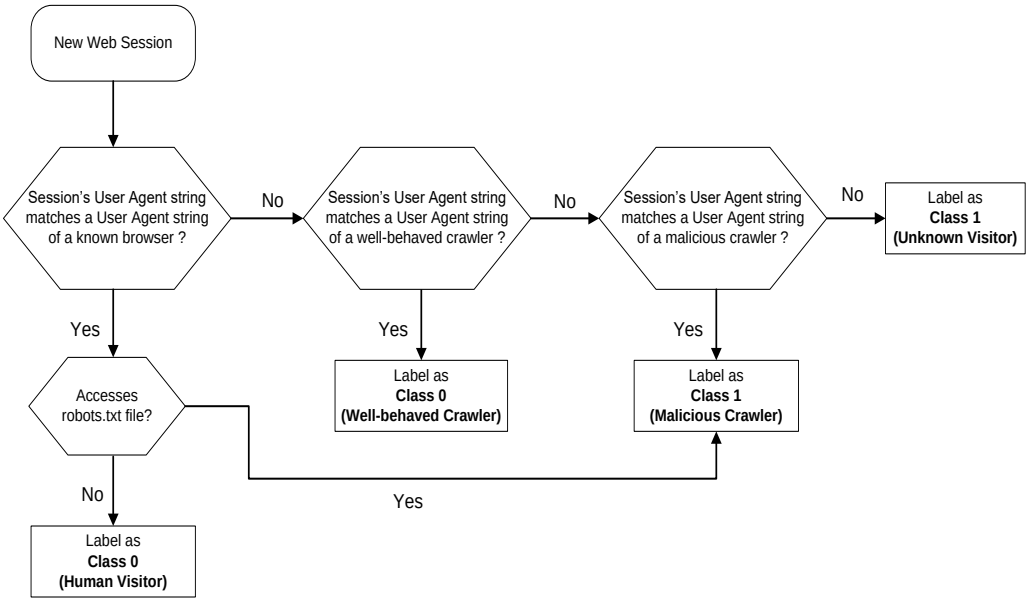


Figure 4.3 – Dataset labelling flow chart for Experiment 2

4.2.2 Experimental Design

In the previous section, we have described the process of dataset preparation (including session identification, feature extraction and vector labelling) as performed by our log analyzer. Assuming the correctness of the labelling process, the main goal of our work is to examine the classification accuracy of seven selected supervised learning algorithms when applied to the prepared dataset, as well as to evaluate the effectiveness of utilized web-session features in improving the algorithms' classification accuracy. In this section, we outline the details of the experimental study that followed the dataset preparation process.

4.2.2.1 Experimental Setup

In both experiments we have conducted two types of tests: one test involving only features 1-8 and the other involving all 10 features, as discussed in the previous section. Subsequently, the results of the two tests are compared in order to examine whether features 9 and 10 can improve the accuracy rate of the classification algorithms.

4.2.2.2 Web Server Access Logs

Note that we were unable to employ standard datasets for HTTP/web traffic mining, such as [158], [159] and/or [160]. These datasets are incomplete for the following reasons:

1. A unique identifier (e.g., IP address) for each source of HTTP/web requests is absent from all three datasets and/or
2. HTTP packet contents are omitted, e.g., user agent strings in [158] and [159], etc..

Therefore, in order to be able to group HTTP/web requests into a valid user web session, we had to rely on the following three (complete) datasets:

1. Web server access log files provided by York University's Computer Science and Engineering (CSE) department, i.e., www.cse.yorku.ca (referred to as *CSE* dataset in the remainder of the document).
2. Web server access log files provided by York University's main Internet domain, i.e. www.yorku.ca (referred to as *YORKU* dataset in the remainder of the document).
3. Web server access log files provided by SharpSchool [161] - web portal consisting of close to 1500 distinct K-12 school web sites across U.S. and Canada.

The purpose of performing our analysis on both, a smaller CSE and larger YORKU and SharpSchool web domains, was to evaluate whether our analysis can be generalized to larger, i.e., different web domains.

A typical entry in all three server access log files resembles the following line of data:

122.248.163.1 - - [09/Nov/2011:04:37:38 -0500] "GET /course_archive/2008-09/W/3421/test/testTwoPrep.html HTTP/1.1" 200 5645 Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)

Each file entry contains information in the following order from left to right: IP address of the source of the request (122.248.163.1), the timestamp of the request (09/Nov/2011:04:37:38 -0500), the HTTP method (GET), the file on the server that was requested (/course_archive/2008-09/W/3421/test/testTwoPrep.html), the response code from the server (200), the size of the data retrieved from the server (5645 bytes) and user agent field (Mozilla/5.0 compatible; Googlebot/2.1; +http://www.google.com/bot.html). The information provided in individual entries is employed by the log analyzer in all three stages of dataset preparation: session identification, feature-vector extraction, and dataset labelling.

The CSE log file stores detailed information about user web-based access into the domain www.cse.yorku.ca during an 2-week interval – Fall 2013. There are a total of over 7 million log entries in the file. Tables 4.2 and 4.3 list the number of sessions and class label distributions generated by the log analyzer from CSE log data for experiments 1 and 2, respectively. The YORKU log file stores detailed information about user web-based access into the domain www.yorku.ca during a 10 day interval – last 10 days of April 2012. There are a total of over 50 million log entries (14GB worth of data) in the file. Tables 4.4 and 4.5 list the number of sessions and class label distributions generated by the log analyzer from the YORKU log data for experiments 1 and 2, respectively. The SharpSchool log files store detailed information about user web-based access into various web sites as part of the SharpSchool web portal during a 7 day interval in June of 2013. There are a total of over 75 million log entries (18GB worth of

data) in the file. Tables 4.6 and 4.7 list the number of sessions and class label distributions generated by the log analyzer from the SharpSchool log data for experiments 1 and 2, respectively.

Table 4.2 – Class distributions in datasets generated from *CSE* logs and used in Experiment #1

	Session Samples
Total Number of Sessions	98793
Total # of Sessions with Class Label = 0	92149
Total # of Sessions with Class Label = 1	6644 (up-sampling = 13.8)

Table 4.3 – Class distributions in datasets generated from *CSE* logs and used in Experiment #2

	Session Samples
Total Number of Sessions	100151
Total # of Sessions with Class Label = 0	98793
Total # of Sessions with Class Label = 1	1358 (up-sampling = 7.8)

Table 4.4 – Class distributions in datasets generated from *YORKU* logs and used in Experiment #1

	Session Samples
Total Number of Sessions	716868
Total # of Sessions with Class Label = 0	707854
Total # of Sessions with Class Label = 1	9014 (up-sampling = 78.4)

Table 4.5 – Class distributions in datasets generated from *YORKU* logs and used in Experiment #2

	Session Samples
Total Number of Sessions	721173
Total # of Sessions with Class Label = 0	716868
Total # of Sessions with Class Label = 1	4305 (up-sampling = 167)

Table 4.6 – Class distributions in datasets generated from *SharpSchool* logs and used in Experiment #1

	Session Samples
Total Number of Sessions	776050
Total # of Sessions with Class Label = 0	650526
Total # of Sessions with Class Label = 1	125524 (up-sampling = 5.26)

Table 4.7 – Class distributions in datasets generated from *SharpSchool* logs and used in Experiment #2

	Session Samples
Total Number of Sessions	793845
Total # of Sessions with Class Label = 0	776050
Total # of Sessions with Class Label = 1	17795 (up-sampling = 43)

4.2.2.3 Classification Algorithms

The detection of web crawlers is evaluated with the following seven supervised classifiers: C4.5 [153], RIPPER [162], Naïve Bayesian [163], Bayesian Network [163], K-Nearest Neighbour [164], LibSVM (Support Vector Machines) [165] and Multilayer Perceptron (Neural Networks) [155]. The implementation of each algorithm is provided in the WEKA software package [166] and the parameter settings for each algorithm are shown in Table 4.8. This table also lists the basic methodology employed by each algorithm. All input vectors were normalized between 0 and 1, prior to being fed to the seven classifiers.

Table 4.8 – The classifier methodology and Weka parameter settings in our experiments

Classifiers	Methodology	Weka Parameters
C4.5 (weka.classifiers.trees.J48)	Decision Trees	binarySplits = false; confidenceFactor = 0.25; minNumOfObj = 2; numFolds = 3; seed = 1; subtreeRaising = true; unpruned = false; useLaplace = false
RIPPER (weka.classifiers.rules.JRip)	Decision Rules	checkErrorRate = true; folds = 3; minNo = 2.0; optimizations = 2; seed = 1; usePruning = true
Naïve Bayesian (weka.classifiers.bayes.NaiveBayes)	Bayesian Classification	useKernelEstimator = false; useSupervisedDiscretization = false
Bayesian Network (weka.classifiers.bayes.BayesNet)	Bayesian Classification	estimator = (SimpleEstimator -A 0.5); searchAlgorithm = (K2 -P 1 -S BAYES); useADTree = false
K-nearest Neighbours (weka.classifiers.lazy.IBk)	Lazy Learners	KNN = 1; crossValidate = false; distanceWeighting = (No distance weighting); meanSquared = false; nearestNeighbourSearchAlgorithm = (EuclideanDistance -R first-last); windowSize = 0
LibSVM (weka.classifiers.functions.LibSVM)	Support Vectors Classification	SVMType = C-SVC; cacheSize = 40.0; coef0 = 0.0; cost = 1.0; degree = 3; doNotReplaceMissingValues = false; eps = 0.001; gamma = 0.0; kernelType = (Radial Basis Function); loss = 0.1; normalize = false; nu = 0.5; probabilityEstimates = false; seed = 1; shrinking = true
Multilayer Perceptron (weka.classifiers.functions.MultilayerPerceptron)	Neural Networks	hiddenLayer = a; learningRate = 0.3; momentum = 0.2; nominalToBinaryFilter = true; normalizeAttributes = true; normalizeNumericClass = true; reset = true; seed = 0; trainingTime = 500; validationSetSize = 0; validationThreshold = 20

4.2.2.4 10-fold Cross Validation and Dataset Up-sampling

A simple evaluation of imbalanced datasets based on accuracy, i.e. the percentage of correct classifications, can be misleading. To illustrate this, assume a dataset with 100 cases out of which 90 cases belong to the majority class and 10 cases belong to the minority class. Then a classifier that classifies

every case as a majority class will have 90% accuracy, even though it failed to detect every single target of the minority class.

It is evident from Tables 4.2-4.7 that our original datasets suffered from serous class imbalance. In order to overcome this problem, and be able to conduct a more meaningful performance evaluation, we have applied the process of dataset up-sampling[7]. The up-sampling factor of the underrepresented class 1 samples is shown in Tables 4.2-4.7.

We also employed, as recommended in [155], stratified 10-fold cross-validation. In 10-fold cross-validation, the training samples are randomly partitioned into 10 mutually exclusive subsets or "folds," D_1, D_2, ... , D_{10}, each of approximately equal size. Training and testing is performed 10 times. In iteration i, partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model. That is, in the first iteration, subsets D_2, : : : , D_{10} collectively serve as the training set in order to obtain a first model, which is tested on D_1; the second iteration is trained on subsets D_1, D_3, : : : , D_{10} and tested on D_2; and so on.

4.2.2.5 Recall, Precision and F_1 score

In order to test the effectiveness of our classifiers, we have adopted the following three metrics: recall, precision, and the F_1 score [136]. The definitions of these three metrics are presented in Appendix B. The exact expressions for the three metrics, as used in Experiment 1 and 2, are presented in (4.3) to (4.9). Note that in both experiments, we evaluate the precision and recall scores for the underrepresented class only, i.e., Class 1. (Recall from section 4.2.1.4 that in Experiment 1 Class 1 comprises well-behaved crawlers,

[7] Up-sampling is a data mining pre-processing technique that balances the class distribution in the dataset by duplicating the training examples belonging to the class with fewer samples. The amount of up-sampling can be controlled by the number of training examples that are duplicated.

and in Experiment 2 it comprises malicious crawlers and unknown visitors.) This is justified by the fact that the main objective of our work is to be able to identify automated web-crawler visitors to a web site, and in particular web-crawlers that exhibit malicious behaviour and/or intent. However, we also calculate the precision (accuracy of classification) for both classes as well (presented in (4.8) and (4.9) as utilized for experiments 1 and 2, respectively).

It is also worth nothing from (4.7) that F_1 score summarizes the first two metrics into a single value, in a way that both metrics are given equal importance. Namely, the F_1 score penalizes a classifier that gives high recall but sacrifices precision and vice versa. For example, a classifier that classifies all examples as positive has perfect recall but very poor precision. Recall and precision should therefore be close to each other, otherwise the F_1 score yields a value closer to the smaller of the two. The definition of these metrics is given below:

Experiment 1: Precision (class 1 only) =

$$\frac{\#\text{ of well-behaved crawler sessions correctly classified}}{\#\text{ of truely well-behaved sessions}+\#\text{ of sessions incorrectly classified as well-behaved crawler sessions}} \quad (4.3)$$

Experiment 2: Precision (class 1 only) =

$$\frac{\#\text{ of (malicious or unknown) sessions correctly classified}}{\#\text{ of truely malicious/unknown sessions}+\#\text{ of sessions incorrectly classified as malicious/unknown}} \quad (4.4)$$

Experiment 1: Recall (class 1 only) =

$$\frac{\#\text{ of well-behaved crawler sessions correctly classified}}{\#\text{ of truely well-behaved sessions}+\#\text{ of actual well-behaved sessions not classified as well-behaved sessions}} \quad (4.5)$$

Experiment 2: Recall (class 1 only) =

$$\frac{\#\text{ of (malicious or unknown) sessions correctly classified}}{\#\text{ of truely malicious/unknown sessions}+\#\text{ of actual malicious or unknown sessions not classified as malicious/unknown sessions}} \quad (4.6)$$

$$F_1 = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \quad (4.7)$$

Experiment 1: Precision for both classes =

$$\frac{\#\text{ of (human or well}-\text{behaved) sessions correctly classified}}{\text{total \# of sessions}} \quad (4.8)$$

Experiment 2: Precision for both classes =

$$\frac{\#\text{ of (human, well}-\text{behaved, malicious or unknown) sessions correctly classified}}{\text{total \# of sessions}} \quad (4.9)$$

4.2.2.6 Entropy Metrics: Information Gain and Gain Ratio

In addition to ranking the classifiers, we also rank the most important dataset features by employing attribute/feature selection methods such as information gain and gain ratio. The ranking provides the purity test of the two proposed features. Basically, a higher ranking, by either of the two metrics, implies that a feature is more valuable to a classifier in separating sessions into classes.

a) Information Gain

The information gain for a feature A is a measure of the information needed to classify the new previously unseen sessions in the log file by partitioning the dataset on feature A. In a dataset composed of nominal features, the information gain is calculated as follows (from [155]). Let the partition D be the collection of all web sessions extracted from a web access log file. The expected information needed to classify a session in D is given by (from [155]):

$$\text{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \quad (4.10)$$

where m is the number of classes in D, p_i is the ratio (i.e., probability) that an arbitrary session in D belongs to class i and is estimated by the number of sessions classified with class i out of the total number of sessions in D. Info(D) (also known as the entropy of D) is the amount of information needed to identify the class label of a session in D.

Let us suppose that we partitioned sessions in D on some feature A having v distinct values, $\{a_1, a_2, \ldots, a_v\}$. Feature A can be used to split D into v partitions or subsets, $\{D_1, D_2, \ldots, D_v\}$, where D_j contains those sessions in D that have outcome a_j of A. Ideally, we would like this partitioning to be a collection of sessions that map to a unique class in D (i.e., we would like for each partition to be pure). However, it is quite unlikely that the partitions will be pure (e.g., a partition may contain a collection of sessions from different classes rather than from a single class). The amount of information we still need (after the partitioning based on feature A) in order to arrive at an exact classification is measured by (from [155])

$$\mathrm{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \mathrm{Info}(D_j) \tag{4.11}$$

where $|D_j|$ is the number of sessions in partition D_j, $|D|$ is the total number of sessions in the set D and $\mathrm{Info}(D_j)$ is defined in Expression 4.10. $\mathrm{Info}_A(D)$ is the expected information required to classify a session from D based on the partitioning by feature A. Clearly, different partitions of A will result in different values of $\mathrm{Info}_A(D)$. The partition that achieves the minimum entropy is the most optimal, as it implies the best possible purity of partitions. Information gain is then defined as the difference between the original information requirement (i.e., based on just the proportion of classes), that is Expression 4.10, and the new requirement (i.e., obtained after optimal partitioning on feature A), that is Expression 4.11. The information gain formula is, (from [155])

$$\mathrm{Gain}(A) = \mathrm{Info}(D) - \mathrm{Info}_A(D) \tag{4.12}$$

In other words, Gain(A) is the expected reduction in the information requirement caused by knowing the value of feature A. Or, in other words, Gain(A) tell us how much would be gained by branching on A. The feature A with the highest information gain, (Gain(A)), is chosen/ranked as the most pure feature.

Note that in our dataset, all 10 features are numerical. Hence, partitioning the sessions in D on a feature A with v distinct values is a non-trivial task. Namely, as suggested in [155], the procedure for finding the optimal (i.e., best/smallest) $Info_A(D)$ requires that the "best" split-point for a numerical feature A be found. We can accomplish this by first sorting the values of a numerical feature A in increasing order. Typically, the midpoint between each pair of adjacent increasingly ordered values is considered as a possible split-point. E.g., given t values of A, t−1 possible splits are evaluated. For example, the midpoint between the values a_i and a_{i+1} of A is $\frac{a_i + a_{i+1}}{2}$**.** Next, for each possible split-point for A, we evaluate $Info_A(D)$, where the number of partitions is two, that is v = 2 (or j = 1..2) in (4.11). D_1 is the set of sessions in D satisfying A ≤ *split point*, and D_2 is the set of sessions in D satisfying A > *split point*. Once the minimal $Info_A(D)$ is calculated based on one of the possible split points, the gain for that feature can be calculated using the (4.12).

b) Gain Ratio

The information gain measure is biased toward tests with many outcomes. That is, the gain will be larger for features having a large number of distinct values. Therefore, in our analysis, we additionally rank features with an extension of information gain metric known as gain ratio, which attempts to overcome the bias of information gain metric. It applies a kind of normalization to information gain using a "split information" value defined analogously with Info(D) as (from [155]):

$$\text{SplitInfo}_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right) \quad (4.13)$$

This value represents the potential information generated by splitting the training data set, D, into v partitions, corresponding to the v outcomes of a test on feature A. In our dataset, v = 2. Then, the gain ratio is defined as (from [155])

$$\text{GainRatio(A)} = \frac{\text{Gain(A)}}{\text{SplitInfo(A)}} \tag{4.14}$$

4.2.2.7 Significance of the Difference Test (t-test)

In addition, the effectiveness of the new attributes in classifying visitor's sessions is further evaluated by applying the significance of difference test or t-test. Namely, after we generate the classification results with all seven classifiers, we separate the sessions in 2 groups, true negatives and true positives. The sessions are grouped into true negatives if both the log analyzer and classification algorithms label the session with class label 0. On the other hand, the sessions are grouped into true positives if both the log analyzer and classification algorithms label the session with class label 1.

Next, we calculate means and variances for each of 10 features in both groups of sessions and perform the significance of the difference test (alternatively called the t-test) with 95% confidence interval. The calculation of the significance of the difference test is based on (4.15) (from [167]):

$$t = \frac{|\text{mean}_1(f) - \text{mean}_2(f)|}{\sqrt{\frac{\text{Var}_1(f)}{n_1} + \frac{\text{Var}_2(f)}{n_2}}} \tag{4.15}$$

Note that we cannot assume the normality of the feature values distributions. However, as central limit theorem proves, the distribution of means of two group means, $\text{mean}_1(f)$ and $\text{mean}_2(f)$, in repeated

sampling, converges to a normal distribution, irrespective of the distribution of f in the population. Typically, the rule of thumb is that the sample size greater than 20 is sufficient to satisfy this condition.

In (4.15) $mean_1$ and $mean_2$ are means of the feature values in two groups, Var_1 and Var_2 are the variances of the feature values in two groups, and n_1 and n_2 are the number of elements in two groups. The degrees of freedom value used in the t-test is: (from [167])

$$\text{degrees of freedom} = \frac{\left(\frac{\text{Var}_1(\text{f})^2}{\text{n}_1} + \frac{\text{Var}_2(\text{f})^2}{\text{n}_2}\right)^2}{\frac{\left(\frac{(\text{var}_1(\text{f}))^2}{\text{n}_1}\right)^2}{\text{n}_1 - 1} + \frac{\left(\frac{(\text{var}_2(\text{f}))^2}{\text{n}_2}\right)^2}{\text{n}_2 - 1}} \tag{4.16}$$

Note that (4.16) is used because we assume that $Var_1 \neq Var_2$ and $n_1 \neq n_2$. The mean values between the two groups of sessions will be significantly (statistically) different with 95% confidence if the t score (4.15) is greater than 1.96 and degrees of freedom (4.16) are greater than or equal to 200. In general, for a fixed confidence percentage, there exists an inverse relationship between the value of the degrees of freedom calculation and the t score threshold. The exact t score threshold based on the calculated degrees of freedom can be determined by consulting the t-table found in [168].

4.2.3 Classification Results

In this section we present and discuss the results of our two experiments. Namely, in the next two subsections, we give a detailed summary of the results of Experiments 1 and 2, respectively. In the third subsections, we derive additional observation and conclusions from the presented results.

4.2.3.1 Experiment 1

The motivation for this experiment was to test the classification accuracy of the seven data-mining algorithms, as well as to evaluate whether features 9 and 10, can improve the accuracy in classifying sessions as either belonging to a human user or a well-behaved web crawler.

a) Recall, Precision and F_1 Score (class 1 only)

The comparisons of the recall, precision and F_1 scores for the CSE dataset with features 1-8 and with all 10 features are shown in Figures 4.4, 4.5 and 4.6, respectively. The results generated from the YORKU and SharpSchool data for the same three metrics are shown in Figures 4.7-4.12. As explained in 4.1.2.5, we only show the precision and recall scores for the underrepresented class, i.e., Class 1. It is evident from the presented graphs that the use of all 10 features generally improves both, the recall and precision scores, for all classifiers except for the Bayesian Network in CSE and SharpSchool data (there is no improvement in either recall or precision), Naive Bayesian in YORKU data and Neural Network in Sharpschool data where the precision is slightly lower with 10 features. Similarly, in three datasets, the use of 10 features results in improved F_1 score (between 0.5% up to nearly 10%) in six out of seven examined algorithms.

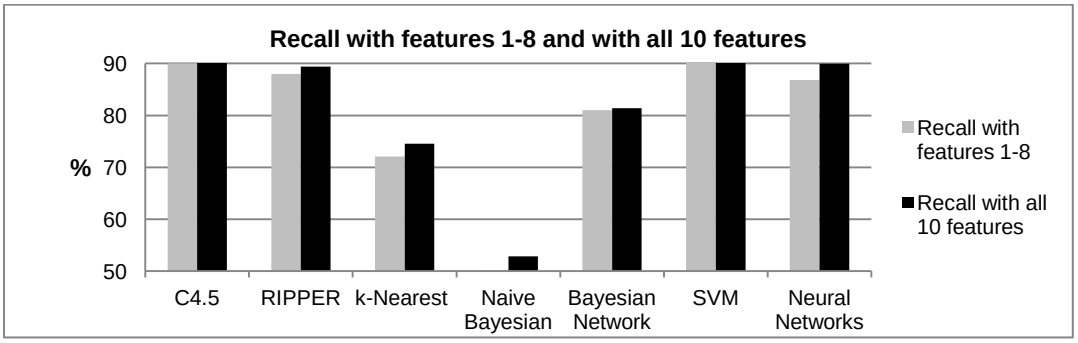


Figure 4.4 – Recall scores for various classifiers trained on the *CSE* datasets that contain only features 1-8 and all 10 features

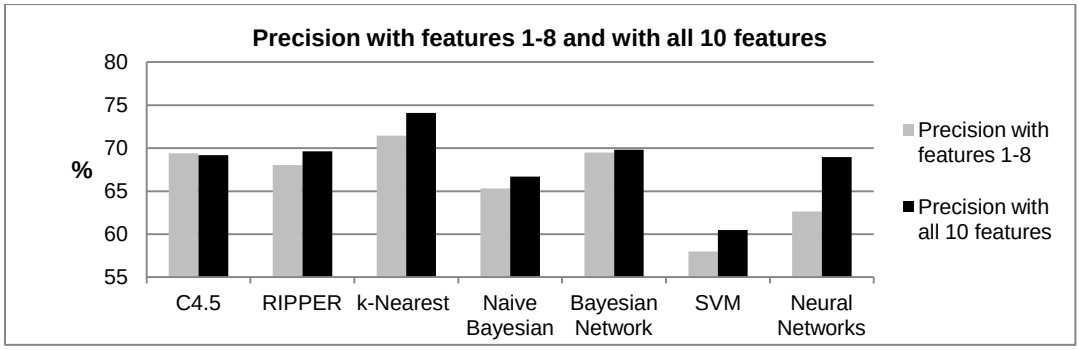


Figure 4.5 – Precision scores for various classifiers trained on the *CSE* datasets that contain only features 1-8 and all 10 features

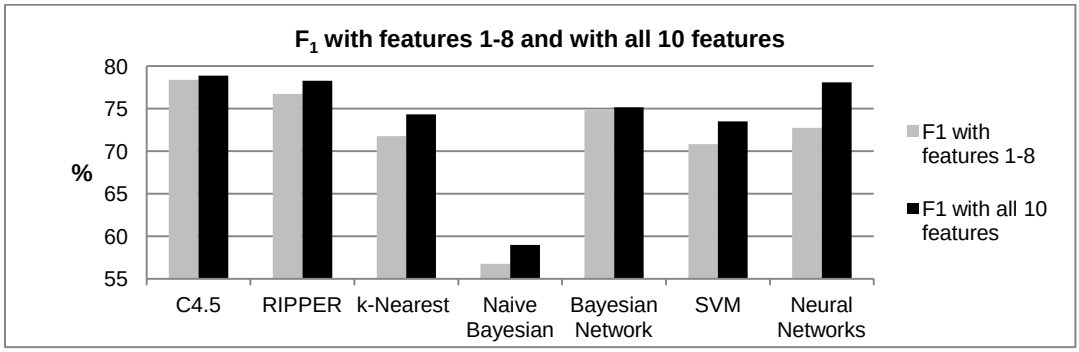


Figure 4.6 – F_1 scores for various classifiers trained on the *CSE* datasets that contain only features 1-8 and all 10 features

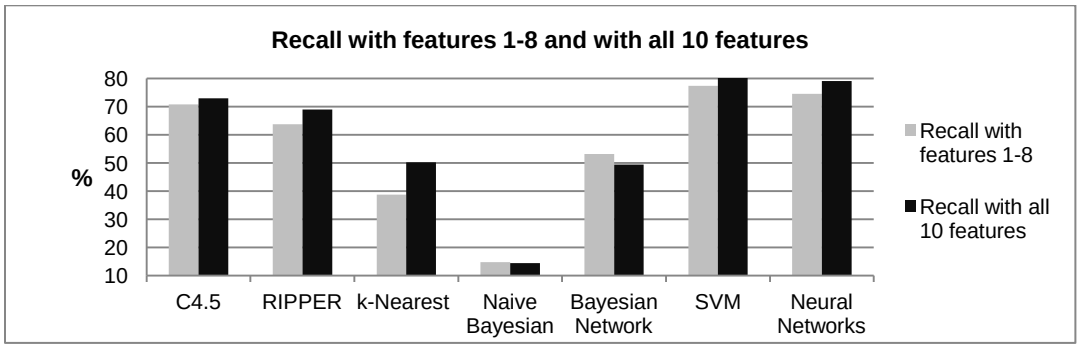


Figure 4.7 – Recall scores for various classifiers trained on the *YORKU* datasets that contain only features 1-8 and all 10 features

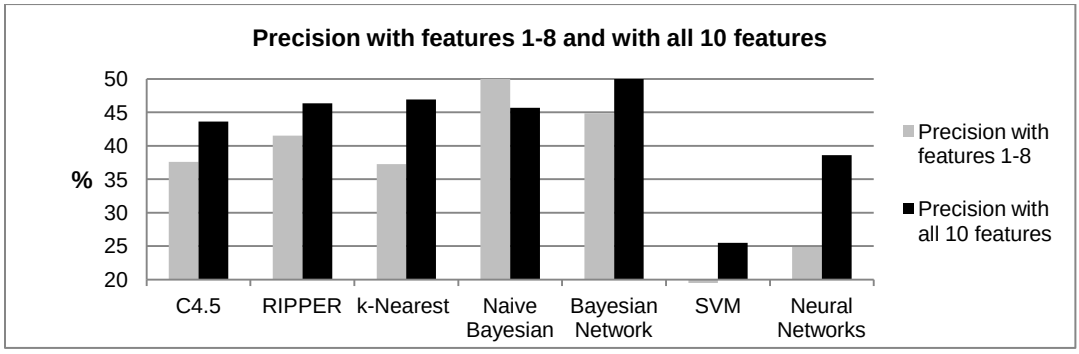


Figure 4.8 – Precision scores for various classifiers trained on the *YORKU* datasets that contain only features 1-8 and all 10 features

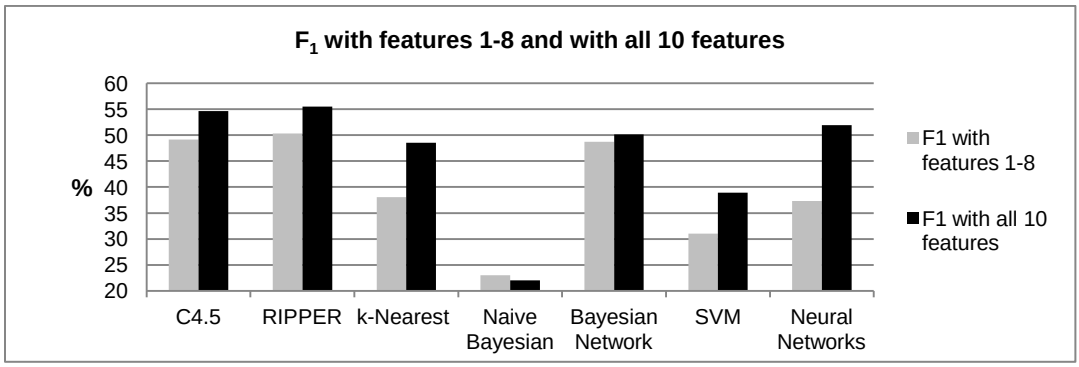


Figure 4.9 – F_1 scores for various classifiers trained on the *YORKU* datasets that contain only features 1-8 and all 10 features

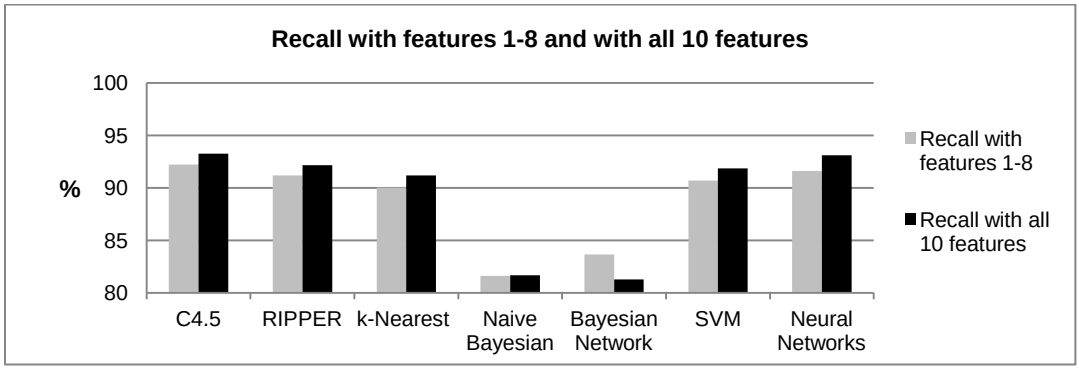


Figure 4.10 – Recall scores for various classifiers trained on the *Sharpschool* datasets that contain only features 1-8 and all 10 features

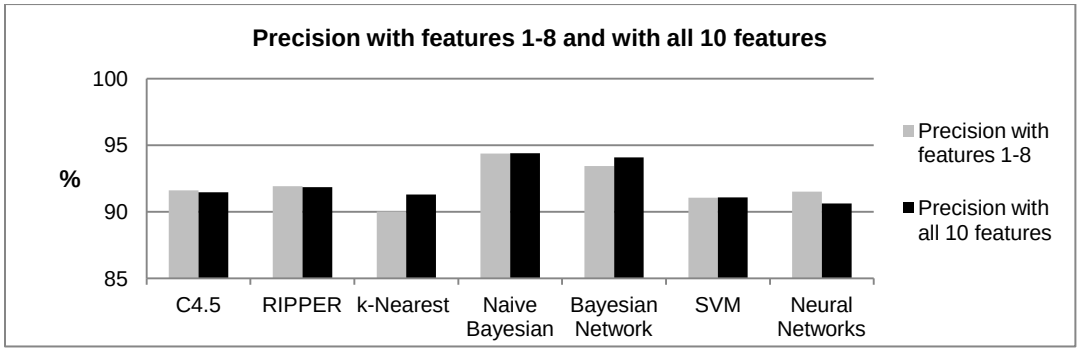


Figure 4.11 – Precision scores for various classifiers trained on the *Sharpschool* datasets that contain only features 1-8 and all 10 features

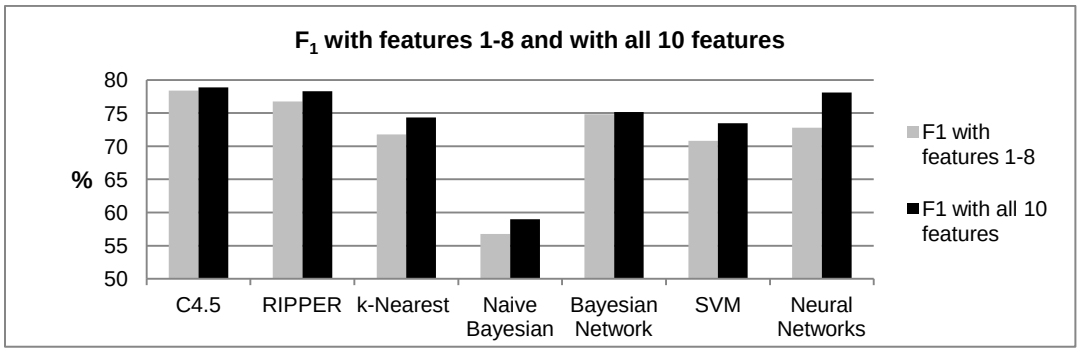


Figure 4.12 – F_1 scores for various classifiers trained on the *Sharpschool* datasets that contain only features 1-8 and all 10 features

b) Classification Precision for Both Classes (i.e. Classification Accuracy)

The Figure 4.13 shows the comparison of the classification accuracy rate (Equation 4.8) when the seven classification algorithms are trained on both 8- and 10- feature version of the entire CSE dataset. The Figure 4.14 and 4.15 shows the results generated from the YORKU and SharpSchool dataset for the same metric, respectively. As expected, due to class imbalance, the classification accuracy is very high (at 93% or above) for all seven classification algorithms in CSE and SharpSchool results. The class imbalance is even higher in the case of YORKU-dataset where the classification accuracy approaches 100%. However,

as can be observed, there is a slight improvement in accuracy rate when all 10 features are used for six out of seven algorithms in both datasets.

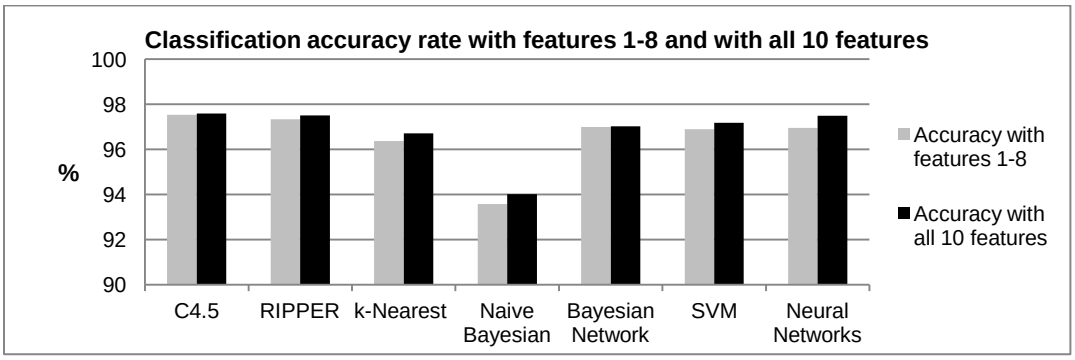


Figure 4.13 – Classification accuracy rates for various classifiers trained on the *CSE* datasets that contain only features 1-8 and all 10 features

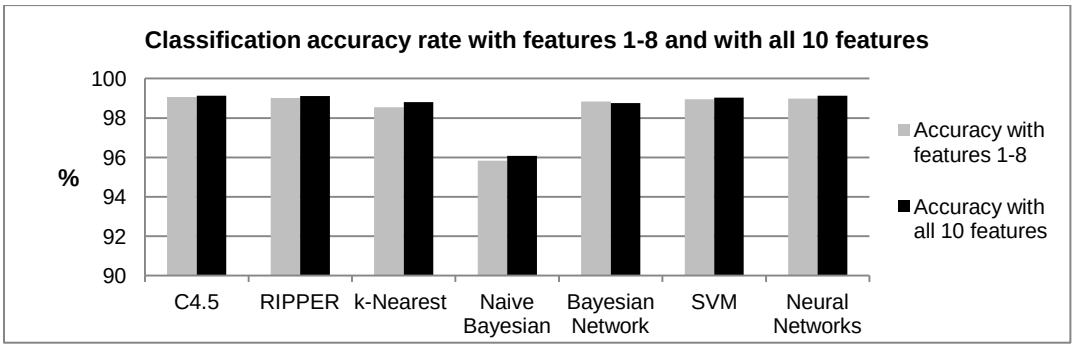


Figure 4.14 – Classification accuracy rates for various classifiers trained on the *YORKU* datasets that contain only features 1-8 and all 10 features

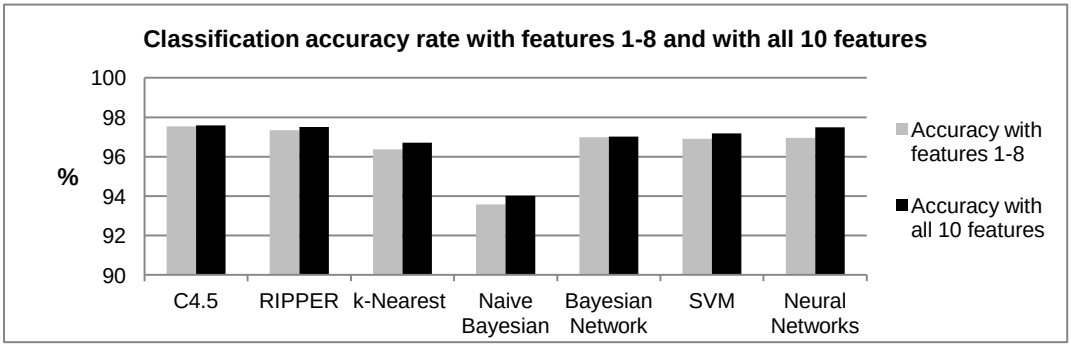


Figure 4.15 – Classification accuracy rates for various classifiers trained on the *SharpSchool* datasets containing only features 1-8 and all 10 features

c) Entropy-based Feature Rankings

Tables 4.9, 4.10 and 4.11 show the rankings of all 10 features in terms of the information gain and gain ratio metrics sorted in the descending order for the CSE, YORKU and SharpSchool datasets, respectfully. The ranking scores derived from the 3 datasets are almost identical. As expected, the percentage of unassigned referrers is one of the most valuable features for classifying sessions. Recall, the referrer parameter is typically only assigned by a browser of the user visiting the web site and is left blank if the visitor is a web crawler.

Also, note that both the information gain and gain ratio metrics similarly rank the 10 features in both datasets. The only major/noticeable difference between the two sets of rankings is that the percentage of HEAD requests feature is ranked much higher by the gain ratio than by the information gain metric. As explained in section 4.2.2.6, the gain ratio prefers features that have more unbalanced split and that information gain prefers features that have more balanced split among the two classes. This implies that the percentage of HEAD requests has a very unbalanced split among the two classes.

Table 4.9 – Attribute rankings in terms of Information Gain and Gain Ratio metrics (ordered top down from best to worst) for CSE dataset

Information Gain	Gain Ratio
1. % of Unassigned Referrers	1. Popularity Index
2. Popularity Index	2. % of Unassigned Referrers
3. % of Sequential HTTP requests	3. % of HEAD requests
4. Standard Deviation of Page Depth	**4. % of Sequential requests**
5. Click Number	**5. Standard Deviation of Page Depth**
6. HTML to Image ratio	6. Click Number
7. % of PDF documents	7. % of PDF documents
8. % of Error requests	8. % of Error requests
9. # of Server to Client Bytes	9. HTML to Image ratio
10. % of HEAD requests	10. # of Server to Client Bytes

Table 4.10 – Attribute rankings in terms of Information Gain and Gain Ratio metrics (ordered top down from best to worst) for YORKU dataset

Information Gain	Gain Ratio
1. Popularity Index	1. % of Unassigned Referrers
2. % of Unassigned Referrers	2. % of HEAD requests
3. % of Sequential requests	3. Popularity Index
4. Standard Deviation of Page Depth	**4. % of Sequential requests**
5. HTML to Image ratio	**5. Standard Deviation of Page Depth**
6. % of Error requests	6. % of Error requests
7. Click Number	7. Click Number
8. # of Server to Client Bytes	8. % of PDF documents
9. % of PDF documents	9. HTML to Image ratio
10. % of HEAD requests	10. # of Server to Client Bytes

Table 4.11 – Attribute rankings in terms of Information Gain and Gain Ratio metrics (ordered top down from best to worst) for SharpSchool dataset

Information Gain	Gain Ratio
1. % of Unassigned Referrers	1. % of Unassigned Referrers
2. Popularity Index	2. HTML to Image ratio
3. Click Number	3. Popularity Index
4. % of Sequential requests	4. Click Number
5. Standard Deviation of Page Depth	5. % of PDF documents
6. # of Server to Client Bytes	**6. Standard Deviation of Page Depth**
7. % of Error requests	**7. % of Sequential HTTP requests**
8. HTML to Image ratio	8. # of Server to Client Bytes
9. % of PDF documents	9. % of Error requests
10. % of HEAD requests	10. % of HEAD requests

It is also interesting to note that the two new features introduced in this study are rather highly positioned in all of the presented rankings, ultimately proving that these attribute - % of Sequential Requests and Standard Deviation of Page Depth - can be very helpful in determining whether the session belongs to a human user or to a benign web-crawler.

d) Significance of the Difference Test

The significance of the differences of values between true positive and true negative sessions for all 10 features can be confirmed by applying the significant difference of the mean test (Equation 4.15 or the t-test) [168]. As can be observed in Table 4.12, the mean values of features 9 and 10 are significantly different between true positive and true negative sessions extracted from the CSE web log files in the case of all seven classifiers (bold-lettered values indicate significant difference with 95% confidence). The mean values of the other features (except features 5 and 7) are also significantly different between the two groups of sessions.

The significance of the differences of values between true positive and true negative sessions extracted from the YORKU web log data are shown in Table 4.13. As can be observed, the results are highly similar with t-scores being slightly smaller in Table 4.13 than in the Table 4.12. The generally smaller t-scores with YORKU data can be probably attributed to the fact that visitors to the much larger web domain are generally more diverse than the visitors that visit a small sub-domain. Nevertheless, all features (except for % of PDF requests) that are significantly different between true positive and true negative sessions in Table 4.12 are also significantly different between true positive and true negative sessions in Table 4.13. The SharpSchool dataset generate significance of the differences results, presented in Table 4.14, are similar to CSE and YORKU results.

