A Research on Detection and Classification of Automated HTTP Communication
(HTTP 自動通信の検出と分類に関する研究**)**

防衛大学校理工学研究科後期課程

電子情報工学系専攻　　情報知能メディア学教育研究分野

トラン　コン　マン

平成２９年３月

Dedicated to

My family

Declaration

The work in this thesis is based on research carried out at the Software Laboratory of Computer, Intelligent and Media System of Electronics and Information Engineering, Graduate School of Science and Engineering, National Defense Academy, Japan. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

研究成果の概要

近年，様々なアプリケーションの実装基盤として HTTP が用いられるようになってきた．利用者が必要とするサービスのほとんどがウェブ上で提供されるため，HTTP はインターネット上で最も多用されるプロトコルと認知されている．このため多くのネットワーク環境において，HTTP 通信は制限されることなく自由に利用できる．HTTP は，個別の要求と応答のみが定義されており，会話的な相互通信を行わないため，アプリケーションが連続的かつ会話的な非同期通信を必要とする場合は，クライアントは能動的かつ自動的にサーバへ接続要求を送信し続ける必要がある．この論文では，このような通信を行うソフトウェアを自動化ソフトウェア (autoware)，生成された通信を自動化トラフィックと呼ぶ.

自動化トラフィックには，OS の更新などの良性のもの，ボットネットの C&C 通信などの悪性のもの，およびアドウェアなどのグレーのものがある．ユーザのコンピュータやネットワークに危害を与えるのはマルウェアだけでなく，アドウェアなどのグレーウェアの可能性もある．一部のアドウェアは，ユーザのウェブアクセスの習慣や設定を監視し，その情報をターゲット広告を目的とした第三者に送信する．この自動化の結果として，ウェブアクセスの通信はユーザのウェブ操作とは無関係に生起することになる．正当なユーザーは，未知のトラフィックや制御されていないトラフィックのためにインターネット上でランザクションを行うため，常にリスクに直面している．近年，サイバー犯罪者は，不正なアドウェア，スパイウェア，ボットなどの悪意のある HTTP オートウェアを拡散させるための通信媒体としてウェブを

活用するようになってきている．

このような問題への従来の対策のひとつとして，アンチウィルス（AV）製品は，コンテンツおよびシグネチャベースのマルウェア検出技術を用いてアプリケーションの良性または悪性を判定する．しかし，近年のさまざまな調査の結果，シグネチャベースの手法を使用して検出されないマルウェアが数多く存在し，主要な AV エンジンは最近のマルウェアのわずか 30 〜 70% しか検出できていないことが明らかとなった．本研究では，コンテンツベースの検出手法における問題に対処するため，ネットワークトラフィックの分析と分類による手法を用いる．

これまでの HTTP 環境におけるマルウェア検出は，ボットネットを対象としたものが多い．このような検出法は，一般的なマルウェアを対象としているため，正当なソフトウェアに類似した様々なアドウェアなどのグレーウェアを識別することができない．したがって，HTTP 自動通信を分析し，それらのアクティビティを検出および分類する必要がある．そこで本研究では，自動通信の振る舞いの特徴量を観測，分析することにより，ホストベースおよびネットワークベースでの HTTP 自動通信を検出，分類する手法を提案する．

ホストベースの検出アプリケーションモデルは，メモリとリソースの制限を伴う単一の PC に適用できる利点がある．ホストベースのシステムを導入することにより，マルウェア感染の可能性のあるトラフィックを低減できるため，ネットワーク全体のリスク軽減に役立つ．このアプリケーションモデルは，ユーザがネットワークトラフィックを監視して，不審なトラフィックをフィルタリングすることができる．

ネットワークレベルの検出手法は，特定の URL ではなく，特定の目的のための URL グループに着目する．この分析結果は，ネットワークやシステムの管理者がユーザにはほとんど知られていない HTTP 自動トラフィックを検知することができる．その結果から，HTTP オートウェアによって引き起こされる内部脅威を早期に検知することができる．

様々な種類のオートウェアの通信が混合した実ネットワークトラフィックを用いて実験を行った結果，高い識別精度が得られ，手法の有効性を確認した．また，提案手法を実環境に実装する場合のアプリケーションモデルについても考察，提案している．

今後の課題は誤警報を減らすことと達成されている結果を広げることである．ネットワークベースの手法において，クラスタ化されなかった疑わしい URL を通常の URL と識別するために，新しい特徴量について検討する必要がある．また，いくつかのソフトウェアはセキュアチャネルを使用する傾向があるため，悪意のある HTTPS 通信の検出について検討する必要がある．

Outline of Research Results

HTTP is recognized as the most widely used protocol on the Internet since applications are being transferred more and more by developers onto the web, and users can find everything they need through web services. For this reason, HTTP based communication is always allowed in most of networks. Utilization of HTTP based software is blooming and reaching all Internet users. HTTP is designed to defines only individual requests and responses protocol and it does not perform interactive communication protocol. Therefore, HTTP based autoware have to actively send requests to its servers in order to perform the communication with their servers.

Automated communication can be good such as OS updates, or malicious such as botnet command and control (C&C) or gray such as adware. In computer system, not just malware harm the users computers and network system but also grayware such as adware. Adware monitor web access habits or preferences and transmits that information to the third parties that use it for target advertising [1]. As a result of this automation, communication is no longer strictly driven by user actions, and legitimate users alway face with risks since they do their transaction on the Internet because of the unknown or uncontrolled traffic. In recent years, cyber criminals turn to fully exploit web as a medium of communication environment to lurk forbidden action or to spread HTTP malicious autoware such as fraudulent adware, spyware or bot.