Table 4.12 – T-scores for the difference test on mean values of all 10 features between true positive and true negative sessions extracted from *CSE* web log data (note: bold-lettered numbers indicate the significant difference with 95% confidence)

Classifiers	1 t-score	2 t-score	3 t-score	4 t-score	5 t-score	6 t-score	7 t-score	8 t-score	9 t-score	10 t-score
C4.5	**11.68**	**4.06**	**4.46**	**4.71**	0.88	**25.17**	**2.09**	**77.76**	**23.05**	**16.88**
RIPPER	**11.61**	**4.00**	**4.21**	**4.69**	1.01	**25.65**	1.91	**79.55**	**22.81**	**16.75**
k-Nearest Neighbour	**11.51**	**4.06**	**4.24**	**4.53**	1.12	**24.41**	**2.37**	**71.62**	**22.45**	**16.11**
Naive Bayesian	**11.97**	**5.22**	**4.47**	**5.35**	1.56	**26.46**	1.96	**78.29**	**21.95**	**15.79**
Bayesian Network	**11.52**	**4.07**	**4.16**	**5.01**	0.98	**25.69**	1.57	**80.79**	**23.4**	**17.19**
SVM	**12.69**	**4.08**	**4.14**	**4.69**	0.7	**25.41**	1.23	**79.4**	**23.08**	**17.08**
Neural Network	**11.37**	**4.00**	**4.49**	**4.58**	0.75	**25.26**	2.00	**78.2**	**23.24**	**17.48**

Table 4.13 – T-scores for the difference test on mean values of all 10 features between true positive and true negative sessions extracted from *YORKU* web log data (note: bold-lettered numbers indicate the significant difference with 95% confidence)

Classifiers	1 t-score	2 t-score	3 t-score	4 t-score	5 t-score	6 t-score	7 t-score	8 t-score	9 t-score	10 t-score
C4.5	**5.34**	**3.71**	1.55	**2.44**	0.71	**22.69**	1.7	**37.25**	**3.72**	**5.4**
RIPPER	**5.09**	**4.41**	1.78	**2.55**	0.78	**21.43**	1.52	**37.99**	**3.82**	**5.31**
k-Nearest Neighbour	**4.35**	**4.42**	1.05	**2.75**	0.71	**22.03**	1.51	**35.29**	**3.43**	**5.34**
Naive Bayesian	**4.13**	**4.59**	1.51	**3.35**	0.89	**23.58**	1.96	**35.54**	**2.46**	**5.15**
Bayesian Network	**5.68**	**4.5**	1.29	**2.66**	0.81	**22.88**	1.57	**41.21**	**3.84**	**5.36**
SVM	**6.04**	**3.7**	1.14	**1.99**	0.78	**22.56**	**2.13**	**51.27**	**3.2**	**5.7**
Neural Network	**5.51**	**3.52**	1.07	**2.3**	0.69	**23.47**	1.86	**41.87**	**3.93**	**5.52**

Table 4.14 – T-scores for the difference test on mean values of all 10 features between true positive and true negative sessions extracted from *SharpSchool* web log data (note: bold-lettered numbers indicate the significant difference with 95% confidence)

Classifiers	1 t-score	2 t-score	3 t-score	4 t-score	5 t-score	6 t-score	7 t-score	8 t-score	9 t-score	10 t-score
C4.5	**104.32**	**3.3**	1.33	**5.29**	0.3	**15.6**	**105.74**	**43.81**	**12.47**	**107.88**
RIPPER	**110.24**	**3.31**	1.29	**5.3**	0.29	**15.74**	**117.6**	**44.1**	**12.44**	**109.67**
k-Nearest Neighbour	**94.62**	**3.25**	1.31	**5.28**	0.31	**15.53**	**96.52**	**43.39**	**12.41**	**106.52**
Naive Bayesian	**125.9**	**3.39**	1.51	**6.19**	0.52	**15.57**	**140.79**	**46.39**	**13.63**	**112.56**
Bayesian Network	**126.8**	**3.37**	1.377	**5.78**	0.44	**15.74**	**123.3**	**46.24**	**13.35**	**113.55**
SVM	**115.08**	**3.12**	1.08	**4.53**	0.25	**15.08**	**108.93**	**44**	**9.78**	**111.65**
Neural Network	**120.07**	**3.27**	1.31	**5.3**	0.29	**15.69**	**131.98**	**44.02**	**12.22**	**110.04**

4.2.3.2 Experiment 2

In this section, we present the results derived in the second experiment of our supervised learning study. The goal of the experiment was to test the classification accuracy of the seven data-mining algorithms, as well as to evaluate whether features 9 and 10 can improve the accuracy in classifying sessions as belonging to malicious web crawlers and unknown visitors.

a) Recall, Precision and F_1 Score (Class 1 Only)

The comparisons of the recall, precision and F_1 scores for the CSE dataset are shown in Figures 4.16, 4.17 and 4.18, respectively. The results generated from the YORKU and SharpSchool datasets for the same three metrics are shown in Figures 4.19-4.24. Again, we only show the precision and recall scores for the underrepresented class, i.e., Class 1. In both datasets, it is evident from the presented results that in five out of seven classifiers the use of all 10 features results in noticeably higher recall, precision, and F_1 scores. Although, note that precision scores are quite low (i.e., the false positive rate is fairly high for class 1) implying that the supervised learning cannot accurately detect the difference in the browsing behaviour

between sessions belonging to malicious and unknown visitors and sessions belonging to human visitors and well-behaved crawlers.

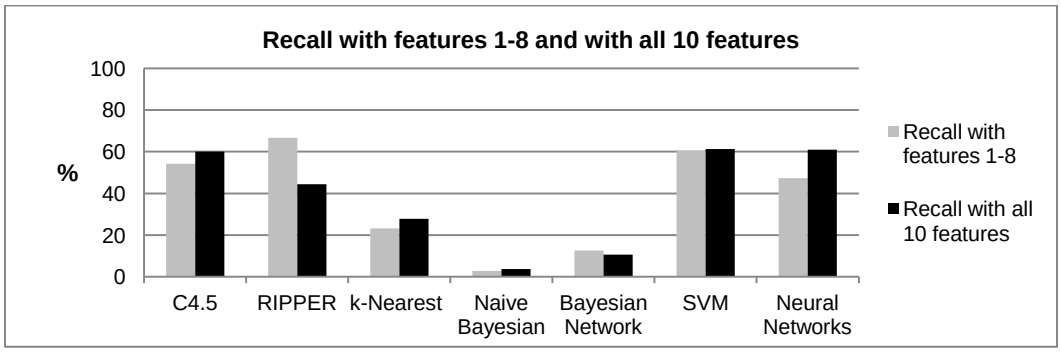


Figure 4.16 – Recall scores for various classifiers trained on the *CSE* datasets that contain only features 1-8 and all 10 features

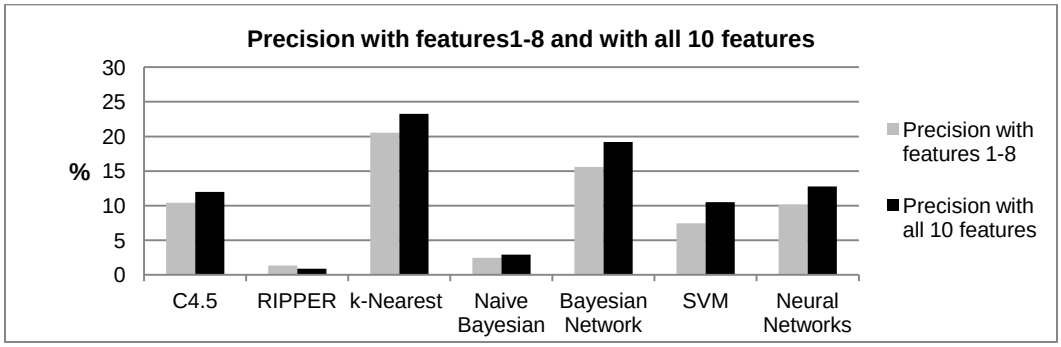


Figure 4.17 – Precision scores for various classifiers trained on the *CSE* datasets that contain only features 1-8 and all 10 features

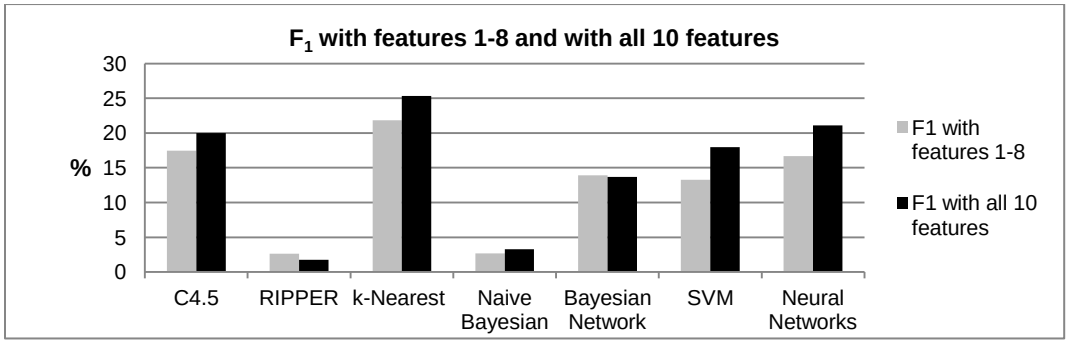


Figure 4.18 – F_1 scores for various classifiers trained on the *CSE* datasets that contain only features 1-8 and all 10 features

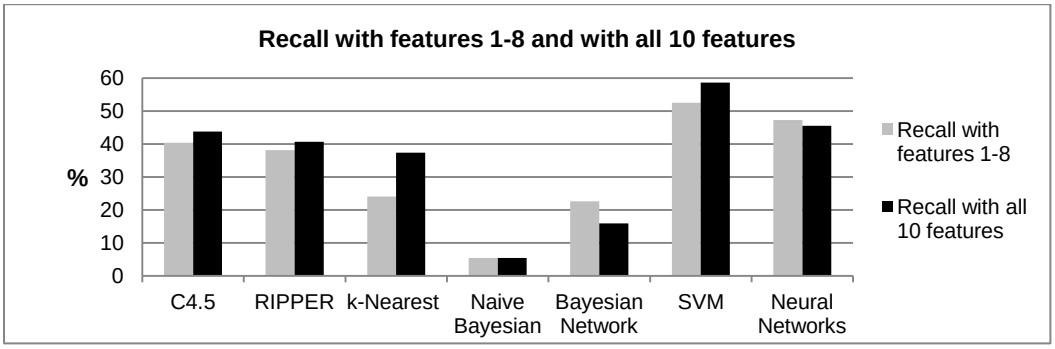


Figure 4.19 – Recall scores for various classifiers trained on the *YORKU* datasets that contain only features 1-8 and all 10 features

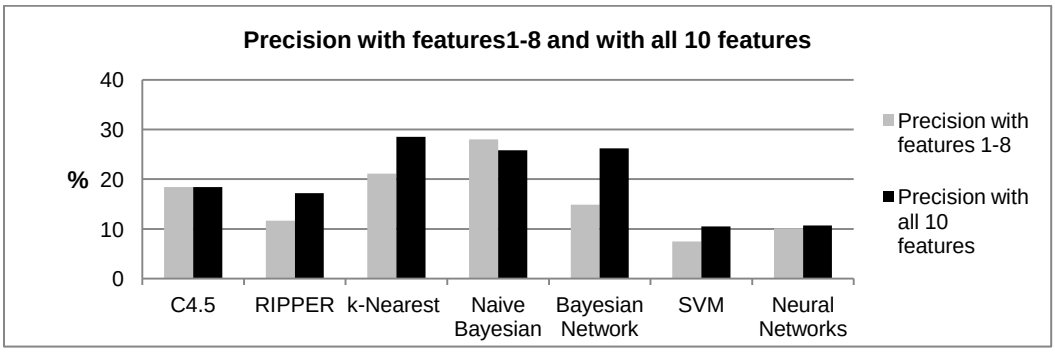


Figure 4.20 – Precision scores for various classifiers trained on the *YORKU* datasets that contain only features 1-8 and all 10 features

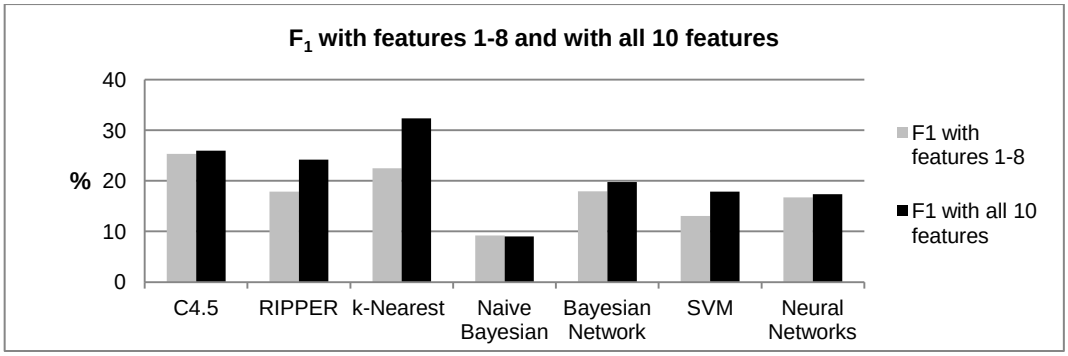


Figure 4.21 – F_1 scores for various classifiers trained on the *YORKU* datasets that contain only features 1-8 and all 10 features

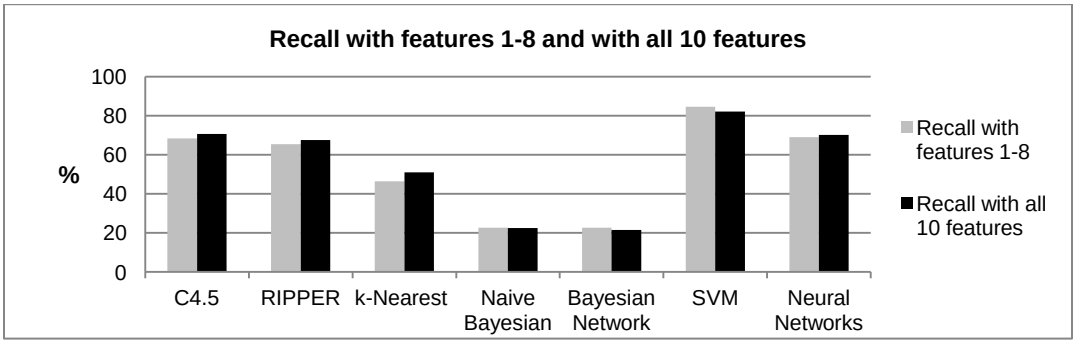


Figure 4.22 – Recall scores for various classifiers trained on the *SharpSchool* datasets that contain only features 1-8 and all 10 features

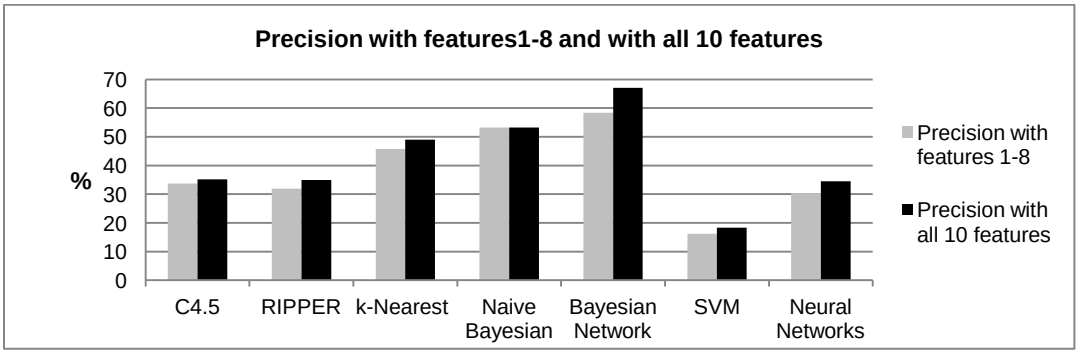


Figure 4.23 – Precision scores for various classifiers trained on the *SharpSchool* datasets that contain only features 1-8 and all 10 features

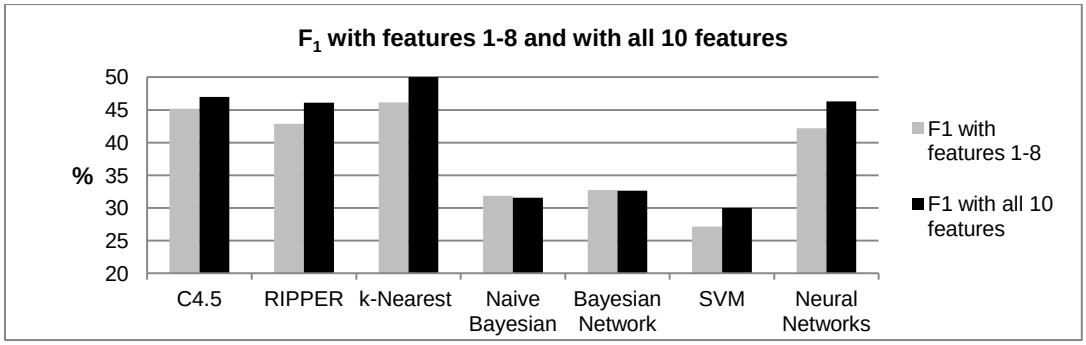


Figure 4.24 – F_1 scores for various classifiers trained on the SharpSchool datasets that contain only features 1-8 and all 10 features

b) Classification Precision for Both Classes (i.e. Classification Accuracy)

The Figure 4.25 shows the comparison of the classification accuracy rate (Equation 4.9) when the seven classification algorithms are trained on the entire CSE dataset. The Figure 4.26 and 4.27 shows the results generated from the YORKU and SharpSchool dataset for the same metric, respectively. Again, as expected, due to class imbalance, the classification accuracy is very high (at 95% or above) for all seven classification algorithms in both datasets. However, as can be observed, there is a slight improvement in accuracy rate when all 10 features are used in five out of seven examined algorithms.

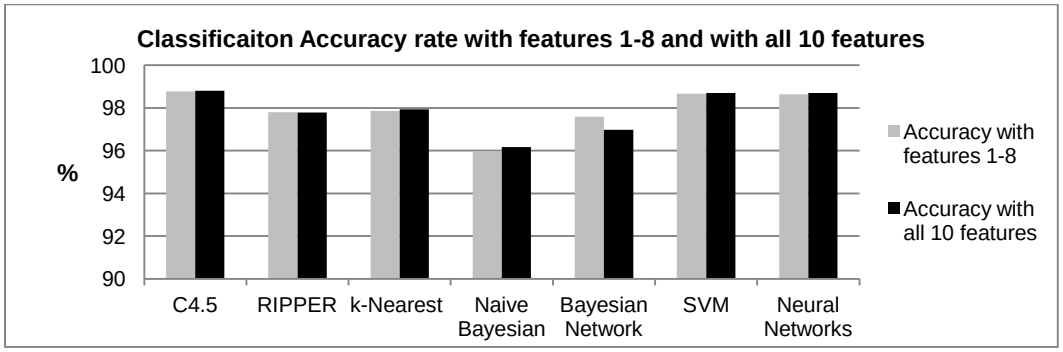


Figure 4.25 – Classification accuracy rates for various classifiers trained on the CSE datasets containing only features 1-8 and all 10 features

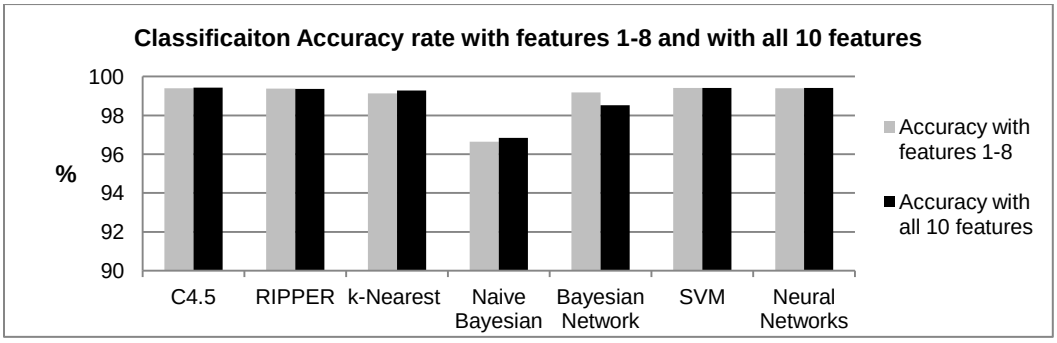


Figure 4.26 – Classification accuracy rates for various classifiers trained on the *YORKU* datasets containing only features 1-8 and all 10 features

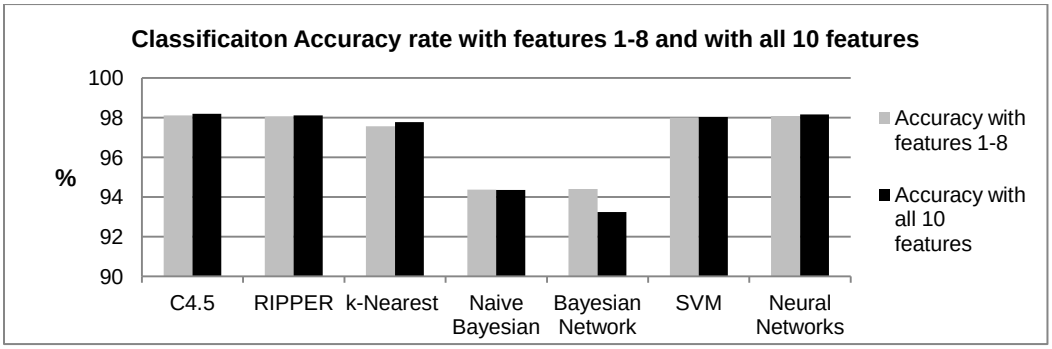


Figure 4.27 – Classification accuracy rates for various classifiers trained on the *SharpSchool* datasets containing only features 1-8 and all 10 features

c) Entropy-based Attribute Rankings

Lastly, Tables 4.15, 4.16 and 4.17 show the rankings of 10 features in terms of the information gain and gain ratio metrics for the sessions extracted from CSE, YORKU and SharpSchool datasets, respectfully. Again, the ranking scores derived from all three datasets follow the similar trend. Also, the ranking scores are similar to those from the first experiment.

The major/noticeable difference between the three sets of rankings is that the two features, percentage of HEAD requests and percentage of erroneous requests, are ranked much higher by the gain ratio than by the information gain metric. As explained in section 4.2.2.6, this implies that the percentage of HEAD requests and the percentage of erroneous requests have a very unbalanced split among the two classes.

Additionally, the two new metrics, standard deviation of page depth and percentage of sequential HTTP requests, do not perform as well in comparison to the first experiment. Note that given the low precision of all supervised learning algorithms in this particular case (malicious vs. non malicious visitors), these ranking should be taken with 'some reservations' (i.e., we cannot expect them to be 100% reliable).

Table 4.15 – Attribute rankings in terms of Information Gain and Gain Ratio metrics (ordered top down from best to worst) for CSE dataset

Information Gain	Gain Ratio
1. % of Unassigned Referrers	1. % of HEAD requests
2. Popularity Index	2. Click Number
3. Click Number	3. % of Unassigned Referrers
4. % of Sequential HTTP Requests	4. Popularity Index
5. # of Server to Client Bytes	5. % of Error requests
6. HTML to Image ratio	6. % of PDF documents
7. Standard Deviation of Page Depth	**7. % of Sequential HTTP requests**
8. % of PDF documents	8. # of Server to Client Bytes
9. % of HEAD requests	9. HTML to Image ratio
10. % of Error requests	**10. Standard Deviation of Page Depth**

Table 4.16 – Attribute rankings in terms of Information Gain and Gain Ratio metrics (ordered top down from best to worst) for YORKU dataset

Information Gain	Gain Ratio
1. Popularity Index	1. % of HEAD requests
2. % of Unassigned Referrers	2. Popularity Index
3. % of Sequential HTTP Requests	3. % of Unassigned Referrers
4. Click Number	4. % of Error requests
5. HTML to Image ratio	5. % of PDF documents
6. # of Server to Client Bytes	6. Click Number
7. Standard Deviation of Page Depth	**7. % of Sequential HTTP requests**
8. % of PDF documents	8. # of Server to Client Bytes
9. % of HEAD requests	9. HTML to Image ratio
10. % of Error requests	**10. Standard Deviation of Page Depth**

Table 4.17 – Attribute rankings in terms of Information Gain and Gain Ratio metrics (ordered top down from best to worst) for SharpSchool dataset

Information Gain	Gain Ratio
1. Popularity Index	1. % of HEAD requests
1. % of Unassigned Referrers	2. % of Error requests
3. Click Number	3. Popularity Index
4. % of HEAD requests	4. % of Unassigned Referrers
5. % of Error requests	5. HTML to Image ratio
6. Standard Deviation of Page Depth	6. Click Number
7. # of Server to Client Bytes	**7. Standard Deviation of Page Depth**
8. % of Sequential requests	8. # of Server to Client Bytes
9. HTML to Image ratio	**9. % of Sequential HTTP requests**
10. % of PDF documents	10. % of PDF documents

d) Significance of the Difference Test

The significance of the differences of results between true positive and true negative sessions extracted from the CSE web log data are shown in Table 4.18. As can be observed, the mean values of features 9 and 10 are significantly different between true positive and true negative sessions in the case of four out of seven classifiers (bold-lettered values indicate significant difference with 95% confidence). However, the most significantly different features between the two groups of sessions are popularity index, HTML-to-Image ratio and unassigned referrer. Additionally, the results show that differences between the features values among the two groups of sessions in this experiment are not as significantly different as is the case in the first experiment.

The significance of the differences results between true positive and true negative sessions extracted from the YORKU and SharpSchool web log data are shown in Tables 4.19 and 4.20. In general, the results are somewhat different between the three datasets. The features that are significantly different in the three

datasets are click number and popularity index. In general, fewer features are significantly different in this experiment than in the first experiment.

Table 4.18 – T-scores for the difference test on mean values of all 10 features between true positive and true negative sessions extracted from *CSE* web log data (note: bold-lettered numbers indicate the significant difference with 95% confidence)

Classifiers	1 t-score	2 t-score	3 t-score	4 t-score	5 t-score	6 t-score	7 t-score	8 t-score	9 t-score	10 t-score
C4.5	**9.30**	**2.60**	0.8	**6.14**	1.82	**66.89**	2.35	**5.76**	0.69	0.67
RIPPER	**17.52**	**17.10**	**8.45**	**5.56**	1.44	**67.04**	3.27	**5.24**	**29.33**	1.82
k-Nearest Neighbour	**2.56**	**15.43**	1.00	0.58	1.83	**5.21**	0.70	**8.71**	1.52	0.88
Naive Bayesian	**2.09**	**14.93**	**2.24**	1.84	0.73	**6.37**	**2.13**	**16.77**	**3.13**	**3.22**
Bayesian Network	**26.76**	**2.26**	**2.04**	0.35	1.91	**11.00**	1.33	**6.67**	**2.41**	**2.02**
SVM	**13.74**	**15.83**	**2.32**	0.75	1.02	**3.8**	1.15	**6.49**	1.25	1.45
Neural Network	**12.31**	**9.45**	1.47	**6.76**	1.83	**8.62**	2.84	**5.79**	1.87	**3.39**

Table 4.19 – T-scores for the difference test on mean values of all 10 features between true positive and true negative sessions extracted from *YORKU* web log data (note: bold-lettered numbers indicate the significant difference with 95% confidence)

Classifiers	1 t-score	2 t-score	3 t-score	4 t-score	5 t-score	6 t-score	7 t-score	8 t-score	9 t-score	10 t-score
C4.5	**3.61**	**4.96**	1.9	0.86	1.17	**43.21**	1.6	**7.66**	1.81	0.67
RIPPER	**5.33**	**4.16**	1.81	1.78	1.44	**32.83**	1.32	**7.43**	**2.2**	1.82
k-Nearest Neighbour	**3.38**	**6.87**	0.9	0.59	1.83	**9.43**	1.27	**10.07**	1.26	0.88
Naive Bayesian	**2.73**	**3.72**	1.18	1.34	0.73	**3.97**	0.68	**12.47**	**2.4**	**3.22**
Bayesian Network	**4.54**	**6.23**	0.64	0.92	1.91	**19.42**	1.13	**15.08**	**2.11**	**2.02**
SVM	**6.21**	**5.48**	**1.99**	0.81	1.02	**8.96**	0.99	**6.25**	**2.01**	1.45
Neural Network	**6.95**	**5.53**	**2.88**	1.52	1.83	**35.89**	1.56	**24.46**	**2.5**	**2.5**

Table 4.20 – T-scores for the difference test on mean values of all 10 features between true positive and true negative sessions extracted from *SharpSchool* web log data (note: bold-lettered numbers indicate the significant difference with 95% confidence)

Classifiers	1 t-score	2 t-score	3 t-score	4 t-score	5 t-score	6 t-score	7 t-score	8 t-score	9 t-score	10 t-score
C4.5	**34.19**	0.75	1.14	**5.85**	**4.79**	1.12	**12.45**	**15.04**	**7.84**	**18.61**
RIPPER	**38.67**	0.61	1.04	**5.68**	**5.27**	1.36	**72.88**	**21.05**	**9.94**	**21.38**
k-Nearest Neighbour	**34.3**	0.77	0.91	**5.19**	**4.48**	1.56	**13.47**	**16.04**	**8.13**	**21.43**
Naive Bayesian	**58.56**	0.98	0.65	**6.88**	**4.48**	**2.45**	**37.37**	**27.84**	**12.33**	**34.07**
Bayesian Network	**60.14**	0.93	0.55	**6.09**	**4.43**	**2.62**	**50.94**	**28.85**	**13.62**	**33.72**
SVM	**42.87**	0.87	1.5	**5.98**	**5.38**	1.79	**42.54**	**22.97**	**11.68**	**26.54**
Neural Network	**38.87**	0.71	1.4	**5.85**	**5.41**	1.39	**37.65**	**19.14**	**9.63**	**19.95**

4.2.4 Discussion and Observations

The classification results of Experiment 1 are quite close to what we have expected to see. Namely, the characteristics of web site usage by well-behaved web crawlers are inherently different from the usage by human users in terms of the features examined in our study. Hence when faced with the problem of differentiating between such two diverse groups of web users (i.e. their respective feature vectors), all of the examined classification algorithms end up producing highly favourable results.

The classification task in Experiment 2 is more complex than that in Experiment 1 because the classification algorithms are now required to differentiate between four different types of sessions (see Figure 4.28). The results in the second experiment prove that the browsing behaviour of malicious and unknown sessions (as an aggregate group) is quite similar to the browsing behaviour of human users and well-behaved crawlers (as another aggregate group), and thus present a far more challenging task to all

examined algorithms. Consequently, the obtained classification results are far less favourable than those observed in Experiment 1.

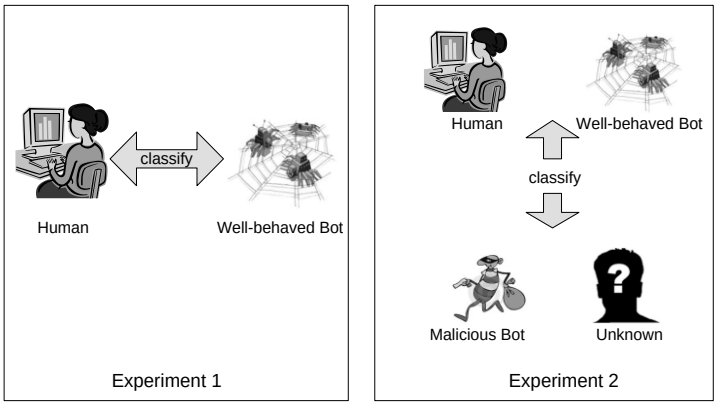


Figure 4.28 – Classification experiments

4.3 Summary

The following three general conclusions were derived from our study:

- The classification accuracy of classification algorithms such as Neural Networks, C4.5, RIPPER and k-Nearest Neighbour algorithms is close to 100% in the first experiment. From the data mining perspective, the identification/classification and separation of well-behaved crawlers and human visitors is very much a feasible task. These results are in line with the results derived in the previous studies on web crawler classification (such as [56], [154], [136], [147]).
- The two new features proposed - the consecutive sequential requests ratio and standard deviation of page request depths - are highly ranked among the other features used in the study by the information gain. The new features are also explicitly shown to improve the classification accuracy, recall, precision and F_1 score of most evaluated algorithms, in both conducted experiments.

- As evident in the second experiment of our study, the browsing behaviours of majority of malicious and unknown visitors as well as well-behaved crawlers and human users is not significantly different. Therefore, from the data mining perspective, their identification/classification is a more difficult task than a simple classification of known well-behaved crawlers. Specifically, effective identification/classification of web crawlers that attempt to mimic human behaviour remains a challenging classification problem. A particular challenge in detection of (malicious) crawlers that attempt to mimic human behaviour is the existence of information that is easily alterable and, as such, can lead to inaccurate classification results. (Note that all 10 features defined in section 4.2.1.2 can ultimately be spoofed; yet, there exists no simple mechanism that can reliably detect whether any spoofing has taken place.) Supervised machine-learning techniques that rely on alterable data – especially for the purposes of data labeling – are particularly susceptible to the problem of web-visitor misclassification. In contrast to supervised learning, unsupervised learning is generally known to be less sensitive to the presence of sporadic data outliers (i.e. presence of sporadically alterable features), and more effective in providing unbiased look and understanding of the underlying data set. Due to this very promise of more robust and unbiased classification, we turn to unsupervised learning in the next stage of our study, as described in the next chapter.

Chapter 5 Detecting Differences in Browsing Behavior of Web Visitors with Unsupervised Clustering Algorithms

In this chapter, we present our second study on detection of suspicious web visitors using two unsupervised neural network (NN) learning algorithms: the Self-Organizing Map (SOM) [137] and Modified Adaptive Resonance Theory 2 (Modified ART2)[8] [169]. (The results of this study have been published in [9] and [10].) In section 5.1, we describe our work on static web domains that employs unsupervised neural network algorithms to cluster web visitors using the same features employed in chapter 4. Again, to the best of our knowledge, this is the first study that has used unsupervised, i.e. unbiased learning, to detect differences in browsing behaviour between malicious and benign web site visitors. In section 5.2, we propose a framework for detecting differences in browsing behaviour between non-outlier and outlier human and malicious web visitors by employing non-parametric correlation analysis. Lastly, in section 5.3, we present our concluding remarks.

[8] Modified ART2 is a variation of the original ART algorithm [143]. Its advantages over the original algorithm are: 1) stable learning that results in gradually increasing/merging clusters, and 2) learning/clustering that can be terminated either when the radius of the formed clusters reaches some predetermined size, or when the number of formed clusters reaches some predetermined number.

5.1 Study on Unsupervised Clustering of Web Visitors Sessions

Among many possible unsupervised NN algorithms, we have chosen to work with the SOM and Modified ART2, for the following reasons:

- The goal of the SOM algorithm is to cause the underlying neural network to respond similarly to similar input patterns. From the practical point of view, the SOM algorithm is well known for: a) its *topology preservation* ability, which implies that similar input samples activate topologically close neurons, b) its ability to produce natural clustering, i.e. clustering that is robust to statistical anomalies, and c) superior visualization of high-dimensional input data in 2D-representation space. Note that we chose SOM over an alternative clustering visualization technique such as Multi-Dimensional Scaling (MDS) [170]. As shown in [171] and [172], both SOM and MDS generate similar mappings for the principle of data visualization. Note that unlike MDS, SOM algorithm is well known for its *topology preservation* ability. The topology-preserving property enables extraction and visualization of relative mutual relationships among the data. The exhaustive overview of application of various visual clustering techniques on our data is beyond the scope of this study.
- The ART2 algorithm also deploys the concept of competitive learning, but in combination with the so-called *winner-takes-all* rule, ultimately producing fundamentally different clustering results. Namely, in contrast to SOM, the ART2 algorithm is able to: a) preserve the balance between retaining previously learned patterns and learning new ones – a feature known as *stability-plasticity property*; and b) identify statistically underrepresented but significant clusters, thus being greatly suited for imbalanced datasets.

In the view of the previous works, the novelty of our research is twofold. Firstly, to the best of our knowledge, this is the first study that applies unsupervised learning to the problem of web-visitor

categorization, ultimately aiming to promote effective differentiation between malicious web-crawlers and other (non-malicious) visitor groups to a web site. (Note, in two other studies deploying SOM [138], [173], and one other study deploying ART [144], only human web-visitors have been considered, and little to no attention has been given to automated web crawlers. Secondly, this is the first study that specifically focuses on the detection of "outlier" points (i.e., web visitors) such as malicious crawlers that exhibit human-like browsing behaviour.)

5.1.1 Pre-processing of Web Server Logs

In this study, the process of session identification is exactly the same as in previous study presented in section 4.2.1. The dataset labelling process, as performed by our log analyser, is illustrated in Figure 5.1. Recall from section 4.1.3 that in unsupervised machine learning cluster labelling is performed after the learning process derives the cluster distributions.

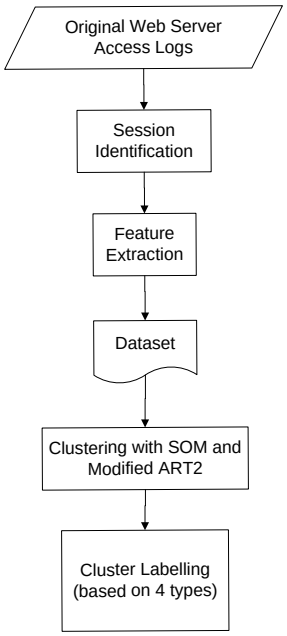


Figure 5.1 – Web server access log pre-processing

5.1.1.1 Features

In order to be consistent with our previous experiments, in this unsupervised learning study we have utilized the same 10 features as in the previous supervised learning study.

5.1.1.2 Dataset Labelling

Once the training dataset (comprising feature-vector representations) is generated, the log analyzer labels each feature-vector as belonging to one of the following 4 categories: *human visitors*, *well-behaved web crawlers*, *malicious crawlers* and *unknown visitors*. As in the previous study, the log analyzer maintains a table of user agent fields of all known (malicious or well-behaved) web crawlers and browsers. This table can be built from the data found on web sites [64], [150], [151]. The dataset labelling process is shown in Figure 5.2.

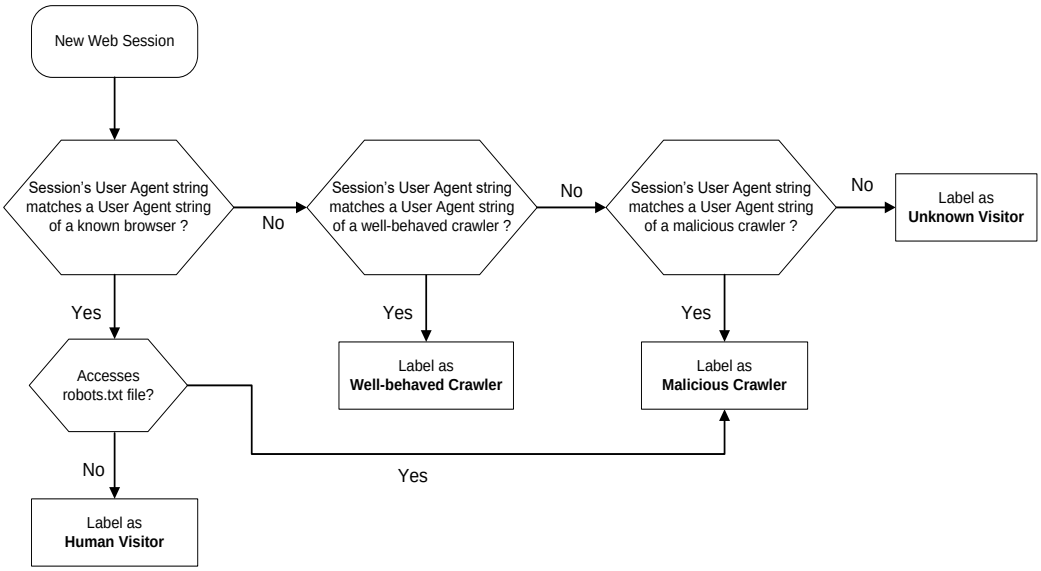


Figure 5.2 – Dataset labelling flow chart

As explained previously, in unsupervised learning, the goal of data labelling is to facilitate our understanding and validation of the results that are to be obtained by the actual clustering process. Namely, through quick association of feature-vectors corresponding to a cluster with their pre-assigned labels, we hope to be able to obtain a better understanding of the cluster's nature and significance.

5.1.2 Experimental Design

5.1.2.1 Training Data

In this unsupervised learning study we have used the same log files as in the supervised learning study described in section 4.2.2.2. Tables 5.1, 5.2, 5.3 list the number of sessions and class label distributions generated by the log analyzer and extracted from the CSE, YORKU and SharpSchool web log datasets, respectfully.

Table 5.1 – Class distributions in the CSE dataset

	Number of Sessions
Total # of Human Sessions	92149
Total # of Well-behaved Crawler Sessions	6644
Total # of Malicious Crawler Sessions	448
Total # of Unknown Visitor Sessions	910
Total #	100151

Table 5.2 – Class distributions in the YORKU dataset

	Number of Sessions
Total # of Human Sessions	707854
Total # of Well-behaved Crawler Sessions	9014
Total # of Malicious Crawler Sessions	860
Total # of Unknown Visitor Sessions	3445
Total #	721173

Table 5.3 – Class distributions in the SharpSchool dataset

	Number of Sessions
Total # of Human Sessions	650526
Total # of Well-behaved Crawler Sessions	125524
Total # of Malicious Crawler Sessions	7166
Total # of Unknown Visitor Sessions	10629
Total #	793845

5.1.2.2 Clustering Algorithms

The detection of web crawlers was evaluated with the following two unsupervised neural network algorithms: SOM and Modified ART2. The implementation of SOM algorithm is provided within MATLAB as a part of Neural Network Toolbox software package. We have chosen a SOM comprising 100 neurons in 10-by-10 hexagonal arrangement. The map was trained with 200 epochs. The Modified ART2 implementation was based on the pseudo-code outlined in [169]. The algorithm parameters were set to: $\rho_{max} = 1.5$, $\Delta\rho = 0.005$ and $n_{max} = 1$. These parameters ensure that cluster generation will be stable regardless of the initial ordering of the input data. All input vectors were normalized prior to being fed to SOM and Modified ART2.

Note that the size of the SOM map and number of clusters in Modified ART2 were selected strategically to help us understand and visualize the distribution of input sessions (i.e., web browsing behavior) and to simplify the identification of outlier web sessions. Also, a larger SOM map was computationally very expensive and would not have helped us identify outlier neurons/web sessions in a more meaningful way.

5.1.3 Clustering Results

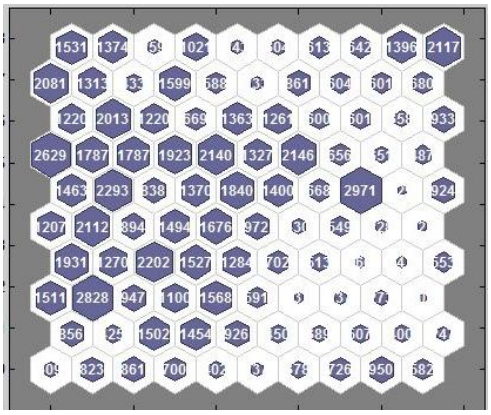
In this section, we present the results obtained by clustering data sets with SOM and Modified ART2 unsupervised neural network algorithms.

5.1.3.1 SOM Results

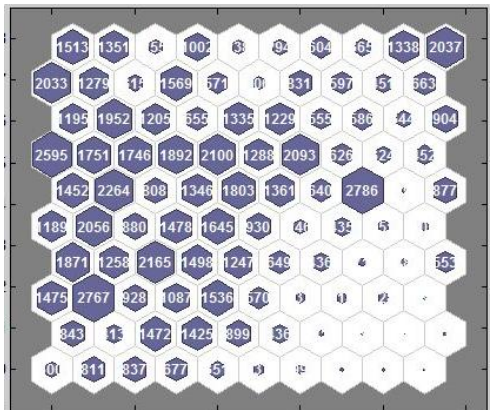
Figures 5.3-5.5 display the SOM clustering results obtained with CSE, YORKU and SharpSchool web log datasets, respectfully. On each of the shown 10-by-10 neuron maps, the size of the blue region inside a neuron's hexagon is proportional to the number of session hits for that neuron, i.e. number of sessions

whose feature vectors are the closest to the (same) given neuron. The exact number of a neuron's session hits is also explicitly provided within the neuron's hexagon region.

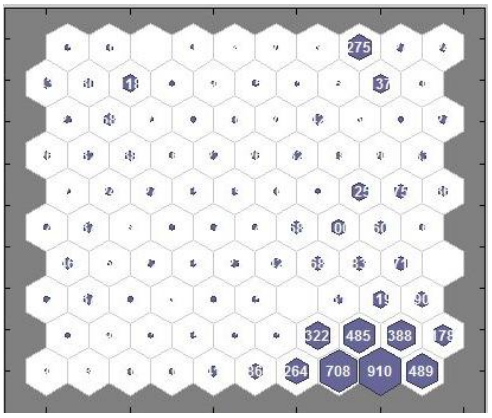
The maps in Figures 5.3.a), 5.4.a) and 5.5.a) show the neuron hits for all sessions and thus helps us visualise the actual distribution of the training dataset (i.e. helps us get an idea about the number, size and spatial proximity of the dataset's most dominant clusters). Figures 5.3-5.5.b)-e) show the neuron hits for sessions that were pre-labelled as belonging to human, well-behaved crawler, malicious crawler and unknown visitor, respectively.