One of traditional approach is to detect the benign or malicious application

such as Anti-Virus (AV) products. They are the most common content and signature based malware detection techniques. These types of AV software employ signature based detection to identify variants of known malware. As a consequence, the signature generation and update cycle cause an inherent delay in protecting users against new variants of malware. Additionally, with the aim of limiting AV engines effectiveness, malware authors have developed increasingly sophisticated evasion techniques such as packing and polymorphism, aimed at circumventing detection by AV engines. Various studies figure that many undetected malware binaries by using signature-based techniques, and major AV engines just detect only 30% to 70% of recent malware. Overcoming the issue of contents based detection studies, network traffic analysis and classification are suggested and approached.

In HTTP environment, most of the work concerning malware detection focuses on such as botnet. The same type of detection methods would be unsuccessful when dealing with grayware, for example adware, which a part of them are more similar to legitimate software than such types of malware. Nevertheless, recently grayware represents a serious threat to privacy and, as such, the research on grayware is also important, especially in terms of detection approaches. In network, the traffic of all kind of software are merged transparent with each others. The classification and detection of their purposes are really serious challenge because of similarity in their requests format and structures.

This raises the demand for analyzing HTTP autoware communication behavior to detect and classify malicious and normal activities via HTTP traffic. Hence, in this research, based on many studies and analysis of the autoware communication behavior,a new method using minor features at application layer and access graph to detect and classify HTTP autoware communication at host and network level are presented, and they are summarized as bellow:

At host level, based on the observation of communication pattern of HTTP

autoware, it is proposed a detection method of suspicious HTTP-based autoware. The behavior of autoware is observed and analyzed through two parameters periodic access and access rate. The observation shows that these two features of suspicious autoware are stable than others normal autoware. In another word, there is almost no variation in the graph of periodic access and access rate in a period of time. The advantage of host-based application is that it can be applied for single PC (without interconnected network) with limitation of memory and resource. In addition, the design in host space within interconnected network will help to reduce the risk for the whole network since the traffic are possible generated from malware infection can be banned from the source. It is self-surveillance for users. The application will help users to monitor or watch their traffic to identify suspicious one and do the filtering before they are allowed to join the network from their PC.

At network level, based on the study and analysis of the autoware communication behavior, a method clustering and identifying HTTP automated communication is proposed. In that, multiple traffic from many clients are observed and behavior of each kind of autoware are recognized. In that, malicious HTTP-based bots always connect to their command and control server periodically in order to get the commands and updates. The number of requests from malicious bots are not high as normal autoware (e.g. updater and downloader) which just generate requests with a long interval than unusual malicious bots. Malicious access is recognized by scoring the its access speed. Malicious bots often connect to one control domain and to a specific server resource during a given period of time. Difference with that, unwanted HTTP applications, or grayware, such as annoying adware or spyware, often report back to or request new information from many external resources. Therefore, they keep communicating to their numerous advertising sites or URLs to update pop-up or advertisement and commercial content areas. Autoware will behave the same communication pattern to its difference URLs. if they are requested at the same or approximately equivalent timing, access graph of URLs from a specified autoware are presented similar. In addition, autoware requested to many URLs with the same timing, so the access duration to these URLs is approximately equal. It means that the first and the last requests' timing to these URLs are almost the same with others. Suspicious autoware, for instance adware, since they update contents, like other autoware they access to many URLs however they will collect data from many URLs of multiple sites which own various domain names. This is not alike with normal autoware, such as a electronic newspaper, since it self-refresh the contents of presenting page by accessing to many URLs but with only one domain name. A suspicious autoware starts with the time of human computer start. Therefore, it expected that the access duration of a suspicious autoware might be similar with user computer interaction. The results in network level are not focused for a specific URLs but for group of URLs for a specific purpose. The findings assist network and system administrator clarify the HTTP automated traffic which are almost unknown to users, from there the internal threats caused by HTTP autoware might be inspected early.

All methods are experimented by using real traffic and give a good results since mixture traffic of all kind autoware are still identified and detected. In that, suspicious and other autoware traffic are detected in experiment data for host-based proposed method. In network based proposed method, in general, 100% URLs are clustering and identifying into normal, suspicious and malicious groups. The accuracy rate reaches at 91.18% and error rate constitutes 8.82%.

Host based or network level methods are also able to be applied in the real environment. Especially, used features are minor which are easily extract from application layer traffic. All experiments show good results. In that, network proposed method can be implemented in distributed processing environment such as Map Reduce of Hadoop. These will help increasing the performance in detection process. the implementation of big data application for network level proves that it shows good performance since it takes the logarithm complexity in processing.

Although, false alarm is still exists, the research propose a new method to detect and classify to all kind of HTTP automated traffic which are not just malicious but also suspicious and normal.

The key necessary improvement in the future work is to reduce the false alarm and to extend the achieved results. For this to be done, new features are considered to be added, that are related to the URLs properties to improve the accuracy in suspicious and normal identifying for unclustered URLs in network based method.

Along with that, deep machine learning method is also considered in order to improve the achievement of current result. In addition, suspicious and malicious software always effort to spread out them-self on other clients in the same network. A proposal method in order to cluster of clients based on their automated communication is considered, in which, system or network administrator can score the activity of malware and also early detect the threats from internal. In addition, malicious HTTPS detection need to be considered since all the communication has trend to establish in secure channel.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The Internet, nowadays, has become more and more an importance part of everyday life of people. Every services, which users need, are able to find on the Internet such as online banking, advertising and shopping. Internet services have expanded its role from a basic communication network to interconnected set of among things. Therefore, desktops and servers become more complicated, they employ an increasing amount of automatic, non-user initiated communication [2]. Automated communication can be good such as OS updates, or malicious such as botnet command and control (C&C) or gray such as adware. In computer system, not only malware but also grayware such as adware can harm the users' computers and network system. Adware monitor web access habits or preferences and transmits that information to the third parties that use it for target advertising [1]. As a result of this automation, communication is no longer strictly driven by users' actions, and legitimate users always face risks since they perform their transaction on the Internet because of the unknown or uncontrolled traffic, as presented in Figure 1.1.