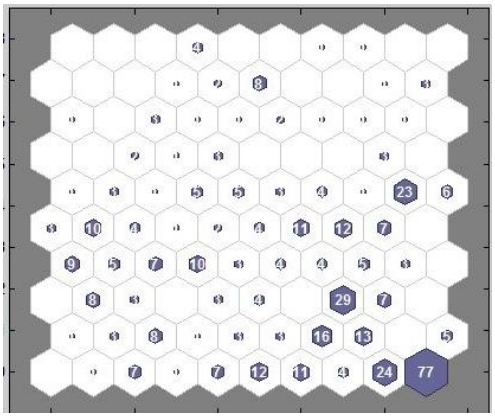
5.3.a) *All* sessions neuron hits



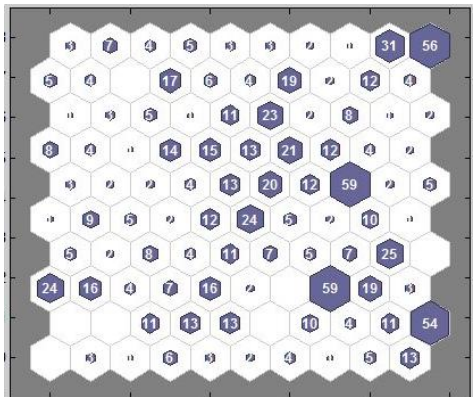
5.3.b) *Human visitor* sessions hits



5.3.c) *Well-behaved crawler* sessions hits

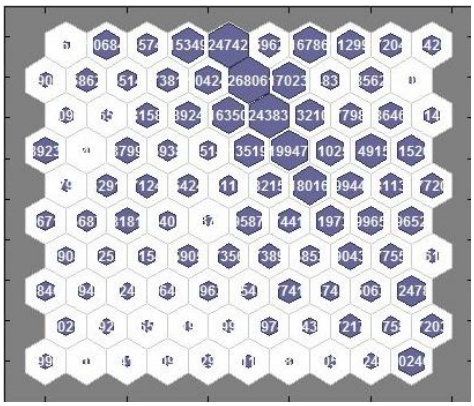


5.3.d) *Malicious crawler* sessions hits

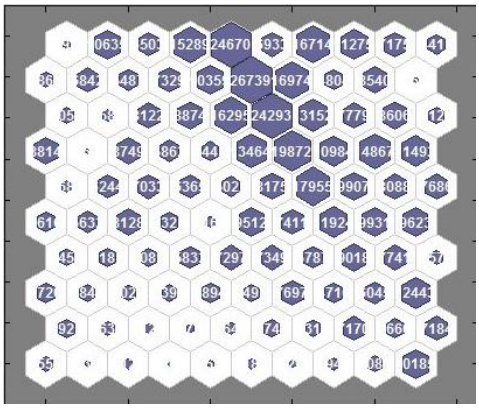


5.3.e) *Unknown visitor* sessions associations with neurons

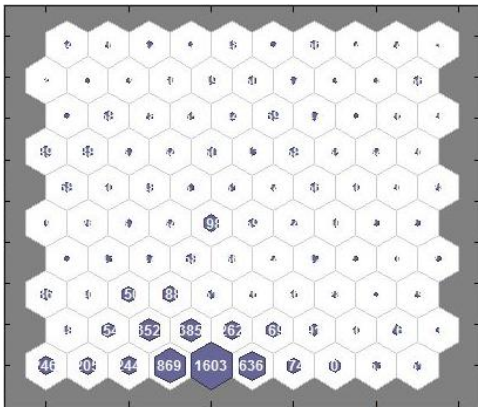
Figure 5.3 – Session hits per neuron visualized in 2 dimensional SOM map (*CSE* web log data)



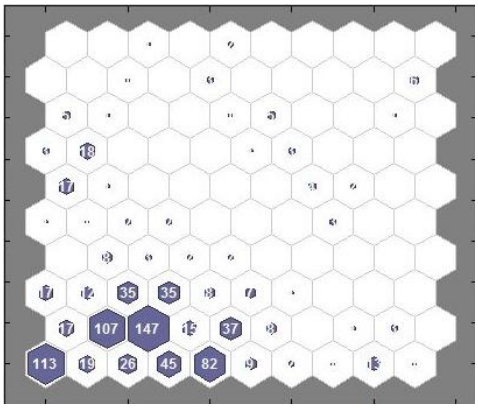
5.4.a) *All* sessions neuron hits



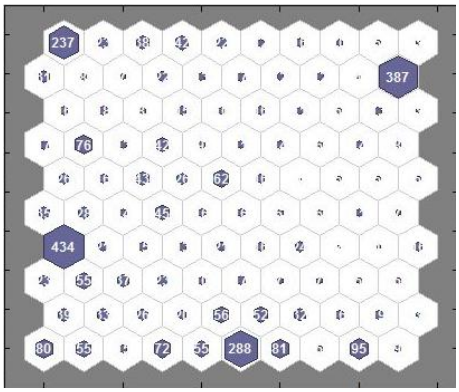
5.4.b) *Human visitor* sessions hits



5.4.c) *Well-behaved crawler* sessions hits

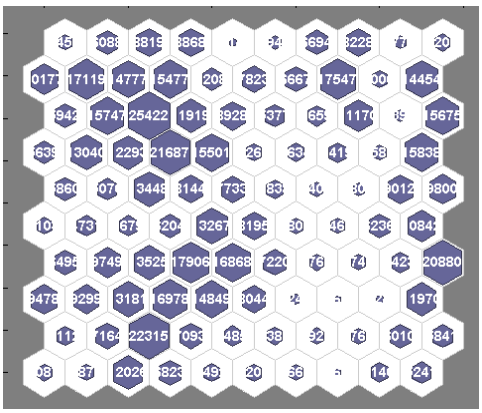


5.4.d) *Malicious crawler* sessions hits

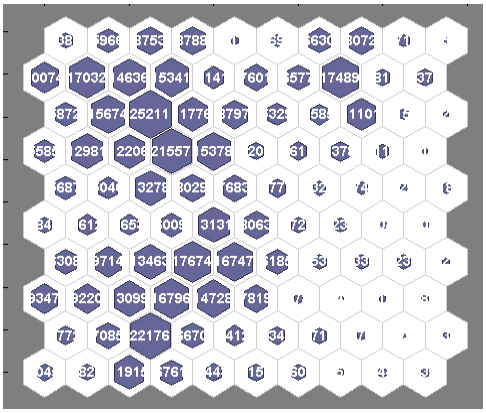


5.4.e) *Unknown visitor* sessions associations with neurons

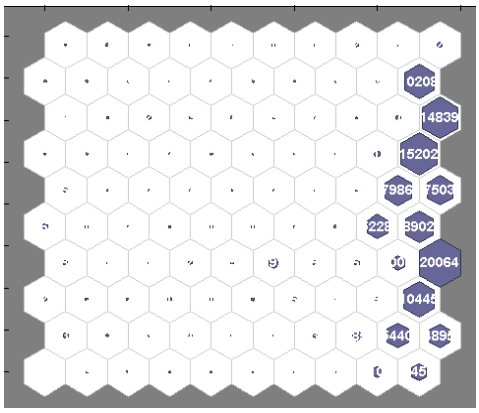
Figure 5.4 – Session hits per neuron visualized in 2 dimensional SOM map (*YORKU* web log data)



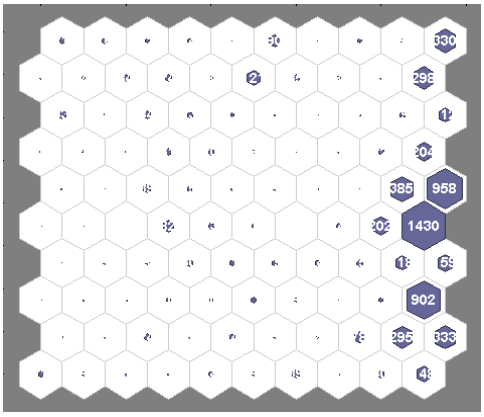
5.5.a) *All* sessions neuron hits



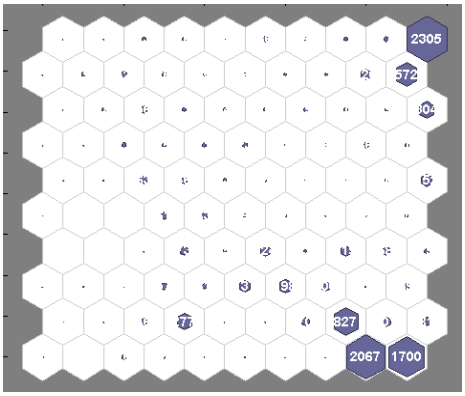
5.5.b) *Human visitor* sessions hits



5.5.c) *Well-behaved crawler* sessions hits



5.5.d) *Malicious crawler* sessions hits



5.5.e) *Unknown visitor* sessions associations with neurons

Figure 5.5 – Session hits per neuron visualized in 2 dimensional SOM map (*SharpSchool* web log data)

From the obtained maps, the following interesting conclusions can be drawn:

- *Human vs. crawlers sessions*: Based on the distribution of fired neurons in Figures 5.3.b)-d), 5.4.b)-d) and 5.5.b)-d), there appears to be a reasonably good separation between human visitor sessions and web-crawler sessions (both malicious and well-behaved). Namely, in CSE web log data, while crawler sessions are almost exclusively associated with neurons in the lower right corner of the map, human sessions are spread over a large area of the map, with most human sessions firing the neurons in the upper left corner of the map. Similarly, in YORKU and SharpSchool web log data, same conclusions can be derived. It might be worth pointing out that the large spread of fired neurons in the map of Figures 5.3.b), 5.4.b) 5.5.b) is not an indicator of greater variability in humans sessions compared to other session groups. Instead, it is the result of the statistical dominance of training-data corresponding to human sessions - see Tables 5.1-5.3. (As indicated in the introduction, the SOM algorithm produces results that are dependent on the input data density; hence, data clusters with higher density tend to 'win-over' a larger number of SOM neurons, regardless of their inter-cluster variance.)
- *Sessions that are labeled as human but 'behave' like malicious crawlers:* A detailed inspection of Figures 5.3.b) and 5.3.d) reveals that, in spite of the well-formed separation between human and malicious web-crawlers, a percentage of web visitor sessions that declare themselves as regular (human) visitors[9] end up firing neurons in the region (or close to the region) dominated by malicious web-crawlers – lower right corner of the map. In the view of our data-labeling algorithm from section 5.1.1.2, this observation raises the question whether those sessions, in fact, correspond to malicious crawlers whose aim is to bypass web site security by simply not accessing the robots.txt file and/or falsifying the value of user agent string field. Recall, the initial accessing of the robots.txt file would ideally be performed by automated crawlers visiting a web site, but cannot be enforced. Similarly, user

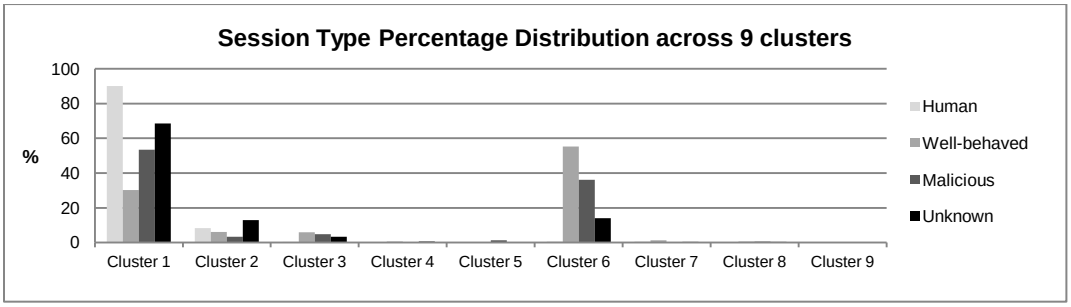
[9] Recall from section 4.2.1.4 that we consider all sessions that carry the name of a well-known browser in the user agent field and that do not access the robots.txt to be human sessions.

agent string appears as a parameter in HTML requests, and can be relatively easily altered. Same conclusions can be made with the clustering results shown in Figure 5.4.b) and d) and 5.5.b) and d).

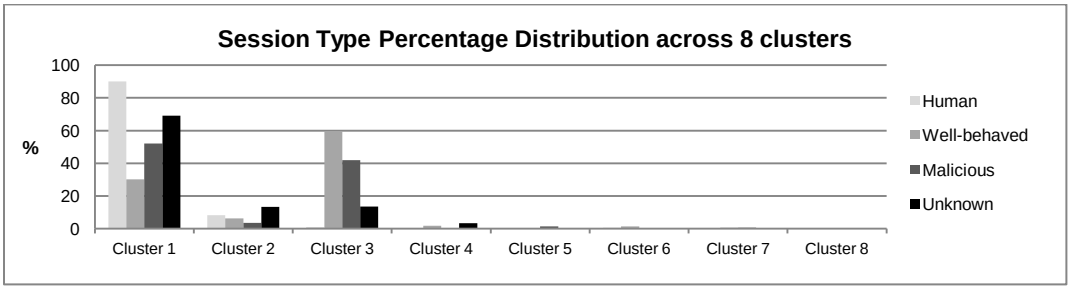
- *Sessions that are labeled as malicious crawlers or unknown visitors but 'behave' like humans:* A detailed inspection of Figures 5.3.b), 5.3.d) and 5.3.e) also reveals that a number of sessions that are identified as belonging to malicious crawlers or unknown visitors end up firing neurons in the region dominated by human generated sessions – left and top half of the map. It is reasonable to assume that these visitors are indeed malicious crawlers or unknown crawlers, as it is unlikely that any regular human visitor would change the value of its agent string into 'malicious crawler', thus risking to be blocked by the web site. Same conclusion, as stated above, can be made with the YORKU and SharpSchool clustering results in Figures 5.4 and 5.5. Accordingly, this observation implies that the behaviour of some malicious crawlers and most unknown visitors - those that fire the nodes in the left half of the map - is very similar to the behaviour of regular users. It should be obvious that such malicious crawlers and unknown visitors are potentially very dangerous. Namely, had they attempted to falsify the value of agent string (i.e. declare themselves as regular visitors), they would have 'perfectly' blended into the population of regular human visitors, and would be very hard to detect by the web site's security system.

5.1.3.2 Modified ART2 Results

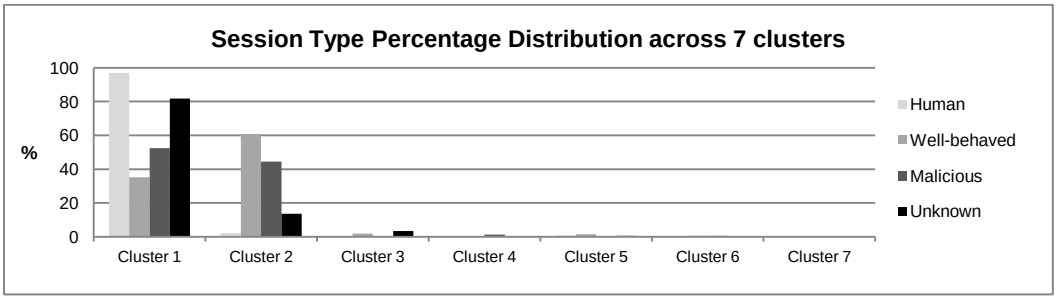
Figures 5.6-5.8 display the results of CSE, YORKU and SharpSchool dataset clustering using Modified ART2, respectfully. Each plot displays the ratio of each session type (human, well-behaved crawler, malicious crawler and unknown visitor) per cluster placement. A session is placed in cluster *i*, if its 10-dimensional vector representation is the closest (measured in the Euclidean distance) to the centroid of cluster *i* among all other clusters. The plots a), b), c), d), e), f), g) and h) display the sample results when Modified ART2 algorithm generates 9, 8, 7, 6, 5, 4 and 3 clusters of sessions, respectively.



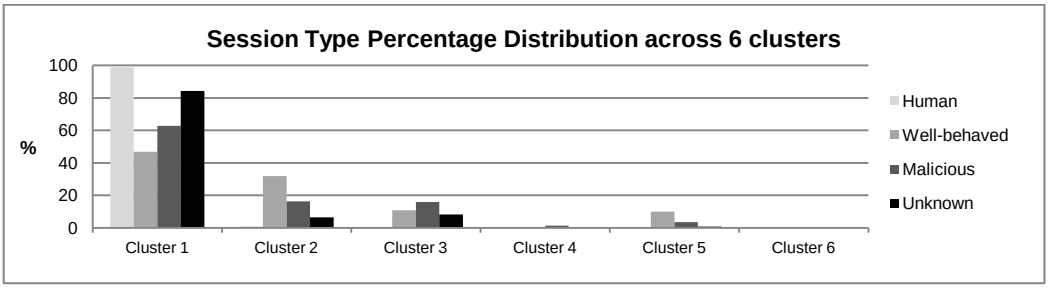
5.6.a) Distribution across 9 clusters



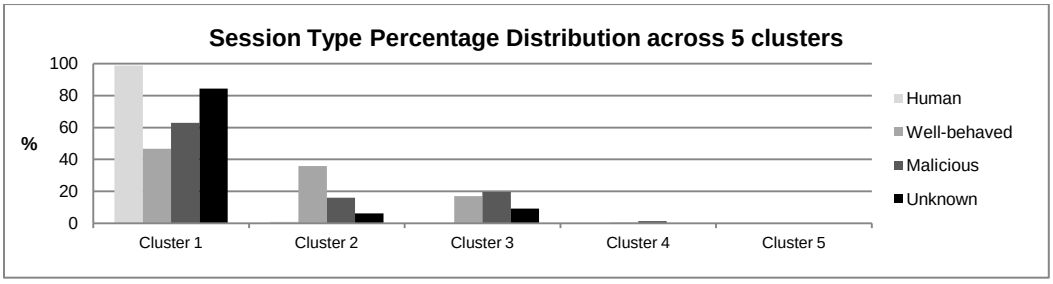
5.6.b) Distribution across 8 clusters



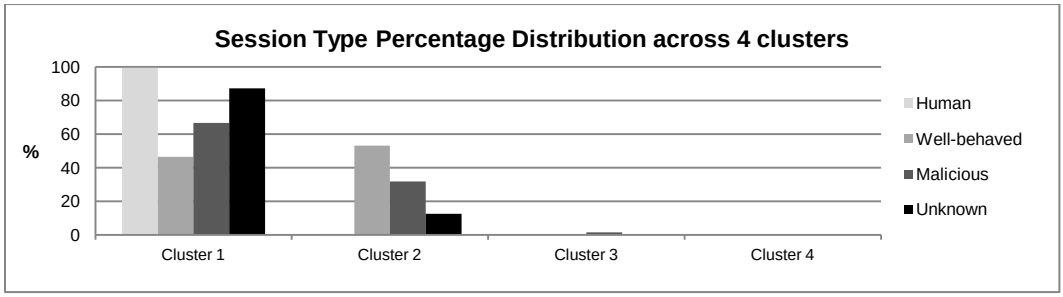
5.6.c) Distribution across 7 clusters



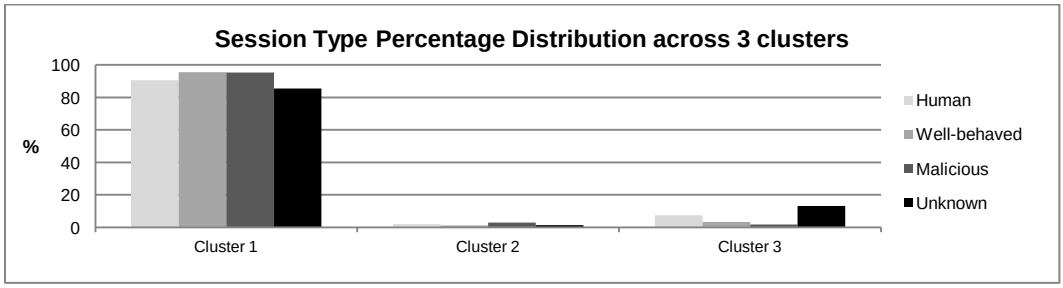
5.6.d) Distribution across 6 clusters



5.6.e) Distribution across 5 clusters

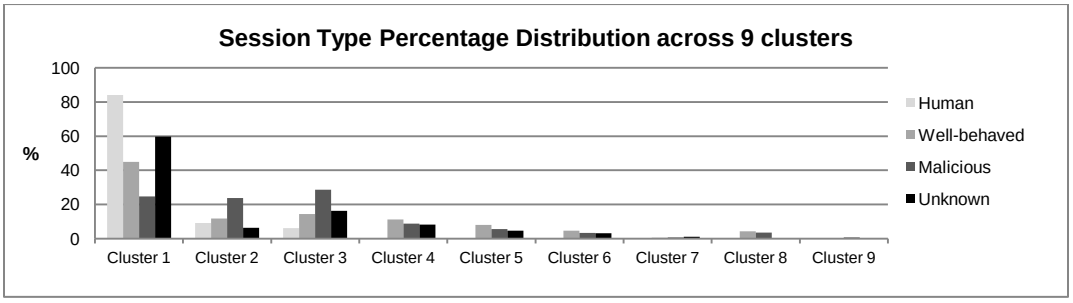


5.6.f) Distribution across 4 clusters

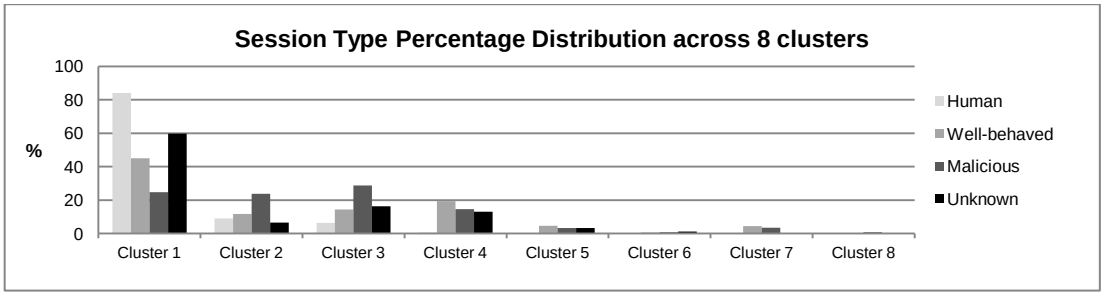


5.6.g) Distribution across 3 clusters

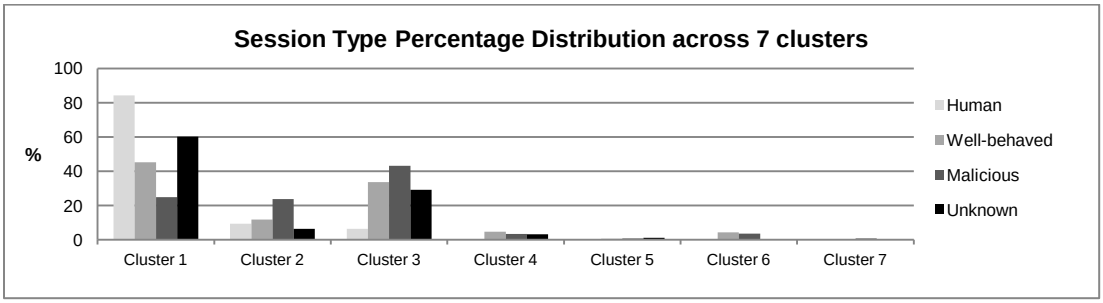
Figure 5.6 – Session type percentage distribution in *CSE* web log data



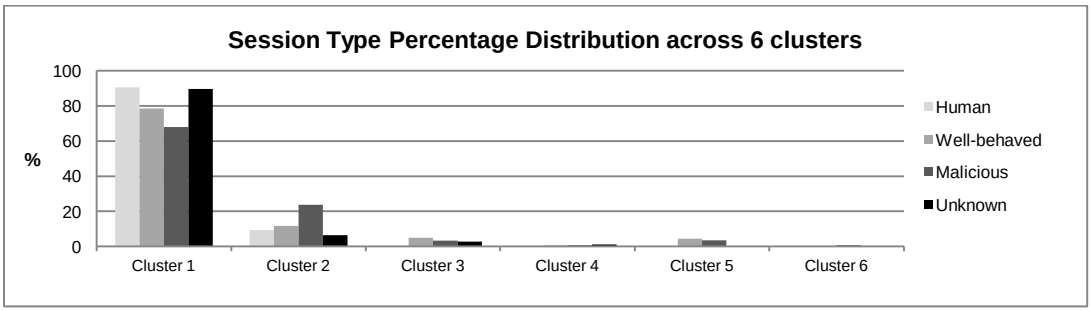
5.7.a) Distribution across 9 clusters



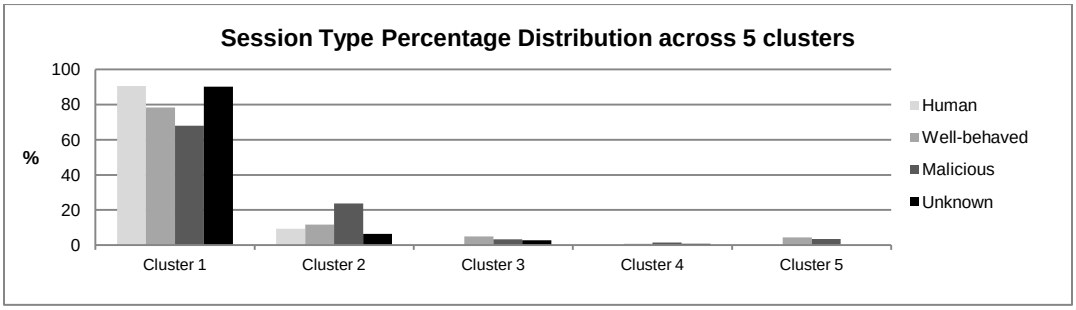
5.7.b) Distribution across 8 clusters



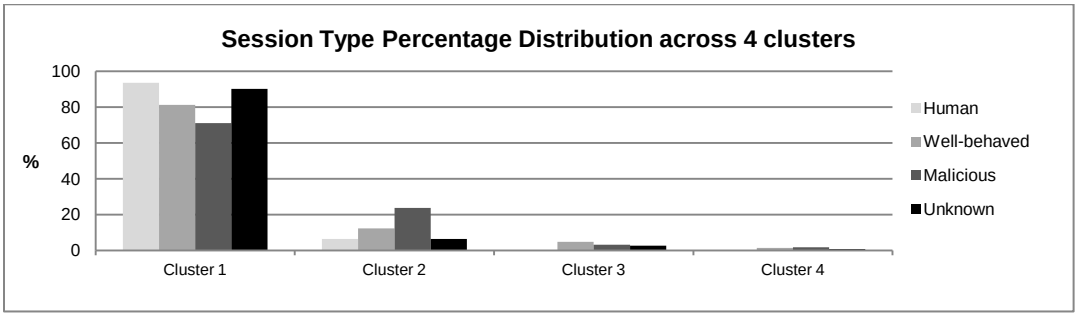
5.7.c) Distribution across 7 clusters



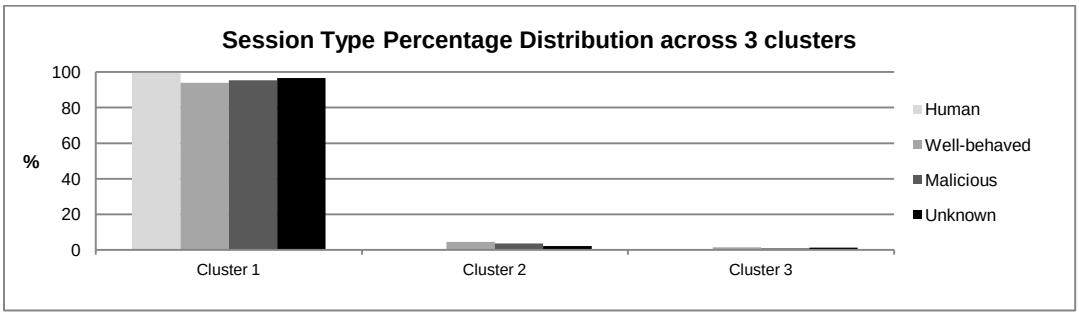
5.7.d) Distribution across 6 clusters



5.7.e) Distribution across 5 clusters

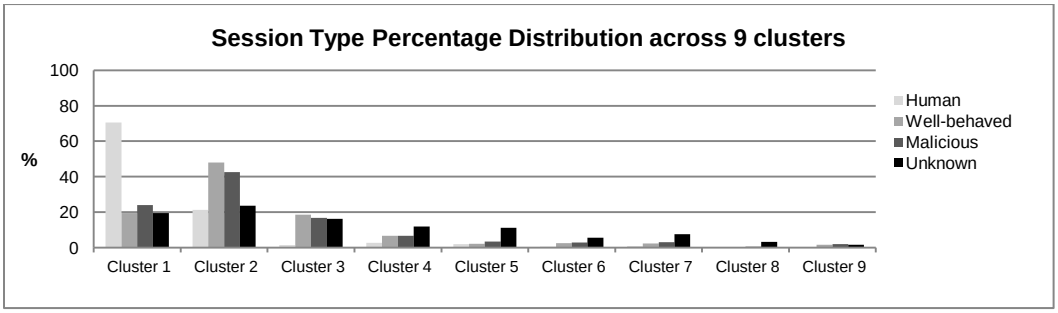


5.7.f) Distribution across 4 clusters

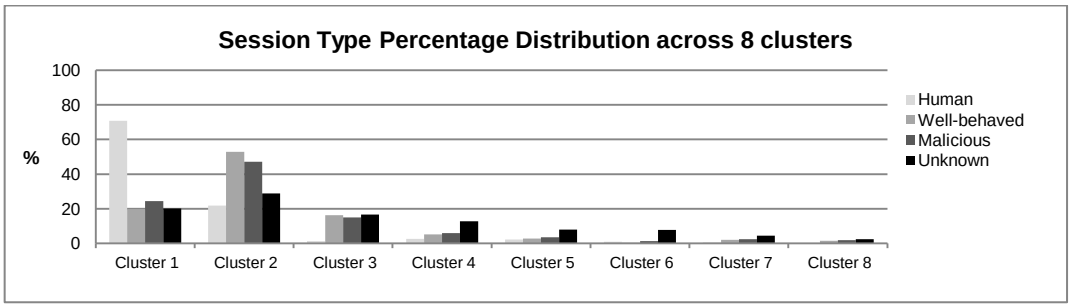


5.7.g) Distribution across 3 clusters

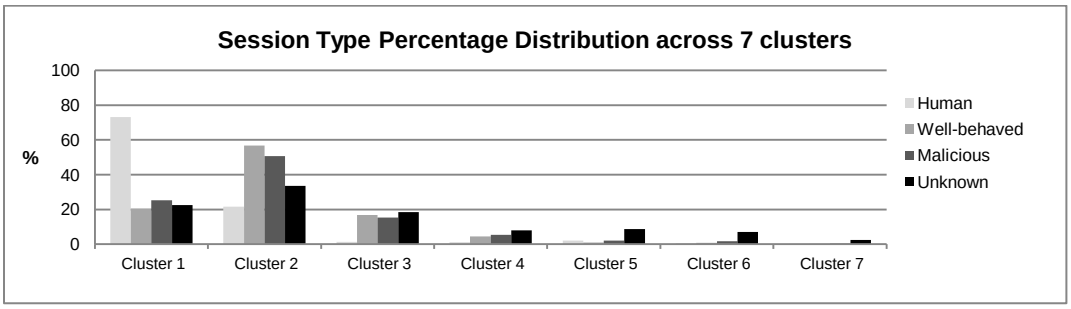
Figure 5.7 – Session type percentage distribution in *YORKU* web log data



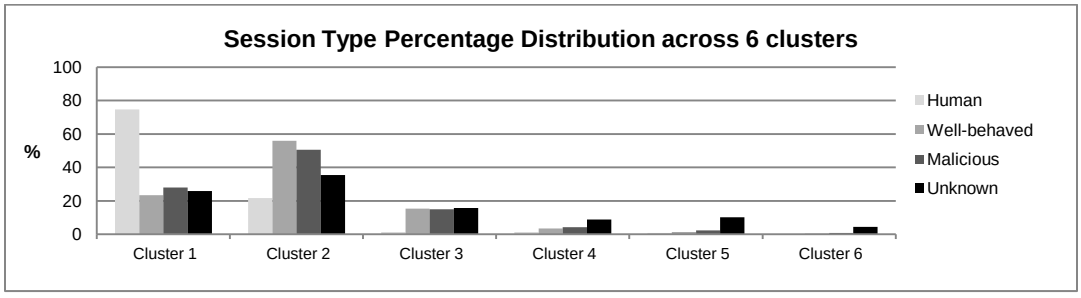
5.8.a) Distribution across 9 clusters



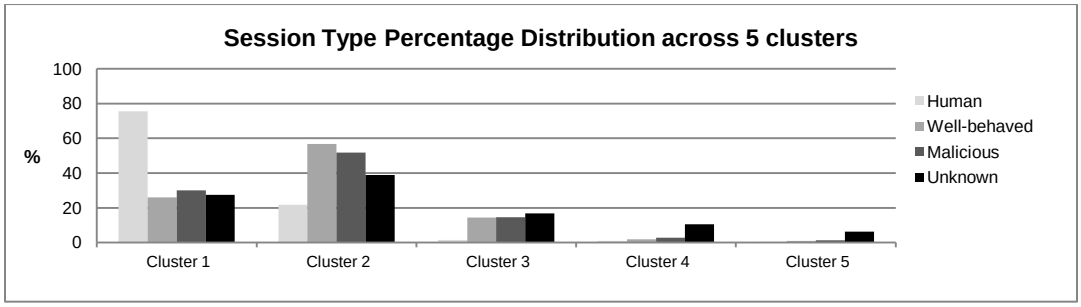
5.8.b) Distribution across 8 clusters



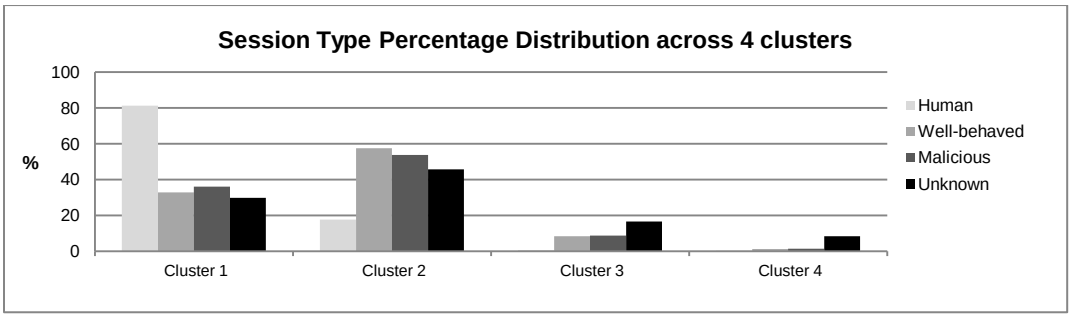
5.8.c) Distribution across 7 clusters



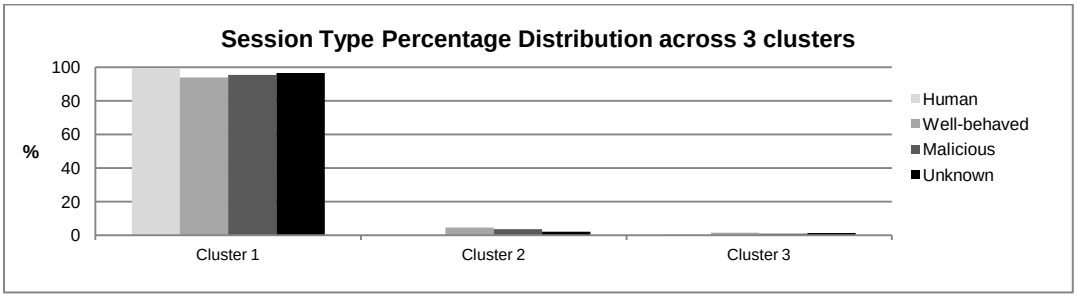
5.8.d) Distribution across 6 clusters



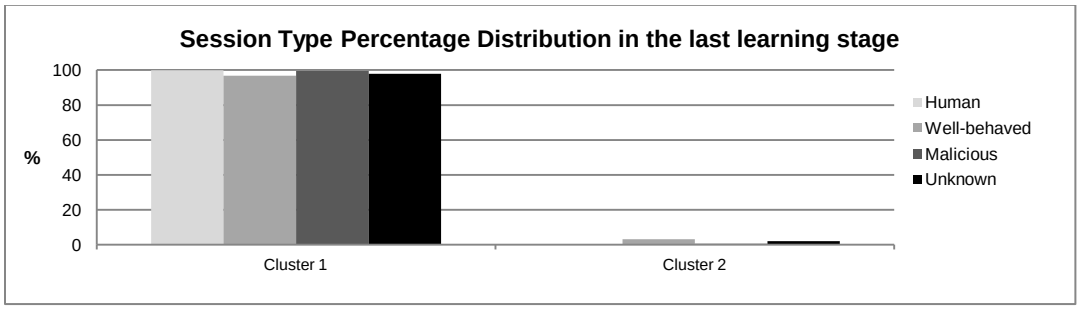
5.8.e) Distribution across 5 clusters



5.8.f) Distribution across 4 clusters



5.8.g) Distribution across 3 clusters



5.8.h) Distribution in the last learning stage

Figure 5.8 – Session type percentage distribution in SharpSchool web log data

While the results generated with SOM are useful for obtaining information about the spatial distribution, i.e. proximity, of data clusters, Modified ART2 gives us an insight into the inter-cluster variance. (As indicated in the introduction, Modified ART2 creates equal-size clusters and is not influenced by statistical irregularities in the training dataset.) With this in mind, and by expecting Figures 5.6-5.8, we derive the following conclusions:

- *Cluster Relationships*: We plot the session type distributions in Figure 5.6 starting from 9 clusters and moving down to a single cluster in order to show how clusters (and session types) are merged together in the last 8 learning stages (epochs) of an application of Modified ART2 algorithm on our dataset. By observing the seven plots and noting the session type percentages among the clusters, we can conclude the following. Firstly, sessions grouped together in cluster 1 and belonging to malicious crawlers, unknown visitors and human visitors, are, in terms of the Euclidean space of their feature vectors, in very close proximity to each other, i.e. the malicious crawlers and unknown visitors have very similar browsing behaviour to the human visitors. Secondly, the malicious crawler sessions and unknown visitor sessions grouped together in clusters 2-5 in Figure 5.6.a)-e) are significantly distant from sessions in cluster 1 and hence exhibit a significantly different browsing behaviour than sessions belonging to cluster 1.
- *Cluster Relationships in YORKU and SharpSchool logs*: The cluster relationships are similar to the results shown in Figure 5.6. In general, the sessions that are grouped together inside cluster 1 in the Figure 5.7.a) remain together for the final 8 learning stages in Figures 5.7.b)-f). The rest of the sessions grouped together in clusters 2-10 in the Figure 5.6.a) are mostly distributed among clusters 2-5 in the final learning stages.
- *Human sessions*: Over 90% of human sessions fall into cluster 1 in Figures 5.6.a)-e), over 85% of human sessions fall into cluster 1 in Figures 5.7.a)-e) and over 65% of human sessions fall into cluster

1 in Figures 5.8.a)-e), thus suggesting a relatively small behavioural variance of this user group in both datasets. In practical terms, this implies that human users tend to follow very similar web browsing patterns.

- *Unknown sessions*: Over 80% of unknown sessions in Figures 5.6.a)-e), over 60% of unknown sessions in Figures 5.7.a)-e) and over 60% of unknown sessions in Figures 5.8.a)-e) belong to the same clusters as human sessions, namely cluster 1. This confirms our hypothesis from the previous section, that most unknown sessions follow very human-like browsing characteristics.
- *Malicious web-crawler sessions*: Out of all session groups, malicious crawlers (and well-behaved crawlers) exhibit the greatest variability in both datasets – they are spread over most of the clusters in Figures 5.6-5.8 with most being assigned to cluster 1. It is interesting to observe that 50%-60% of malicious web-crawler sessions are assigned to cluster 1, together with human visitors in Figures 5.6 a)-e) and 20% - 60 % of malicious web-crawler sessions are assigned to cluster 1, together with human visitors in Figures 5.7.a)-e) and 5.8.a)-e). This again confirms our earlier hypothesis, that some malicious web crawlers behave very-much like regular users, and in the case of a falsified user agent string value their detection would have presented a particular challenge.

5.1.4 Outlier Analysis

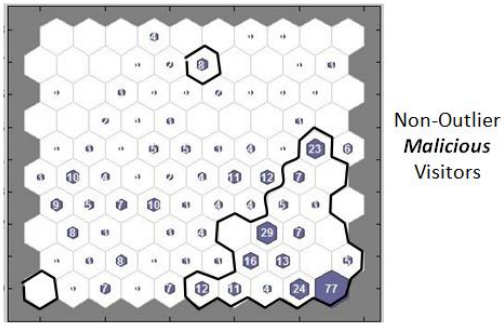
In the preceding section, we have identified three groups of sessions that could present a particular challenge for web intrusion detection systems:

1) Sessions that are labeled as malicious crawlers but 'behave' like humans - *outlier malicious sessions*. It is very important to detect/identify such sessions since with higher level of sophistication (i.e., by employing a browser-type user agent string and not accessing robots.txt file) they could easily blend in with actual/regular human sessions and become undetectable.

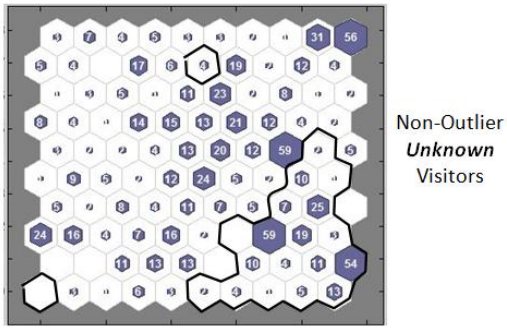
2) Sessions that are labeled as unknown visitors but 'behave' like humans *outlier unknown sessions*. As in the case of outlier malicious sessions, the outlier unknown sessions could be also challenging to detect/identify if they had employed a higher level of sophistication as described above.
3) Sessions that are labeled as humans but 'behave' like malicious crawlers or unknown visitors *outlier human sessions*. These sessions, in fact, already present a detection challenge since there is a possibility that they might actually belong to malicious crawlers that are spoofing their user agent string to hide their true identity. Web administration staff would be very interested in detection of such sessions.

The positions of outlier and non-outlier web sessions in the SOM diagrams are shown in Figures 5.9-5.11. The procedure for separating web sessions into outliers and non-outliers is as follows:

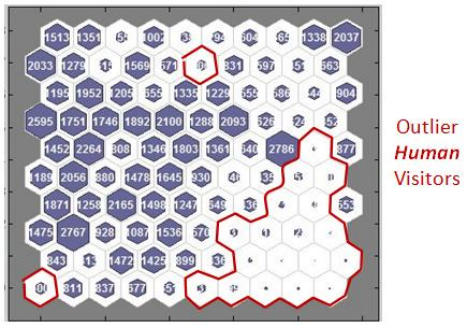
1. The neurons are sorted in ascending order based on the number of human web sessions clustered within their region.
2. The human web sessions clustered within the first 20 neurons after the ordering are marked as outlier human web sessions and the remainder are marked as non-outliers.
3. The non-outlier malicious and unknown web sessions are identified as belonging to the 20 "outlier" neurons identified in step 2. The remainder are marked as outliers.



a) Non-outlier and outlier malicious sessions

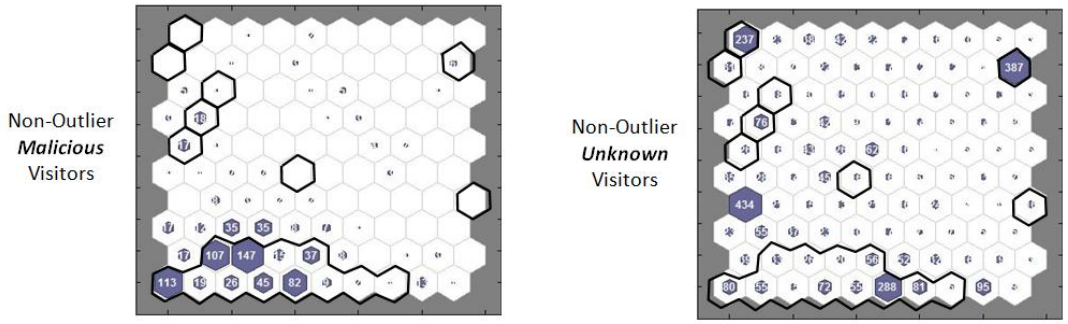


b) Non-outlier and outlier unknown sessions



c) Non-outlier and outlier human sessions

Figure 5.9 – Positions of outlier and non-outlier sessions in the SOM generated from CSE dataset

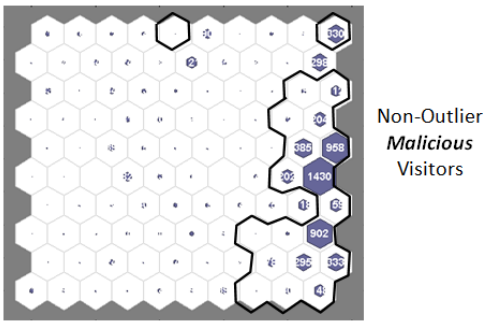


a) Non-outlier and outlier malicious sessions

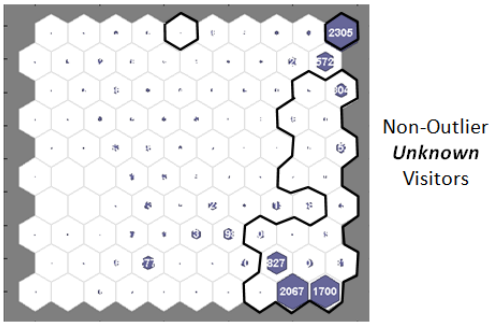
b) Non-outlier and outlier unknown sessions

c) Non-outlier and outlier human sessions

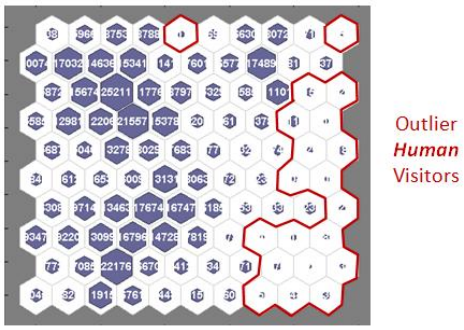
Figure 5.10 – Positions of outlier and non-outlier sessions in the SOM generated from YORKU dataset



a) Non-outlier and outlier malicious sessions



b) Non-outlier and outlier unknown sessions



c) Non-outlier and outlier human sessions

Figure 5.11 – Positions of outlier and non-outlier sessions in the SOM generated from SharpSchool dataset

In this section we undertake a detailed analysis of the underlying characteristics of these three groups of sessions, which we will refer to as *outlier sessions* in the remainder of this chapter. Our analysis of outlier sessions entails:

1) Identifying the user agent strings (UASs) and the respective source IP addresses associated with human, malicious and outlier unknown sessions.
2) Applying the significance of the difference test between individual feature values of outlier sessions and non-outlier session types. We believe that from the point of view of web administrators, they would be most interested in discovering malicious and unknown sessions once they start appearing as human (i.e., by using legitimate-looking browser-based UASs). In this scenario, the administrator should, from the entire group of human sessions, be able to quickly identify those that are, in fact,

malicious. Accordingly, in the case of malicious and outlier unknown sessions, we would identify features that make them *most distinguishable/distant* relative to the non-outlier human sessions. Contrarily, in the case of human outlier sessions, web administrators would be most interested in identifying features that make them *most similar* relative to actual non-outlier malicious sessions.

5.1.4.1 Outlier Malicious Sessions

In order to gain a better understanding of the malicious crawler sessions that 'behave' like humans - i.e., outlier malicious sessions, we identified and grouped together all outlier malicious crawler sessions in the SOM generated from the CSE log data (see Figure 5.9.a)). Note that these sessions are also associated with cluster 1 in Figure 5.6 a)-c). There has been a total of 235 such sessions. Secondly, we grouped together all outlier malicious crawler sessions in the SOM generated from the YORKU log data (see Figure 5.10.a)). Note that these sessions are also associated with cluster 1 in Figure 5.7 a)-c). There has been a total of 177 such sessions. Thirdly, we identified together all outlier malicious crawler sessions in the SOM generated from the SharpSchool log data (see Figure 5.11.a)). Note that these sessions are also associated with cluster 1 in Figure 5.8 a)-c). There has been a total of 1427 such sessions.

a) User Agent Strings and Source-IP Addresses of Malicious Crawlers that 'behave' Like Humans

Tables 5.4-5.6 display the UASs of outlier malicious sessions extracted from CSE, YORKU and SharpSchool web log datasets, respectfully. These three tables also show the source IP addresses of user sessions, as well as the overall number of sessions that have employed the given UASs. (Note that in most cases, multiple web sessions originated from the same IP address.) According to the information posted on http://www.botsvsbrowsers.com/ , http://www.useragentstring.com and http://www.user-agent-string.info web sites, these sessions are either known as malicious (e.g., sogou web crawler), belong to unknown bots that access the robots.txt file (all other user agent strings describing crawlers/bots in Tables 5.4-6) or employ valid browser-type UASs but suspiciously access the robots.txt file.

In the case of the outlier malicious sessions in all three datasets, more than three quarters of these outlier malicious sessions employ valid browser-type (Firefox, IE or Chrome browser's) UASs. (Note that these are marked as malicious because they access the robots.txt file.) As documented in [52], the DDoS agents are known to employ valid UASs. Therefore, it is possible that these sessions could, in fact, be malicious DDoS bots.

Also note in Table 5.4 that 14 (6%) of the malicious crawler outlier sessions have a blank UAS. As explained in section 4.2.1.4, these sessions are labelled as malicious and are not placed in the unknown session group because they access the robots.txt file. We have observed, from the clustering results of SOM and Modified ART2 that these malicious sessions with a blank UAS turn up in the same area as non-outlier human sessions. Therefore, web administrators should take a note of all sessions with a blank UAS.

Also note that "sogou" is a UAS that appears in all three Tables 5.4-5.6. However, there are also a number of unique UASs that appear exclusively in each of the three datasets.

Table 5.4 – Common types of user agent strings among outlier malicious crawlers in CSE web log

User Agent String	Originating IP Addresses	Number of Outlier Malicious Sessions with this User Agent String
Valid Firefox User Agent String	199.246.40.54 (Canada), 99.228.37.126 (Canada), 130.45.78.140 (United States),, 117.211.118.7 (India), 115.75.75.14 (Vietnam), 92.255.64.120 (Russian Federation), 91.197.223.93 (Ukraine), 58.147.172.162 (Bangladesh), 203.141.92.14(Japan), 82.137.200.48 (Syrian Arab Republic), 76.67.100.9 (Canada), 218.248.5.222 (India), 193.138.110.138 (Poland), 14.52.133.143 (Korea, Republic of), 174.95.169.181 (Canada), 62.128.6.5 (Germany)	136
Valid Chrome User Agent String	205.211.168.16 (Canada), 82.192.87.31 (Netherlands)	57
Empty User Agent String	199.187.125.26 (United States), 78.46.78.40 (Germany), 123.30.175.68 (Vietnam), 67.251.82.134 (United States), 66.249.68.4 (United States)	14
Valid Internet Explorer String	62.24.252.133 (United Kingdom), 78.113.135.126 (France), 110.172.152.159 (India), 118.192.35.60 (China), 46.208.73.19 (United Kingdom)	11
ahrefsbot	213.186.127.2 (Ukraine)	6
sogou	123.126.50.76 (China)	3
heritrix	79.175.162.122 (Iran, Islamic Republic of)	3
topyx-crawler	194.199.60.68 (France)	1
linkchecker/6.9	24.89.182.19 (United States)	1
quickobot	110.234.114.50 (India)	1
dibot	188.40.99.137 (Germany)	1
qryos	202.71.101.68 (Malaysia)	1

Table 5.5 – Common types of user agent strings among outlier malicious crawlers in YORKU web log

User Agent String	Originating IP Addresses	Number of Outlier Malicious Sessions with this User Agent String
Valid Firefox UA string	220.181.61.187(China), 24.78.200.51(Canada), 70.162.138.18(United States), 174.89.44.95(Canada), 130.63.236.137(Canada), 99.235.7.6(Canada), 188.165.235.21(France), 173.34.108.78(Canada), 149.255.33.35(N/A), 149.255.33.34(N/A), 123.125.169.88(China), 69.58.178.56(United States)	84
Valid Internet Explorer UA string	62.24.252.133(United Kingdom), 80.40.134.120(United Kingdom), 41.131.253.57(Egypt), 199.187.122.98(United States), 41.189.43.115(Cote D'Ivoire)	51
blekkobot	199.87.252.52(United States), 199.87.252.49(United States), 199.87.252.102(United States)	10
ahrefsbot	213.186.127.2(Ukraine)	7
pear http_request	74.115.1.48(Anonymous Proxy), 74.115.0.27(Anonymous Proxy), 74.115.0.38(Anonymous Proxy)	4
sogou	123.126.68.17(China)	3
careerbot	178.77.126.55(Germany)	2
orbitspiderbot	176.9.90.105(Germany)	2
mozilla/5.0 ()	1.202.221.3(China)	2
squirrobot	23.20.159.217(United States), 23.20.94.137(United States)	2
Blank UA string	123.126.51.51(China), 98.111.48.78(United States),	2
findcanbot	173.13.143.78(United States)	1
acoonbot	95.211.139.1(Netherlands)	1
mail.ru	217.69.133.30(Russian Federation)	1
linkfluence	94.23.0.209(France)	1
citeseerxbot	130.203.133.123(United States),	1
itim	123.30.175.68(Vietnam)	1
cdlwas_bot	128.48.120.140(United States)	1
obot	206.253.226.14(United States)	1

Table 5.6 – Common types of user agent strings among outlier malicious crawlers in SharpSchool web log data.

User Agent String	Originating IP Addresses	Number of Outlier Malicious Sessions with this User Agent String
genieo/1.0	various IPs from US	335
sogou	various IPs from China	118
daumoa	various IPs from South Korea	116
Valid IE UA string	various IP s from China, US	170
Valid Chrome UA string	various IP s from China, US	147
Valid Firefox UA string	various IP s from Turkey, China, US	133
ncbot	85.25.137.24(Germany)	46
pear http_request	74.115.1.48(Anonymous Proxy), 74.115.0.27(Anonymous Proxy), 74.115.0.38(Anonymous Proxy)	18
apple-pubsub/65.28	207.29.222.245(United States)	3
davclnt	65.217.172.18(United States)	1
bl.uk_lddc_bot	194.66.232.88(United Kingdom)	1
empty User Agent String	204.100.126.254(United States),	1

b) Significance of the Difference Test

In our second experiment, we have applied the significance of the difference test on individual feature values between outlier malicious sessions that 'behave' like human and the non-outlier human sessions. The calculation of the significance of the difference test (i.e. t-test) is based on the Expression (4.15). The results of the significance of the difference test for the CSE, YORKU and SharpSchool web log data are shown in Tables 5.7-5.9, respectfully. The tables also display the mean and variance for each feature value in the two groups of sessions. Based on the overall t-test results, the malicious sessions that 'behave' like humans are shown to NOT exhibit human-like browsing characteristics in terms of the mean value of all features except feature 5 in the CSE dataset (relatively small percentage of HEAD requests by outlier malicious sessions) and feature 7 in the YORKU dataset (relatively low amount of data downloaded from

the server by outlier malicious sessions). Hence, it is our recommendation that web administrators pay special attention to all features except feature 5 and 7.

Note the differences in the mean/variance results between the three datasets. This shows that differences in behaviour of malicious bots can be significant. Note also that this is in line with the clustering results generated with Modified ART2 algorithm.

Table 5.7 – Mean, variance and significance of the differences test results on feature values between non-outlier human sessions and outlier malicious sessions extracted from CSE web log data.

Features	Significant Difference?	Mean / Variance Non-outlier Human Sessions	Mean / Variance Outlier Malicious Sessions
1. Click Rate	Yes	0.34/0.48	0.14/0.082
2. Html to image ratio	Yes	0.72/3.93	60.59/138038
3. % of PDF documents	Yes	0.11/0.073	0.14/0.056
4. % of Error requests	Yes	0.048/0.011	0.12/0.022
5. % of HEAD requests	**No**	**0.0013/0.00024**	**0.002/0.0009**
6. % of Unassigned Referrers	Yes	0.044/0.011	0.13/0.039
7. Number of Bytes Requested	Yes	5.25/0.87	5.4/0.98
8. Popularity Index	Yes	7.21/4.85	5.5/7.51
9. Std. Dev. of Page Depth	Yes	0.58/0.2	0.89/0.34
10. % of Sequential requests	Yes	0.65/0.047	0.51/0.047

Table 5.8 – Mean, variance and significance of the differences test results on feature values between non-outlier human sessions and outlier malicious sessions extracted from YORKU web log data.

Features	Significant Difference?	Mean / Variance Non-outlier Human Sessions	Mean / Variance Outlier Malicious Sessions
1. Click Rate	Yes	0.4/0.35	0.26/0.17
2. Html to image ratio	Yes	0.33/1.17	0.82/6.41
3. % of PDF documents	Yes	0.019/0.015	0.17/0.072
4. % of Error requests	Yes	0.01/0.002	0.042/0.014
5. % of HEAD requests	Yes	0.00051/0.00034	0.0357/0.031
6. % of Unassigned Referrers	Yes	0.031/0.026	0.19/0.072
7. Number of Bytes Requested	**No**	**5.33/0.58**	**5.26/0.84**
8. Popularity Index	Yes	11.52/2.27	5.27/10.86
9. Std. Dev. of Page Depth	Yes	0.63/0.05	0.79/0.1
10. % of Sequential requests	Yes	0.46/0.03	0.33/0.06

Table 5.9 – Mean, variance and significance of the differences test results on feature values between non-outlier human sessions and outlier malicious sessions extracted from SharpSchool web log data.

Features	Significant Difference?	Mean / Variance Non-Outlier Human Sessions	Mean / Variance Outlier Malicious Sessions
1. Click Rate	Yes	4.36/44.63	0.79/7.46
2. Html to image ratio	Yes	0.001/0.007	0.81/0.32
3. % of PDF documents	Yes	0.002/0.00093	0.001/0.00038
4. % of Error requests	Yes	0.03/0.00093	0.12/0.035
5. % of HEAD requests	Yes	0.00027/0.000085	0.0047/0.00049
6. % of Unassigned Referrers	Yes	0.042/0.0062	0.15/0.033
7. Number of Bytes Requested	Yes	5.83/0.42	6.16/0.41
8. Popularity Index	Yes	6.5/13.7	6.88/9.8
9. Std. Dev. of Page Depth	Yes	1.93/0.26	1.91/0.24
10.% of Sequential requests	Yes	0.46/0.02	0.42/0.025

5.1.4.2 Outlier Unknown Sessions

We have also tried to gain a better understanding of unknown visitor sessions that 'behave' like humans - i.e., outlier unknown sessions. Namely, we grouped together all outlier unknown crawler sessions in the SOM generated from the CSE log data (see Figure 5.9.b). Note that these sessions are also associated with cluster 1 in Figure 5.6.a)-c). There has been a total of 564 such sessions. Next, we grouped together all outlier unknown crawler sessions in the SOM generated from the YORKU log data (see Figure 5.10.b). These same sessions are also associated with cluster 1 in Figure 5.7.a)-c). There has been a total of 2093 such sessions. Lastly, we grouped together all outlier unknown crawler sessions in the SOM generated from the SharpSchool log data (see Figure 5.11.b). These same sessions are also associated with cluster 1 in Figure 5.8.a)-c). There has been a total of 1854 such sessions.

a) User Agent Strings and Source-IP Addresses of Malicious Crawlers that 'behave' Like Humans

Tables 5.10-5.12 display the user agent string information of outlier malicious sessions extracted from CSE, YORKU and SharpSchool web log datasets, respectfully. According to information posted on

http://www.botsvsbrowsers.com/, http:/www.user-agent-string.info and http://www.useragentstring.com web sites, these sessions are labelled by an unrecognizable user agent string.

As can be observed in Table 5.10, 92% of outlier unknown sessions, extracted from CSE web log data, are observed to have a blank UAS. As explained in section 4.2.1.4, these sessions are labelled as not malicious and instead are placed in the unknown session group because they do not access the robots.txt file. We have observed, however, that in the SOM map some of these unknown sessions with a blank UAS turn up in the same area as outlier malicious sessions. This is a strong indication that such unknown sessions are also likely malicious crawlers that are pretending to be human, but end up in the unknown group just by virtue of avoiding to access the robots.txt file. Therefore, web administrators should take a note of all sessions with a blank UAS.

We can also observe a large number of blank UASs in YORKU and SharpSchool web log data. There is also a number of unique UA strings that appear exclusively in each of the three datasets (e.g., inc in CSE, \ in YORKU and rarely used in SharpSchool).

Table 5.10 – Common types of user agent strings among outlier unknown visitors in CSE web log

User Agent String	Originating IP Addresses (Geographical Location of the IP Addresses)	Number of Outlier Unknown Sessions with this User Agent String
Empty User Agent String	99.34.147.95 (United States), 188.54.201.36 (Saudi Arabia), 113.140.84.109 (China), 122.174.93.244 (India), 187.193.235.228 (Mexico), 112.210.167.77 (Philippines), 41.235.127.108 (Egypt), 158.94.65.231 (United Kingdom), 94.71.171.85 (Greece), 113.246.187.169 (China), 119.152.122.232 (Pakistan), 117.205.59.210 (India), 69.165.160.43 (Canada), 173.206.174.224 (Canada), 97.101.246.66 (United States), 117.200.53.9 (India), 173.248.212.71 (Canada), 173.35.244.203 (Canada), 76.68.16.33 (Canada), 75.68.161.147 (United States), 188.159.128.60 (Iran) … and 300 additional unique IP addresses	519
htc_maple	199.7.156.34 (Canada)	16
Java(JDK version)	134.59.2.173(France), 90.218.5.14(United Kingdom), 67.194.86.169(United States), 130.236.182.150(Sweden), 96.22.105.67(Canada), 79.117.69.200(Romania), 193.170.75.225(Austria), 173.250.139.224(United States), 192.107.175.11(United States), 99.226.218.75(Canada), 150.214.108.217(Spain), 202.4.178.38(India)	10
iuc	65.255.37.199 (United States)	3
zmeu	188.165.213.213(France), 211.233.71.244(Korea, Republic of)	2
ivborw	173.252.20.27 (Canada), 117.196.99.238 (India),	2
pagepeeker.com	188.40.84.81 (Germany)	2
gvfs	132.180.194.76(Germany), 132.67.104.205(Israel)	2
Bot, links, midori, webprocess, juc, ossproxy, sogouframework, cuteftp	69.84.154.155 (United States), 72.28.188.180 (United States), 109.204.40.122 (United Kingdom), 175.152.2.167 (China), 113.210.253.174 (Malaysia), 121.14.162.110 (China), 175.140.19.107 (Malaysia), 165.246.166.240 (Korea, Republic of), 124.121.241.216 (Thailand),	1 (each)

Table 5.11 – Common types of user agent strings among outlier unknown visitors in YORKU web log

User Agent String	Originating IP Addresses (Geographical Location of the IP Addresses)	Number of Outlier Unknown Sessions with this User Agent String
Empty User Agent String	14.98.60.93(India), 184.144.76.55(Canada),, 206.192.70.55(United States), 23.21.180.100(United States), 79.125.12.71(Ireland) and additional IP addresses from 50 different countries	780
facebookplatform	69.171.228.248(United States)	361
Java(JDK version)	65.97.144.106(United States), 72.21.217.65(United States), 72.21.217.2(United States), 206.123.170.188(Canada), 24.231.88.130(Canada)	89
site24x7	208.69.56.166(Canada)	56
yahooexternalcache	8.137.88.40(United States)	33
jakarta	206.187.5.200(United States)	21
ossproxy	99.235.106.207(Canada), 108.162.161.76(United States), 74.14.116.138(Canada), 67.82.228.143(United States),	19
htc_maple	207.245.236.101(Canada)	15
vonchimpenfurlr	184.173.98.68(United States)	12
\	174.18.92.139(United States), 188.29.43.99(United Kingdom), 65.103.210.96(United States), 72.152.192.209(United States), 184.36.251.73(United States), 188.29.121.168(United Kingdom),	7
goo_search	218.213.143.193(Hong Kong)	4
360se, apple-pubsub, rget, unwindfetchor, testagent, msdw, gbplugin, juc, goscraper, user-agent:, tencenttraveler, ning, contribute, iuc, nitro pdf download, linux, pbbrbot, ucweb, \x95mozilla	Various International IP addresses	696

Table 5.12 – Common types of user agent strings among outlier unknown visitors in SharpSchool web log

User Agent String	Originating IP Addresses (Geographical Location of the IP Addresses)	Number of Outlier Unknown Sessions with this User Agent String
apple-pubsub/65.28	various IPs from US	617
web sitepulse checker	72.32.122.236(United States), 204.15.198.210(United States),	266
Empty UA string	various IPs from US, 162.72.192.195(N/A), 175.42.85.247(China), 175.44.11.164(China), 123.151.148.155(China), 91.207.4.117(Ukraine), 142.4.100.60(Canada),	206
windows-update-agent	various IPs from US	100
spiceworks	64.68.251.210(United States), 74.94.10.58(United States), 24.247.236.163(United States)	97
nsplayer	various IPs from US	61
twitterfeed	various IPs from US	47
kaseya network monitor	24.247.88.44(United States)	24
ossproxy	various IPs from US	24
vb project	218.213.143.193(Hong Kong)	17
rarely used	151.224.71.45(N/A), 70.230.150.249(United States), 174.255.128.137(United States), 108.54.174.36(United States), 166.137.156.26(United States), 173.18.1.33(United States), 166.137.156.28(United States), 72.204.29.21(United States), 166.137.209.144(United States), 70.63.132.150(United States), 24.167.85.117(United States), 198.228.200.158(United States), 71.235.122.107(United States), 166.147.121.148(United States), 166.137.156.25(United States),	15
zte-z431, pantechp6030, ios/6.1.3 (10b329) dataaccessd, feedbot, windows-media-player, windows-rss-platform, <uau= c=rm-255-smith/> , (null), kwc-torino/, r355[tf26843546020563439 1000000012525083723], sophosautoupdate, azbul.net, user-agent: user-agent:, nsplayer, netfront, shockwave flash, queryseekerspider, acrobat viewer,one browser, camelhttpstream, cricket,	Various International IP addresses	380

sam455/1.0[tf268435460815 71798048021001563419746 4] , <ua u= c=d4h92vm1/>, go http package		

b) Significance of the Difference Test

The results of the significance of the difference test for the sessions generated with CSE, YORKU and SharpSchool web log data are shown in Tables 5.13-15, respectfully. Based on the overall t-test results in three datasets, the unknown sessions that 'behave' like humans are shown to NOT exhibit human-like browsing characteristics in terms of all features except relatively small percentage of erroneous requests and relatively high variance of the HTML-to-image ratio by outlier unknowns.

Note the differences in the mean/variance results between the three datasets. This shows that differences in behaviour of unknown bots can be significant.

Table 5.13 – Mean, variance and significance of the differences test results on feature values between non-outlier human sessions and outlier unknown sessions extracted from *CSE* web log data.

Features	Significant Difference?	Mean / Variance Non-outlier Human Sessions	Mean / Variance Outlier Unknown Sessions
1. Click Rate	Yes	0.34/0.48	0.45/0.56
2. Html to image ratio	Yes	0.72/3.93	0.32/0.54
3. % of PDF documents	Yes	0.11/0.073	0.23/0.15
4. % of Error requests	**No**	**0.048/0.011**	**0.04/0.012**
5. % of HEAD requests	Yes	0.0013/0.00024	0.0069/0.0036
6. % of Unassigned Referrers	Yes	0.044/0.011	0.02/0.01
7. Number of Bytes Requested	Yes	5.25/0.87	5.44/1.03
8. Popularity Index	Yes	7.21/4.85	6.49/3.73
9. Std. Dev. of Page Depth	Yes	0.58/0.2	0.49/0.22
10. % of Sequential requests	Yes	0.65/0.047	0.68/0.052

Table 5.14 – Mean, variance and significance of the differences test results on feature values between non-outlier human sessions and outlier unknown sessions extracted from *YORKU* web log data.

Features	Significant Difference?	Mean / Variance Non-outlier Human Sessions	Mean / Variance Outlier Unknown Sessions
1. Click Rate	Yes	0.4/0.35	0.16/0.12
2. Html to image ratio	**No**	**0.33/1.17**	**2.33/3990**
3. % of PDF documents	Yes	0.019/0.015	0.051/0.034
4. % of Error requests	Yes	0.01/0.002	0.049/0.022
5. % of HEAD requests	Yes	0.00051/0.00034	0.17/0.1
6. % of Unassigned Referrers	Yes	0.031/0.026	0.018/0.056
7. Number of Bytes Requested	Yes	5.33/0.58	5.04/2.55
8. Popularity Index	Yes	11.52/2.27	9.67/9.58
9. Std. Dev. of Page Depth	Yes	0.63/0.05	0.59/0.14
10. % of Sequential requests	Yes	0.46/0.03	0.43/0.11

Table 5.15 – Mean, variance and significance of the differences test results on feature values between non-outlier human sessions and outlier unknown sessions extracted from *SharpSchool* web log data.

Features	Significant Difference?	Mean / Variance Non-outlier Human Sessions	Mean / Variance Outlier Unknown Sessions
1. Click Rate	Yes	4.36/44.63	1.48/10.27
2. Html to image ratio	**No**	**0.001/0.007**	**0.0016/0.00029**
3. % of PDF documents	Yes	0.002/0.00093	0.0011/0.000074
4. % of Error requests	Yes	0.03/0.00093	0.064/0.033
5. % of HEAD requests	Yes	0.00027/0.000085	0.0015/0.00015
6. % of Unassigned Referrers	Yes	0.042/0.0062	0.076/0.012
7. Number of Bytes Requested	Yes	5.83/0.42	5.73/1.3
8. Popularity Index	Yes	6.5/13.7	6.85/13.33
9. Std. Dev. of Page Depth	Yes	1.93/0.26	1.65/0.56
10. % of Sequential requests	Yes	0.46/0.02	0.49/0.042

5.1.4.3 Outlier Human Sessions

Another interesting group of sessions are sessions that are labelled as human but 'behave' like malicious crawlers - i.e., outlier human sessions. Recall that in the terms of the CSE web log data, these are the sessions that end up firing neurons in the lower right quarter/corner of the map (see Figure 5.9.c)) and are also associated with clusters 2-9 in Figure 5.6.a)-c). There has been a total of 8122 such sessions. In the

terms of the YORKU web log data, these are the sessions that end up firing neurons in the lower left quarter/corner of the map (see Figure 5.10.c)). Note that most of these sessions are also associated with clusters 2-9 in Figure 5.7.a)-c). There has been a total of 109645 such sessions. In the terms of the SharpSchool web log data, these are the sessions that end up firing neurons in the far right end of the map (see Figure 5.11.c)). Note that most of these sessions are also associated with clusters 2-9 in Figure 5.8.a)-c). There has been a total of 122467 such sessions.

a) User Agent Strings and Geographical Location of Source-IP Addresses of Human Crawlers that 'behave' Like Malicious

Due to privacy reasons we cannot list the IP addresses of the most common types of UASs among the human outlier sessions. However, we can list various types of browsers (derived from UASs) which were utilized by human visitors that 'behave' like malicious crawlers, and those include: Mozilla Firefox, Internet Explorer, Safari, Google Chrome and Opera. Also there are a number of visitors that have utilized mobile platforms such as smart phones and tablet PCs.

The geographical distributions of human outlier sessions extracted from CSE, YORKU and SharpSchool web log data are presented in Tables 5.16-5.18. Note that the information on geographic locations of the origin IP addresses was retrieved from the http://www.geobytes.com web site. By examining CSE and YORKU tables, we can observe that majority of sessions are originating from Canada with much smaller proportions originating from other countries such as China, United States, India, UK, Iran, Thailand, Australia, Japan, Germany, Brazil, Bangladesh, Pakistan, Russia, Ukraine, France, and South Korea. The sessions originating from Canadian IP addresses are mostly from Toronto area with great majority of those originating from the York University's workstations and on-campus computers. It is very possible that some of these outlier human sessions are actual human users with browsing characteristics similar to that of malicious crawlers or that some of the 'on campus' machines – especially laptops – are infected with

malware that, in fact, generates these requests in a fashion similar to malicious bots. It could also be the case that some of the identified geographical locations are, in fact, inaccurate since the malicious bots are known to frequently employ proxies. In SharpSchool dataset, the majority of visitors are from U.S. since the web portal hosts mostly web sites for U.S. K-12 schools.

Table 5.16 – Geographical location distribution of outlier human sessions in CSE web log

Country Origin of IP addresses	%
Canada	53
China	10
United States, India	7
UK, Iran, Thailand, Germany, Brazil, Bangladesh, Pakistan, Ukraine, France, Korea	3
Others	~24% (113 unique countries)

City Origin of IP addresses	%
Toronto and York University Domain	35
Jinan	7
New Delhi, Beijing	3
London, Bangalore, Oxford, Tehran	1
Other	~ 41% (1000 unique cities)
Unknown	13

Table 5.17 – Geographical location distribution of outlier human sessions in YORKU web log

Country Origin of IP addresses	%
Canada	55
United States	16
UK	5
India, China	2 (each)
Australia, Germany, Brazil, France, Russia, Japan, Pakistan	1 (each)
Unknown	1
Others	~8% (185 unique countries)

City Origin of IP addresses	%
Toronto and York University Domain	42
Various US cities	14
London, York	1
New Delhi, Beijing	1
Bangalore, Oxford, Tehran	1
Other	~ 28% (7000 unique cities)
Unknown	13

Table 5.18 – Geographical location distribution of outlier human sessions in SharpSchool web log

Country Origin of IP addresses	%
United States	64
Unknown	15
Japan	7
China	5
Germany	2
Russia, Canada, France, Ukraine	1 (each)

City Origin of IP addresses	%
Unknown	42
Various US cities	37
Beijing, Seoul, Shenzen	1
Others	< 1 (each)
Unknown	42

b) Significance of the Difference Test

The results of the significance of the difference test for the sessions generated with CSE, YORKU and SharpSchool web log data are shown in Tables 5.19-5.21, respectively. Again, for each feature, we display the mean and variance values. Based on the presented results for the CSE datasets, the human sessions that 'behave' like malicious crawlers are shown to exhibit atypical (i.e., non-human-like) browsing characteristics in terms of mean values of features 2 (relative low HTML to image file retrievals), 5 (relatively high percentage of HEAD request), and 7 (relatively low number of bytes sent to the client by the server). In case of YORKU data set, only one such feature has been identified - (relatively high percentage of HEAD request), while no such features have been identified for SharpSchool dataset.

Table 5.19 – Mean, variance and significance of the differences test results on feature values between non-outlier malicious sessions and outlier human sessions extracted from *CSE* web log data.

Features	Significant Difference?	Mean / Variance Non-Outlier Malicious Sessions	Mean / Variance Outlier Human Sessions
1. Click Rate	Yes	0.062/0.039	0.21/0.27
2. Html to image ratio	**No**	**2.08/438.9**	**0.82/3.34**
3. % of PDF documents	Yes	0.21/0.045	0.11/0.078
4. % of Error requests	Yes	0.096/0.035	0.12/0.08
5. % of HEAD requests	**No**	**0.017/0.013**	**0.012/0.0055**
6. % of Unassigned Referrers	Yes	0.86/0.071	0.2/0.08
7. Number of Bytes Requested	**No**	**5.11/1.24**	**5.1/1.2**
8. Popularity Index	Yes	3.65/5.41	5.2/4.7
9. Std. Dev. of Page Depth	Yes	1.17/0.43	0.69/0.3
10. % of Sequential requests	Yes	0.37/0.12	0.54/0.048

Table 5.20 – Mean, variance and significance of the differences test results on feature values between non-outlier malicious sessions and outlier human sessions extracted from *YORKU* web log data.

Features	Significant Difference?	Mean / Variance Non-Outlier Malicious Sessions	Mean / Variance Outlier Human Sessions
1. Click Rate	Yes	0.13/0.15	0.45/0.56
2. Html to image ratio	Yes	1.77/183.9	0.41/259.3
3. % of PDF documents	Yes	0.054/0.009	0.009/0.002
4. % of Error requests	Yes	0.12/0.034	0.027/0.012
5. % of HEAD requests	**No**	**0.0017/0.0014**	**0.0005/0.00018**
6. % of Unassigned Referrers	Yes	0..45/0.11	0.24/0.047
7. Number of Bytes Requested	Yes	5.03/0.85	5.25/0.39
8. Popularity Index	Yes	1.94/2.94	6.48/6.22
9. Std. Dev. of Page Depth	Yes	0.89/1.52	0.58/0.21
10. % of Sequential requests	Yes	0.52/0.08	0.6/0.04

Table 5.21 – Mean, variance and significance of the differences test results on feature values between non-outlier malicious sessions and outlier human sessions extracted from *SharpSchool* web log data.

Features	Significant Difference?	Mean / Variance Non-outlier Malicious Sessions	Mean / Variance Outlier Human Sessions
1. Click Rate	Yes	0.30/0.5	0.93/5.75
2. Html to image ratio	Yes	0.009/0.0025	0.021/0.235
3. % of PDF documents	Yes	0.005/0.0017	0.025/0.018
4. % of Error requests	Yes	0.11/0.031	0.15/0.088
5. % of HEAD requests	Yes	0.052/0.037	0.021/0.015
6. % of Unassigned Referrers	Yes	0.99/0.004	0.76/0.11
7. Number of Bytes Requested	Yes	4.98/0.43	5.24/0.78
8. Popularity Index	Yes	11.49/5.34	10.7/23.53
9. Std. Dev. of Page Depth	Yes	0.62/0.15	1.01/0.59
10. % of Sequential requests	Yes	0.62/0.03	0.55/0.073

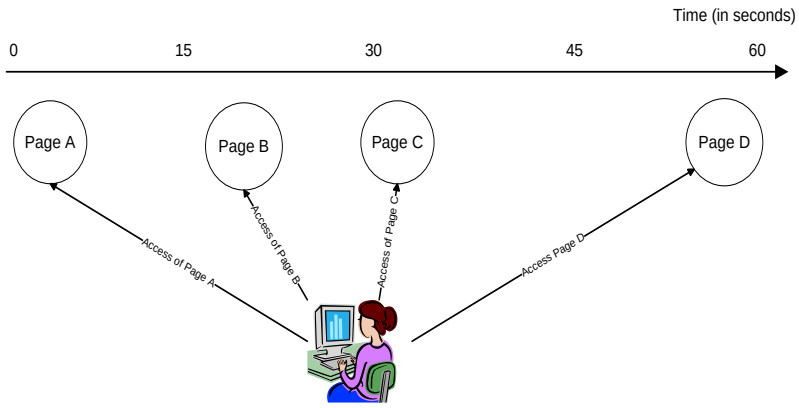
5.1.5 Main Conclusions

In general, there exists a reasonably good separation between malicious and non-malicious web users in terms of their browsing behaviour. And, while mostly human visitors tend to follow rather similar browsing patterns, automated web crawlers (and in particular malicious web crawlers) exhibit a range of browsing strategies. Moreover, in all three datasets we have employed in our study, over 20% of malicious

web crawlers and over 40% of unknown visitors employ browsing strategies that are somewhat similar to those of regular human web-visitors. Clearly, with a higher level of sophistications, these crawlers could pose a serious challenge for current web site security systems, especially those that perform simple screening of their visitors, e.g., by examining the value of user-agent string and/or looking for attempts to access the robots.txt file.

5.2 Detecting Differences in Browsing Behavior of Web Visitors with Non-Parametric Correlation Analysis

The results from the previous section have shown that unsupervised learning can be applied to detect potentially dangerous "outlier" web visitors. Now, it is worth noting that our work presented so far has been conducted without considering the time-dimension (i.e., time-domain) characteristics of the identified web-sessions. However, as illustrated in Figure 5.12, it seems reasonable to expect that time-domain related browsing statistics is significantly different between human- and machine-generated sessions, and hence this statistics can be used to additionally aid the process separation/identification of human vs. machine, or benign vs. malicious web-visitors.



Human visitor

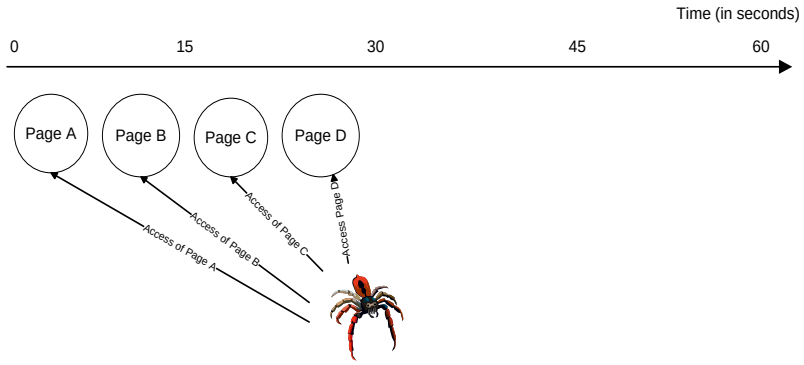


Figure 5.12 – Comparison of the timing differences in web page access behaviour between a human visitor and a web crawler

Here are the two simple illustrations as for how the inclusion of time-domain analysis may benefit the task of web user classification:

- For instance, two sessions might comprise exactly the same sequence of web pages, visited in exactly the same order. However, the information on whether these pages are accessed in a rapid sequence or over a longer period of time, is clearly of great significance. Namely, a rapid sequence access is likely to belong to a crawler, while sessions that span over a longer period of time are likely to belong to humans.
- The time spent viewing each individual page is another important parameter to consider. Namely, as previously discussed in chapter 3, crawlers are likely to spend the same or constant amount of time viewing, i.e., visiting a web page regardless of the amount of contextual information that the given page provides or its actual byte-wise size. On the other hand, the amount of time that humans spend viewing a page will be highly correlated to the contextual importance and byte-wise size of that page relative to others.

Based on the above, it seems reasonable to conclude that the task of distinguishing between human and automated web-visitors will likely benefit from a time-domain analysis of web-browsing session.

In this section, we outline the methodology and results of our analysis involving time-domain characteristics of web sessions. In particular, in section 5.2.1, we introduce the model of human browsing behaviour that deploys three well-known statistical distributions. Using this model, in section 5.2.2, we present the correlation analysis of browsing behaviour characteristics of various identified web visitor groups. Finally, in section 5.2.3, we present several practical recommendations arising from this work.

5.2.1 Introduction

In the study presented in [57], authors state that the human browsing characteristics can be modelled by three statistical distributions:

1) Zipf-Mandelbrot distribution for web page popularities, which models users' page selection behaviour;
2) Pareto distribution for page viewing times by an individual browser;
3) Inverse Gaussian distribution for the overall length of a browsing sequence (i.e., number of 'clicks' in a session).

(Note that, indirectly, features 1) and 3) have already been employed in our supervised/unsupervised studies, and are known to be very effective at distinguishing between human and machine-generated sessions. Feature 2) has not been previously used for the purposes of web-user classification.)

In the following sections, we present a short overview of these three statistical distributions and the way they have been studied in the past literature.

5.2.1.1 Zipf-Mandelbrot Distribution for Web Page Popularities

To introduce the concept of web-page popularity (in a given web site), let us assume that there are N (N > 0) web-pages in total. The popularity of a web page can be seen as a measure of the frequency (i.e., probability) at which this page is accessed relative to other N-1 pages. Now, let random variable W be the requested web page, and Pr(W = i) be the access probability of page w_i. If we further assume that the pages are sorted by their popularity, then in the case of most web sites the page popularity distribution is shown to follows Zipf-Mandelbrot distribution (a generalized Zipf distribution):

$$\Pr(W = i) = \frac{\Omega}{(i+q)^{\alpha}} \tag{5.1}$$

where i is the rank of the data, α (α > 0) is the bias factor, which characterizes the length of the tail of the distribution, and q (q ≥ 0) is the plateau factor, which make the probability of the highest ranked pages flatted. The observed value of the exponent α varies from one traffic trace (i.e., one web domain) to another. That is, the request distribution does not follow the strict Zipf's law (for which α = 1), but instead follows a more general Zipf-like distribution with varying α.

There are a number of studies, published in the last 18 years, that confirm the above hypothesis - that the web page popularities closely follow the Zipf-Mandelbrot distribution. Table 5.22 presents a summary of various datasets analyzed in the previous studies and also lists the reported exponent α. The documented values of α parameter range from 0.47 to 2.5 with most being below 1.0.

Table 5.22 – Summary of previous research analysis on web page popularity

Reference	α	Date of the log file	Type of web site
[57]	1.31	June, 2010	Chinese backbone network
[58]	0.77	August, 1996	Digital Equipment Corporation web proxy traces
	0.69	November, 1996	Home IP service offered by UC Berkeley
	0.78	May, 1997	Computer Science Department at University of Pisa
	0.73	January, 1998	Questnet, a regional ISP in Australia
	0.64	December, 1997	National Lab for Applied Networking Research
	0.83	June, 1998	FuNet, a regional ISP network serving the academic communities in Finland
[174]	1.0-1.5	January, 001-December, 2001	Mobile Internet News Service
[175]	1.35	June, 2004	University of Stuttgart web servers traces
[176]	0.85	October, 1996	web servers at the Computer Science Department at Boston University
[177]	1.35	November, 1994-March, 2005	web servers from the Computer Science Department at Boston University
[178]	0.96	January, 1994	Digital Equipment Corporation web cache
[179]	0.65	April, 1998 – May, 1998	web servers from the Department of Computer Science at Boston University
[180]	~1	September, 1998	web servers at INRIA, a French national research institution
[181]	1.6	November, 1998-April, 2001	HP Laboratories web servers
[182]	1.23	July, 1999-April, 2001	HP Laboratories web servers
[183]	0.6	September, 2004	Internet Service Provider, unknown location
[184]	0.47	April, 2000	University of Washington web servers
[185]	0.25-2.5	September, 2000	web servers at UC at Berkeley
[186]	~1.0	November, 2001-June, 2002	web Servers from the Department of Computer Science at University of Toronto, web servers from National Technical University of Athens in Athens, Greece, web Servers from the Department of Computer Science at University of Cyprus
[187]	0.7, 0.97	January, 2006 – February, 2006	ISP in Brazil
[188]	1.156	December, 2007 – January, 2008	UOL, video content provider located in Brazil

5.2.1.2 Pareto Distribution for Web Page Viewing Times

Let the random variable V represent the page viewing time for a particular web page, and let v_m be the minimum viewing time for all web pages in a given web site. Consequently, the probability density function of web-page viewing times in the given web site is shown in (5.2),

$$\Pr(V = v) = \frac{\alpha \cdot v_m^{\alpha}}{v^{(\alpha+1)}} \quad (5.2)$$

where α is called the Pareto index. In previous studies such as [57], [189], [175], [190], [191], [192] and [193], the observed minimum viewing time v_{min} is on average 30 seconds, while exponent α varies between 0.31 and 1.5. Table 5.23 presents a summary of various datasets analyzed in the previous studies and also lists the reported exponent α and parameter v_{min}.

Table 5.23 – Summary of previous research analysis on web page viewing time

Reference	α / v_{min}	Date of the log file	Type of web site
[57]	1.31 / N/A	June, 2010	Chinese backbone network
[189]	1.5 / 30 sec	January, 1994-February, 1995	web servers from the Computer Science Department at Boston University
[175]	0.7 / 10 sec	June, 2004	University of Stuttgart web servers traces
[190]	0.9 / 56	August, 1995	web servers at GTE Laboratories in Waltham, Massachusetts
[191]	1.5 / 30	Various Log Files from 1998	web Servers at Boston University
[192]	0.31 – 1.15 / 29-99	Various Log Files from 2004	web servers from Automobile Manufacturing Company
[193]	N/A/ / 30 sec	Not available, paper published in 2009	web servers at Network Center of Sun Yat-Sen University
[194]	0.9 / N/A	December, 1989	web servers from University of California at Berkeley

5.2.1.3 Inverse Gaussian Distribution for Number of Clicks in a Session

Let L be the number of links that a visitor visits (i.e., follows or clicks) on a web site during a single web session, then the probability density function of L is known to follow the distribution below:

$$\Pr(L = l) = \sqrt{\frac{\lambda}{2\pi l^3}} \exp\left[\frac{\lambda(l-\mu)^2}{2\mu^2 l}\right], l = 1, 2, \ldots \quad (5.3)$$

where the average value of L is, E[L] = μ, variance Var[L] = μ^3 / λ and $\lambda > 0$ is the shape parameter describing the length of the distribution's tail. Note that as $\lambda \rightarrow \infty$, inverse Gaussian distribution approximates to the normal Gaussian distribution.