The threats of Internet transaction can be internal and external. External

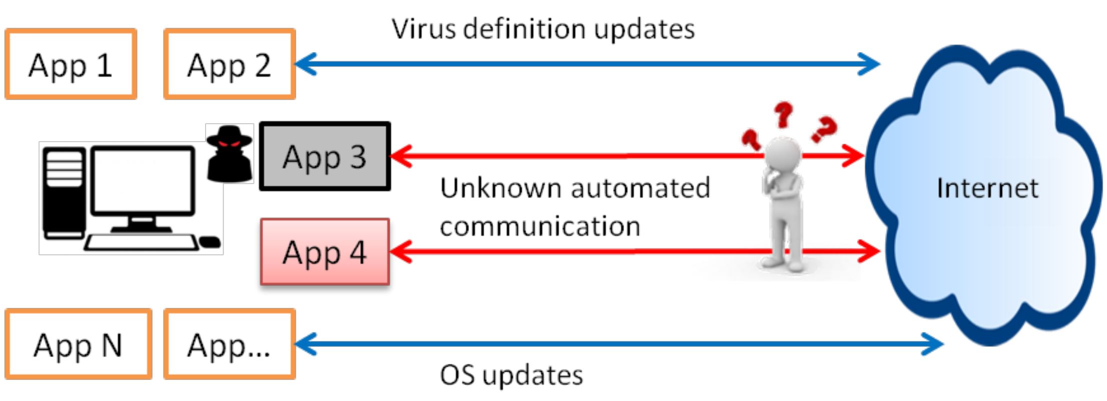


FIGURE 1.1: A lot of non-human (automated) traffic is unknown and unnotice with users.

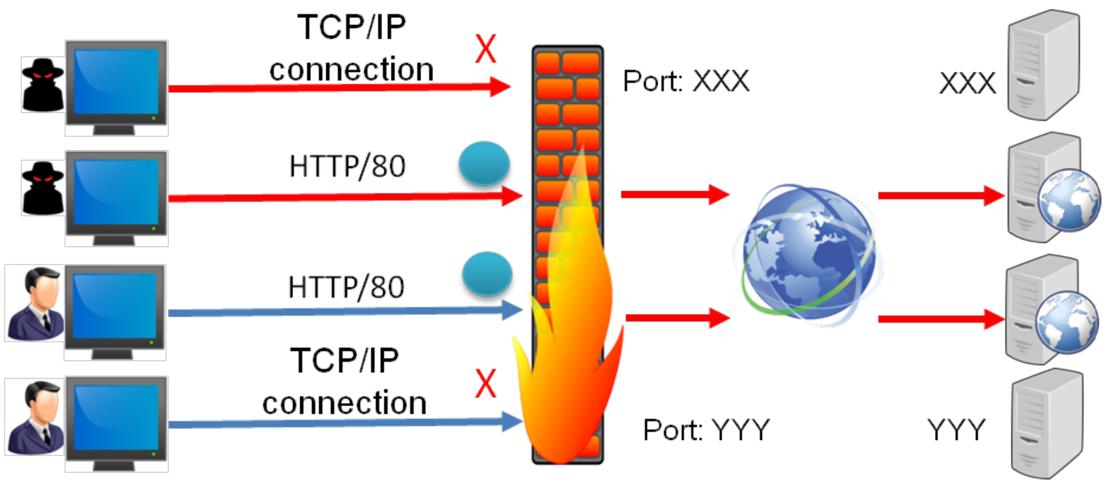


FIGURE 1.2: HTTP is always allowed in network because of its flexibility and commonality.

threats to the cyber-infrastructure of an organization are constantly evolving and it is clear that organizations spend a majority of their security countermeasure in protecting malicious attacks from external. However, one of the toughest and most insidious problems in information security, and indeed in security in general, is that of protecting against attacks from an insider [55].

One of the effective method on defend network from inside, system or network security administrators block all unnecessary outbound protocol or direct connections from TCP/IP and just permit outgoing communication only over selected protocols. In many decades, the flexibility and interoperability of HTTP make users progressively explore it in a much wider range of applications. It is recognized as the most widely used protocol on the Internet since users can find everything they need there when applications are being transferred more and more by developers onto the web. Therefore, HTTP is always allowed the network perimeter. In addition, in web environment, digital spies and thieves can cover their identities, conceal their physical locations, and create the malicious code from side to side. Consequently, cyber criminals or the Internet spiders have tendency to turn to fully exploit web technology and use it as a medium for communication to lurk malware (malicious software) or variety of illicit activities. In this case, as can be seen in Figure 1.2, malicious and other HTTP traffics will be passed through firewall because of its popularity.

Due to fast growing of malware and virus threats, destructive payloads can be inadvertently introduced to the network by employees since they look at video or file-sharing websites, playing games or using social media which can emblemed rootkit or unwanted software. When these unauthorized HTTP software infect into an user network, they will act like robot and mimic normal behavior web access and bypass network firewall or IDS. From there, they could turn to be a agent to receive command from outside and generate serious attacks from inside. Therefore, application layer attacks, including HTTP based attacks, still pose an ever serious threat to network security for years such as low-rate DoS attack in [3]. Application layer attack is difficult to be detected since it always comes after a technically legitimate connection has been established.

As usual, HTTP traffic can be classified into two categories: human traffic and non-human traffic or automated traffic. Human traffic is generated by users when they use normal web browsers (e.g. Internet Explorer...) to access websites which they want or need. On the contrary, non-human traffic comes from automated software (autoware).