A number of the past studies have documented values of μ and λ parameters to show that the browsing length of human visitors follows an inverse Gaussian distribution. Table 5.24 presents a summary of various datasets analyzed in the previous studies and also lists the reported μ and λ parameters.

Table 5.24 – Summary of previous research analysis on web session's browsing length

Reference	μ / λ	Date of the log file	Type of web site
[195]	2.98 / 6.24	December, 1997	AOL web servers
	3.86 / 6.08	August, 1997	Xerox Corporation
[180]	5.96 / 3.0	September, 1998	web servers at INRIA, a French national research institution
[196]	3.65 / 2.69	September, 2002	European web Portal
[197]	4.75 / 3.08	September, 1999	msnbc.com
[198]	Only μ available: 2.08, 2.12, 2.26, 2.5, 2.72, 2.82, 2.85, 2.93, 3.1	Not available, paper published in 2007	Educational, Commercial web sites in Chile, US, Spain and Italy

5.2.1.4 Motivation

As previously argued in chapter 3, it will be fairly difficult for an attacker to estimate the page popularity and page viewing times of regular human visitors of either a dynamic or a static web site. We expect that the distribution of web page popularities, page viewing times and sessions' sequence length will be significantly different between human and malicious/unknown web visitors.

Therefore, we propose an extended analytical framework to evaluate the potential of our methodology to separate non-outlier human visitors from outlier malicious/unknown visitors. (Recall that non-outlier and

outlier visitors were defined in section 5.1.4). Namely, we employ two correlation tests, *Kolmogorov–Smirnov* test 2 (K-S2 test) and *Mann-Whitney Rank Sum* test, to measure the differences in distributions and medians, respectively, of web page popularity rankings, page viewing times and number of pages visited during a session. From the point view of web administrators, we believe that they would be most interested in discovering malicious or unknown sessions once they start appearing as human (i.e., by using proper browser-based UASs). In that case, the administrator should, from the entire group of human sessions, be able to quickly identify those that are, in fact, malicious by evaluating whether their browsing behaviour is *significantly different* from the browsing behaviour of non-outlier human visitors in terms of the 3 features discussed above. On the other hand, in the case of outlier human sessions, web administrators would be mostly interested in evaluating whether their browsing behaviour is (or is not) *significantly different* from non-outlier malicious behaviour, in which case the likelihood that these sessions are actually malicious would be rather small.

Additionally, as a part of our study, we compare the parameters (α, v_{min}, μ and λ) of Zipf, Pareto and Inverse Gaussian distributions derived with our fairly recent web log traces against the distribution parameters previously reported in the research literature. (Note that the traffic traces are 8 years or older in previous research works listed in Tables 5.22-5.24.)

5.2.2 Correlation Analysis of CSE, YORKU and SharpSchool Datasets

In this section, we present the results of the correlation analysis of outlier and non-outlier web sessions extracted from CSE, YORKU and SharpSchool web server logs.

5.2.2.1 Time series data

For the purposes of correlation analysis we have collected three sets of data: the web page popularity rankings, the web page visiting/viewing times and browsing lengths of each session. The datasets were collected for non-outlier human sessions, outlier human sessions, non-outlier malicious sessions, outlier malicious sessions and outlier unknown sessions. Note that in our datasets we only include requests for primary web pages - as defined in section 3.2.3. We do not include the secondary content requests as these are usually indirectly requested by a user when downloading a HTML page via a web browser.

5.2.2.2 Correlation metrics

In this study we employ the following 2 nonparametric statistical tests (from [167]):

1) Mann-Whitney Rank Sum Test which detects the significant difference between two sample medians.
2) Kolmogorov-Smirnov Test of 2 Independent Samples, which measures the significant difference at any point along the two cumulative frequency distributions derived from two samples.

The short overview of these two techniques is presented in Appendix C.

Note that these two correlation tests apply two different techniques to evaluate the differences between the given samples. By applying both techniques, we tend to provide a more holistic evaluation of the statistical differences between various session types examined in our analysis.

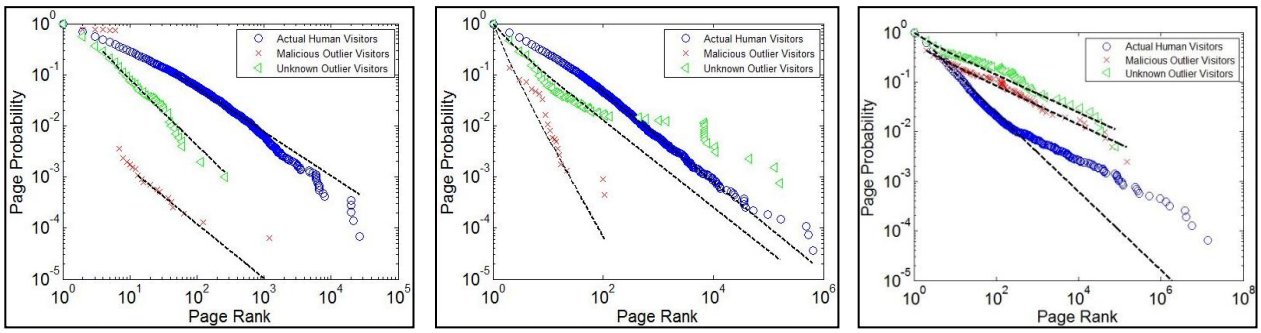
5.2.2.3 Results of the correlation analysis

In this section, we present the results of the correlation analysis of CSE, YORKU and SharpSchool web log data. We also plot the three distributions in log-log space (Figures 5.13 to 5.18). Log-log space reveals

the differences in the various exponential distribution families that are just not apparent by viewing a non-log-log plot because all such distributions have long tails. Also, we document the parameters of the three statistics and show the results of the two statistical tests: Mann-Whitney Rank Sum and Kolmogorov-Smirnov 2 independent sample test. The main conclusions derived from the observed results are presented in the last section of this chapter, 5.2.3.

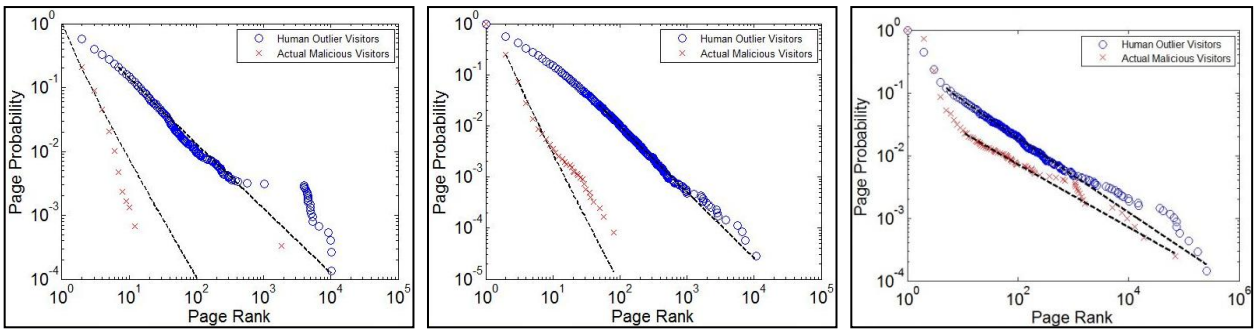
a) Web Page Popularity Rankings

The Mann-Whitney Rank Sum and Kolmogorov-Smirnov tests were employed to investigate whether the differences in the web page popularity ranks are significant in Figures 5.13 and 5.14. Note that in these graphs, the empirical distribution of web page popularity ranks for various session types are shown with either blue, red or green markings, depending on the session type. The dashed black lines represent the fitted PMFs of data with slope equal to exponent α in the Zipf-Mandelbrot's PMF. Now, if we examine each graph individually in Figures 5.13 and 5.14, we can observe that the fitted PMFs of data (dashed lines) are quite varied between different session types.



a) CSE data b) YORKU data c) Sharpschool data

Figure 5.13 Log-log representation of Zipf-Mandelbrot's PMFs describing the web page popularity ranks for non-outlier human, outlier malicious and outlier unknown visitors



a) CSE data b) YORKU data c) Sharpschool data

Figure 5.14 – Log-log representation of Zipf-Mandelbrot's PMFs describing the web page popularity ranks for outlier human and non-outlier malicious visitors

Table 5.25 – Correlation metric scores between various session types extracted from CSE web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z-score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**-5.38**	**0.36 / 0.015**
Non-outlier Human vs. Outlier Unknown	**-14.53**	**0.22 / 0.044**
Non-outlier Human vs. Non-outlier Malicious	**-36.2**	**0.36 / 0.029**

Table 5.26 – Correlation metric scores between various session types extracted from YORKU web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**-45.55**	**0.534 / 0.03**
Non-outlier Human vs. Outlier Unknown	**-19.1**	**0.25 / 0.038**
Outlier Human vs. Non-outlier Malicious	**-69.14**	**0.34 / 0.014**

Table 5.27 – Correlation metric scores between various session types extracted from Sharpschool web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**3.56**	**0.177 / 0.068**
Non-outlier Human vs. Outlier Unknown	**-4.51**	**0.21 / 0.096**
Outlier Human vs. Non-outlier Malicious	**-18.65**	**0.29 / 0.027**

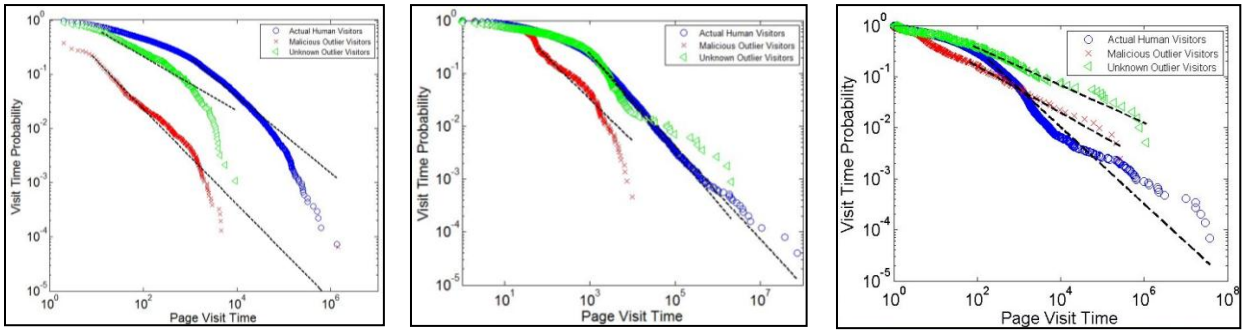
The α parameters (part of Equation 5.1) for Zipf-Mandelbrot's PMFs shown in Figures 5.13 and 5.14 are displayed in Table 5.28. These two tables also list the size of each sample. The α parameters are derived utilizing the method of maximum likelihood described in [199].

Table 5.28 – α parameters of Zipf-Mandelbrot's PMF and sample sizes for various session types extracted from CSE/YORKU/SharpSchool web logs

Session Types		CSE web log		YORKU log		SharpSchool log	
		α	**Sample size**	**α**	**Sample size**	**α**	**Sample size**
Human Sessions	Non-outlier	1.86	14447	1.88	27209	1.79	15694
	Outlier	2.01	7501	2.29	35176	1.59	6894
Malicious Sessions	Non-outlier	2.81	2981	3.6	12279	1.5	4087
	Outlier	2.06	15530	2.9	2227	1.39	410
Unknown Sessions	Outlier	2.26	1009	1.85	1310	1.38	201

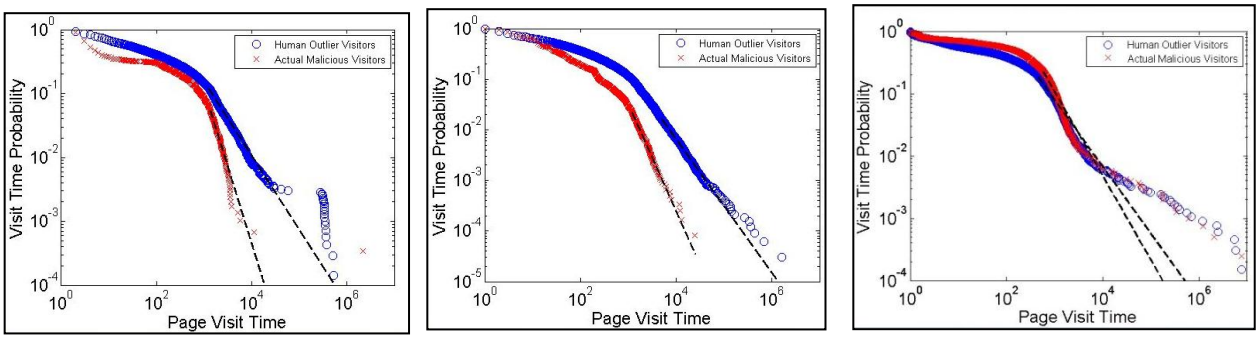
b) Web Page Viewing Time Results

The distributions of web page viewing times for non-outlier human, outlier malicious and outlier unknown visitors are shown in Figure 5.15 and for outlier human and non-outlier malicious visitors are shown in Figure 5.16. As in the previous graphs, the dashed black lines represent the fitted PDFs of data with slope equal to exponent α in the Pareto's PDF.



a) CSE data b) YORKU data c) SharpSchool data

Figure 5.15 – Log-log representation of Pareto's PDFs describing the web page visiting times for non-outlier human visitors, outlier malicious and outlier unknown visitors



a) CSE data b) YORKU data c) SharpSchool data

Figure 5.16 – Log-log representation of Pareto's PDFs describing the web page visiting times for outlier human and non-outlier malicious visitors

Table 5.29 – Correlation metric scores between various session types extracted from CSE web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z-score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**119.3**	**0.62 / 0.016**
Non-outlier Human vs. Outlier Unknown	**-14.7**	**0.22 / 0.046**
Outlier Human vs. Non-outlier Malicious	**-23.48**	**0.33 / 0.03**

Table 5.30 – Correlation metric scores between various session types extracted from YORKU web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**-27.77**	**0.36 / 0.03**
Non-outlier Human vs. Outlier Unknown	**-2.86**	**0.1 / 0.04**
Outlier Human vs. Non-outlier Malicious	**-22.58**	**0.18 / 0.015**

Table 5.31 – Correlation metric scores between various session types extracted from SharpSchool web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**8.84**	**0.28/0.068**
Non-outlier Human vs. Outlier Unknown	**-1.38**	**0.11/0.1**
Outlier Human vs. Non-outlier Malicious	**-14.27**	**0.16/0.027**

The α and v_{min} parameters (part of Equation 5.2) of Pareto's PDFs shown in Figures 5.15 and 5.16 are displayed in Table 5.32. The α and v_{min} parameters are derived utilizing the method described in [199].

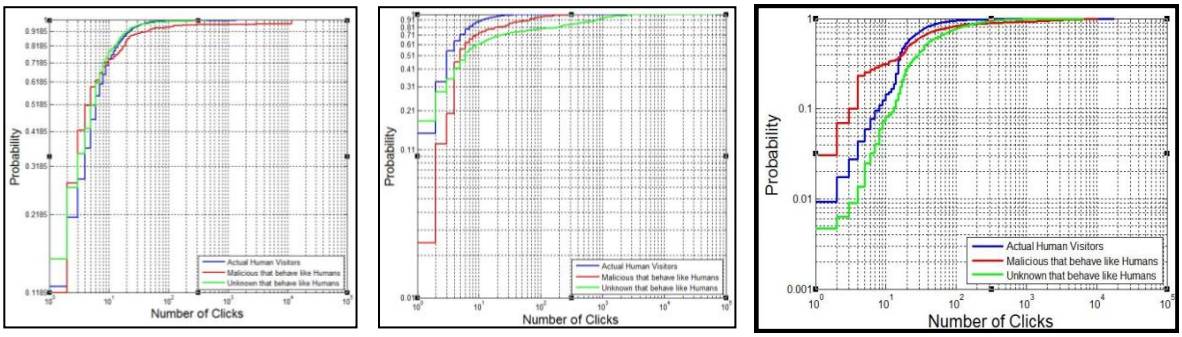
Table 5.32 – α and v_{min} parameters of Pareto's PDF and sample sizes for various session types extracted from CSE/YORKU/SharpSchool web logs

Session Types		CSE web log			YORKU web log			SharpSchool web log		
		α	**v_{min} (s)**	**Sample size**	**α**	**v_{min} (s)**	**Sample size**	**α**	**v_{min} (s)**	**Sample size**
Human Sessions	Non-outlier	1.71	1086	13613	1.88	1356	25139	1.75	154	14573
	Outlier	2.16	1124	6981	2.32	4544	32846	2.07	405	6596
Malicious Sessions	Non-outlier	3.4	1302	2936	3.12	1125	12016	2.35	540	4035
	Outlier	1.88	8	15480	1.79	58	2168	1.45	58	406
Unknown Sessions	Outlier	1.5	13	934	1.94	900	1130	1.37	68	196

c) Browsing Sequence Length Results

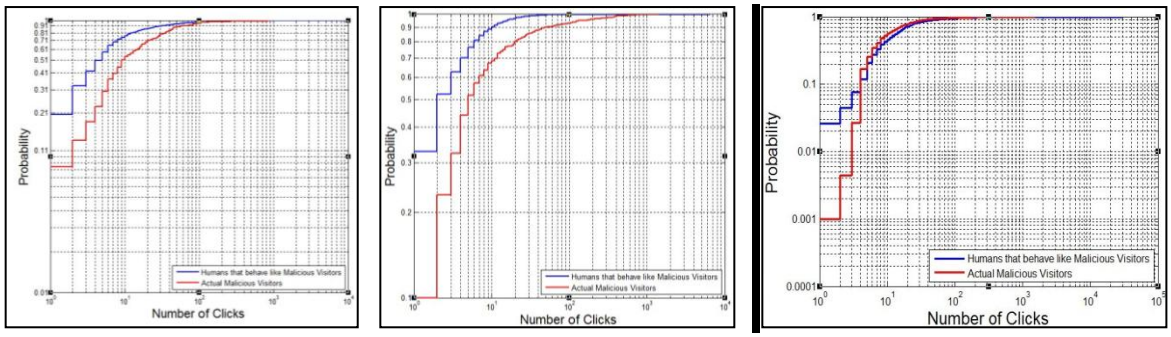
The distributions of browsing sequence lengths for non-outlier human visitors, outlier malicious visitors and outlier unknown visitors are shown in Figure 5.17 and for outlier human visitors and non-outlier malicious visitors are shown in Figure 5.18.

The results of the two statistical tests between the session types extracted from CSE/YORKU/SharpSchool web log data are shown in Tables 5.33, 5.34 and 5.35 respectfully. The significance of the difference with the browsing length metric is much less pronounced than with the other metrics discussed in the previous two sections. For instance, note that the U and K-S statistic scores in Tables 5.33-5.35 are somewhat smaller proportionally than the U and K-S statistics scores reported in the previous two sections. This implies that the length of a sequence belonging to a non-outlier and outlier visitor is not as different as in the case of web page popularity rankings and web page viewing times.



a) CSE data b) YORKU data c) SharpSchool data

Figure 5.17 – Log-log representation of Inverse Gaussian's CDFs describing the browsing lengths of the sessions for non-outlier human visitors, outlier malicious visitors and outlier unknown visitors



a) CSE data b) YORKU data c) SharpSchool data

Figure 5.18 – Log-log representation of Inverse Gaussian's CDFs describing the browsing lengths of the sessions of human outlier visitors and non-outlier malicious visitors

Table 5.33 – Correlation metric scores between various session types extracted from CSE web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z-score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**-1.97**	**0.14 / 0.09**
Non-outlier Human vs. Outlier Unknown	**-2.92**	**0.066 / 0.065**
Outlier Human vs. Non-outlier Malicious	**8.43**	**0.28 / 0.1**

Table 5.34 – Correlation metric scores between various session types extracted from YORKU web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**8.24**	**0.35 / 0.11**
Non-outlier Human vs. Outlier Unknown	**19.38**	**0.28 / 0.034**
Outlier Human vs. Non-outlier Malicious	**16.98**	**0.3 / 0.058**

Table 5.35 – Correlation metric scores between various session types extracted from SharpSchool web log data (bold-lettered values indicate that the two samples are uncorrelated with 95% confidence when the given correlation metrics are applied)

Session Type Comparisons	Mann-Whitney z-score	Kolmogorov-Smirnov K-S statistic / K statistic
Non-outlier Human vs. Outlier Malicious	**-1.6**	**0.2/0.043**
Non-outlier Human vs. Outlier Unknown	**-29.37**	**0.31/0.031**
Outlier Human vs. Non-outlier Malicious	**13.79**	**0.11/0.02**

The μ and λ parameters (from Equation 5.3) of Inverse Gaussian's CDFs shown in Figure 5.17 and 5.18 are displayed in Table 5.36. The μ and λ parameters are derived utilizing the maximum likelihood estimation function provided with Matlab software package.

Table 5.36 – μ and λ parameters of Inverse Gaussian's CDF and sample sizes for various session types extracted from CSE, YORKU and SharpSchool web logs

Session Types		CSE web log			YORKU web log			SharpSchool web log		
		μ	**λ**	**Sample size**	**μ**	**λ**	**Sample size**	**μ**	**λ**	**Sample size**
Human Sessions	Non-outlier	9.63	5.53	67899	5.73	4.84	563661	56.1	18.05	617476
	Outlier	25.61	3.03	6849	5.1	3.13	63636	48.05	8.88	24240
Malicious Sessions	Non-outlier	31.36	5.92	190	32	4.08	715	24.58	11.66	6060
	Outlier	360.7	3.2	211	21.57	5.81	165	362.8	8.07	1017
Unknown Sessions	Outlier	10.04	4.45	435	271.8	3.27	1622	120.03	28.7	1898

5.2.3 Summary of the Results and Concluding Remarks

We have made the following main conclusions from the results shown in Tables 5.25-5.36 and Figures 5.13-5.18:

1. **Non-outlier human vs. outlier malicious/unknown sessions.** The results derived from applying the Mann-Whitney Rank Sum test show that the medians of web page popularity rankings, page viewing

times and browsing sequence lengths metrics are significantly different between non-outlier human visitors and outlier malicious/unknown sessions. Furthermore, the results derived from applying the Kolmogorov-Smirnov 2 independent sample test show that the web page popularity rankings, page viewing times and browsing sequence lengths metrics of non-outlier human visitors and outlier malicious/unknown sessions are derived from different populations. As such, these results have a great practical significance. Namely, they suggest that in the case that the outlier malicious sessions were marked by a spoofed user agent string – which would make them less 'obvious' and not as easily identifiable by the SOM algorithm – the use of the correlation analysis would provide for an effective way of distinguishing them from true human sessions.

2. **Non-outlier malicious vs. outlier human sessions.** The results derived from applying the Mann-Whitney Rank Sum test show that the medians of the three statistical metrics are significantly different between outlier human visitors and non-outlier malicious sessions. The results derived from applying the Kolmogorov-Smirnov 2 independent sample test show that the three statistical metrics of outlier human visitors and non-outlier malicious sessions are also derived from different populations. These results may be an indication that the outlier human session, as identified by the SOM algorithm (see section 5.1.3.1), are not actually malicious but instead may be generated by legitimate human visitors that happen to exhibit non-typical human browsing behaviour.
3. Lastly, by examining Tables 5.28, 5.32 and 5.36, we have discovered that the parameters of Zipf-Mandelbrot, Pareto and Inverse Gaussian distributions are significantly different from the same distribution parameters reported in the past research literature (refer to Tables 5.22-5.24). This result provides further evidence that browsing characteristics of human visitors and various crawler visitors vary among different web domains and also over time.

5.3 Summary

Our work, up to this point, has focused on detecting suspicious web visitors offline[10] and assuming a very stable/static web domains. Same can be said about previous research on this problem, as already outlined in Sections 3.3 and 3.4. However, such an approach will not be effective in real-time where the web sessions continuously arrive at a very high data rate. In real-time detection of suspicious web visitors, it will not be possible/feasible to store the entire dataset for the classification/clustering tasks and to repeatedly examine the data instances in the dataset. Most importantly, in dynamic web domains, some of the data will have to discarded over time and may be harmful in the clustering process as the web content of the web site is changed or updated. Recall our discussion from chapter 3 that as the web content changes the browsing behaviour of visitors of that site is expected to change as well.

Thus, in the next chapter, we present our next-generation system for real-time detection of various HTTP-based DDoS attacks in both static and dynamic web domains.

[10] In offline machine learning, the classification model is built once, after the training phase, and assuming the presence of the entire dataset that can be examined repeatedly.

Chapter 6 Next-Generation System for HTTP-based DDoS Defence

In this chapter, we present our system for detection of HTTP-based DDoS attacks. In section 6.1, we provide a general overview of our system and also provide a brief summary of the methodology used to detect the most sophisticated attacks - outlier detection algorithms for data streams with concept drift. In section 6.2, we present a detail overview of the 3-stages of detection employed in our system. In section 6.3, we present the details on the performance evaluation of our system and finally, in section 6.4, we summarize our main findings.

6.1 System Overview

Our system employs a 3-stage approach for detection and protection of *trivial*, *intermediate* and *advanced* variants of HTTP-based DDoS attacks, respectively (recall Table 3.1). The high-level workflow of our proposed application-layer anti-DDoS system is shown in Figure 6.1. An overview of different possible categories of browsing behaviour (both human and bot), and the specific stages of our system in which each of these categories are likely to be dealt with, is presented in Figure 6.2. (Note that Figure 6.2 is the modified version of Figure 3.3).

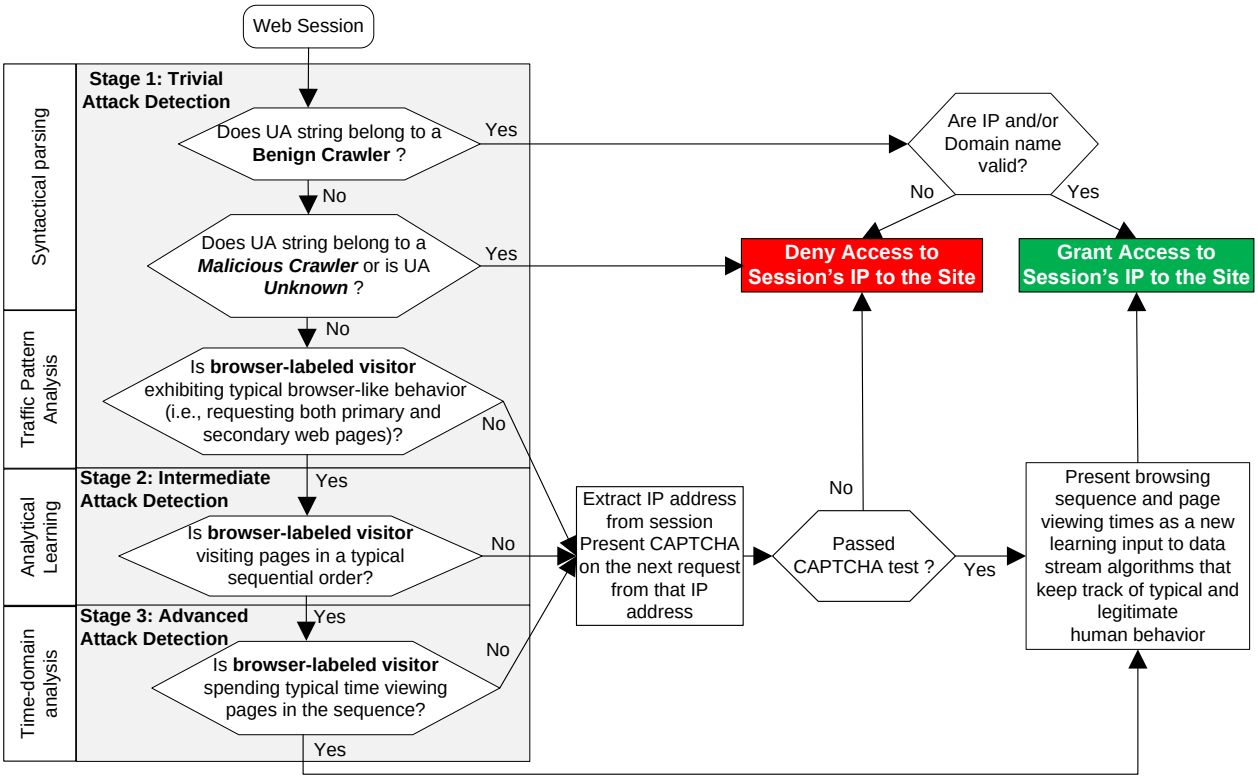


Figure 6.1 – The workflow of the next-generation anti-DDoS system

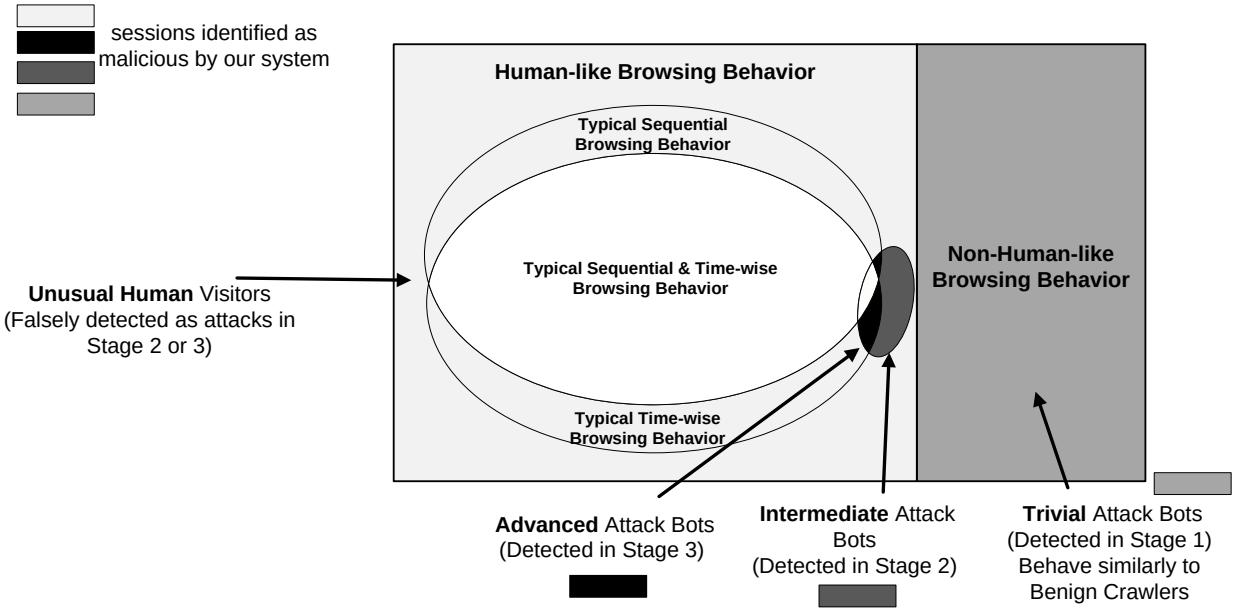


Figure 6.2 – Types of bots detected in our system and their browsing behaviour characteristics

As indicated in Figure 6.1, all categories of *trivial* (non-browser-like) bot behaviour are likely to be detected in Stage 1. This is accomplished by employing syntactical log parsing and traffic pattern analysis techniques described in sections 3.2.5 and 4.1. Bots that exhibit *intermediate attack* behaviour (i.e., generate browser-like HTTP requests, but these requests arrive in a sequence that is atypical of normal/legitimate human browsing) are likely to be detected in Stage 2. The most sophisticated types of attacks (*advanced attacks*), in which the bots generate browser-like HTTP requests in sequences that resemble legitimate human browsing but with atypical page stay-times[11], are likely to be detected in Stage 3. The last two stages employ analytical techniques from outlier detection in data streams with concept drift. The brief overview of these techniques and our motivation for using them are described in section 6.1.2.

Note that given that our system operates on the principle of 'outlier detection', there is a possibility that some actually legitimate but very atypical instances of truly human sessions could initially be labeled as 'outlier' (i.e., suspicious) by our system. As shown in Figure 6.1, users generating such sessions will be presented with a challenge-response test (e.g., CAPTCHA), in order to resolve any uncertainty about their true status. If CAPTCHA test is successfully passed, the 'outlier' label will be removed from the respective session and, furthermore, the sessions will be used to refine what currently constitutes legitimate human behaviour. We would like to emphasize, however, that our system resorts to the use of CAPTCHA only in case of human sessions that are shown to considerably deviate from the majority of previously seen human sessions.

[11] Recall, automated estimation of true human-like web-page stay-times is a particularly challenging task for a bot/attacker, as the stay-time on a web-page depends not only on the byte-wise amount of information found in the given page but also on the contextual popularity/importance of that information to the human population visiting the given page.

6.1.1 Threat Model and System Limitations

In this section, we outline the characteristics of the threat model and limitations of our system.

6.1.1.1 Adversary/Bot Sophistication

The trivial, intermediate and advanced HTTP-based DDoS bots will be detected in one of the 3 stages of the system. Namely, trivial bots will be detected in the first stage, intermediate bots in the second stage and advanced bots in the third stage. As such, it is assumed that all three types of bots either do not visit pages in a typical sequential order of human web visitors or do not spend typical time viewing pages in the sequence as human web visitors.

6.1.1.2 Expected Attack Traffic Rate

It is assumed that our system can provide protection from attacks where the attack rate is as much as 99 times the rate of legitimate (human) traffic. Note that this implies that out of every 100 packets received by the system, 99 packets will be generated by bots while only 1 packet will be generated by a human visitor. The severity of the attack rates were based on our own assumptions as previous surveys on real-world DDoS attacks (from Akamai [45] and Arbor Networks [4]) did not provide the ratio of attack traffic to legitimate traffic. Additionally, it is assumed that the total amount of both legitimate and attack traffic is less than or equal to the overall system traffic capacity.

6.1.1.3 Expected Attack Duration

We assume that the DDoS attack duration will be less than 60 minutes. Note that according to the Arbor Network survey on real-world DDoS attacks [4], 91% of DDoS attacks stopped within 60 minutes.

6.1.1.4 Expected Botnet Size

Also, we assume that the attack traffic will be spread out over a number of Botnet machines. Namely, it assumed that each bot will generate attack traffic at a rate similar to regular human visitors. Therefore, in order to generate attack traffic much larger than the legitimate traffic, a large Botnet has to be employed. Note that the reason for this assumption is to justify the motivation for our system as DDoS attacks where an individual bot generates a significantly larger traffic rate than an individual human web visitors can be easily detected with traditional DDoS defenses based on hard-thresholds, as discussed in chapter 3. Previous research [190] has shown that inter-arrival time between web requests of human visitors is on average 1 minute.

6.1.1.5 CAPTCHA Engine Assumptions

As shown in our workflow diagram of our system in Figure 6.1, suspicious traffic is redirected to the CAPTCHA Engine, which is responsible for making the final decision on whether to grant or not to grant further access to the suspicious web visitor. Please note, in case of web sites with server-only processing, each visitor identified as suspicious/outlier by our system would be asked to solve CAPTCHA on the very next request/session to the web server. On the other hand, in case of web sites (i.e., web pages) that assume client-side processing/scripting (e.g., through use of AJAX), a client-side script could be used to immediately prompt the user to solve a CAPTCHA test.

Note that after successfully passing the CAPTCHA test, the visitor could be exempted from solving another CAPTCHA puzzle for a specified amount of time, even if the visitor's subsequent sessions are labelled as outliers by the system. This step will ensure that the actual human visitors will not be additionally bothered by CAPTCHA puzzles in the case, e.g., they repeatedly request the brand new content very recently added to the web site (i.e., their subsequent outlier-labeled sessions will not trigger additional CAPTCHA tests).

6.1.1.6 Types of Web Sites Protected by Our System

It is assumed that our system would provide protection to web sites of non-trivial size. Namely, we assume that a website has a significant number of unique primary web pages and paths through the web site that a visitor can traverse by following hyperlinks from the top main web page. This assumption is made to ensure that bot designers cannot easily guess what are typical browsing sequences that human visitors would follow on a web site. For instance, the 3 web sites that we were able to evaluate our system against (see section 6.3) were of non-trivial size as they have at least 8419 unique paths through the web site and also over 117,060 unique primary web pages.

6.1.1.7 System Deployment

As illustrated in Figure 6.3, our anti-DDoS system could be deployed as a dedicated device or in a combination with a scrubbing/cloud anti-DDoS center (recall the definition of this deployment type from Table 2.2). In the case of a cloud deployment scenario, the traffic to the web site could be rerouted for further processing inside the scrubbing center once the detection system detects the presence of an attack, e.g., when there are a number of "outlier" web visitors that have failed the CAPTCHA test.

Regardless of the deployment scenario, a visitor that fails the CAPTCHA test will have its IP address added to the list of black-holed/blocked IP addresses inside the dynamic Firewall (similarly to [68]). The subsequent requests from that IP address/visitor will be dropped by the Firewall.

Additionally, we assume that our system will be deployed in parallel with the existing commercial anti-DDoS defenses, as overviewed in Section 3.4.2. This approach will ensure that trivial high-volume flooding attacks (such as Random recursive HTTP GET attack discussed in Chapter 3) will be detected before they can inundate our CAPTCHA engine with web requests and cause it to crash. The specifics regarding this parallel deployment are beyond the scope of this thesis.

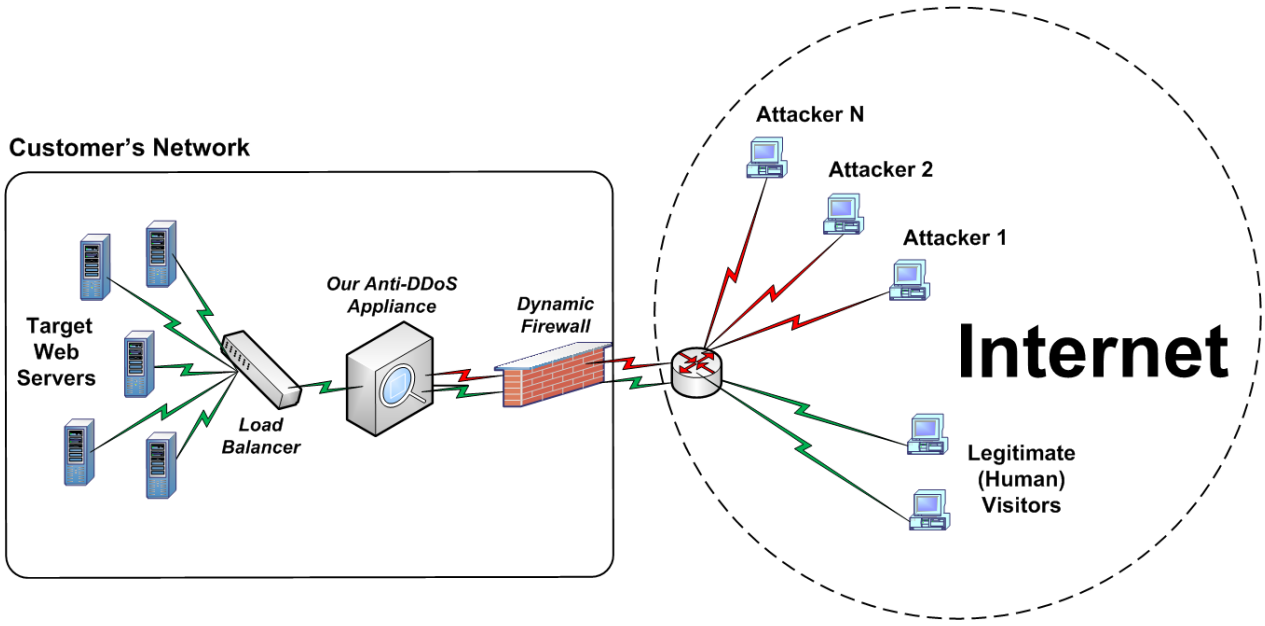


Figure 6.3 - A possible deployment scenario for our next-generation anti-DDoS system

6.1.2 Outlier Detection in Data Streams with Concept Drift - Overview and Motivation

The majority of research in data mining is devoted to static off-line environments, where the hidden patterns of interest are stable (i.e., do not change over time) and each data input can be repeatedly presented to the learning system. However, over the last decade, there has been an increasing interest in managing the so-called data streams − massive, continuously-generated sequences of data that are likely to evolve over time [200]. Real-world examples of data that fall in the category of data streams are: web click-stream data, computer network monitoring data, readings from sensor networks and stock quotes, etc..

The main characteristics of data streams imply the following constraints on the design and operation of algorithms that deal with the processing and learning from data streams (from [201] and [202]):

1. *Bounded Storage* − It is often impractical if not impossible to store all the data from a data stream. In many applications, only small summaries of a data stream can be computed and stored, and the rest of the information must be discarded. The common approach to handling this constraint is by employing a summarization structure, such as:

   (a) Sliding time window ([203] and [204]) − a time-ordered list of *N* most recent data instances observed over the data stream.

   (b) Micro-cluster [205] − a synopsis data structure that, instead of storing all points associated with a cluster, only stores the aggregate statistical information about the data points mapped to the given cluster.

   (c) Density Grid [206] − a partitioning of n-dimensional data space which, similarly to micro-cluster, only maintains the aggregate statistical information about the data mapped onto different cells of the grid.

2. *Real-Time*, *Single-Pass* – The overall volume as well as the arrival speed of data stream instances often implies that each particular element can be processed only once, and in real time. Data stream systems/algorithms must be able to efficiently identify whether the new input is an outlier based on the available past (limited) information. This constraint is generally handled by employing an efficient indexing data structure for rapid processing of arriving data instances, such as M-trees ([203] and [204]) or hash tables [206].

3. *Concept Drift* − The distribution generating the inputs can change/evolve over time due to a drift in the underlying concept. I.e., data from the past may become irrelevant or even harmful (i.e., misguiding) for the current summary. Data stream systems/algorithms must be able to recognize and adequately adjust to such a phenomenon. There are two major types of concept drifts: sudden and gradual. In a sudden drift a new input distribution abruptly emerges and quickly overtakes the previously-observed input distribution while in a gradual drift this processes is more subtle and happens over a longer period

of time. The problems associated with the concept drift phenomenon are typically addressed by one of the following approaches:

(a) Employing a sliding time window where all elements are given the same weight ([203] and [204]). This technique is more suited for slow/gradual drifts where the new and the old input-generating distributions are more equally to be 'intertwined' in time. However, a more suitable approach would be to apply a decaying weight function over the elements in the sliding time window.

(b) Assigning a decaying weight function to the synopsis data structure such as a micro-cluster [205] or a density-grid [206]. The weight of a micro-cluster/grid is directly related to the arrival time of points placed inside the micro-cluster/grid. Also, a micro-cluster/grid is removed from the list of existing clusters if its weight falls below a specified threshold. This approach ensures that old and no-longer-relevant clusters do not influence the future learning as the data stream input distribution evolves over time.

6.1.2.1 Data Stream with Concept Drift as Pertaining to Detection of Application Layer DDoS in Dynamic Web Domains: Motivation for Our Approach

In web domains that experience high volumes of data and undergo dynamic changes, the problem of HTTP-based DDoS defense against the current and next generation attacks naturally falls within the framework of *unsupervised learning for data streams with concept drift*. Namely, a) the notion of '*data streams with concept drift*' is clearly applicable to this problem due to the very nature of the assumed domain (high-volume and dynamically changing), b) the problem is '*unsupervised*', as the knowledge of what constitutes normal vs. attack behaviour is not known in advance.

Now, while building/training an application-layer anti-DDoS system for a dynamic web domain that receives high volumes of web requests, it may be redundant to keep track of all patterns of past user

behaviour. Namely, in dynamic web domains the browsing behaviour of legitimate/human visitors is likely to change over time, and therefore only the most recently observed trend(s) will be helpful in correctly distinguishing between the typical/non-outlier human and any atypical/outlier malicious bot behaviour.

From the implementation point of view, distinguishing between the emergence of new legitimate (i.e., benign) human sessions and new malicious sessions could be a challenging task. Specifically, as the nature of legitimate user behaviour changes, the new emerging groups/profiles will inevitably, at least initially, be identified as outliers - the same way that the new emerging attack profiles/sessions will also be identified as new outliers. For that reason, the sub-category of *unsupervised data stream with concept drift* algorithms most suited to deal with the given problem (identification and separation of benign vs. malicious outliers) are algorithms that specialize in outlier detection.