HTTP is designed to defines only individual requests and responses proto-

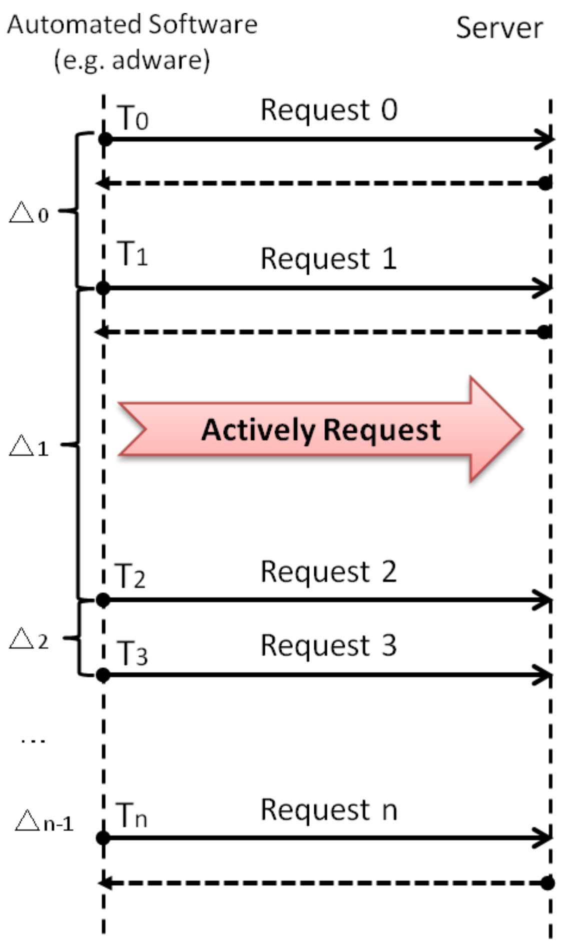


FIGURE 1.3: HTTP autoware always actively requests to their servers to perform the communication because of HTTP properties.

col and it does not perform interactive communication. If a HTTP application requires continuously and interactively asynchronous communication, the client necessary to continue sending the connection request to the server actively and automatically as illustrated in Figure 1.3. As a result, HTTP network communication is not just rigorously controlled by users intention and HTTP based automated software is blooming in utilizing in reaching Internet users. Automated traffic can be controlled and beneficial for user, however a lot of them are unknown and noticed with users, as described in Figure 1.1. Normal traffic, which are generated from normal autoware to benefit users usage purposes. This may include traffic for operating system update, virus definition updates. Gray traffic may generated from software which are not obviously malicious. However

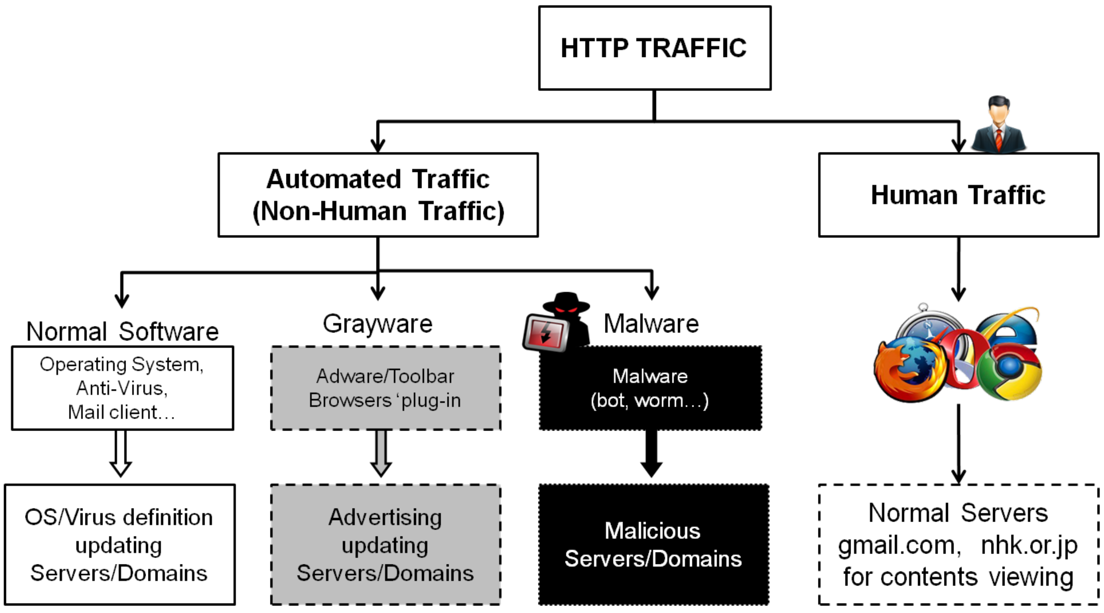

HTTP TRAFFIC
Automated Traffic (Non-Human Traffic)
Human Traffic
Normal Software
Operating System, Anti-Virus, Mail client...
Grayware
Adware/Toolbar Browsers 'plug-in
Malware
Malware (bot, worm...)
OS/Virus definition updating Servers/Domains
Advertising updating Servers/Domains
Malicious Servers/Domains
Normal Servers gmail.com, nhk.or.jp for contents viewing

FIGURE 1.4: HTTP traffic and automated software categories.

this kind of traffic is unknown and unnoticed to user, not all of them are bad for user but some of them can be annoying or even harmful to the user. In the dark side, suspicious traffic can generated to download unwanted contents to local such as from adware for advertising purpose or spyware for either collect information for marketing purposes or to deliver advertisements to web pages. Although this kind of software is legitimate, it can be installed on users computer without their knowledge or intention. Malicious traffic are generated from malicious autoware such as botnet. A good sample is Zeus and its variant. They are very dangerous malicious botnet [4, 5] on the Internet recent years and it is readily available for sale and also traded in underground forums [5]. Therefore, traffic in HTTP environment are merge of communication generated by all kind of normal, grey and malicious autoware. This raises the demand for analyzing HTTP autoware communication behavior to detect and classify malicious and normal activities via HTTP traffic.

One of traditional approach is to detect the application associated withe generated traffic such as Anti-Virus (AV) products. They are the most common content and signature based malware detection techniques. These types of AV software employ signature based detection to identify variants of known malware. As a consequence, the signature generation and update cycle cause an inherent delay in protecting users against new variants of malware [8]. Additionally, with the aim of limiting AV engines effectiveness, malware authors have developed increasingly sophisticated evasion techniques such as packing and polymorphism, aimed at circumventing detection by AV engines [6, 56]. Oberheide et al. [7] figure many undetected malware binaries by using signature-based techniques, and major AV engines just detect only 30% to 70% of recent malware. As the same content, Rajab et al. [8] show that less than 40% of malicious binaries can be detected by four AV engines in their experiment. Overcoming the issue of contents based detection studies, network traffic analysis and classification are suggested and approached. Hence, in this research, based on many studies and analysis of the autoware communication behavior, a new method to detect and classify HTTP autoware communication at host and network level are presented.

1.2 Traffic Classification Techniques

Review of traffic classification is presented in [9], and some of them are summarized in the following. The large body of literature about traffic classification [10–17, 19–28, 43] is a further evidence of the great interest of the research community towards this topic. In the first days of the Internet, identifying the application associated with some network packets was not an issue whatsoever: protocols were assigned to well-known transport layer ports by IANA [29]. Therefore, Port-based classification [22, 23, 25] simply extracted such value from the packet header and then look it up in the table containing the port-application associations. Unfortunately Port-based classification has become largely unreliable [23, 30]. In fact, in order to circumvent control by ISPs, modern applications, especially P2P ones, either use non-standard ports, or pick a random port at startup. Even worse, they hide themselves behind ports of other protocols

–this might enable bypassing firewalls as well. While port-based classification may still be reliable for some portion of the traffic [19], nevertheless it will raise undetectable false-positive (e.g., a non-legitimate application hiding beyond well-known port numbers) and falsenegative (e.g., a legitimate application running on non-standard ports) classifications.

To overcome this problem, Payload-based classifiers [14, 16, 20, 23, 27] were proposed. They inspect the content of packets well beyond the transport layer headers, looking for distinctive hints of an application protocol in packet payloads. We actually split this family of classification algorithms in two subcategories, Deep packet inspection (DPI) techniques that try to match a deterministic set of signatures or regular expressions against packet payload, and Stochastic packet inspection (SPI), rather looking at the statistical properties of packet content.

DPI has long provided extremely accurate results [23] and has been implemented in several commercial software products as well as in open source projects [31] and in the Linux kernel firewall implementation [32]. The payload of packets is searched for known patterns, keywords or regular expressions which are characteristic of a given protocol: the website of [32] contains a comprehensive lists of well known patterns. Additionally, DPI is often used in intrusion detection systems [25] as a preliminary step to the identification of network anomalies. Besides being extremely accurate, DPI has been proved to be effective from the very first payload packets of a session [33, 34], thus being particularly convenient for early classification.

Despite its numerous advantages, DPI has some significant drawbacks. First the computational cost is generally high, as several accesses to packet memory are needed and memory speed is long known to represent the bottleneck of modern architectures [35]. String and regular expression matching represent an additional cost as well: although there exist several efficient algorithms and data structures for both string matching and regular expression, hardware implementation (e.g. FPGA), ad hoc coprocessors (e.g. DFA) possibly massively parallel (e.g., GPU) are often required to keep up with current transmission speed [36]. These hardware-based approaches have been analyzed and used to improve the performance of machine learning algorithms, traffic classification approaches, and platforms for network security [37–42]. Yet, it is worth noting that while [42] estimate that the amount of GPUs power can process up to 40 Gbps worth of traffic, bottlenecks in the communication subsystem between the main CPU and the GPU crushes the actual performance down to a mere 5.2 Gbps [42]. Similarly, Network Processors [39] and [40] achieve 3.5 Gbps and 6 Gbps of aggregated traffic rate at most. As will see, statistical classification outperforms these classification rates without requiring special hardware. Another drawback of DPI is that keywords or patterns usually need to be derived manually by visual inspection of packets, implying a very cumbersome and error prone trial and error process. Last but not least, DPI fails by design in the case of encrypted or obfuscated traffic.

Stochastic packet inspection (SPI) tries to solve some of these issues, for instance by providing methods to automatically compute distinctive patterns for a given protocol. As an example, authors of [20] define Common Substring Graphs (CSG): an efficient data structure to identify a common string pattern in packets. Other works instead directly apply statistical tools to packet payload: authors of [16] directly use the values of the first payload bytes as features for machine learning algorithms; in [14], instead, a Pearson Chi-square test is used to study the randomness of the first payload bytes, to build a model of the syntax of the protocol spoken by the application. Additionally, this last algorithm is able to deal with protocols with partially encrypted payload, such as Skype or P2P-TV applications.

Authors of [43], instead, propose a fast algorithm to calculate the entropy of the first payload bytes, by means of which they are able to identify the type of content: low, medium and high values of the entropy respectively correspond to text, binary and encrypted content. Authors argue that, even if this is a very rough repartition of traffic and moreover some applications are very likely to use all of these kinds of content, nonetheless such information might reveal useful to prioritize some content over the others (e.g. in enterprise environments, binary transfers corresponding to application updates to fix bugs deserve an high priority). Yet, SPI is still greedy in terms of computational resources, requiring several accesses to packet payload, though with simpler operations (i.e., no pattern matching).

While both [14, 43] use entropy-based classification, a notable difference is represented by the fact that in [14] entropy is computed for chunks of data across a stream of packets, while [43] computes entropy over chunks within the same packet.