It should be observed, however, that the outlier detection algorithms alone are effective in (only) identification of newly-emerging outliers. However, to facilitate effective separation of newly identified outliers into benign vs. malicious, in our work we propose (very limited) use of CAPTCHA test. If this session is an instance of emerging and truly human behaviour, then upon successfully solved CAPTCHA test the session (i.e., the respective outlier) will be labeled ‘valid’. After a number of subsequent alike sessions also pass the CAPTCHA test, their 'outlier' status will change to non-outlier and they will start forming a proper/non-outlier clusters. Once this happens, all subsequent sessions belonging to (i.e., falling within) this cluster will no longer be presented with the CAPTCHA test. On the other hand, as discussed in chapter 3, malicious bots pretending to be human visitors by spoofing the UAS, are generally not able to fully model typical human behaviour, and will be failing the CAPTCHA test. Consequently, such sessions will end up being labeled as ‘malicious’ outliers. By confirming that these outlier web visitors are in fact malicious bots with the help of CAPTCHA tests, our technique ensures that in the case of high-volume

DDoS attacks - where the bot sessions constitute the majority of incoming sessions - such sessions will be 'rejected' and will not be employed to refine, i.e., distort the profile(s) of typical valid human browsing behaviour.

We would like to emphasize that our system falls in the category of semi-supervised machine learning algorithms as it only performs machine learning using feedback from the CAPTCHA engine.

6.1.2.2 Techniques for Outlier Detection in Data Streams with Concept Drift

The taxonomy of outlier detection algorithms for data streams with concept drift is presented in Table 6.1 (adapted from [207]).

Each algorithm is characterized by the type of technique employed to address the bound memory, real-time/single pass and concept drift constraints of data streams. Also, we present the time and processing complexity for labeling an arriving data stream instance as an outlier and the space complexity of the utilized summarization model/synopsis data structure. Note that this is the comprehensive list of outlier detection algorithms for data streams that address all of the 3 constraints of data streams: bounded storage, single-pass processing and concept drift handling.

Table 6.1 – The taxonomy of outlier detection algorithms for data streams with concept drift

n: number of data points/instances, m: number of micro-clusters, g: number of grids

Category	Algorithm	Bounded Storage & Concept Drift	Single-pass / Real-time Technique	Time Complexity	Space Complexity
Distance-based	**COD** [203]	fixed sliding time window model with equal weights	M-tree	O(logn)	O(n)
Density-based	**ILOF** [204]	fixed sliding time window model with equal weights	M-tree	O(nlogn)	$O(n^2)$
	Denstream [205]	Weighted decaying function over micro-clusters	Micro-cluster	O(m)	O(m)
	D-Stream [206]	Weighted decaying function over density-grids	Density grid	O(1)	O(g)

As shown in Table 6.1, the four algorithms employ unique strategies to detect outliers in data streams with concept drift. All four algorithms satisfy the three constraints of data streams: bounded memory, real-time/single pass processing and concept drift reactivity. (The four algorithms are described in more detail in Appendix D). Our system employs modified versions of COD, ILOF, Denstream and D-Stream to detect intermediate and advanced HTTP-based DDoS attacks in stages 2 and 3. In the following sections we describe the 3 stages of our system in more detail.

6.2 Implementation of the 3 Stages of Our System

In this section, we present the inner-workings of each stage of our system.

6.2.1 Stage 1: Trivial Attack Detection

The purpose of Stage 1 is to detect (i.e., eliminate from further processing and analysis) the three types of *trivial attack* sessions, using the methodology previously discussed in section 3.2 in chapter 3. Sessions that are identified as malicious by Stage 1 are eliminated from further processing - i.e., are not passed to Stage 2. The workflow diagram illustrating the series of steps in Stage 1 is shown in Figure 6.4.

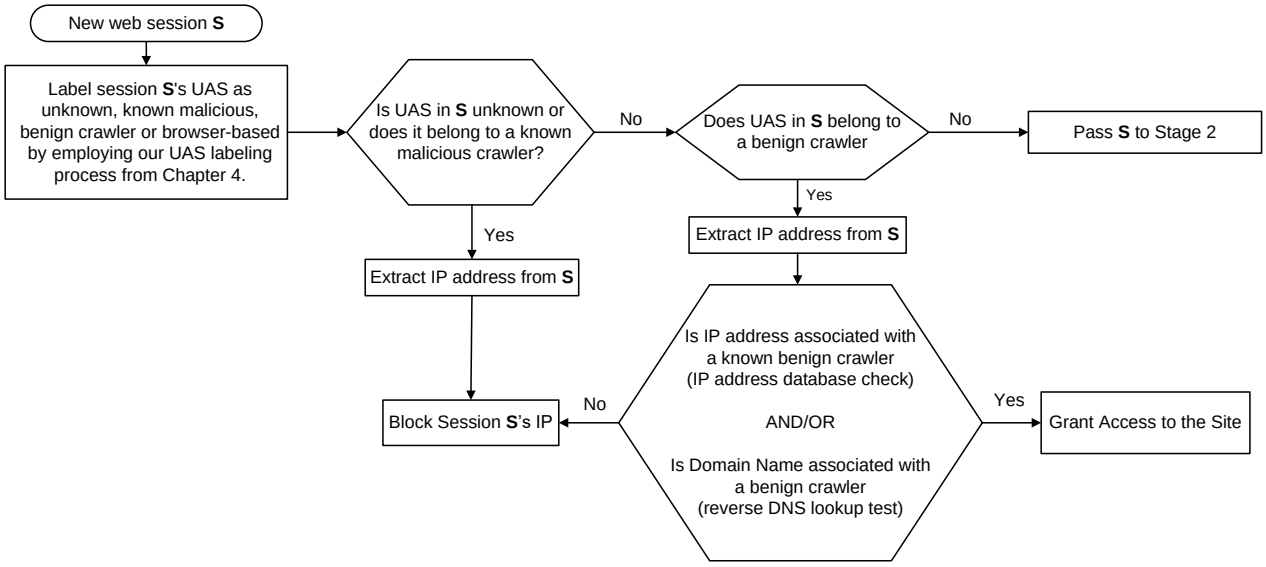


Figure 6.4 – The outline of the steps in the Stage 1

In general, our system can be adapted to optimally operate for each particular web site and its respective visitor population. For example, if the web site being protected is an university web site, web administrator has the option to allow benign known search engine crawlers to index their web site. Alternatively, if the web site being protected is an online content management application, with visitors that are exclusively users of the application, web administrator has the option to choose to block everyone but human visitors since there is no need for search engines crawlers to index this type of a domain.

6.2.2 Stage 2: Intermediate Attack Detection

In Stage 2, the system specializes in capturing *intermediate attack* sessions. As indicated in section 3.2.5, these types of attack sessions could be detected by examining their chronological browsing sequences (i.e., the sequences of actually requested web pages). Thus, in Stage 2, the system first extracts the chronological browsing sequence (CBS) from each session passed to it from Stage 1. Subsequently, on

such formed CBS, Stage 2 employs data stream algorithms from Table 6.1 (COD, ILOF and our novel density-based algorithm for outlier CBS detection we refer to as Density Stream for Sequences (DSS) algorithm - described in 6.2.2.2) in order to update what constitutes the currently typical/legitimate CBS profile(s) as well as to keep track of new/emerging CBS profiles. The workflow diagram illustrating the series of steps in Stage 2 is shown in Figure 6.5.

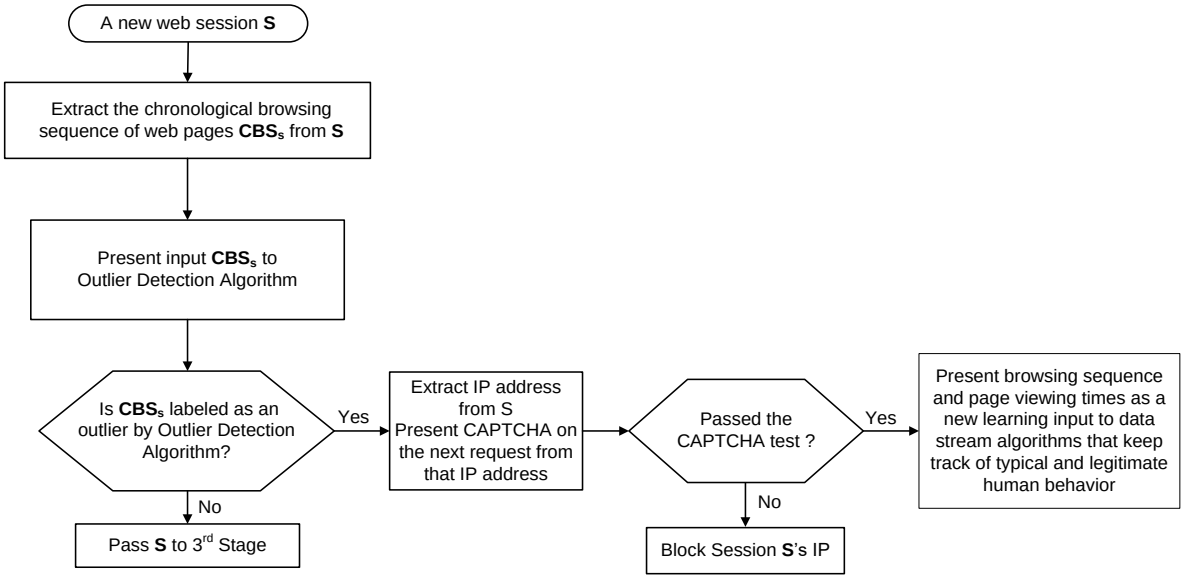


Figure 6.5 – The outline of steps in Stage 2

As indicated in Figure 6.5, for CBSs that are marked as 'outlier' (suspicious) by Stage 2, the respective sessions are not passed to Stage 3, and the respective users are presented with a CAPTCHA. Clearly, if those users happen to be malicious bots, they will fail the CAPTCHA test and be denied a subsequent access to the site. Otherwise, if those users are legitimate human visitors, and they successfully pass the CAPTCHA test, their respective sessions/sequences will be incorporated into the typical human browsing profile(s).

6.2.2.1 Chronological Browsing Sequence of Web Page Requests

Firstly, we formally define the CBS. Assume that a web site has m distinct primary pages or URLs. Let us also index these pages in decimal notation sequentially as 1, 2, ..., m. Then, the chronological browsing sequence of n visited primary web pages during web session S is defined in (6.1).

$$CBS_S = [a_1, a_2, ..., a_n] \quad (6.1)$$

where $a_k \epsilon \{1, 2, ..., m\}$ is the k^{th} page visited in the sequence in web session S and $1 <= n <= m$. Recall the definition of a web session that no primary web page request may repeat in a web session. Therefore, the maximum length of the CBS can be at the most equal to the number of pages on the web site m. Note that CBS has been employed previously in intrusion detection studies such as [208], [209] and [210].

6.2.2.2 Outlier Detection Algorithm for Chronological Browsing Sequences

The input to this stage is the stream of CBSs extracted from incoming web sessions. Possible choices of algorithms for detecting outlier sequences include COD-Sequence, ILOF-Sequence and our own DSS algorithm - outlined in the following sections.

a) COD-Sequence - Modified COD Algorithm for Sequence Outlier Detection

The COD-Sequence algorithm is the slightly modified version of the original COD algorithm from [203] that employs the similarity distance metric called normalized length of the longest common subsequence[12] (NLLCS) between a pair of CBSs in the sliding time window. (Note that this distance metric has been

[12] The *longest common subsequence (LCS) problem* is defined as the search for the subsequence common to all sequences in a set of two or more sequences. In mathematics, the subsequence is defined as a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. For example, the LCS of two strings "ACBDEGCEDBG", "BEGCFEUBK" is the string "BEGCEB" of length 6. The algorithm for calculating the LCS between two sequences x and y can be implemented with a dynamic programming algorithm in O(|x|+|y|) time, where |x| and |y| are the lengths of two sequences [237].

employed for detection of outlier sequences in intrusion detection problems in [211], [212] and [213].) The similarity between two browsing sequences CBS_i and CBS_j, is computed as follows:

$$NLLCS(CBS_i, CBS_j) = 1 - \frac{|LCS(CBS_i, CBS_j)|}{\sqrt{|CBS_i||CBS_j|}} \quad (6.2)$$

The value of $NLLCS(CBS_i, CBS_j)$ ranges from 1, if CBS_i and CBS_j have no common subsequence, and 0, if CBS_i and CBS_j are identical.

b) ILOF-Sequence - Modified ILOF Algorithm for Sequence Outlier Detection

Similarly, ILOF-Sequence algorithm is the original ILOF algorithm from [203] that employs the NLLCS similarity measure (Expression 6.2) instead of Euclidean distance to represent the distances between the CBSs in the sliding time window.

c) DSS - Density Stream for Sequences Algorithm Based on the Denstream and D-Stream Algorithms

The DSS algorithm combines the micro-cluster synopsis structure from Denstream with decaying factor function from D-Stream for detection of outlier input sequences. Note the original Denstream algorithm cannot be applied on sequences since the computation of the center and radius of the micro-cluster requires that the input is composed of continuous values that satisfy the triangle inequality. To overcome this limitation, we have proposed DSS - a version of the D-Stream algorithm for sequences that employs the following modifications:

Micro-clusters

In DSS, a micro-cluster MC_i consists of a representative CBS denoted as MC_i.RCBS (similar to [211]), decaying cluster density function D(c, t) based on the density grid function from D-Stream (Expressions D.14-D.16 in Appendix D) and radius ε. MC_i.RCBS is the first CBS added to the new micro-cluster (refer

to Figure 6.6). Furthermore, in DSS, we do not label micro-clusters as p- or o-micro-clusters. Instead, each micro-clusters is labeled as dense or sparse as in the D-Stream algorithm.

Online Learning Phase (Insertion of new input CBS_i)

The online learning phase follows the training phase and consists of the insertion of the new input CBS into the micro-clusters. As in the original Denstream algorithm, the number of training data points processed during the training phase is a user-defined parameter.

The algorithm for insertion of new CBSs into micro-clusters in DSS is shown in Figure 6.6. A CBS S_i is merged to the closest micro-cluster MC_m in which the dissimilarity between S_i and MC_m.RCBS is less than radius ε. As in the COD-Sequence and ILOF-Sequence algorithms, NLLCS similarity measure (shown in (6.2)) is employed to calculate similarity/dissimilarity between sequences. If the closest MC_m is not within radius ε , a new micro-cluster MC_n is created with MC_n.RCBS set to S_i.

Maintenance of Micro-clusters

The decaying density function of a micro-cluster in DSS is the same as the density function of a grid in the D-Stream algorithm. Namely, the algorithm promotes sparse and sporadic micro-clusters to dense micro-clusters whenever a micro-cluster's density satisfies the condition in (6.3) and downgrades a dense micro-clusters to sparse micro-clusters if condition in (6.4) is satisfied. The expressions (6.3) and (6.4) are slightly modified version of (D.17) and (D.18), respectively:

$$D(g, t) \geq C_l * \text{AvgMCDensity} \tag{6.3}$$

$$D(g, t) < C_l\left(1 - \lambda^{t-t_g+1}\right) * \text{AvgMCDensity} \tag{6.4}$$

where AvgMCDensity is the average density of micro-clusters at time t. Note that in the original D-Stream algorithm, the average density $\frac{1}{N(1-\lambda)}$ is constant, while in DSS, N changes as micro-clusters are added or removed. Therefore, AvgMCDensity has to be recalculated as new micro-clusters are added or removed.

Insert(CBS_i)
Let MC be the set of micro-clusters
find closest micro-cluster MC_m in MC to CBS_i
  if NLLCS(CBS_i, MC_m.RCBS) < ε
    merge CBS_i to MC_m by updating MC_m's D(c, t) (given in (5.16))
  end if
end for
if CBS_i was not merged with any MC_m in MC
  create new micro-cluster MC_n
  MC_n.RCBS = CBS_i
  add MC_n to MC
end if

Figure 6.6 – The insertion step in DSS

Also, periodically, the algorithm verifies if sparse and sporadic micro-clusters should be discarded. The periodicity of the check T_p is defined in (D.19). The sparse and sporadic micro-cluster is removed if condition in (6.3) is satisfied for two T_p periods.

6.2.3 Stage 3: Advanced Attack Detection

In Stage 3, our system specializes in capturing *advanced attack* sessions. Now, as explained in section 3.2.5 in chapter 3, in *advanced attacks* the attacker is expected to perform a thorough analysis of the target web site, and be in the position to generate a sequence of requests that appear as (possibly) produced by a

legitimate human visitor. E.g., a request for one web page will be followed by a request for another page that is directly linked to it. However, the sequence of requested web-pages is not the only feature that characterizes truly human browsing behaviour. Namely, the actual 'page viewing time' is another unique feature of human browsing – a feature that may be impacted by a multitude of very different and complex factors. We believe that the accurate modeling of page viewing times will remain a particularly challenging problem for the attacker, as it will require a very thorough understanding of: a) the web site's actual subject matters, b) the relevance of those subjects to the site's visitor population, c) the impact the visual organization of information on each web page has on the amount of time a human visitor needs to absorb the given information (before moving to the next page), etc.

Given the above, the detection of *advanced attack* sessions in Stage 3 is based on time-domain analysis of visitors' browsing behaviour. The workflow diagram illustrating the series of steps in Stage 3 is shown in Fig. 6.7.

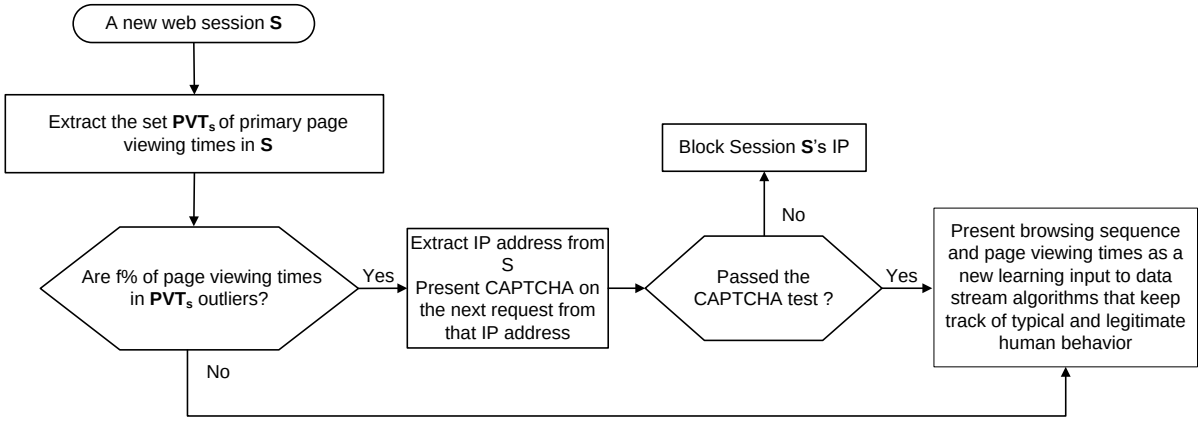


Figure 6.7 – The outline of the steps in Stage 3

6.2.3.1 Page Viewing Times

The collection of page viewing times in a session S is defined as follows. The set PVT is the collection of primary web page viewing times in web session S where each element t(j) in PVT is the inter-arrival time between requests j and j+1 in session S and where j and j+1 are one of the m pages on the web site. Note that if the visitor viewed j>1 pages in a session, we can extract the page viewing times for the first j-1 pages requested.

6.2.3.2 Modeling Typical and Atypical Page Viewing Times

The input to this stage of our system is the stream of PVTs extracted from incoming web session. In particular, the system deploys algorithms from Table 6.1 to identify outlier/low-frequency page viewing times in attacks sessions that are previously marked as typical by Stage 2. Note that the page viewing times must be clustered on per-page basis for each web page on the web site. This implies that the system in this stage must maintain clustering results (i.e., sliding time windows, micro-clusters or density grids) for each individual web page. Also, for each input web session, the system calculates the fraction of page viewing times in that session that are identified as outliers by the algorithm. If this fraction exceeds a system-defined threshold, the session is identified as an outlier (potentially malicious) by Stage 3, and the respective user is requested to solve a CAPTCHA test at the next visit to the web site.

6.3 Evaluation of the System

In this section we present the results of the performance evaluation of the system.

6.3.1 Dataset and Simulation Environment

The system is evaluated against the three sets of real-world web server access log files:

1. Web log traces provided by York University's Computer Science and Engineering (CSE) department.
2. Web log traces from Lewis County School board in Iowa, U.S.A.
3. Web log traces from East Allen County School (EACS) board in Indiana, U.S.A.[13].

As already explained in Chapter 4, we were unable to employ the standard datasets with HTTP packet contents. The CSE log files contain detailed information about user web-based access into the domain www.cse.yorku.ca during the 9-week interval – in June and July of 2014. A total of about 460000 sessions can be extracted from the file. The Lewis log files contain user web-based access into the domain www.lewiscentral.org during the 5-week interval – between mid-August 2013 and October 2013. A total of about 80000 sessions can be extracted from the file. The EACS log files contain user web-based access into the domain www. eacs.k12.in.us during the 5-week interval – between mid-August 2013 and October 2013. A total of about 70000 sessions can be extracted from the file.

The system itself, including the four algorithms from Table 6.1, are implemented in Java. The implementation code for Denstream and COD are borrowed from MOA framework [214]. The simulations were executed on a computer with Intel i7 processor and 8GB of RAM.

6.3.2 Experimental Design

The following 3 synthesized attacks are employed in our evaluation:

- ***Attack 1:*** In this attack, an attacker collects the set of pages directly accessible (through a single hyperlink) from the top (i.e., index.html) page of the CSE/Lewis/EACS web site. Next, the attacker synthesizes a browsing sequence by randomly selecting pages from this set. The attacker also randomly

[13] Note that Lewis and EACS web sites experience the most inbound traffic among over 1500 web sites that are part of Sharpschool web domain.

selects the number of requests in a sequence to be between 1 and 10. The page viewing are sampled with exponential distribution with exponential mean of 1 minute (based on the average page viewing time reported in [190] and employed in [94]), 5 minutes and 10 minutes, respectively.

- ***Attack 2:*** In this attack, the attacker estimates the page popularities by querying the Google search engine. (Note that this type of an attack is described in [215].) The motivation for this attack is that an outsider, completely deprived of any internal insight on the popularity of individual web pages within a particular web domain, would clearly have to rely on external third-party resources (such as Google search) to gather this information. For instance, the web page popularity score can be estimated by the count of returned hits in the Google search for the given page. As in 1), the attacker selects the set of pages directly accessible from the top page of the CSE/Lewis/EACS web site and then orders them in terms of their popularity by the web page popularity score derived from the Google search engine. Next, the attacker synthesizes a browsing sequence by randomly selecting web pages from the set of 10 most popular web pages as derived in the previous step. As in 1), the sequence length is randomly chosen between 1 and 10 and page viewing times are sampled from the three above mentioned exponential distribution.
- ***Attack 3:*** In this attack, the attacker's sequence is synthesized to start from the top (i.e., index.html) web page of the CSE/Lewis/EACS web site and the remaining web pages in the sequence are selected by following a random link between the web pages. As in 1) and 2) the page viewing times are sampled from the exponential distribution and the sequence length is randomly chosen between 1 and 10 web pages or until the sequence reaches the web page with no outgoing link to another CSE/Lewis/EACS page.

Note that all 3 attacks assume that each page request is followed by a series of secondary requests for its embedded content. As such, all 3 attacks are at least of intermediate attacker/bot level. Attack 1 is an intermediate attack as the attacker is not making an attempt to mimic a sequence of web page requests that

would be made by a human visitor while viewing content on a web site. Attacks 2 and 3 are advanced attacks as the attacker is making an attempt to mimic a typical sequential browsing behavior of human web visitors. (Refer to definitions of 3 attack types and typical sequential browsing behavior of human web visitors from Chapter 3.)

To the best of our knowledge, attacks 2 and 3 have never been evaluated against a DDoS defensive system in previous research literature. A variation of the attack 1 have been evaluated in [70], [84], [94] and [99]. Table 6.2 lists the parameters of the synthesized attacks. The progressively challenging attack scenarios have been chosen to illustrate the effectiveness and limits of the presented methodology.

Table 6.2 – The parameters of synthesized attack scenarios

Attack	Number of Web Pages in a Sequence	Sequence Generation Process	Page Viewing Times
1	uniformly distributed between 1 and 10	Randomly select web pages directly accessible from the top (i.e., index.html) web page of CSE/Lewis/EACS web sites	Exponentially distributed with mean of 1, 5 and 10 minutes
2		Randomly select 10 most popular web pages as calculated from Google search engine	
3		Follow random path through the web site by following links between web pages	

Note that the input datasets for each individual experiment are generated by uniformly interleaving the respective 'synthetic' attack sessions into the sample CSE/Lewis/EACS dataset. The percentage of synthesized attack sessions in each dataset is set at 20%, 50% and 99%[14] of all sessions with a browser-labeled UAS in the dataset. In each attack scenario the *same* outlier detection algorithm is employed in both Stage 2 and Stage 3 (e.g., COD-sequence and original COD, ILOF-sequence and original ILOF). **In the case of Denstream and D-Stream scenarios, it is assumed that our DSS was employed in the**

[14] Note that the dataset where 20% of all sessions are attack sessions implies that for every 100 sessions in the dataset, 20 are attack sessions and 80 are actual human sessions. In the dataset where 50% of all sessions are attack session implies that for every 100 sessions in the dataset, 50 are attack sessions and 50 are actual human sessions. In the dataset where 99% of all sessions are attack session implies that for every 100 sessions in the dataset, 99 are attack sessions and 1 is an actual human session.

Stage 2 because of the similarities between the three algorithms. In each scenario, the algorithms in Stages 2 and 3 were trained for the first 15% of total number of regular human sessions in the dataset before the performance results were collected.

6.3.3 Performance Criteria

The attack detection performance of the four algorithms is evaluated by employing the following metrics:

- True Positive Rate (TPR) - percentage of human-generated sessions correctly classified as 'legitimate' by the system (i.e., sessions are identified as belonging to non-outlier clusters in the algorithms).
- True Negative Rate (TNR) - the percentage of synthesized attack sessions correctly classified as 'malicious' by the system (i.e., sessions are identified as belonging to outlier clusters in the algorithms).
- The four algorithms are also evaluated based on the average amount of time required to process simulated attack scenarios.

The following aspects of the operation of the system are also investigated:

- The relationship between the attack sophistication and TPR and TNR rates.
- The relationship between the page viewing time modeling and TPR and TNR rates.
- The relationship between the attack intensity (i.e., the number of attack sessions embedded in the dataset) and TPR and TNR rates.
- The relationship between the four data stream algorithm parameters and TPR and TNR rates.
- The amount of concept drift observed in the 3 datasets.

6.3.4 Experimental Results

The TPR and TNR for 3 simulated attack scenarios, where 20%, 50% and 99% of all sessions in the CSE dataset are attack sessions, are shown in Tables 6.4, 6.5 and 6.6, respectively. The equivalent results pertaining to the Lewis dataset and the EACS dataset are shown in Tables 6.7 to 6.9 and Tables 6.10 to 6.12, respectively. For each dataset, there are 108 distinct scenarios in total. For each scenario, the average TPR/TNR is shown derived from 3 trials.

Note that for each data stream algorithm the maximum achieved TPR and TNR rates are displayed. The deployed algorithm parameters, listed in Table 6.3, were determined empirically. The same algorithm parameters were employed for 20%, 50% and 99% attack scenarios.

Table 6.3 – The parameters of outlier detection algorithms

Algorithm	Parameter Settings
COD-Sequence (Stage 2)	k=2, R=0.2 and N=30000 (in Lewis/EACS datasets) and N=150000 in CSE dataset
COD (Stage 3)	k=1750, R=25, and N=30000 (in Lewis/EACS datasets) and N=150000 in CSE dataset, fOT[15]=0.75
ILOF-Sequence (Stage 2)	k=2, and N=30000 (in Lewis/EACS) and N=150000 in CSE dataset and L[16]=0.1
ILOF (Stage 3)	k=1750, N=30000 (in Lewis/EACS) and N=150000 in CSE dataset, L=0.05 and fOT=0.75
DSS (Stage 2)	ε=0.2 λ=0.95, δ=0.3, C_l=0.5
Denstream (Stage 3)	μ=0.5, λ=0.25, δ=0.5, ε=25 and fOT=0.5
D-Stream (Stage 3)	λ=0.95, δ=0.3, C_l=1000 and fOT =0.5

[15] fOT is the threshold of the fraction of page viewing times in the session that are identified as outliers - recall the description of this parameter in section 6.2.3.2.

[16] Based on the approach from [236], the input instance to ILOF algorithm is labeled as an outlier if its local outlier factor (LOF) value is in the top L percentile of all observed LOF values.

Table 6.4 – The TPR/TNR values in attack 1-3 scenarios with CSE dataset where attack sessions comprise 20% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	88	86	88	90	93	93	93	93
1.0	2	85	82	85	84	92	93	93	94
1.0	3	75	73	75	73	93	93	94	94
5.0	1	89	86	88	90	93	94	94	94
5.0	2	86	82	86	85	93	94	95	96
5.0	3	75	74	75	73	94	95	95	96
10.0	1	90	87	90	91	94	95	95	94
10.0	2	87	83	85	86	94	95	96	97
10.0	3	76	74	76	74	95	96	97	97

Table 6.5 – The TPR/TNR values in attack 1-3 scenarios with CSE dataset where attack sessions comprise 50% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	86	85	86	86	93	93	93	93
1.0	2	85	82	85	84	91	91	93	93
1.0	3	71	70	72	72	90	91	92	91
5.0	1	86	86	86	87	93	94	94	95
5.0	2	86	83	85	86	92	91	93	93
5.0	3	74	71	74	73	91	91	93	91
10.0	1	87	88	89	89	94	95	95	95
10.0	2	87	84	86	86	93	92	94	94
10.0	3	75	72	75	74	92	92	93	92

Table 6.6 – The TPR/TNR values in attack 1-3 scenarios with CSE dataset where attack sessions comprise 99% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	78	77	78	79	92	92	92	93
1.0	2	77	76	77	78	91	91	92	92
1.0	3	62	61	62	63	90	89	90	90
5.0	1	80	80	80	81	94	93	94	94
5.0	2	78	77	78	79	93	92	93	92
5.0	3	62	61	64	64	91	90	91	91
10.0	1	81	81	81	82	94	93	94	94
10.0	2	81	79	80	81	94	92	93	94
10.0	3	64	63	65	65	91	91	92	92

Table 6.7 – The TPR/TNR values in attack 1-3 scenarios with Lewis dataset where attack sessions comprise 20% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	92	91	91	92	95	94	94	95
1.0	2	91	90	90	90	94	94	94	95
1.0	3	72	71	74	73	92	91	92	92
5.0	1	93	93	93	93	96	95	95	96
5.0	2	92	91	92	92	95	94	95	96
5.0	3	73	72	74	74	93	92	93	93
10.0	1	94	93	94	94	97	96	96	97
97	2	93	92	92	93	96	95	96	97
10.0	3	74	73	75	75	94	93	93	94

Table 6.8 – The TPR/TNR values in attack 1-3 scenarios with Lewis dataset where attack sessions comprise 50% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	88	86	89	89	94	92	94	94
1.0	2	87	86	88	87	93	92	94	94
1.0	3	68	68	68	69	91	89	91	91
5.0	1	91	88	90	90	95	94	95	95
5.0	2	88	87	89	88	94	93	94	94
5.0	3	70	69	70	70	92	91	92	92
10.0	1	92	89	92	92	96	95	96	96
10.0	2	89	88	90	89	95	94	95	95
10.0	3	72	71	72	72	93	92	93	93

Table 6.9 – The TPR/TNR values in attack 1-3 scenarios with Lewis dataset where attack sessions comprise 99% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	79	77	79	81	94	92	93	94
1.0	2	77	76	78	79	92	91	93	93
1.0	3	58	57	59	59	90	90	90	90
5.0	1	80	79	80	80	94	94	94	95
5.0	2	78	77	78	79	93	92	93	93
5.0	3	59	58	60	59	91	90	91	92
10.0	1	81	80	81	82	95	93	94	95
10.0	2	81	79	80	81	94	92	93	93
10.0	3	60	60	60	60	92	91	92	92

Table 6.10 – The TPR/TNR values in attack 1-3 scenarios with EACS dataset where attack sessions comprise 20% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	91	90	90	91	96	95	95	95
1.0	2	90	89	89	89	94	93	94	95
1.0	3	73	71	74	74	92	91	92	92
5.0	1	92	92	92	92	96	95	96	97
5.0	2	91	90	91	91	95	94	95	95
5.0	3	74	72	75	75	93	92	93	93
10.0	1	92	93	93	93	96	96	96	97
10.0	2	91	92	92	92	96	94	95	96
10.0	3	75	73	76	76	94	93	93	94

Table 6.11 – The TPR/TNR values in attack 1-3 scenarios with EACS dataset where attack sessions comprise 50% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	88	86	88	89	94	92	94	94
1.0	2	86	86	87	87	93	92	94	94
1.0	3	69	69	69	69	91	89	91	91
5.0	1	91	90	90	90	95	94	95	95
5.0	2	88	87	88	88	94	93	94	94
5.0	3	71	71	71	71	92	91	92	92
10.0	1	92	90	92	92	96	95	96	96
10.0	2	89	88	90	89	95	93	95	95
10.0	3	74	72	74	73	93	92	93	93

Table 6.12 – The TPR/TNR values in attack 1-3 scenarios with EACS dataset where attack sessions comprise 99% of all sessions

Mean Exponential Page Viewing Time (min)	Attack #	True Positive Rate				True Negative Rate			
		COD	ILOF	Denstream	D-Stream	COD	ILOF	Denstream	D-Stream
1.0	1	80	77	79	81	93	91	93	93
1.0	2	78	76	78	79	92	91	93	93
1.0	3	57	57	58	59	90	89	90	90
5.0	1	80	79	80	81	94	93	94	94
5.0	2	78	77	78	80	94	92	93	93
5.0	3	58	57	59	59	91	90	91	92
10.0	1	82	80	81	82	95	94	95	95
10.0	2	81	79	80	81	94	93	93	93
10.0	3	59	60	60	60	92	91	91	92

6.3.4.1 The Relationship Between the Attack Sophistication and TPR and TNR Rates

In general, we observe that for all four algorithms and in all three attack scenarios, TNR is generally very good – above 90%. TPR is the most favorable in case of Attack 1 (above 77%), and least favorable in case of Attack 3 (above 57%). Recall, the TPR of 57% implies that 43% of actual human visitors, falsely identified as attackers, are presented with CAPTCHA (see Figure 6.1). This observation can be explained as follows:

In Attack 1 scenario, the attacker synthesizes CBSs by randomly selecting web pages from a relatively large pool of web pages (see section 6.3.2). Consequently, the generated CBSs are likely to be very different from other (e.g.) human-generated CBSs. As such, these CBSs are likely to be placed in the low-density (i.e., outlier) clusters/grids in Stage 2. On the other hand, in Attack 2, the attacker synthesizes the CBSs by randomly selecting web pages from a smaller set of the 10 most popular web pages as identified by Google search engine. Therefore, in this scenario, the attacker-generated CBSs are likely to be more similar to other (e.g.) human-generated CBSs than in the case of attack 1 scenario. As such, these CBSs

are also likely to be misplaced into a non-outlier cluster/grid in Stage 2. In the Attack 3 scenario, the attacker-generated CBSs have an even higher chance of matching against actual human-generated CBSs than in Attacks 1 and 2, since the attacker is synthesizing the CBSs as if browsing the web site in a manner of true human visitors. Consequently, the sensitivity of the system is further lowered relative to the first two scenarios.

Furthermore, in the case of the most sophisticated Attack 3 scenario , the TPR results in Lewis and EACS scenarios are slightly lower than in the CSE scenarios. The differences in the number of possible paths or CBSs between the Lewis/EACS and CSE web domains can explain this outcome. Namely, Lewis and EACS web sites have 8419 and 10643 unique paths through the web site that a visitor can traverse by following the hyperlinks from the top main web page, respectively. However, the CSE web site has 25741 such paths. As we have explained, in Attack 3 the attacker is browsing the web site similar to actual human visitors by following links from the main top web page of the web site. Under the assumption that all other parameters of the simulation are equal, the attacker-synthesized CBSs in Attack 3 scenarios have a higher chance of matching against the CBSs generated by actual human visitors in the case of the Lewis or EACS web domains than in the case of CSE web domain.

Recall our discussion from chapter 5 that the density-based algorithms (ILOF, Denstream and D-Stream) are less sensitive to noise than COD, a distance-based algorithm, since they can discover clusters of arbitrary shape [206]. This is a possible explanation for COD's slightly worse TPR/TNR in the case of some test scenarios. The slightly worse performance of both COD and ILOF could also be attributed to the fact that these algorithms employ a sliding window where all the points in the window are given an equal weight (unlike Denstream and D-Stream algorithms that employ a decaying function over the previously

observed data points). However, as shown in Figure 6.8 a-c), on average, across all simulated scenarios, all four algorithms perform fairly similarly.

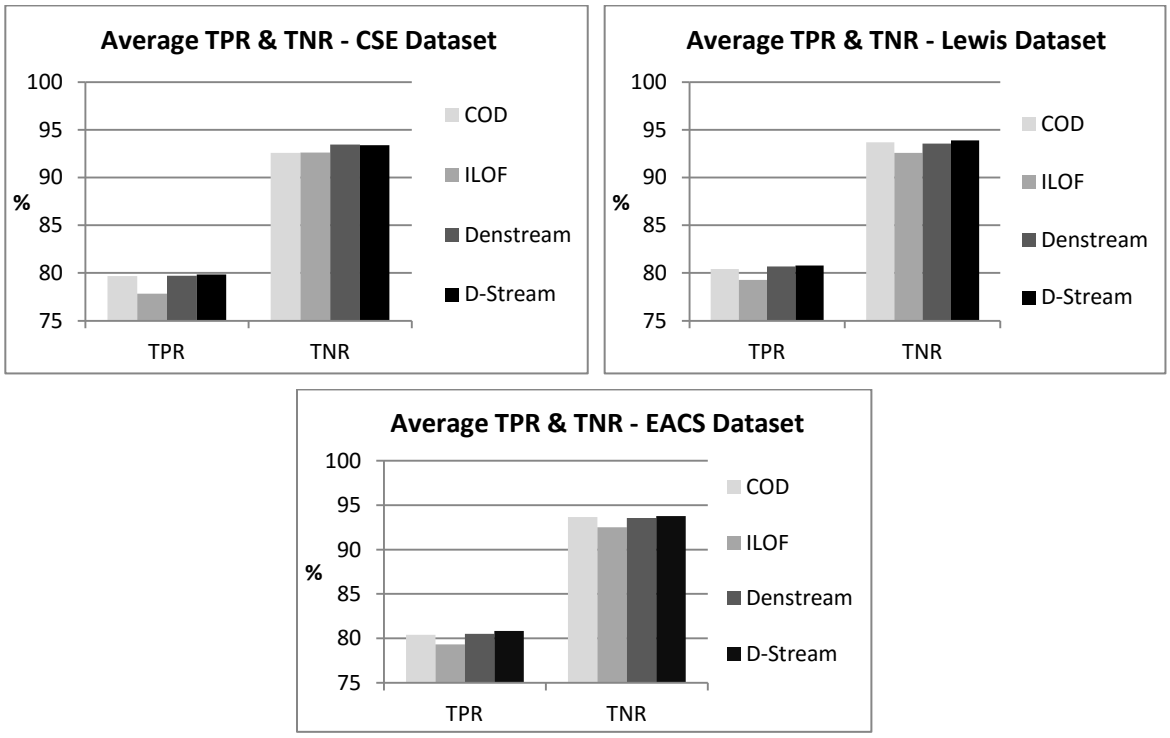


Figure 6.8 – Average TPR/TNR across all simulated scenarios in CSE, Lewis and EACS datasets

6.3.4.2 The Relationship Between the Page Viewing Time Modeling and TPR and TNR Rates

By examining Tables 6.4-6.12, we can conclude the model of attacker behaviour with longer average page viewing time (i.e., the longer average inter-arrival time between attacker's web requests in synthesized sessions) results in slightly higher TNR rate across all of the 3 web domains. Note that TNR rates are lowest in Figures 6.4-6.12 when the page viewing times are sampled with exponential distribution with mean of 1 minute and highest when the page viewing times are sampled with exponential distribution with mean of 10 minutes. This is somewhat of an expected result, as previous reported in [190], on average, human visitors spend close to 1 minute viewing web pages. Note that our study has also confirmed these

results - in CSE web domain, the average per-page viewing time for human visitors is 62 seconds and slightly higher in EACS and Lewis web domains at 89 and 100 seconds, respectively.)

6.3.4.3 The Relationship Between the Attack Intensity and TPR and TNR Rates

By examining Tables 6.4-6.12, we notice that as the number of attack sessions embedded into the dataset increases from 20% to 99%, i.e., as the intensity of the attack increases, the TPR rates decrease by 10%-15% while TNR rates stay nearly constant. This observation implies that, as the dominance of attack sessions in comparison to regular human sessions increases, we can observe some "distortion" of the legitimate web visitor CBS and PVT profiles. However, the distortion is not drastic since majority, i.e., over 90 % of attack sessions, are outliers in attack scenarios shown in Tables 6.4-6.12 and as such are not used to refine the profile of typical browsing behaviour (see again Figure 6.1).

Note that the spike of 10-15% in CAPTCHA rates occurs temporarily for the duration of the attack. As the attack sessions are identified as outliers by the system with at least 90% accuracy, the bots, as part of the attack-generating Botnet, quickly generate an outlier sessions and are presented with a CAPTCHA puzzle. Since they also fail, i.e., cannot solve, the CAPTCHA puzzle, their IP addresses quickly become blocked by the system. After, the bots are blocked, the legitimate web visitors refine the profile back to its original state and the TPR rate returns back to the rate prior to the start of the attack.

6.3.4.4 The Relationship Between the Algorithm Parameters and TPR and TNR Rates

The same algorithms parameters (see Table 6.3) were employed in all scenarios and the results are fairly similar. Our decision to choose/use these exact values of the given parameters was based on the following observations:

- In the case of COD-sequence and ILOF-sequence small k values under 5 in Stage 2 and larger k values greater than 10 in Stage 3 result in the highest values of TPR and TNR. Also, in the case of these two algorithms, the radius below 0.5 in Stage 2 and radius between 25 and 100 in Stage 3 result in the highest values of TPR and TNR.
- In the case of DSS, ε values below 0.5 and C_l values below 0.5 result in the highest values of TPR and TNR. Note that these parameters are in line with radius parameters in COD-sequence and ILOF-sequence.
- In the case of Denstream and D-Stream, the radius ε between 25 and 100 and density threshold C_l between 100 and 3000 result in most favorable TPR/TNR.
- The size of the sliding window in COD and ILOF did not significantly affect the TPR/TNR results due to low concept drift - see next section 6.3.4.5. We evaluated these two algorithms with the size of the window between 50K and 400K in increments of 50K and we the observed that the TPR/TNR results varied +/- 3%.
- Similarly, the λ parameters in DSS, Denstream and D-Stream did not show to have a significantly affect on the TPR/TNR rates due to low concept drift.

These parameter settings indicate that non-outlier clusters with a larger radius or higher number of cluster members, i.e., generally larger k, R/ε, C_l and L parameters in Stage 3 and non-outlier clusters with a smaller radius or smaller number of cluster members, i.e., generally smaller k, R/ε, C_l and L parameters in Stage 2, result in more favorable TPR/TNR rates.

6.3.4.5 Concept Drift in the CSE, Lewis and EACS Datasets

In order to gain a better understanding of 1) the parameter settings that result in the highest TPR/TNR and 2) the amount of concept drift, i.e., the amount of new/emerging instances of browsing behaviour in the 3 datasets, we collected additional statistics from the web logs.

a) CBS Concept Drift

Firstly, we collected for each day, the number of CBSs that have some or no subsequences in common with the set of CBSs observed during the previous days. These results, for the CSE, Lewis and EACS datasets, are shown in Figures 6.9, 6.10 and 6.11, respectively. Note that the line points in Figures 6.9-6.11 indicate the number of CBSs, collected on day i, that were at least 0.0, 0.25, 0.5, 0.75, 0.9 and 1.0 distant, in terms of the NLLCS distance metric, from the set of CBSs collected during the previous i-1 days. The results were collected for exclusively human-generated web sessions. (Note that the fluctuations in number of CBSs observed per day in Figures 6.9-6.11 are due to natural traffic variations over the course of the weeks - i.e., web site has more visitors on a weekday than on a Saturday or Sunday.)