Statistical classification [11, 12, 21, 26, 44–48] is based on the rationale that, being the nature of the services extremely diverse (e.g., Web vs VoIP), so will be the corresponding traffic (e.g., short packets bursts of full-data packets vs long, steady throughput flows composed of small-packets). Such classifiers exploit several flow-level measurements, a.k.a. features, to characterize the traffic of the different applications [21,26,47]: a comprehensive list of a large number of possible traffic discriminators can be found in the technical report [49]. Finally, to perform the actual classification, statistical classifiers apply data mining techniques to these measurements, in particular machine learning algorithms.

Unlike payload-based techniques, these algorithms are usually very lightweight, as they do not access packet payload and can also leverage information from flow-level monitors such as [50]. Another important advantage is that they can be applied to encrypted traffic, as they simply do not care what the content of packets is. Nevertheless, these benefits are counterbalanced by a decrease in accuracy with respect to DPI techniques, which is why statistical-based algorithms have not evolved to commercial products yet. Still, researchers claim that in the near future operators will be willing to pay the cost of a few errors for a much lighter classification process.

This class of algorithms can further be divided in a few subclasses according to the data mining techniques employed and to the protocol layer of the features used. Concerning the first criterion, on one hand, unsupervised clustering of traffic flows [21] (e.g., by means of the K-means algorithm) does not require training and allows to group flows with similar features together, possibly identifying novel unexpected behaviors; on the other hand, supervised machine learning techniques [19,48] (e.g., based on Naive Bayes, C4.5 or Support Vector Machines) need to be trained with already classified flows, but are able to provide a precise labeling of traffic. Regarding the protocol layer, we have classifiers employing only flow-level features [47] (e.g., duration, total number of bytes transferred, average packet-size), as opposed to algorithms using packet-level features [11, 12] (e.g., size and direction of the very first packets of a flow). The former ones are usually capable of late (in some cases only post-mortem), coarse-grained classification, whereas the latter ones can achieve early, fine-grained classification.

Finally, Behavioral classification [17, 28, 51] moves the point of observation further up in the network stack, and looks at the whole traffic received by a host, or an (IP:port) endpoint, in the network. By the sole examination of the generated traffic patterns (e.g., how many hosts are contacted, with which transport layer protocol, on how many different ports) behavioral classifiers try to identify the application running on the target host. The idea is that different applications generate different patterns: for instance, a P2P host will contact many different peers typically using a single port for each host, whereas a Web server will be contacted by different clients with multiple parallel connections.

Some works [17, 28] characterize the pattern of traffic at different levels of detail (e.g., social, functional and application) and employ heuristics (such as the number of distinct ports contacted, or transport-layer protocols used) to recognize the class of the application running on a host (e.g., P2P vs HTTP). Works taking the behavioral approach to its extreme analyze the graph of connections between endpoints [52, 53], showing that P2P and client-server application generate extremely different connection patterns and graphs. They prove also that such information can be leveraged to classify the traffic of these classes of services even in the network core. A second group of studies [10,15], instead, propose some clever metrics tailored for a specific target traffic, with the purpose of capturing the most relevant properties of network applications. Combining these metrics with the discriminative power of machine learning algorithms yields extremely promising results. The Abacus classifier [10] belongs to this last family of algorithms, and it is the first algorithm able to provide a fine-grained classification of P2P applications.

Behavioral classifiers have the same advantages of statistical-based classifiers, being lightweight and avoiding access to packet payload, but are usually able to achieve the same accuracy with even less information. Such properties make them the perfect candidate for the most constrained settings. Moreover given the current tendency toward flow-level monitors such as NetFlow [50], the possibility to operate on the sole basis of behavioral characteristics is a very desirable property for classifiers. Based on these review in [9] and the target of research, behavior approaches are focused. In which, automated traffic of multiple autoware are analysis from there features or parameters, which presented their behavior, are extracted to classify and detect automated traffic.

1.3 Research Objectives

HTTP traffic are simpled categories in Figure 1.4. In that, HTTP traffic can be classified into two categories: human traffic and non-human traffic or automated traffic. Human traffic is generated by users when they use normal web browsers (e.g. Internet Explorer...) to access websites which they want or need. The characteristic of type of traffic is the accessed websites are known for users and they know what kind of data/information they will get. On the contrary, non-human traffic come from autoware can be classified into three types: normal software such as anti-virus updater, mail client, browser's toolbar; grayware encompasses adware, spyware, joke programs; malware acting as HTTP based bot. This kind of autoware accesses to unknowing servers without user intention. The distinction of normal and malicious activity from HTTP traffic is really serious challenge when sophisticated malware generate HTTP traffic requests similar with normal HTTP requests. When these malicious autoware infect into a user network, they will act like robot and mimic normal behavior web access and bypass network firewall or IDS. Furthermore, in a large private network, detection and also classification between types of HTTP autoware traffic are really great challenge when huge requests are generated each day.

Studies about malware detection over network data include multiple approaches. However, it continues to be a serious challenge since operational methods of malware are constantly changing. Overcoming the issue of contents based detection studies, many studies suggest to use network traffic analysis approaches. Botnets detection methods are presented in [57–63]. In host-based approach, Lee et al. [58] introduced a parameter based on one of the pre-defined characteristics of HTTP-based botnet. They suggested a Degree of Periodic Repeatability (DPR) to show the pattern of regular connections (i.e. pull style) of HTTP-based botnet to certain servers. Meisam Eslahi at el. [59] proposed an approach to reduce the false alarm HTTP botnet detection, in this research, high access rate traffic, which might be other security threats, is filtered out. Lee et al. [58] and Meisam Eslahi et al. [59] did not give the details of formula or method to calculate and measure each parameter.