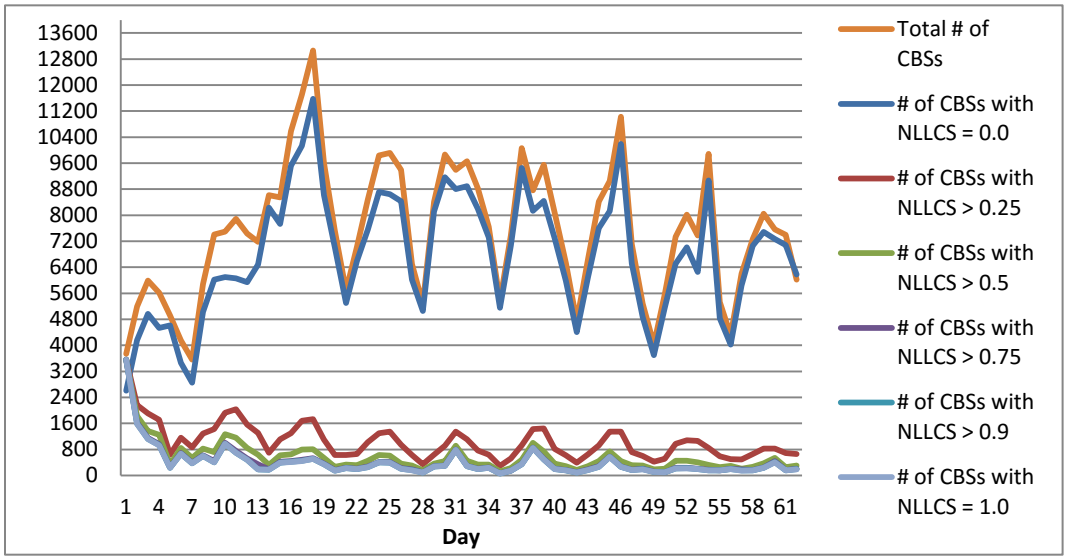


Figure 6.9 – The number of CBSs, collected on day i, that were at least 0.0, 0.25, 0.5, 0.75, 0.9 and 1.0 distant, in terms of the NLLCS distance metric, from the set of CBSs collected during the previous i-1 days in CSE dataset

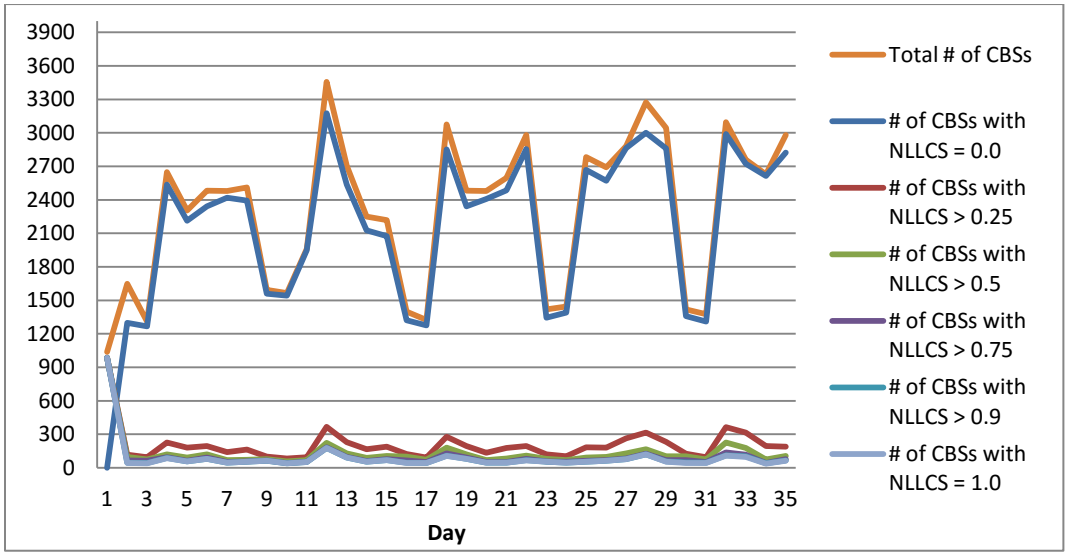


Figure 6.10 – The number of CBSs, collected on day i, that were at least 0.0, 0.25, 0.5, 0.75, 0.9 and 1.0 distant, in terms of the NLLCS distance metric, from the set of CBSs collected during the previous i-1 days in Lewis dataset

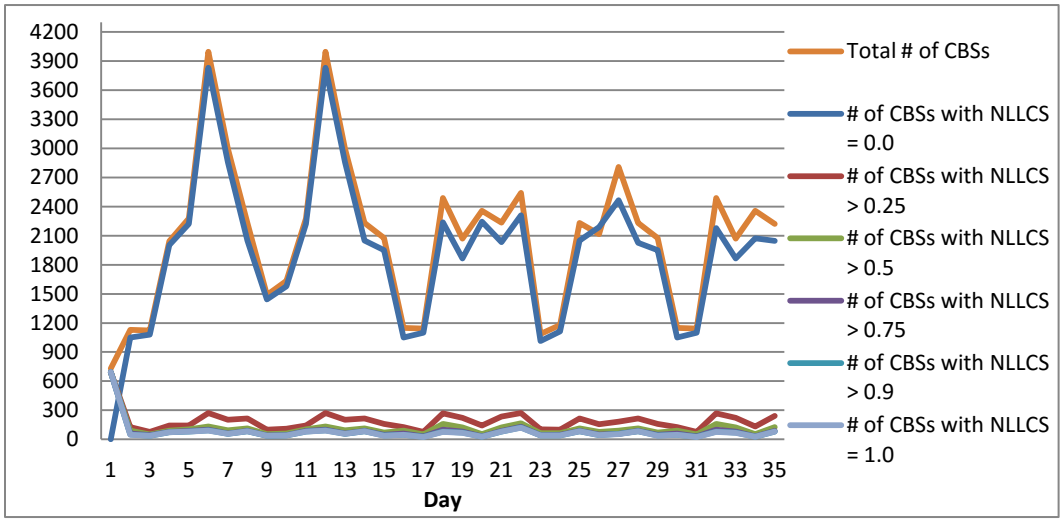


Figure 6.11 – The number of CBSs, collected on day i, that were at least 0.0, 0.25, 0.5, 0.75, 0.9 and 1.0 distant, in terms of the NLLCS distance metric, from the set of CBSs collected during the previous i-1 days in EACS dataset

In all three Figures 6.9-6.11, we can observe the following:

- There exists a steady rate of about 1-3% of completely new CBSs per day, i.e., these CBSs are 1.0 distant in terms of NLLCS from all of the CBSs observed on the previous days. However, ~ 90% of the new per-day incoming CBSs identically match against at least one of CBSs observed in the previous days, i.e., these CBSs are 0.0 distant in terms of NLLCS from at least one previously observed CBS.
- The trend lines in Figures 6.9-6.11 could explain why the R and ε parameters with values under 0.5 in COD-sequence and DSS scenarios, respectively, produce favorable TPR/TNR results. Namely, almost all new per-day arriving CBSs are within radius 0.5 from the CBSs observed in the previous i-1 days. Also, based on the obtained TNR results in the simulation scenarios, it appears that a great majority of attacker-generated CBSs are more than 0.5 NLLCS distance from the human-generated web sessions.

b) PVT Concept Drift

Additionally, we looked into the amount of variation in per-page viewing times in the 3 datasets. The mean page viewing time, the median of the standard deviation of per-page viewing time and the mean of the standard deviation of per-page viewing times across the individual web pages in the 3 datasets are plotted in Figure 6.12. (Note that these results were collected exclusively for web pages that were accessed at least 100 times in the web logs.) Again, the results were collected from only human-generated web sessions. The results indicate a substantial variation in the amount of time human visitors spend viewing a web page over the course of the data collection time period. As can be seen from 6.12, the viewing time for a web page, on average, deviates 3-4 minutes from the mean viewing time for that web page.

These results could explain why the non-outlier clusters with a larger radius or higher number of cluster members, i.e., generally larger k, R/ε, C_l and L parameters in Stage 3, produce more favorable TPR/TNR rates. Namely, the attacker's modeling of page viewing times with the exponential distribution with mean

of 1 minute (close to actual mean page viewing time observed in our 3 datasets), would have a good chance of correctly guessing the page viewing times for some of the web pages. Also, because of the large deviation of true human per-page viewing times there will be a greater chance that some of the attacker's guessed page viewing time will be close to the regular human-generated page viewing time for a particular web page. Therefore, larger k, R/ε, C_l and L parameters for the non-outlier clusters in Stage 3 result in a higher chance at identifying attacker's sessions.

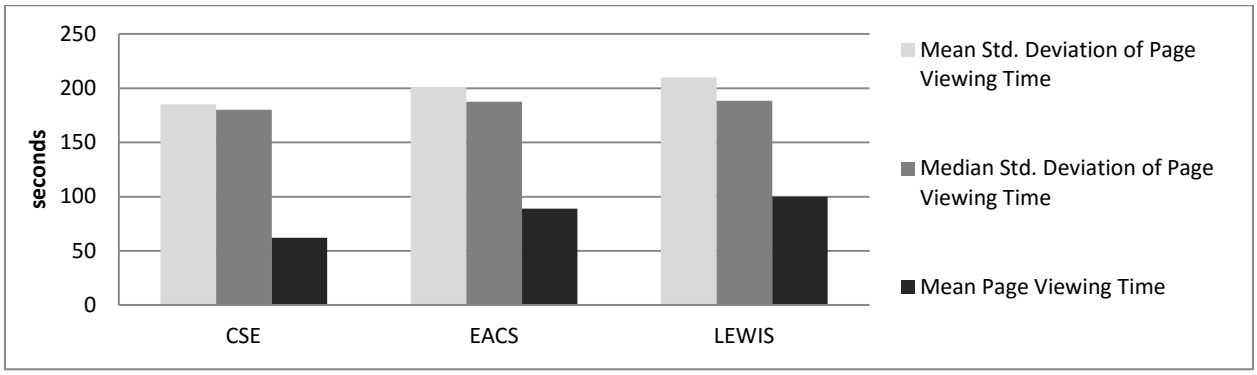


Figure 6.12 – The mean per-page viewing time, median of the standard deviation of per-page viewing time and mean of the standard deviation of per-page viewing time in CSE, Lewis and EACS datasets

c) Recommendations for Algorithm Parameter Settings in More Dynamic Web Domains

The 3 datasets have generally very low concept drift, especially in the terms of the new emerging CBSs (see Figures 6.9-6.11 where most of the CBS observed on day i are identical to CBS observed on days 1,.., i-1 in terms of the NLLCS metric). Other web domains can have a more evolving content. In such web domains, the number of CBSs that are at least 0.25, 0.5, 0.75 and 1.0 distant from previously observed CBSs could be higher. As such, our system could employ a larger radius parameters, i.e., R and ε parameters, in both Stages 2 and 3, to ensure similar TPR rates as documented in our analysis above.

Additionally, to deal with the challenges of more dynamic web sites, the system operation can be further optimized as follows. Namely, during an absence of a DDoS attack, the outlier sensitivity of the system can be reduced by adjusting the radius parameters in COD-sequence, COD, Denstream and DSS algorithms, and L parameter in ILOF to slightly larger values or slightly lower values in the case of C_l parameter in D-Stream than the ones reported in Table 6.3. Note that the absence of a DDoS attack can be defined as the absence of web visitors that have failed a CAPTCHA test for some pre-specified time period. Then, as soon as the system detects a web visitors that has failed a CAPTCHA test, which would likely be a DDoS bot initiating an attack on the target web site, the system can reset these algorithm parameters to the optimal values as listed in Table 6.3. This procedure would ensure that the system achieves very high TPR rates, i.e., the number of regular human web visitors that are bothered by CAPTCHA puzzle would be reduced further in the absence of an attack, while at the same time, the system will quickly react to the onset of an attack by dynamically changing the parameters to higher outlier threshold values. In our simulations, we noticed that setting the radius and k parameters in COD, Denstream and DSS to 0.5 and 1, respectively, Cl to100, and L parameter in ILOF-sequence to 0.1 resulted in TPR rates above 95% while keeping TNR rates above 40% in the case of all 3 attack scenarios.

6.3.4.6 Processing Time Efficiency of the Four Outlier Algorithms

The average processing time of synthesized datasets with each algorithm is shown in Figure 6.13. As expected, the ILOF algorithm has the worst processing time performance due to its worst time complexity (recall Table 6.1). Note that the ILOF algorithm computes the KNN for each new input instance added to the sliding time window of size n in O(logn) time using the M-tree data structure. Also, each time an input instance x is added or removed from the sliding window, every input instance y in the sliding window, for which x is added or removed from its KNN, respectively, must re-compute its LOF value (see Appendix D). The running time complexity of this step, in the worst case, is O(nlogn).

The processing time of COD, Denstream and D-Stream algorithms is shorter since they employ either an indexing data structure (i.e., M-tree in COD) or synopsis data structure (i.e., micro-cluster and density grid in Denstream and D-Stream, respectively). The worst running time complexity for labeling a new instance in the case of COD, Denstream and D-Stream is O(logn), O(m) and O(g), respectively (refer again to Table 6.1).

Figure 6.13 – The processing time efficiency of COD, ILOF, Denstream and D-Stream

6.4 Summary

In this chapter, we presented and evaluated the most significant contribution of this thesis: a new system for detection of both current and next-generation (i.e., future) HTTP-based DDoS attacks. Our DDoS defense system is also the first of its kind to tackle the problem of real-time HTTP-based attacks detection in both static and dynamic web domains. The results demonstrate that, in the case of an attack generated by the most sophisticated, advanced HTTP-based bots, the proposed system is capable of detecting 90% of such malicious traffic. While at the same time, during such an attack by these advanced HTTP-based bots, that are yet to be observed in the real-world, our system ensures that at most 43% of human web visitors will be bothered with solving CAPTCHA puzzles.

In the next chapter, we summarize the main conclusions of our research and outline possible future work.

Chapter 7 Conclusions and Future Work

In the first part of this thesis, we reviewed the current state-of-the-art in automated benign and malicious web crawler's browsing behaviour. This research has achieved the following milestones. Firstly, we have shown that supervised machine learning can be employed to 1) classify user sessions as belonging to either automated benign web crawlers or human visitors, and 2) identify which browsing features are the most useful in identifying various visitors groups. Secondly, we also have shown that unsupervised machine learning can be utilized to obtain a better insight into the types and distribution of visitors to a public web site based on their link-traversal behaviour, as well as to identify suspicious outlier malicious/unknown visitors that exhibit human browsing behaviour and suspicious outlier human visitors that mimic malicious browsing behaviour on a web site. Thirdly, we demonstrated that time series analysis can be applied to perform a further fine-tuned separation of truly malicious sessions from suspicious outlier sessions by exploiting the underlying time-wise browsing behaviour differences between the human and the machine-generated session types.

In the second part, we presented and evaluated a novel real-time system for detection of various current and possibly future sophisticated HTTP-based DDoS attacks. The system was tested on real-world traces against synthesized attack scenarios. The simulation results demonstrate the effectiveness of the proposed

system to correctly detect over 90% of the synthesized attack sessions while limiting the number of actual human visitors presented with CAPTCHA below 43% across all attack scenarios.

While the aim of this thesis was to provide some definite answers related to the detection of current and future application layer DDoS attacks in static and dynamic web domains, there are many possible extensions to the work conducted so far, including the following:

1. The sensitivity of the system could be improved by maintaining an additional profile of malicious web visitors. Note that, in the worst case scenario, the sensitivity of the system can be as low as 57%, i.e., 43% of actual human web visitors are asked to solve CAPTCHA. In order to further minimize the number of regular human visitors that are presented graphical puzzles by our system, our system could maintain a profile (i.e., clustering results) of web visitors that fail CAPTCHA test for fine-tuned separation of outlier human-generated web sessions from outlier bot-generated web sessions. In the two-profile system, the visitors are presented CAPTCHA *only if*: 1) Their web sessions are identified as outliers in the comparison to the profile of typical browsing behaviour of regular human visitors (i.e. the profile employed in the original version of the system shown in Figure 6.1) and 2) Their web sessions are identified as non-outliers in the comparison to the profile of the browsing behaviour of web visitors that fail the CAPTCHA test - i.e., the browsing behaviour profile derived from malicious DDoS bots. The web visitors whose web sessions are outliers in comparison to the typical browsing behaviour of human visitors and are also outliers in comparison to the profile of the browsing behaviour of web visitors that fail the CAPTCHA test are granted uninterrupted access to the site.
2. Another issue in the current version of the system is how and when to serve CAPTCHA puzzles in the light of the possibility that either: 1) it could take a very long time for the "outlier" user to revisit the site, or 2) the "outlier" user is one-time-only web visitor (i.e., he/she never revisits the web site). For

instance, in CSE, Lewis and EACS web logs, the percentage of one-time-only[17] web visitors among all web visitors was 50%, 29% and 27%[18], respectively. A possible solution could entail interrupting the incoming/incomplete web sessions with a CAPTCHA as soon as the CBS or PVT becomes suspicious in order to resolve the true identity of the visitor before the visitor leaves the web site.

3. The system's performance could be evaluated by employing additional web log data from dynamic web domains such as news web sites and online blogging sites. Specifically, we could investigate how the sensitivity and specificity of the system are affected when the same algorithm parameters, employed in our evaluation of the system, are also applied to detect the 3 attacks in additional set of highly dynamic web domains.
4. So far, we have studied the detection of unusually behaving human web visitors. A possible direction for the future research would be to extend our analysis to the detection of unusually behaving benign crawlers or other non-DDoS malicious crawlers (e.g., spam, web site scraping bots or unknown crawlers identified in chapter 5) by employing the proposed anti-DDoS system.
5. Also, the system could be evaluated with additional synthesized attacks. For instance, the simulation of the system performance with additional variations of Attack 2 scenario could be undertaken by selecting the most popular pages with a help of additional search engines or Wayback machine [216]. Also, another page viewing time modeling can be employed where the mean Exponential page viewing time would be greater for the web pages with greater amount of embedded information, i.e., web pages with more text or image content. For instance, the size of the embedded content in web pages can be estimated as the total size of the web page file in bytes.

[17] The one-time-only web visitors are defined as human web site visitors, identified by their origin IP address in the web logs, that generated only a single session for the duration of the web collection time period.

[18] In the calculation of these statistics, we only considered web visitors that generated a web session in the first n-1 weeks of web log data, where n is the total number of weeks of web log data for that dataset. The reasoning for this approach was to ignore web visitors that generated the single web session for the first time during the last week of web log collection time period and who typically visit the web site infrequently, i.e., once-a-week.

Bibliography

[1] R. Kumar and A. Tomkins, "A Characterization of Online Browsing Behavior," in *World Wide Web*, Raleigh, North Carolina, USA, 2010, pp. 561-570.

[2] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer, "Web Page Revisitation Revisited: Implications of a Long-term Click-stream Study of Browser Usage," in *In proceedings of CHI '07*, San Jose, California, USA, 2007, pp. 597-606.

[3] J. Lewis. (2012, August) Forbes.com. [Online]. http://www.forbes.com/sites/ciocentral/2012/05/08/figuring-ddos-attack-risks-into-it-security-budgets/#5e6bf1c41cfc

[4] (2016) Arbor Networks. [Online]. https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf

[5] "The Growing Threat of Application-Layer DDoS Attacks," Arbor Networks, Ann Arbor, MI, USA, Whitepaper 2012. [Online]. Arbor Networks, “The Growing Threat of Application-Layer DDoS Attacks”, Whitepaper, 2012.

[6] D. Drinfeld, N. Vlajic, and D. Stevanovic, "Bots for Flash-Crowd DDoS that Mimic Human Behavior: Are We There Yet?," *InderScience International Journal of Internet Technology and Secured Transactions (IJITST).*, Accepted.

[7] D. Stevanovic, A. An, and N. Vlajic, "Detecting Web Crawlers from Web Server Access Logs with Data Mining Classifiers," in *In the proceedings of the 19th International Symposium on Methodologies for Intelligent Systems*, Warsaw, Poland, June, 2011.

[8] D. Stevanovic, A. An, and N. Vlajic, "Feature Evaluation for Web Crawler Detection with Data Mining Techniques," *Elsevier Expert Systems with Applications*, vol. 39, no. 10, pp. 8707-8717, August 2012.

[9] D Stevanovic, N Vlajic, and A An, "Unsupervised Clustering of Web Sessions to Detect Malicious and Non-malicious Website Users," in *In the proceedings of the 2nd International Conference on Ambient Systems, Networks and Technologies*, Niagara Falls, Canada, September, 2011.

[10] D. Stevanovic, N. Vlajic, and A. An, "Detection of Malicious and Non-malicious Website Visitors Using Unsupervised Neural Network Learning," *Applied Soft Computing*, vol. 13, no. 1, pp. 698-708, January 2013.

[11] D. Stevanovic and N. Vlajic, "An Integrated Approach to Defence Against Degrading Application-Layer DDoS Attacks," in *International Conference on Security and Management*, Las Vegas, NV, 2013, pp. 1-7.

[12] D. Stevanovic and N. Vlajic, "Application-Layer DDoS in Dynamic Web-Domains: Building Defenses against Next-Generation Attack Behavior," in *In proceedings of IEEE Conference on Communications and Network Security*, San Francisco, CA, 2014, pp. 1-2.

[13] D. Stevanovic and N. Vlajic, "Next Generation Application-Layer DDoS Defences: Applying the Concepts of Outlier Detection in Data Streams with Concept Drift," in *In proceedings of the IEEE International Conference on Machine Learning and Applications*, Detroit, MI, USA, 2014.

[14] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms*. Upper Saddle River, NJ, United States of America: Pearson Education Inc., 2005.

[15] C. Wilson, "Botnets, Cybercrime, and Cyberterrorism: Vulnerabilities and Policy Issues for Congress," Foreign Affairs, Defense, and Trade Division, United States Governemnt, CRS Report for Congress 2008.

[16] K. M. Finklea and C. A. Theohary, "Cybercrime: Conceptual Issues for Congress and U.S. Law Enforcement," Congressional Research Service, 2012.

[17] D. Holden. (2014, July) SC Magazine. [Online]. http://www.scmagazineuk.com/the-science-behind-ddos-extortion/article/362050/

[18] D. Goodin. (2014, June) Arstechnica. [Online]. http://arstechnica.com/security/2014/06/under-ddos-feedly-buckles-but-defies-attackers-extortion-demands/

[19] J. E. Dunn. (2013, December) Techworld. [Online]. http://www.techworld.com/news/security/ddos-blackmailers-branded-playground-bullies-for-attack-on-casino-firm-3494516/

[20] J. Nazario. (2008, August) The Arbor Networks Security Blog. [Online]. http://www.arbornetworks.com/asert/2008/07/georgia-on-my-mind-political-ddos/

[21] J. Vijayan. (2013, January) Computer World. [Online]. http://www.computerworld.com/s/article/9235636/Experts_unsure_whether_Iran_is_behind_bank_DDoS_attacks_

[22] E. Schurman and J. Brutlag, "The User and Business Impact of Server Delays, Additional Bytes, and HTTP Chunking in Web Search," in *Velocity - Web Performance and Operations Conference*, San Jose, USA, June, 2009.

[23] Forrester Consulting. (2010, January) Banktech. [Online]. http://www.banktech.com/business-intelligence/222500093

[24] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *In Proceedings of ICNP 2002*, Paris, France, 2002, pp. 312-321.

[25] S. M Specht and R. B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures," in *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems*, Tainan, Taiwan, September, 2004, pp. 543-550.

[26] A. Asosheh and N. Ramezani, "Comprehensive Taxonomy of DDoS Attacks and Defense Mechanisms Applying in a Smart Classification," *WSEAS Transaction on Computers*, vol. 7, no. 4, pp. 281-290, April 2008.

[27] (2012, October) CloudFlare. [Online]. https://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack/

[28] W. O. Chee. (2010, November) OWASP. [Online]. https://www.owasp.org/images/4/43/Layer_7_DDOS.pdf

[29] B. Sullivan. (2009, November) MSDN Magazine. [Online]. https://msdn.microsoft.com/en-us/magazine/ee335713.aspx

[30] S. A. Crosby and D. S. Wallach, "Denial of service via algorithmic complexity attacks," in *In proceedings of the 12th Conference on USENIX Security Symposium*, Berkeley, CA, USA, 2003, pp. 3-3.

[31] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attacks and DDoS Defence Mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 4, no. 2, pp. 39-54, April 2004.

[32] C Douligeris and A Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643-666, April 2004.

[33] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Computer Survey*, vol. 39, April 2007.

[34] (2015, April) Snort. [Online]. http://www.snort.org/

[35] C. Manikopoulos and S. Papavassiliou, "Network intrusion and fault detection: A statistical anomaly approach," *IEEE Communication*, vol. 40, no. 10, pp. 76-82, October 2002.

[36] T. M. Gil and M. Poletto, "MULTOPS: a data-structure for bandwidth attack detection," in *In Proceedings of the 10th USENIX Security Symposium*, Boston, MA, 2001.

[37] Arbor Networks. (2011, October) The Peakflow Platform. [Online]. http://www.arbornetworks.com

[38] Information Sciences Institute. (2011, October) DynaBone. [Online]. http://www.isi.edu/dynabone

[39] (2011) IETF.org. [Online]. https://www.ietf.org/mail-archive/web/tls/current/msg07553.html

[40] G. Ormazabal, S. Nagpal, E. Yardeni, and H. Schulzrinne, "Secure SIP: A scalable prevention mechanism for DoS attacks on SIP-based VoIP systems," in *In Proceedings of the 2nd International Conference on Principles, Systems and Applications of IP Telecommunications*, Heidelberg, Germany, 2008, pp. 107-132.

[41] Securiteam. (2011, November) [Online]. http://www.securiteam.com/exploits/2AUQBR5Q0Q.html

[42] D. Goodin. (2014, January) New DoS attacks taking down game sites deliver crippling 100Gbps floods. Web site. [Online]. http://arstechnica.com/security/2014/01/new-dos-attacks-taking-down-game-sites-deliver-crippling-100-gbps-floods/

[43] IBM. (November, 2011) [Online]. http://www-01.ibm.com/support/docview.wss?uid=swg21105201

[44] (2014) Prolexic. [Online]. http://www.prolexic.com/kcresources/attack-report/attack_report_q214/Prolexic-Q22014-Global-Attack-Report-A4.pdf

[45] Akamai Inc. (2015, September) [Online]. http://www.wsta.org/wp-content/uploads/2013/11/Q1-2015-SOTI-Security-Report-Low-Res.pdf

[46] R. Barnett. (2011, August) SpiderLabs. [Online]. https://www.trustwave.com/Resources/SpiderLabs-Blog/%28Updated%29-Mitigation-of-Apache-Range-Header-DoS-Attack/

[47] E. Cambiaso, G. Papaleo, and M. Aiello, "Taxonomy of slow dos attacks to web applications," in *Communications in Computer and Information Science*.: Springer Berlin Heidelberg, 2012, vol. 335, ch. Recent Trends in Computer Networks and Distributed Systems Security, pp. 195-204.

[48] A. Karthigeyan, A. Andavar, and J. Ramya, "Adaptable practices for curbing xdos attacks.," *International Journal of Scientific and Engineering Research*, vol. 3, no. 6, pp. 1-6, June 2012.

[49] (2014, June) Applicure Technologies. [Online]. http://www.applicure.com/solutions/prevent-denial-of-service-attacks

[50] A. Lane. (2013, July) Securosis. [Online]. https://securosis.com/blog/database-denial-of-service-the-attacks

[51] (2014, July) RioRey. [Online]. http://static.squarespace.com/static/53319b01e4b0ec02b601ca49/t/537ab649e4b02004337c19aa/1400550985376/RioRey_Taxonomy_DDoS_Attacks_2.6_2014.pdf

[52] C. Wilson. (2012, April) A DDoS Family Affair: Dirt Jumper bot family continues to evolve. Web blog. [Online]. http://www.arbornetworks.com/asert/2012/04/a-ddos-family-affair-dirt-jumper-bot-family-continues-to-evolve/

[53] J. Nazario. (2007, October) BlackEnergy DDoS Bot Analysis. Document. [Online]. http://atlas-public.ec2.arbor.net/docs/BlackEnergy+DDoS+Bot+Analysis.pdf

[54] M. M Andrade and N. Vlajic, "Dirt Jumper: A New and Fast Evolving Botnet-for-DDoS," *International Journal of Intelligent Computing Research*, vol. 3, no. 3, pp. 330-336, September 2012.

[55] P. Shankdhar. (2013, October) InfoSec Institute. [Online]. http://resources.infosecinstitute.com/dos-attacks-free-dos-attacking-tools/

[56] P. N. Tan and V. Kumar, "Discovery of Web Robot Sessions Based on their Navigation Patterns," *Data Mining and Knowledge Discovery*, vol. 6, no. 1, pp. 9-35, January 2002.

[57] S. Yu, Z. Guofeng, S. Guo, X. Yang, and A. V. Vasilakos, "Browsing Behavior Mimicking Attacks on Popular Web Sites for Large Botnets," in *proceedings of 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Shanghai, China, April, 2011, pp. 947-951.

[58] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proceedings of the INFOCOM*, New York, 1999, pp. 126-134.

[59] A. Madrigal. (2013, December) Welcome to the Internet of Thingies: 61.5% of Web Traffic Is Not Human. Web site. [Online]. http://www.theatlantic.com/technology/archive/2013/12/welcome-to-the-internet-of-thingies-615-of-web-traffic-is-not-human/282309/

[60] J. Idziorek and M. Tannian, "Exploiting Cloud Utility Models for Profit and Ruin," in *IEEE 4th International Conference on Cloud Computing*, Washington, DC, 2011, pp. 33-40.

[61] J. Idziorek, M. Tannian, and D. Jacobson, "Attribution of Fraudulent Resource Consumption in the Cloud," in *IEEE Fifth International Conference on Cloud Computing*, Honolulu, Hawaii, 2012, pp. 99-106.

[62] J Idziorek, M Tannian, and D Jacobson, "Detecting Fraudulent Use of Cloud Resources," in *The ACM Cloud Computing Security Workshop*, Chicago, Illinois, 2011, pp. 61-72.

[63] M. H. Sqalli, F. Al-Haidari, and K. Salah, "EDoS-Shield - A Two-Steps Mitigation Technique against EDoS Attacks in Cloud Computing," in *Fourth IEEE International Conference on Utility and Cloud Computing*, Melbourne, Australia, 2011, pp. 49-56.

[64] (2012, May) User-agent-string.info. [Online]. http://user-agent-string.info/

[65] J. Edwards. (2012, June) It's 2012 and Armageddon has arrived. Web article. [Online]. http://www.arbornetworks.com/asert/wp-content/uploads/2012/03/Crypto-Armageddon-Blog1.pdf

[66] D. Schwarz. (2013, December) Arbor Networks. [Online]. http://www.arbornetworks.com/asert/2013/12/a-business-of-ferrets/

[67] C. Wilson. (2011, October) DDoS Watch: Keeping an Eye on Aldi Bot. Web article. [Online]. http://www.arbornetworks.com/asert/2011/10/ddos-aldi-bot/

[68] C. Barna, M. Shtern, M. Smit, V. Tzerpos, and M. Litoiu, "Model-based adaptive DoS attack mitigation," in *In Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, Zurich, Switzerland, 2012, pp. 119-128.

[69] Y. Xuan, S. Shin, M. T. Thai, and T. Znati, "Detecting Application Denial-of-Service Attacks: A Group-Testing-Based Approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 8, pp. 1203-1216, August 2010.

[70] J. Wang, X. Yang, and K. Long, "Web DDoS Detection Schemes Based on Measuring User's Access Behavior with Large Deviation," in *IEEE Globecom*, Houston, TX, USA, 2011, pp. 1-5.

[71] L. Li and G. Lee, "Ddos attack detection and wavelets," *Telecommunication Systems*, vol. 28, no. 3, pp. 435-451, 2005.

[72] L Liu, X Jin, G Min, and L Xu, "Real-Time Diagnosis of Network Anomaly based on Statistical Traffic Analysis," in *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, Liverpool, UK, 2012, pp. 264-270.

[73] S. Lee, G. Kim, and S. Kim, "Sequence-order-independent network profiling for detecting application layer DDoS attacks," *EURASIP Journal on Wireless Communications and Networking*, vol. 1, no. 50, pp. 1-9, 2011.

[74] S. Yadav and S. Selvakumar, "Detection of Application Layer DDoS Attack by Modeling User

Behavior Using Logistic Regression," in *4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)*, Noida, India, 2015, pp. 1-6.

[75] S. Seufert and D. O'Brien, "Machine learning for automatic defence against distributed denial of service attacks," in *In IEEE International Conference on Communications*, Glasgow, Scotland, 2007, pp. 1217-1222.

[76] A. Chonka, J. Singh, and W. Zhou, "Chaos theory based detection against network mimicking ddos attacks," *Communications Letters*, vol. 13, no. 9, pp. 717-719, 2009.

[77] P. Chwalinski, R. Belavkin, and X. Ch, "Detection of Application Layer DDoS Attacks with Clustering and Bayes Factors," in *IEEE International Conference on Systems, Man, and Cybernetic*, 2013, pp. 156-161.

[78] C. Ye, K. Zheng, and C. She, "Application layer DDoS detection using clustering analysis," in *2nd International Conference on Computer Science and Network Technology*, Changchun, China, 2012, pp. 1038-1041.

[79] P. Hayati, V. Potdar, K. Chai, and A. Talevski, "Web spambot detection based on web navigation behaviour," in *International Conference on Advanced Information Networking and Applications*, Perth, Australia, 2010, pp. 797-803.

[80] D. Das, U. Sharma, and D. K. Bhattacharyya, "Detection of HTTP Flooding Attacks in Multiple Scenarios," in *In proceedings of the 2011 International Conference on Communication, Computing & Security* , Odisha, India, 2011, pp. 517-522.

[81] M. Zolotukhin, T. Hamalainen, T. Kokkonen, and J. Siltanen, "Analysis of HTTP requests for anomaly detection of web attacks," in *IEEE 12th International Conference on Dependable, Autonomic and Secure Computin*, Dalian, China, 2014, pp. 406-411.

[82] Y. S. Choi, J. T. Oh, J. S. Jang, and I. K. Kim, "Timeslot Monitoring Model for application layer DDoS attack detection," in *IEEE International Conference on Computer Sciences and Convergence Information Technology (ICCIT)*, Seogwipo, South Korea, 2011, pp. 677-679.

[83] Y. Xie and S- Z. Yu, "Monitoring the Application-Layer DDoS Attacks for Popular Websites," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 15-25, February 2009.

[84] C. Xu, G. Zhao, G. Xie, and S. Yu, "Detection on Application Layer DDoS using Random Walk Model," in *IEEE Communication and Information Systems Security Symposium*, Melbourne, Australia, 2014, pp. 707-712.

[85] C-S. Lin, C-Y. Lee, J-C. Liu, C-R. Chen, and S-Y. Huang, "A detection scheme for flooding attack on application layer based on semantic concept," in *International Computer Symposium* , Tainan, China, 2010, pp. 385-389.

[86] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites," in *Proceedings of the 11th international conference on World Wide Web*, Honolulu, 2002, pp. 293-304.

[87] S Yu et al., "Discriminating DDoS Attacks from Flash Crowds Using Flow Correlation Coefficient," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 6, pp. 1073 - 1080, June 2012.

[88] T. Thapngam, S. Yu, W. Zhou, and G. Beliakov, "Discriminating DDoS Attack Traffic from Flash Crowd through Packet Arrival Patterns," in *The First International Workshop on Security in Computers, Networking and Communications*, Shanghai, China, 2011, pp. 969-974.

[89] J. Wang, X. Yang, and K. Long, "A New Relative Entropy Based App-DDoS Detection Method," in *IEEE Symposium on Computers and Communications*, Riccione, Italy, 2010, pp. 966-968.

[90] H. Liu and M. Kim, "Real-Time Detection of Stealthy DDoS Attacks Using Time-Series Decomposition," in *IEEE International Conference on Communications (ICC)* , Cape Town, South Africa, 2010, pp. 1-6.

[91] L. Wang, Q. Wu, and D. D. Luong, "Engaging edge networks in preventing and mitigating undesirable network traffic," in *In proceedings of the 2007 3rd IEEE Workshop on Secure Network Protocols*, Washington, DC, 2007, pp. 1-6.

[92] V. S. Huang, R. Huang, and M. Chiang, "DDoS Mitigation System with Multi-Stage Detection and Text-Based Turing Testing in Cloud Computing," in *27th International Conference on Advanced Information Networking and Applications Workshops*, Barcelona, Spain, 2013, pp. 655-662.

[93] G. Oikonomou and J. Mirkovic, "Modeling Human Behavior for Defense against Flash-Crowd Attacks," in *In Proceedings of IEEE International Conference on Communications*, Dresden, Germany, 2009, pp. 1-6.

[94] J. Wang, M. Zhang, X. Yang, K. Long, and C. Zhou, "HTTP-sCAN: detecting HTTP-flooding attack by modeling multi-features of web browsing behavior from Noisy dataset," in *In proceedings of the 9th Asia-Pacific Conference on Communications*, Bali, Indonesia, 2013, pp. 677-682.

[95] T. Ni, X. Gu, H. Wang, and Y. Li, "Real-Time Detection of Application-Layer DDoS Attack Using Time Series Analysis," *Journal of Control Science and Engineering*, vol. 2013, pp. 1-6, August 2013.

[96] K. Pandiyarajan, S. Haridas, and K. Varghese, "Transparent FPGA based Device for SQL DDoS Mitigation," in *2013 International Conference on Field-Programmable Technology (FPT)*, Kyoto, Japan, 2013, pp. 82-89.

[97] H. Beitollahi and G. Deconinck, "ConnectionScore: a statistical technique to resist application-layer DDoS attacks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 5, no. 3, pp. 425-442, June 2014.

[98] L. Jie, S. Jianwei, and H. Changzhen, "A Novel Framework for Active Detection of HTTP Based Attacks," in *International Conference on Electric and Electronics*, Nanchang, China, 2011, pp. 411-418.

[99] I. Yatagai, T. Isohara, and I. Sasase, "Detection of http-get flood attack based on analysis of page access behavior," in *In proceedings IEEE Pacific RIM Conference on Communications, Computers, and Signal Processing*, 2007, pp. 232-235.

[100] S. Kandula, D. Katabi, M. Jacob, and A. Berger, "Surviving Organized DDos Attacks That Mimic Flash Crowds," in *USENIX Symposium on Network Systems Design and Implementation*, Boston, MA, May, 2005.

[101] R. Gossweilier, M. Kamvar, and S. Baluja, "What's up CAPTCHA?: a CAPTCHA based on image orientation," in *Proceedings of 18th international conference on World wide web*, Madrid, Spain, 2009, pp. 841-850.

[102] A Basso, "Protecting Web resources from massive automated access," University of Torino, Technical RT114/08, 2008.

[103] K. Park, V. Pai, K. Lee, and S. Calo, "Securing Web Service by Automatic Robot Detection," in *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, Berkeley, CA, 2006, pp. 23-29.

[104] D. Gavrilis, I. Chatzis, and E. Dermatas, "Flash crowd detection using decoy hyperlinks," in *2007 IEEE International Conference on Networking Sensing and Control*, London, UK, April, 2007, pp. 466-470.

[105] D. Brewer, K. Li, L. Ramaswamy, and C. Pu, "A Link Obfuscation Service to Detect Webbots," in *2010 IEEE International Conference on Services Computing*, Miami, FL, 2010, pp. 433 - 440.

[106] D. Pogue. (2012, February) Scientific American. [Online]. http://www.scientificamerican.com/article.cfm?id=time-to-kill-off-captchas

[107] (2011) VouchSafe. [Online]. http://www.vouchsafe.com/all-about-vouchsafe/stopping-captcha-farmers

[108] (2016, March) Palo Alto Networks. [Online]. https://live.paloaltonetworks.com/t5/Documentation-Articles/Application-DDoS-Mitigation/ta-p/54531?attachment-id=230

[109] Barracuda Networks. (2014, August) [Online]. https://www.barracuda.com/products/webapplicationfirewall/features

[110] McAfee. (2016, March) [Online]. http://www.mcafee.com/ca/resources/demos/nsp-demo-dos-ddos.html

[111] Sophos. (2014, August) [Online]. http://www.sophos.com/en-us/medialibrary/PDFs/factsheets/sophosutmnetworkprotectiondsna.pdf

[112] (2016, March) Juniper Networks. [Online]. http://www.juniper.net/techpubs/en_US/junos12.1x44/topics/concept/idp-application-level-ddos-protection-overview.html

[113] (2016, March) Arbor Networks. [Online]. http://www.arbornetworks.com/arbor-partner/alliances/technology-alliances/clean-pipes-2-0

[114] O. Katz. (2015, February) F5 Networks. [Online]. https://f5.com/resources/white-papers/protecting-against-application-ddos-attacks-with-big-ip-asm-a-three-step-solution

[115] (2016, January) Juniper Networks. [Online]. http://jncie.files.wordpress.com/2008/09/801003_protecting-the-network-from-denial-of-service-floods.pdf

[116] HP. (2016, March) [Online]. http://www8.hp.com/h20195/V2/getpdf.aspx/4AA5-3394ENW.pdf?ver=1.0

[117] (2016, March) Checkpoint Software Technologies. [Online]. https://www.checkpoint.com/downloads/product-related/datasheets/ds-ddos-protector-appliances.pdf

[118] (2016, March) RioRey. [Online]. http://www.riorey.com/

[119] Huawei. (2014, August) [Online]. http://enterprise.huawei.com/topic/AntiDDoS_2013_en/index.html

[120] (2016, March) Prolexic. [Online]. http://www.prolexic.com/pdf/Prolexic_PLXabm_Service_Overview_082312.pdf

[121] (2016, March) SonicWall. [Online]. http://us-downloads.quest.com/Repository/support.quest.com/SonicWALL%20TZ%20Series/200/Documentation/SonicOS_5.8_Administrators_Guide.pdf

[122] WatchGuard. (2016, March) [Online]. http://www.watchguard.com/training/vbasics5/Admin/vclasss8.htm

[123] (2016, March) Fortinet's DDoS Defense Technology. [Online]. http://www.fortinet.com/sites/default/files/productdatasheets/FortiDDos-1000B.pdf

[124] (2016, March) Prolexic. [Online]. http://www.prolexic.com/why-prolexic-best-global-ddos-mitigation-network.html

[125] (2016, March) Akamai Technologies Inc. [Online]. http://www.akamai.com/dl/product_briefs/product-brief-site-shield.pdf

[126] W. Guo, S. Ju, and Y. Gu, "Web robot detection techniques based on statistics of their requested URL resources," in *Proceedings of ninth international conference on computer supported cooperative work in design*, Coventry, UK, 2005, pp. 302-306.

[127] N. Geens, J. Juysmans, and J. Vanthienen, "Evaluation of Web robot discovery techniques: a benchmarking," *Lecture notes in computer science*, vol. 4065, pp. 121-130, July 2006.

[128] P. Huntington, D. Nicholas, and H. R. Jamali, "Web robot detection in the scholarly information environment," *Journal of Information Science*, vol. 34, no. 5, pp. 726–741, May 2008.

[129] T. Kabe and M. Miyazaki, "Determining WWW user-agents from server access log," in *Proceedings of seventh international conference on parallel and distributed systems*, Washington D.C., 2000, pp. 173-178.

[130] D. Doran and S. S. Gokhale, "Discovering New Trends in Web Robot Traffic Through Functional Classification," in *Seventh IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, 2008, pp. 275-278.

[131] J. Lee, S. Cha, D. Lee, and H. Lee, "Classification of web robots: An empirical study based on over one billion requests," *Computers & Security*, vol. 28, no. 8, pp. 795-802, November 2009.

[132] X. Lin, L. Quan, and H. Wu, "An automatic scheme to categorize user sessions in modern HTTP traffic," in *In Proceedings of IEEE global telecommunications conference*, New Orleans, Louisiana, 2008, pp. 1-6.

[133] D. Doran and S. S. Gokhale, "Long Range Dependence (LRD) in the Arrival Process of Web Robots," in *In proceedings of International Conference on Computer Technology and Science (ICCTS 2012)*, Singapore, 2012, pp. 176-180.

[134] D. Doran, K. Morillo, and S. S. Gokhale, "A Comparison of Web Robot and Human Requests," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Niagara, Ontario, 2013, pp. 1374-1380.

[135] D. Doran and S. S. Gokhale, "Detecting Web Robots Using Resource Request Patterns," in *11th International Conference on Machine Learning and Applications*, Boca Raton, FL, 2012, pp. 7-12.

[136] A. Stassopoulou and M. D. Dikaiakos, "Web robot detection: A probabilistic reasoning approach," *Computer Networks: The In-ternational Journal of Computer and Telecommunications Networking*, vol. 53, no. 3, pp. 265-278, February 2009.

[137] T. Kohonen, *Self-Organizing Maps*, 3rd ed. New York: Springer-Verlag, Berlin Heidelberg, 2001.

[138] Y. Hiltunen and M. Lappalainen, "Automated Personalization of Internet Users Using Self-Organizing Maps," in *IDEAL*, Manchester, UK, 2002, pp. 31-34.

[139] P. Lichodzijewski, A. N. Zincir-Heywood, and M. I. Heywood, "Dynamic Intrusion Detection Using Self-Organizing Maps," in *Proceedings of the 14th Annual Canadian Information Technology Security Symposium*, Ottawa, Canada, 2002.

[140] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "A hierarchical SOM-based intrusion detection system," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 4, pp. 439-451, June 2007.

[141] T. Horeis, "Intrusion detection with neural networks-combination of self-organizing maps and radial basis function networks for human expert integration," Computational Intelligence Society, Research Report 2003.

[142] Z. Banković, D. Fraga, J. C. Vallejo, and J. M. Moya, "Self-Organizing Maps versus Growing

Neural Gas in Detecting Data Outliers for Security Applications," *Springer - Lecture Notes in Computer Science* , vol. 7209, pp. 89-96, 2012.

[143] G. A. Carpenter and S. Grossberg, "Adaptive Resonance Theory," in *The handbook of brain theory and neural networks*, Michael A Arbib, Ed. Cambridge, MA, USA: MIT Press, 1998, pp. 79-82.

[144] J. Martín-Guerrero, E. Soria-Olivas, P.J.G. Lisboa, A. Palomares, and E. Balaguer-Ballester, "User Profiling from Citizen Web Portal Accesses using the Adaptive Resonance Theory Neural Network," in *IADIS*, San Sabastian, Spain, 2006, pp. 334-337.

[145] M. Amini and R. Jalili, "Network-Based Intrusion Detection Using Unsupervised Adaptive Resonance Theory (ART)," in *In Proceedings of the 4th Conference on Engineering of Intelligent Systems*, Madeira, Portugal, 2004.

[146] H. Liu and V. Keselj, "Combined mining of Web server logs and web contents for classifying user navigation patterns and predicting users' future requests," *Data & Knowledge Engineering*, vol. 61, no. 2, May 2007.

[147] C. Bomhardt, W. Gaul, and L. Schmidt-Thieme, "Web Robot Detection - Preprocessing Web Logfiles for Robot Detection," in *In Proc. SISCLADAG*, Bologna, Italy, 2005.

[148] A. Lourenco and O. Belo, "Catching Web Crawlers in the Act," in *International Conference on Web Engineering*, Palo Alto, CA, 2006, pp. 265-272.

[149] (November, 2011) Robots.org. [Online]. http://www.robotstxt.org/

[150] (2011, August) User-Agents.org. [Online]. http://www.user-agents.org

[151] (2011, August) Bots vs. Browsers. [Online]. http://www.botsvsbrowsers.com/

[152] D Doran and S S Gokhale, "Web robot detection techniques: overview and limitations," *Data Mining and Knowledge Discovery*, pp. 1-28, June 2010.

[153] J. R. Quinlan, *C4.5: Programs for Machine Learning*.: Morgan Kaufmann Publishers, 1993.

[154] J X Yu, O Yuming, C Zhang, and S Zhang, "Identifying interesting visitors through Web log classification," *Intelligent Systems*, vol. 20, no. 3, pp. 55-59, June 2005.

[155] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Diane Cerra, Ed. San Francisco: Elsevier, 2006.

[156] B. Fritzke, "Growing Neural Gas Network Learns Topologies," *Advances in Neural Information Processing Systems*, vol. 7, no. MIT Press, pp. 625-632, 1995.

[157] J. Lee, S. Cha, D. Lee, and H. Lee, "Classification of web robots: An empirical study based on over one billion requests," *Computers & Security*, vol. 28, no. 8, pp. 795-802, November 2009.

[158] (1999, October) KDD Cup 1999 Data. [Online]. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[159] GureKddcup database. [Online]. http://www.aldapa.eus/res/gureKddcup/README.pdf

[160] (2010) CSIC 2010 HTTP Dataset. [Online]. http://users.aber.ac.uk/pds7/csic_dataset/csic2010http.html

[161] (2015, April) Sharpschool. [Online]. http://www.sharpschool.com/

[162] W W Cohen, "Fast effective rule induction," in *ICML 1995*, 1995, pp. 115-123.

[163] G. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," in *Eleventh Conference on Uncertainty in Artificial Intelligence*, San Mateo, 1995, pp. 338-345.

[164] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37-66, 1991.

[165] C. Chang and C. Lin. (2015, December) A Library for Support Vector Machines. [Online].

http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[166] (2010, December) WEKA. [Online]. http://www.cs.waikato.ac.nz/ml/weka/

[167] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 5th ed. USA: CRC Press Taylor & Francis Group, A Chapman and Hall book, 2011.

[168] R.J. Wonnacott and T.H. Wonnacott, *Introductory Statistics*, 4th ed. USA: John Wiley and Sons, 1996.

[169] N Vlajic and Howard C Card, "Vector quantization of images using modified adaptive resonance algorithm for hierarchical clustering," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1147 - 1162, September 2001.

[170] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283-319, 1970.

[171] H. Yin, "On multidimensional scaling and the embedding of self organising maps," *Neural Networks*, vol. 21, no. 2-3, pp. 160-169, March 2008.

[172] B. D. Ripley, *Pattern recognition and neural networks*. New York, United States of America: Cambridge University Press, 1996.

[173] D. Petrilis and C. Halatsis, "Two-level Clustering of Web Sites Using Self-Organizing Maps," *Neural Process Letters*, vol. 27, no. 1, pp. 85-95, February 2008.

[174] Y. Toshihiko, "A Zipf-Like Distribution of Popularity and Hits in the Mobile Web Pages with Short Life Time ," in *Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies*, Taipei, Taiwan, 2006, pp. 240-243.

[175] S Burklen, P J Marron, S Fritsch, and K Rothermel, "User centric walk: An integrated approach for modeling the browsing behavior of users on the web," in *ANSS '05: Proceedings of the 38th annual Symposium on Simulation*, Washington, D.C., 2005, pp. 149-159.

[176] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing reference locality in the WWW," in *IEEE International Conference in Parallel and Distributed Information Systems*, Miami Beach, Florida, 1996, pp. 92-103.

[177] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW client-based traces," Boston University, Boston, MA, Technical Report 95-010, 1995.

[178] S. Glassman, "A caching relay for the world wide web," in *First International Conference on the World-Wide Web*, Geneva, Switzerland, May, 1994, pp. 69-76.

[179] P. Barford, A. Bestavros, A. Bradley, and M. Crovella, "Changes in Web client access patterns: Characteristics and caching implications," *World Wide Web*, vol. 2, no. 1, pp. 15-28, June 1999.

[180] Z. Liu, N. Niclausse, and C. Jalpa-Villanueva, "Traffic model and performance evaluation of Web servers," *Performance Evaluation*, vol. 46, no. 2-3, pp. 77-100, October 2001.

[181] L Cherkasova and M Gupta, "Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads," in *In proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Miami, Florida, 2002, pp. 33-42.

[182] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, "MediSyn: A Synthetic Streaming Media Service Workload Generator," in *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, Monterey, California, 2003, pp. 12-21.

[183] L. Guo, S. Chen, Z. Xiao, and X. Zhang, "DISC: Dynamic Interleaved Segment Caching for Interactive Streaming," in *In Proceedings of the 25th IEEE International Conference on Distributed*

Computing Systems, Columbus, Ohio, 2005, pp. 763 - 772.

[184] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy, "Measurement and Analysis of a Streaming-Media Workload," in *Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, San Francisco, 2001, pp. 1-12.

[185] J. M. Almeida, J. Kruegera, D. L. Eagerb, and M. K. Vernon, "Analysis of Educational Media Server Workloads," in *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, Port Jefferson, New York, 2001, pp. 21-30.

[186] M. D. Dikaiakosa, A. Stassopoulou, and L. Papageorgiou, "An investigation of web crawler behavior: characterization and metrics," *Computer Communications*, vol. 28, no. 8, pp. 880–897, May 2005.

[187] F. Duarte, B. Mattos, A. Bestavros, V. Almeida, and J. Almeida, "Traffic Characteristics and Communication Patterns in Blogosphere," in *In proceedings of the International Conference on Weblogs and Social Media*, Boulder, Colorado, 2007.

[188] F. Benevenuto et al., "Characterization and Analysis of User Profiles in Online Video Sharing Systems," *Information and Data Management*, vol. 1, no. 2, pp. 261-275, June 2010.

[189] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835-846, December 1997.

[190] S. Deng, "Empirical model of WWW document arrivals at access link ," in *IEEE International Conference on Converging Technologies for Tomorrow's Applications*, Dallas, Texas, 1996, pp. 1797 - 1802.

[191] I. C. Y. Ma and J. Irvine, "Characteristics of WAP Traffic," *Wireless Networks*, vol. 10, no. 1, pp. 71-81, January 2004.

[192] D. Ersoz, M. S. Yousif, and C. R. Das, "Characterizing Network Traffic in a Cluster-based, Multi-tier Data Center," in *In the Proceedings of the 27th International Conference on Distributed Computing Systems*, Toronto, Canada, 2007, pp. 59-69.

[193] Y. Xie and S.- Z. Yu, "A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 54-65, February 2009.

[194] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226-244, June 1995.

[195] B. A. Huberman, P. Pirolli, J. E. Pitkow, and R. M. Lukose, "Strong regularities in world wide web surfing," *Science*, vol. 280, no. 5360, April 1998.

[196] M. Halvey, M. T. Keane, and B. Smyth, "Mobile web surfing is the same as web surfing," *Communications of the ACM - Self managed systems*, vol. 49, no. 3, p. 49, March 2006.

[197] J Sanchez and Y He, "Internet Data Analysis for the Undergraduate Statistics Curriculum," *Journal of Statistics Education*, vol. 13, no. 3, November 2005.

[198] R. Baeza-Yates and C. Castillo, "Crawling the Infinite Web," *Journal of Web Engineering*, vol. 6, no. 1, pp. 49-72, March 2007.

[199] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-Law Distributions in Empirical Data," *SIAM*, vol. 51, no. 4, pp. 661-703, November 2009.

[200] J. A. Silva et al., "Data Stream Clustering: A Survey," *ACM Computing Surveys*, vol. 46, no. 1, pp. 1-31, October 2013.

[201] A. Bifet, "Adaptive learning and mining for data streams and frequent patterns," Universitat Politecnica de Catalunya, PhD Thesis 2009.

[202] M. Bolanos, J. Forrest, and M. Hahsler, "Introduction to stream: An extensible Framework for Data Stream Clustering Research with R," , 2011, pp. 1-33.

[203] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Continuous monitoring of distance-based outliers over data streams," in *The IEEE International Conference on Data Engineering (ICDE)*, Hannover, Germany, 2011, pp. 135-146.

[204] D. Pokrajac, A. Lazarevic, and L. J. Latecki, "Incremental Local Outlier Detection for Data Streams," in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Honolulu, Hawaii, USA, 2007, pp. 504-515.

[205] F. Cao, M. Ester, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *In Proceedings of the 6th SIAM International Conference on Data Mining*, Bethesda, Maryland, USA, 2006, pp. 328-339.

[206] Y. Chen and L. Tu, "Density-based clustering for real-ime stream data," in *KDD'07*, New York, NY, 2007, pp. 133-142.

[207] A. Amini, T. Y. Wah, and H. Saboohi, "On Density-Based Data Streams Clustering Algorithms: A Survey," *Computer Science and Technology*, vol. 29, no. 1, pp. 116-141, January 2014.

[208] L. C. Giralte, C. Conde, I. M. De Diego, and E. Cabello, "Detecting denial of service by modelling web-server behaviour," *ElsevierComputers and Electrical Engineering*, vol. 39, no. 7, pp. 2252–2262, October 2013.

[209] J. Velasquez, H. Yasuda, and T. Aoki, "Combining the web content and usage mining to understand the visitor behavior in a web site," in *The Third IEEE International Conference on Data Mining*, Florida, USA, 2003, pp. 669 - 672.

[210] J. Velasquez, H. Yasuda, T. Aoki, and R. Weber, "Acquiring knowledge about user's preferences in a web site," in *International Conference on Information Technology: Research and Education*, Newark, New Jersey, USA, 2003, pp. 375 - 379.

[211] K. Sequeira and M. Zaki, "ADMIT: anomaly-based data mining for intrusions," in *In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, 2002, pp. 386-395.

[212] V. Chandola, V. Mithal, and V. Kumar, "Comparative Evaluation of Anomaly Detection Techniques for Sequence Data ," in *In Proceedings of the 8th IEEE International Conference on Data Mining*, Pisa, Italy, 2008, pp. 743 - 748.

[213] S. Budalakoti, A. Srivastava, R. Akella, and E. Turkov, "Anomaly detection in large sets of high-dimensional symbol sequences," NASA Ames Research Center, Technical Report. NASA TM-2006-2145, 2006.

[214] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601-1604, April 2010.

[215] D. Drinfeld and N. Vlajic, "Smart Crawlers for Flash-Crowd DDoS: The Attacker's Perspective," in *IEEE World Congress on Internet Security*, Guelph, Canada, 2012, pp. 37 - 44.

[216] (2016, March) Internet Archive: Wayback Machine. [Online]. https://archive.org/web/

[217] J B Grizzard, V Sharma, C Nunnery, B B Kang, and D Dagon, "Peer-to-Peer Botnets: Overview and Case Study," in *First Workshop on Hot Topics in Understanding Botnets (HotBots)*, Cambridge, MA, 2007, pp. 1-8.

[218] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *In VLDB Conference*, 1997, pp. 426–435.

[219] D. Yang, E. A. Rundensteiner, and M. O. Ward, "Neighbor-based Pattern Detection for Windows

over Streaming Data," in *12th Intl. Conf. on Extending Database Technology:Advances in Database Technology (EDBT)*, Saint-Petersburg, Russia, 2009, pp. 529-540.

[220] F. Angiulli and F. Fassetti, "Detecting Distance-based Outliers in Streams of Data," in *ACM Conf. on Information and Knowledge Management (CIKM)*, Lisbon, Portugal, 2007, pp. 811-820.

[221] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density Based Local Outliers," in *In Proceedings of the ACM SIGMOD Conference*, Dallas, TX, 2000.

[222] D. K. Tasoulis, G. Ross, and N. M. Adams, "Visualising the cluster structure of data streams," , 81-92, 2007.

[223] C. Ruiz, E. Menasalvas, and M. Spiliopoulou, "C-Denstream:Using domain knowledge on a data stream," , 287-301, 2009.

[224] L. Liu, K. Jing, and Y. Guo, "A three-step clustering algorithm over an evolving data stream," , 160-164, 2009.

[225] J. Ren and R. Ma, "Density-based data streams clustering over sliding windows," , 248-252, 2009.

[226] J. Lin and H. Lin, "A density-based clustering over evolving heterogeneous data stream," , 275-277, 2009.

[227] I. Ntoutsi, A. Zimek, and T. Palpanas, "Density-based projected clustering over high dimensional data streams," , 987-998, 2012.

[228] M. Hassani, P. Spaus, M. M. Gaber, and T. Seidl, "Density-based projected clustering of data streams," , 311-324, 2012.

[229] A. Forestiero, C. Pizzuti, and G. Spezzano, "A single pass algorithm for clustering evolving data streams based on swarm intelligence," *Data Mining and Knowledge Discovery*, vol. 26, no. 1, pp. 1-26, January 2013.

[230] V. Bhatnagar, S. Kaur, and S. Chakravarthy, "Clustering data streams using grid-based synopsis," *Knowledge and Information Systems*, pp. 1-26, June 2013.

[231] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, and K. Zhang, "Density-based clustering of data streams at multiple resolutions," *ACM Trans. Knowledge Discovery from Data*, vol. 3, no. 3, July 2009.

[232] C. Jia, C. Tan, and A. Yong, "A grid and density-based clustering algorithm for processing data stream.," , 517-521, 2008.

[233] J. Ren, B. Cai, and C. Hu, "Clustering over data streams based on grid density and index tree," *Journal of Convergence Information Technology*, vol. 6, no. 1, 2011.

[234] Y. Yang, Z. Liu, and J. Zhang, "Dynamic density-based clustering algorithm over uncertain data streams," , 2664-2670, 2012.

[235] A. Amini and Y. W. Teh, "DENGRIS-Stream: A density-grid-based clustering algorithm for evolving data streams over sliding window," , 206-210, 2012.

[236] H. P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "LoOP: Local Outlier Probabilities," in *In the Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 1649–1652.

[237] D. Gusfield, *Algorithms on strings*, 1st ed. New York, United States of America: Cambridge University Press, 1997.

Appendix A – DDoS Malware

In this section, we give a systematic overview of DDoS-executing malware based on: 1) how they scan for new agents, 2) how they recruit new agents, and 3) how they control the agent population. The overview of DDoS malware is illustrated in Figure A.1.

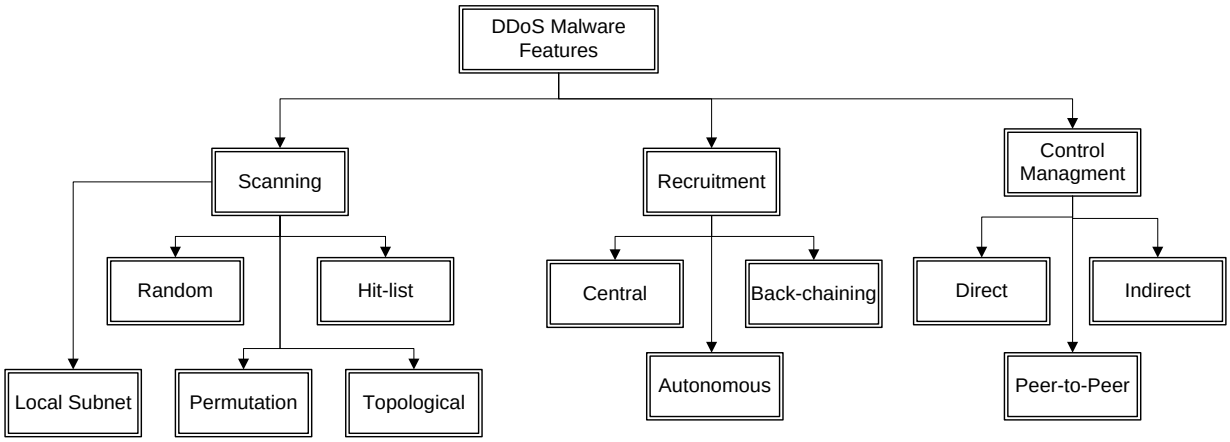


Figure A.1 – DDoS malware features

A.1 Scanning Strategies in DDoS-executing Malware

Before any DDoS attack may be launched, an army of agents must be compromised and recruited (into a Botnet) for participation in an attack. There are five possible approaches to identifying (i.e., scanning) for new agents:

- *Random scanning* – In random scanning each compromised host probes random addresses in the IP address space. This potentially creates a high traffic volume since many machines probe the same addresses.

- *Hit-list scanning* – A machine that performs hit-list scanning, probes all addresses from an externally supplied list. When it detects and compromises a vulnerable machine, it sends one half of its initial hit-list to the compromised machine so that this machine can repeat the same procedure. This technique allows a great propagation speed and no collisions during the scanning phase.
- *Topological scanning* – Topological scanning uses the information on the compromised host to select new targets. For instance, a scanning process may find IP addresses of new hosts on the compromised/infected machine.
- *Permutation scanning* – During permutation scanning, all compromised machines share a common pseudorandom permutation of the IP address space; each IP address is mapped to an index in this permutation. A machine begins scanning by using the index computed from its IP address as a starting point. Whenever it sees an already infected machine, it chooses a new random start point.
- *Local Subnet scanning* – Local subnet scanning can be added to any of the previously described techniques to preferentially scan for targets that reside on the same subnet as the compromised host.

The actual scanning process is executed as follows. The bot scans each host individually and determines the type and version of the operating system running on the host and if the host has any known vulnerabilities that may be exploited. For instance, a host running a specific version of the operating system may be susceptible to well-known vulnerabilities if the system has not been patched. The bot then generates a list of vulnerable hosts and transmits this information to the attacker by employing either an Internet Relay Chat (IRC) channel or by employing HTTP protocol messages.

A.2 Recruiting Strategies in DDoS-executing Malware

Once the attacker has identified and compromised a set of vulnerable machines, his next goal is to inject the actual attack code (i.e. malware) onto these machines. There are four common approaches to malware propagation:

- *Central* – In this method the attack code resides on a central server or a set of servers and each compromised host downloads the malware from this repository.
- *Back-chaining* – In this method the attack code is downloaded from the machine that was used to exploit the system. The infected machine then becomes the source for the next propagation point. Back-chaining propagation is more survivable than central-source propagation since it avoids a single point of failure (central server).
- *Autonomous* – This method avoids the file retrieval step by injecting attack instructions directly into the target host during the exploitation phase. The malware is carried in the payload of the worm and installed on every machine infected by the worm. Autonomous propagation reduces the frequency of network traffic needed for agent mobilization, and thus further reduces chances of attack discovery.
- *Passive* – Passive methods typically involve the attacker sharing corrupt files through e-mail attachments or building web sites that take advantage of known vulnerabilities in a secondary victim's web browser. Upon opening an e-mail attachment or accessing a web site with an embedded DDoS agent code, the secondary victim system is compromised.

A.3 Agent Control Mechanisms in DDoS-executing Malware

After successfully recruiting and infecting an agent network (i.e. network of bots), the attacker must be able to effectively control its Botnet. In particular, the attacker should be able to issue instructions for

attack execution or to be able to upgrade the disseminated malware on each individual agent. Based on the communication mechanisms deployed between agent and handler machines, we distinguish between three types of Botnets: 1) Botnets with direct communication, and 2) Botnets with indirect communication and 3) Botnets with peer-to-peer communication.

A.3.1 Direct Communication

In Botnets with direct communication, the agent and handler machines need to know each other's identity (i.e. IP and port addresses) in order to communicate. This is typically achieved by hard-coding the IP address of the handler machine in the attack code that is installed at the agent machine. In turn, each agent reports its own identity to the handler; hence, the handler ends up collecting the IP addresses of all involved agents. The obvious drawback of this approach is that the discovery of the network's handler can expose the whole DDoS network. Also, since agents and handlers must maintain open ports to facilitate direct intercommunication, they are fairly easy to detect.

A.3.2 Indirect Communication

Due to drawbacks of the direct method, an alternative method utilized by attackers is indirect communication using Internet Relay Chat channels – see section 2.2.1.3. One key property of IRC-based Botnets is the use of IRC channel as a form of centralized pull-based and thus less involved type of communication for the attacker. This property provides the attackers with very efficient communication. However, this property also serves as a major disadvantage to the attacker since it represents the single-point of failure.

A.3.3 Peer-to-Peer Communication

Due to the inherent drawbacks of the previous two communication models, Botnet designers have started building distributed, the so called peer-to-peer, methods of Botnet communication [217]. In peer-to-peer Botnets, there is no centralized point for C&C (i.e., centralize point of failure). Instead, each compromised host acts as both a client and a server. Consequently, even if a significant number of bots in a P2P Botnet are identified and taken off-line, the overall Botnet's operation will be only partially affected.

Appendix B – Recall, Precision and F_1

In this section, we describe the general meaning and relevance of the recall, precision and F_1 metrics to the study presented in chapter 4.

First an introduction to some terminology is necessary. The Table B.1 shows a confusion matrix which depicts how predictions on instances are tabulated.

Table B.1 – The confusion matrix

		Predicted Label	
		Positive	**Negative**
Known Label	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

For simplicity, the assumption is that each instance can only be assigned one of two classes: Positive or Negative (e.g. a web session may be benign or malicious). Each instance (e.g. a web session) has a Known label, and a Predicted label. Some method is used (e.g. machine learning algorithm) to make predictions on each instance. Each instance then increments one cell in the confusion matrix.

A confusion matrix can be summarized using various formulas. The Table B.2 shows the most commonly used formulas.

Table B.2 – Summaries of the measures

Measure	Formula	Meaning
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct.
Recall / Sensitivity	TP / (TP + FN)	The percentage of positive labeled instances that were predicted as positive.
Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct.
F_1	2*Recall*Precision/ Recall + Precision	Summary of Recall and Precision

Different problem domains call for the need to use different measures for summarizing prediction quality.

1. For example, in a data set of 10,000 samples, where 100 of these samples are labeled positive, a predictor that predicts "Negative" for every instance it is presented with evaluates to Precision = 100%, and Accuracy = 99%. This predictor would be entirely useless, and yet these measures show it performs very well. The same predictor would evaluate to Recall = 0%. In this case, Recall seems to be most in tune with how well the classifier is actually performing.
2. If a classifier predicts positive on all instances in the data set in case 1.), then Precision = 1%, Recall = 100% and Accuracy = 1%. In this case, Precision and Accuracy show that this classifier is problematic.
3. F_1 score summarizes the first two metrics into a single value, in a way that both metrics are given equal importance. Namely, the F_1 score penalizes a classifier that gives high recall but sacrifices precision and vice versa. As stated above, a classifier that classifies all examples as positive has perfect recall but very poor precision. Recall and precision should therefore be close to each other, otherwise the F_1 score yields a value closer to the smaller of the two.

Appendix C – Nonparametric Statistical Tests

C.1 Mann-Whitney Rank Sum Test (also known as Wilcoxon-Mann-Whitney Rank Sum Test)

The Mann-Whitney Rank Sum test, when applied on two independent data samples, evaluates whether there is a significant difference between the two sample medians. If so, we can conclude that there is a high likelihood that the samples represent populations with different median values. Initially, the data values in both samples are assigned a ranking score with the highest rank given to the best score (in our case that would be the most popular page in web page popularity statistic or the longest viewed web page in the page viewing time statistic or the longest browsing sequence length in the browsing length statistic). In instances where two or more subjects have the same score, the average of the ranks is assigned to all scores tied for a given rank. Next, once all of the subjects have been assigned a rank, the sum of the ranks for each of the groups is computed. Upon determining the sum of the ranks for both groups, the values U_1 and U_2 statistic are computed by employing (C.1) and (C.2) (from [167]).

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - \sum R_1 \quad \text{(C.1)}$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - \sum R_2 \quad \text{(C.2)}$$

where $\sum R_1$ is the sum of the ranks in sample 1 and $\sum R_2$ is the sum of the ranks in sample 2. The smaller of the two values U_1 versus U_2 is designated as the obtained U statistic. If the sample sizes employed in a study are relatively large, the normal distribution can be employed to approximate the Mann-Whitney U statistic. Although sources do not agree on the value of the sample size which justifies employing the

normal approximation of the Mann-Whitney distribution, they generally state that the size of each sample should be greater than 20. Expression (C.3) (from [167]) provides the normal approximation of the Mann-Whitney U test statistic.

$$z = \frac{U - \frac{n_1 n_2}{2}}{\sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}} \tag{C.3}$$

In order for the median between the two samples to be significantly different, the obtained absolute value of the z-score must be equal to or greater than the tabled critical value at the pre-specified level of significance. The tabled critical two-tailed significance with 95% confidence is z = 1.96.

C.2 Kolmogorov-Smirnov Test for 2 Independent Samples

This nonparametric test compares the cumulative frequency distributions of the two independent samples. If, in fact, the two samples are derived from the same population, the two cumulative frequency distributions would be expected to be identical or reasonably similar to one another. The test protocol for the Kolmogorov-Smirnov test for two independent samples is based on the principle that if there is a significant difference at any point along the two cumulative frequency distributions, the researcher can conclude there is a high likelihood the samples are derived from different populations.

The first step in applying the test is to calculate the so-called Kolmogorov-Smirnov statistic (K-S) which is the supremum of the set of difference between the empirical cumulative distribution functions (empirical CDFs) of the two samples. It is defined as follows (from [167]):

$$K - S_{n_1 n_2} = \sup_x \left| F_{n_1}(x) - F_{n_2}(x) \right| \tag{C.4}$$

where $F_{n_1}(x)$ is the empirical CDF of one of the samples and $F_{n_2}(x)$ is the empirical CDF of the other sample.

The empirical CDF F(X) can be defined as follows. Let (x_1, …, x_n) be independent and identically distributed (i.i.d.) real random variables with the common CDF F(t). Then the empirical distribution function is defined as (from [167]):

$$\widehat{F_n}(t) = \frac{\#\text{ of elements in the sample} \leq t}{n} = \frac{1}{n}\sum_{i=1}^{n} 1\{x_i \leq t\} \quad \text{(C.5)}$$

where $1\{A\}$ is the indicator of event A.

In order to determine whether the empirical CDFs of the two samples are significantly different at α=0.05 level of significance (i.e., with 95% confidence), $K - S_{n_1 n_2}$ must be greater than or equal to the so-called critical value of Kolmogorov-Smirnov test – parameter K. For samples with more than 25 values, K corresponds to (from [167]):

$$K = 1.36 \cdot \sqrt{\frac{n_1 + n_2}{n_1 n_2}} \quad \text{(C.6)}$$

where n_1 and n_2 are the number of values in the two samples.

Appendix D – Unsupervised Outlier Detection Algorithms for Data Streams with Concept Drift

D.1 Continuous Outlier Detection (COD) - Distance-based Outlier Detection Algorithm

The Continuous Outlier Detection or COD is an outlier detection algorithm based on the number of neighbourhood points within a spherical radius. In COD [203], given a set of instances currently in the sliding window, the new instance x is an outlier if there are less than k instances within (Euclidean) distance R from x. Otherwise x is labeled as a non-outlier.

The bounded storage and concept drift constraints in COD are addressed by employing the sliding time window model with equal weight assignment to each instance in the window. The real-time/single-pass constraint of data streams is addressed by employing the M-tree [218] data structure to efficiently (in O(logn) time) execute the range query search for k neighbors of x that are within distance R from it. Note that COD is an improvement of Abstract-C [219] and STORM [220] which provide worse (O(n)) space complexity and worse (O(n)) running-time, respectfully.

The three shortcomings of COD include:

1. The algorithm employs a fixed sliding time window model with equal weights and as such can only react to slow/gradual concept drifts.
2. The algorithm has a high space complexity as it maintains all points in the sliding window.
3. The algorithm assumes the existence of exclusively spherically-shaped clusters (i.e., assumes a specific type of input distribution). Note that spherically-shaped clusters can overlap between each other dependent on the selected cluster radius R. As such, this algorithm may be sensitive to noise (i.e., actual outlier points may be labeled as inliers).

D.2 Density-based Outlier Detection Algorithms

In density-based methods, such as ILOF [204], Denstream [205] and D-Stream [206], clusters are formed as dense regions of input space which are separated from sparse regions of input space (i.e., outlier clusters/areas). The density-based algorithms have the advantage over COD as they can discover clusters of arbitrary shape. As such they may be less sensitive to noise in data streams.

D.2.1 Incremental Local Outlier Factor (ILOF)

The Local Outlier Factor (LOF) is density-based algorithm for finding local high-density and low-density regions of the input space. The main idea of the LOF algorithm [221] is to label each input instance as an outlier with a confidence value called the local outlier factor (LOF). Data points with high LOF are located in low-density regions of the input space and typically represent stronger outliers, unlike data points belonging to uniform (high-density) clusters that usually tend to have lower LOF values. The algorithm for computing the LOFs for a data instance q includes the following steps (from [204]**)**:

1. Compute k-distance(q) as the distance to the k^{th}-nearest neighbor of q.

2. Compute reach-ability distance of data instance q (Expression (D.1)) with respect to data instance p in the sliding window.

$$\text{reach-dist}(q, p) = \max(d(q, p), \text{k-distance}(p)) \tag{D.1}$$

where d(q,p) is Euclidean distance from q to p.

3. Compute local reach-ability density (LRD) of data instance q (Expression (D.2)) as the inverse of the average reach-ability distance based on the k-nearest neighbors (KNN) of the data instance q.

$$\text{LRD}(q) = \frac{1}{\sum_{p \in KNN} \frac{\text{reach-dist}(q,p)}{k}} \tag{D.2}$$

where KNN are neighbors of q that are k-distance or less distant from q.

4. Compute LOF of data instance q (Expression (D.3)) as the ratio of the average local reach-ability density of q's KNN and local reach-ability density of the data instance q.

$$\text{LOF}(q) = \frac{\frac{\sum_{p \in KNN} \text{LRD}(p)}{k}}{\text{LRD}(q)} \tag{D.3}$$

The incremental version of the LOF algorithm, i.e., ILOF [204], computes LOF value for each new input instance by employing the M-tree to perform the KNN queries. However, the drawbacks of ILOF include:

1. It employs a fixed sliding time window for exclusively slow concept drift handling (same as COD).
2. It has the worst time complexity, among the four algorithms, since it requires updating the KNN (and LRD/LOF) of all points that are within k-distance from the new input instance x (in O(nlogn)).

3. The algorithm does not provide any recommendations on how high the LOF value must be for the point to be declared an outlier.
4. Similarly to COD, this algorithm has a high space complexity as it maintains all points in the sliding window.

D.2.2 Denstream

Denstream [205] is a data steam clustering algorithm that maintains two structures — *p-micro-clusters* (potential or non-outlier clusters) and *o-micro-clusters* (outlier clusters). Note that points associated with o-micro-clusters are considered as outliers in the algorithm. For each p- and o-micro-cluster, the algorithm stores the cluster feature vector — a popular summarization technique for data stream clustering. Instead of storing all points included in each cluster, only a vector of the summarization features is maintained.

D.2.2.1 Micro-cluster Feature Vector

The cluster feature vector of a micro-cluster includes the following entries: N, the number of data objects, WLS, the weighted linear sum of the data objects, and WSS, the weighted sum of squared data objects. Each p- and o-micro-cluster structure has an associated weight that indicates its importance based on the temporality (micro-clusters with no recently arrived objects tend to lose importance, i.e. their respective weights continuously decrease over time).

The weight w of a micro-cluster M_i at time T is computed according to Expression (D.4):

$$w = \sum_{j \in M_i} f(T - t_j) \tag{D.4}$$

where $t_1, \ldots, t_j$ are the timestamps of when points $1, \ldots, j$ were mapped to M_i.

The importance of each object decreases according to the fading function in (D.5), parameterized with λ, a user-defined parameter [200].

$$f(t) = 2^{-\lambda t} \tag{D.5}$$

The WLS and WSS for micro-cluster M_i are computed according to Expressions (D.6) and (D.7), respectively. Note that these summarization features can be incrementally updated as the new points are inserted into the cluster.

$$WLS = \sum_{j \in M_i} f(T - t^j)\, x^j \tag{D.6}$$

$$WSS = \sum_{j \in M_i} f(T - t^j)\, x^{j^2} \tag{D.7}$$

From Expressions (D.6) and (D.7), it is possible to compute a micro-cluster's radius r and center c according to Expressions (D.8) and (D.9), respectively.

$$r = \sqrt{\left(\frac{WSS}{w} - \left(\frac{WLS}{w}\right)^2\right)} \tag{D.8}$$

$$c = \frac{WLS}{w} \tag{D.9}$$

D.2.2.2 Insertion and Labeling Process of New Data Inputs

The insertion step in the algorithm of a new object x_j is as follows:

1. Locate the nearest p-micro-cluster P_i and update P_i's N, WLS, WSS, r and c temporarily for now. If the updated r of the nearest P_i is less than a predefined boundary ε (a user-defined parameter) insert x_j into P_i. Note that a point inserted into p-micro-cluster is recognized as an inlier (i.e., non-outlier).
2. Otherwise, the algorithm resets P_i's N, WLS, WSS, r and c to the original values before the arrival of x_j. Instead, the algorithm tries to insert x_j into its closest o-micro-cluster. The insertion process is the same as in step 1. If the insertion of the new point into its closest o-micro-cluster is successful (i.e., r of the closest o-micro-cluster is less than ε) the new point is recognized as an outlier.
3. Lastly, if the insertion to the closest p- or o-micro-cluster was unsuccessful in both cases, a new o-micro-cluster is created with a single point x_j (implying that the point is recognized as an outlier).

Note that algorithm has the initial training phase that constructs the initial set of p- and o-micro-clusters. The number of training data points processed during the training phase is a user-defined parameter.

D.2.2.3 Micro-cluster Maintenance

The algorithm promotes o-micro-clusters to p-micro-clusters whenever o-micro-cluster's weight, after insertion of a new point, exceeds $\beta \times \mu$ (two user-defined parameters). The promotion of an o-micro-cluster to (regular) p-micro-cluster indicates that input distribution (i.e., concept) captured with this o-micro-cluster has now become a dominant new input distribution/concept.

The outdated or sparse p- and o-micro-clusters are discarded as follows. At delimited time periods T_p (as defined in (D.10)), each p-micro-cluster with weight $w < \beta \times \mu$ is removed.

$$T_p = \frac{1}{\lambda} \log\left(\frac{\beta\mu}{\beta\mu - 1}\right) \tag{D.10}$$

Similarly, a o-micro-cluster is removed if its weight w is less than ξ, as defined in (D.11):

$$\xi = \frac{2^{-\lambda(t_c - t_o + T_p)} - 1}{2^{-\lambda T_p} - 1} \quad \text{(D.11)}$$

In (D.11) t_c is the current time and t_o is the creation time of the o-micro-cluster. Note that o-micro-cluster that is never promoted to p-micro-cluster represents the noise in data stream and not the true emerging concept/input distribution in the data stream.

In summary, the Denstream algorithm addresses the constraints of the data stream traffic as follows:

- The bounded memory is handled by employing the micro-cluster synopsis structure and by periodically removing the clusters with older and no-longer-relevant data. Note that the space complexity of Denstream is orders of magnitude smaller than in COD and ILOF since only the summary of data inputs is maintained by the algorithm.
- The real-time constraint is handled by employing a simple incremental update of cluster feature vectors of micro-clusters as new points are presented to the algorithm.
- The concept drift is handled by employing a decaying weight function over the micro-clusters to ensure that clusters with old and no-longer-relevant data points (that are highly likely part of the old and outdated concept) are discarded from the learning process. The new emerging concepts are initially placed in o-micro-clusters and are subsequently placed inside to p-micro-clusters once they become part of the dominant input trend.

A number of extensions of Denstream have been proposed in literature, including [222], [223], [224], [225], [226], [227], [228] and [229], for different application scenarios. Note that the extensions achieve the same or worst time/space complexity and handle the concept drift in similar fashion as Denstream.

D.2.3 D-Stream

D-Stream is another density-based algorithm that clusters regions of input space, called grids, into dense (i.e., non-outlier) and sparse (i.e., outlier) grids. D-Stream and Denstream algorithms have three similar attributes. Firstly, both algorithms summarize input data into a synopsis data structure (micro-cluster in Denstream and density-grid in D-Stream). Secondly, both algorithms address the concept drift in similar fashion by applying the decaying weight function over the synopsis data structure and by removing sparse cluster/grids once their weight falls below a user-specified threshold. Thirdly, both algorithms facilitate incremental and efficient updates to synopsis data structure to address the real-time/single-pass constraint of data streams. The definition of the grid and the outlier detection process in D-Stream are described below.

D.2.3.1 Multi-dimensional Density-Grid

The definition of the multi-dimensional density grid is as follows (from [206] and [230]). Let us assume that input data is bounded by d dimensions and each data instance is defined within the space $S = S_1 \times S_2 \times \cdots \times S_d$. Also, assume that the first x dimensions are continuous and subsequent $y = (d - x)$ dimensions are categorical. Let l_i and h_i, respectively, be the lowest and the highest data values along numeric dimension i, as known to the domain expert[19]. The range $r_i = [l_i , h_i]$ of each numeric dimension i is divided into p_i number of equally-wide intervals $[l_i^1 , h_i^1]$, $[l_i^2 , h_i^2]$, ... , $[l_i^{pi} , h_i^{pi}]$, such that $l_i = l_1$ and $h_i^{pi} = h_i$. Note that categorical data naturally fits in this structure, where the number of intervals p_i for a categorical

[19] For instance, each numerical feature can be normalized to have values between 0 and 1.

attribute is the size of its domain. Domain values of categorical attribute are arbitrarily mapped to continuous natural numbers starting from one, which are used as interval values.

The density grid cell g is defined as a partitioning of S into $S_{1, j1} \times S_{2, j2} \cdot \cdot \cdot \times S_{d, jd}$ and is denoted by its coordinates $j_i = 1, \ldots, p_i$.

The data space S is partitioned into N density grids as defined in (D.12):

$$N = \prod_{i=1}^{d} p_i \tag{D.12}$$

A data instance $x = (x_1, x_2, \ldots, x_d)$ can be uniquely mapped to a density grid cell g as shown in (D.13):

$$g(x) = (j_1, j_2, \ldots, j_d) \text{ where } x_i \in S_{i, ji} \tag{D.13}$$

Each object at time t is associated to a density coefficient that decreases over time, as shown in (D.14), where $\lambda \in (0, 1)$ is a decay factor.

$$D(\mathbf{x}_j, t) = \lambda^{t - t_j} \tag{D.14}$$

The density of a grid cell g at time t, D(g, t), is given by the sum of the adjusted densities of each object that is mapped to g at or before time t (D.15).

$$D(g, t) = \sum_{j \in E(g,t)} D(x, t) \tag{D.15}$$

Note that the algorithm updates the density of a grid D(g, t) incrementally as shown in (D.16) and only when a new data record x_j is mapped to that grid.

$$D(g , t_j) = \lambda^{tj - tl} D(g, t_l) + 1 \quad (D.16)$$

where t_j is the arrival time of the new point to grid g and t_l is the last time grid g has received/absorbed a point.

The grids can be labeled as either dense (non-outlier regions of the input space) or sparse (as outlier regions of the input space). At time t, a grid g is a dense grid if

$$D(g, t) \geq \frac{C_l}{N(1-\lambda)} \quad (D.17)$$

where $C_l > 1$ is a user-defined threshold, N is defined in (D.12) and λ is the decaying factor of inputs. The expression in (D.17) is based on the proof in [206] showing that the average density of the grid can be at most $\frac{1}{N(1-\lambda)}$ while the maximum density of a grid can be at most $\frac{1}{(1-\lambda)}$. This also implies that $N > C_l$ must be true.

At time t, a grid g is a sparse and sporadic grid if

$$D(g, t) < \frac{C_l(1-\lambda^{t-tg+1})}{N(1-\lambda)} \quad (D.18)$$

where $C_l <= 1$ is another user-defined threshold and t_g ($t > t_g$) is the last insertion of a point into grid g.

D.2.3.2 Insertion and Labeling Process of New Data Inputs

In the algorithm, authors propose storing grid cell density summations (as given in (D.16)) in a hash table for efficient updates of densities as new data instances are presented to the algorithm. The hash table (non-uniquely) maps the key (grid coordinates $j_1, j_2, ... , j_d$) to a list of grid cell density summations. The hash table employs a hash function to compute an index from the given key into an array of grid cell density summations. Considering that the number of grid cells may be large, especially in high-dimensional streams, only the density summations for grid cells that have data mapped to them should be stored.

The insertion process for a new instance x_j in D-Stream is as follows:

1. By employing the mapping function shown in (D.13), calculate the coordinates of the grid G_i for x_j.
2. Locate G_i in the hash table by using the computed coordinates as the retrieval key. If G_i does not exist in the hash table, create it.
3. Update G_i's density by employing (D.16).

Note that if x_j is assigned to a dense grid then it is recognized as an non-outlier, otherwise (if it is assigned to a sparse and sporadic grid) it is recognized as an outlier.

D.2.3.3 Density-grid Maintenance

The density grid maintenance in D-stream is similar to micro-cluster maintenance in Denstream. Namely, the algorithm promotes sparse grids to dense grids whenever sparse grid's density satisfies the condition in (D.17) and downgrades dense grids to sparse grids if condition in (D.18) is satisfied.

Also, periodically, the algorithm verifies if sparse (outdated) grids should be discarded. The periodicity of the check T_p is defined in the following expression (D.19):

$$T_p = \left\lfloor \log\left(\max \left\{ \frac{C_l}{C_m}, \frac{N - C_m}{N - C_l} \right\} \right) \right\rfloor \quad \text{(D.19)}$$

where $C_l > 1$ is the same parameter used in (D.17) and (D.18) and $C_m > 1$ is another user-defined threshold. The sparse and sporadic grid is removed if the condition in (D.18) is satisfied for 2 T_p intervals in a row. Note that sparse and sporadic grid that is never promoted to dense grid represents the noise in data stream and not the true emerging concept/input distribution in the data stream.

In summary, the D-stream algorithm addresses the constraints of the data stream traffic as follows:

- The bounded memory is handled by employing the density-grid cell synopsis structure and by periodically removing grids with older and irrelevant data mappings.
- The real-time constraint is handled by employing a simple incremental update of grid cell densities and by employing a hash table to efficiently search for new point mappings.
- The concept drift is handled by employing a decaying weight function over the grid cells to ensure that sparse grid cells with old and no-longer-relevant data point mappings (that are highly likely part of the old and outdated concept) are discarded from the learning process. The new emerging concepts are initially placed in sporadic grids and are subsequently mapped to dense grids once they become part of the dominant input trend.

Similarly to the case of Denstream algorithm, a number of extensions of D-Stream have been proposed in literature, including [231], [232], [233], [234], [235] and [230] for different application scenarios. Again, note that these extensions also achieve the same or worst time/space complexity and handle the concept drift in the same fashion as D-Stream.