Wei Lu et al. in [60], using signature-based techniques, propose a hierarchical framework to automatically discover malicious bot on a large-scale Wi-Fi ISP network, in which the network traffic is classified into different application communities by using payload-signature. These signatures were used to separate known traffic from unknown traffic in order to decrease the false alarm rates. Like other signature-based techniques the proposed classifier is less effective as it is unable to identify new or encrypted patterns [59].

Basil AsSadhan [61] at el. proposed a detection method in which concentrates in C&C communication analysis and find that it exhibits a periodic behavior. In [61], a method which applied discrete time series are analyzed to examine the aggregate traffic behavior in order to detect botnet C&C communication channels traffic. In CoCoSpot of Chiristian et al. [62], they proposed a clustering method to analyze relationships between botnet C&C flows and an approach to recognize botnet command and control channels solely based on traffic analysis features. To achieve this, the authors collected different parameters of the network traffic and consider to response message length from the server. However, in many C&C servers, the length of each response message is not stable or even there is no response in each request. Thus, the detection result might be reduce in that cases. Sung Jin Kim et al. [63] proposed HTTP activity set (HAS) analyzer in detection HTTP C&C. In that, the discriminative features set (low density and high content variability) for distinguishing C&C flows based on HTTP activity set are found and evaluated. In that, user-agent is one of items for HAS. However, attacker tool can modified user-agent with phishing the legitimate content of begin browsers or software. Therefore, the overall accuracy of the method might be degraded. These researches [57–63] focus on botnet communication to C&C server, but actually HTTP threats not just come from malicious bots but also can be from other types of automated software such as HTTP spyware, adware or unauthorized applications.

Seungwon Shin et al. in [64] proposed a framework to detect bot malware at host and network level. At host level, they monitor human-process interactions by using hook technique to capture user mouse and keyboard activities. These hook actions might affect to users PC systems. At network level, a simple way to prevent a malware infected PC sending out the information is to prevent all the direct TCP/IP connection from clients. However allowing HTTP protocol is really leaking hole which might be exploited by HTTP malware. In [64], to over come this issue, they monitored DNS queries to determine C&C server, but actually, many botnets use hacked domain names and its resource as C&C server. Therefore, the detection method might be insufficient.

Some of approaches the use lexical features or keywords extracted from URL and web contents as in [65–67, 81]. However, many other type of malicious web pages which are disguised by domain names or URLs like normal website and can harm users PC systems. In this case, lexical or keywords features might be compromised. Bartlett el at [2] proposed an approach to identify low-rate periodic network traffic and changes in regular communication of autoware. Their research also focus on many types of autoware and monitor TCP flows to detect, but, in this research, the target not just focus only to detect general types of autoware but also on particular URLs where autoware request to. In addition, the proposed methods just collect and process related HTTP traffic at application layer, this will help reduce expensive in process compare with method use TCP packets process.

In this research, in order to clarify automated traffic in HTTP environment and with the efforts using minor and basic features to solve the problem, many studies in analysis HTTP autoware Internet access behavior has been done. From there, characters and properties are extracted from HTTP requests properties such as number or timing of requests.... HTTP automated communication classification and detection method in host and network based are proposed. In general, flows of method is summarized in Figure 1.5. In that, HTTP requests are captured and processed through multiple layers and classified and detected in to many types of traffic such as normal, suspicious, or malicious.

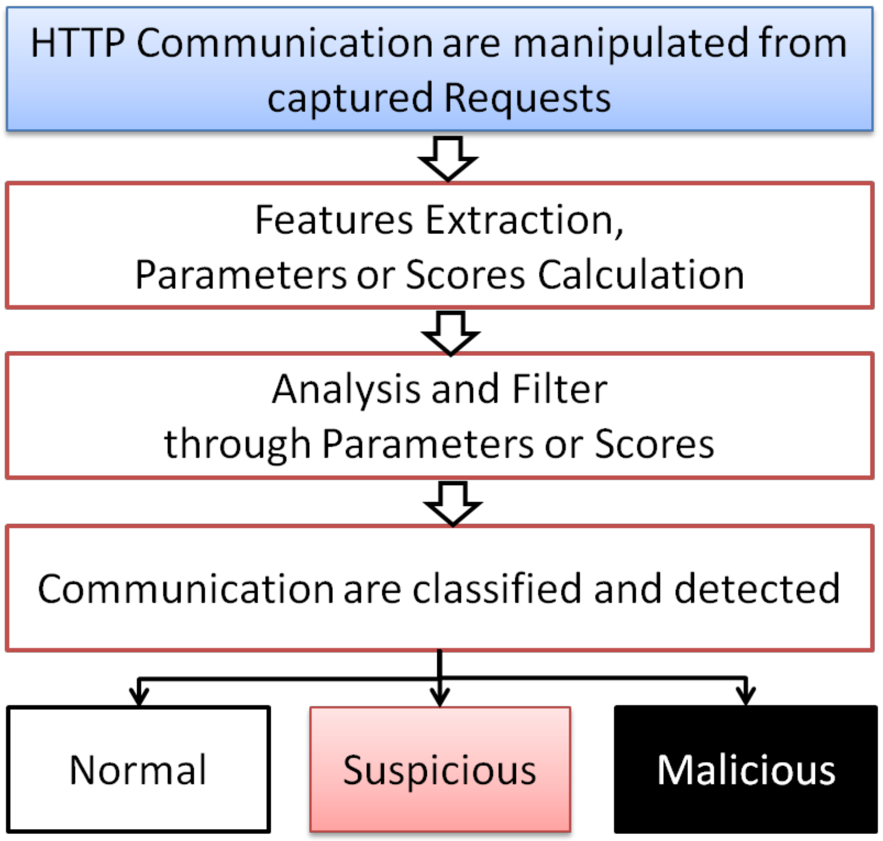

HTTP Communication are manipulated from captured Requests
Features Extraction, Parameters or Scores Calculation
Analysis and Filter through Parameters or Scores
Communication are classified and detected
Normal
Suspicious
Malicious

FIGURE 1.5: General flows of proposed methods in this research.

1.4 Outlines

The dissertation is organized into 5 chapters. chapter 2 to chapter 3 will present about methods in detection and classification of autoware communication at host and network level. All experimental data for each method is real data which are captured in clients or network. The details of each chapter are given below.

Chapter 2 shows a host-based method in detection of suspicious HTTP autoware. The algorithm for this is based on observation of many autoware communication.

Chapter 3, a network level method in clustering and identifying of HTTP automated communication. Chapter 4 presents the application model of two proposed methods and especially a big data based is presented and evaluated with real environment.

Chapter 5 summarizes the results of this work, and provides ideas for future research.

Chapter 2

Host-based Suspicious HTTP Autoware Detection

Malicious autoware can penetrate into users computer in dozen of ways such as plugged into free software, drive-by download attack, or spam link. Consequently, users might not control what or how many auto-wares are installed in their computers. Despite the fact that all types of auto-ware periodically automatically communicate with their servers to keep up the maintenance as analysis in previous chapter, there still the difference in the way communication between normal HTTP-based software and suspicious HTTP based software in some characteristics that are indicated as *periodic access* and *access rate*. In this chapter, by observing the variation of above two features of HTTP based application communication, a simple and new host-based suspicious autoware detection method is proposed. Through which, two parameters, **Autoware Score** and **Suspicious Score** are defined. The method is examined on the real traffic and produces a good result in detection. By in-host type method, it is helpful for users and also administrators in control the autoware in their computers and networks. The method is design suitable for not huge data process, single PC with limitation of memory and resource.

2.1 Methodology

In this section, HTTP based autoware communication characteristics have been reviewed and the detail of method is also interpreted. No public data set is available for use in suspicious detection experiments as opposed to what is available for, e.g., virus and intrusion detection. Therefore, for the purpose of reviewing the different of normal and suspicious auto-ware based on *periodic access* and *access rate* characteristics, four of HTTP based autoware are selected, they are included:

- Two normal toolbars (Toolbar 1, 2): these toolbars are browser's software help people access to their favorite services.
- A cloud service backup tool (Cloud Drive Sync Tool): These kinds of tools will sync users’ files from clients to cloud and vice versa. It helps users access the files or documents from anywhere based on their using cloud service, such as Dropbox or iCloud.
- A Malicious Zeus botnet with its C&C server: Zeus is HTTP-based bot. It has reportedly infected over 3.6 million computers in the United States. Zeus can be updated and directed by the bot master through C&C channel [4, 5].

This set of software is installed in a computer and their requests to server are captured and observed in two days. The data collection information is summarized in Table 2.1. In there, the communicated requests to server activities of malicious Zeus botnet are not as much as normal autoware.

Table 2.1: Data Collection Information for Host-based Method

HTTP-based Autoware	**Number of GET requests**	**Data collection length (2014)**
Toolbar 1	1351	From 21:35:20 Sep 28 To,21:28:06 Sep 30 About 47 hours
Toolbar 2	1205	
Cloud Drive Sync Tool	4079	
Zeus Bot	220	

2.1.1 HTTP based Autoware Communication Characteristics

In this section, by using real above data, two *periodic access* and *access rate* characteristics are compared between normal and malicious autoware.

- **Periodic access:** HTTP-based malicious software, such as botnet, which follow the PULL style where they periodically steadily connect to their server (i.e. command and control server) by GET requests with an interval in order to get the commands and updates [57, 59, 73, 74].

  The observation data for this feature are shown in Figure 2.1. In that X axis is the number of requests and Y axis represents the time different in second of two requests side by side. The graphs in Figure 2.1 illustrates the GET requests interval of selected auto-ware in attempt to keep communication to their servers.

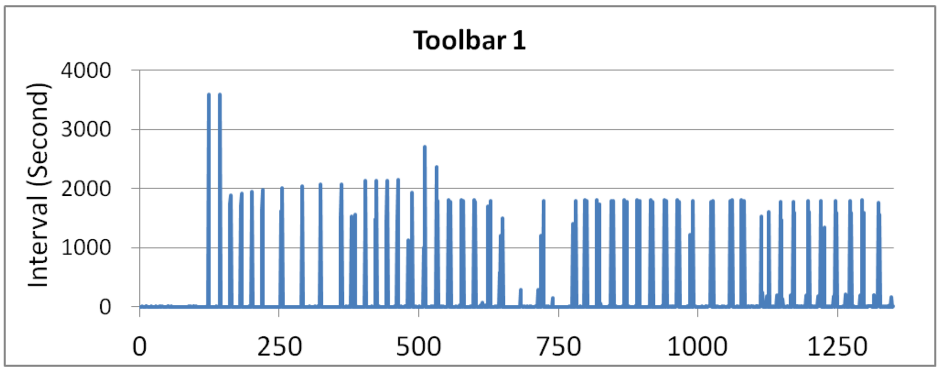
  Toolbar 1 and Toolbar 2's graphs (respectively in Figure 2.1(a), Figure 2.1(b)) are looked similar with each other. They have many long inactive durations which shows as high peaks in diagrams. The Cloud Drive Sync Tool's graph (in Figure 2.1(c)) express high intensity activities by big number of requests. The frequency of requests are also high and interval fluctuates around 50 seconds. Like Toolbar 1 and Toolbar 2, Cloud Drive Sync Tool shows the long inactive durations. In summary, these three normal auto-ware illustrates that they do not keep communication to their servers with steady periodic or interval of time. In contrary, the diagram of malicious Zeus bot (in Figure 2.1(d)) shows that it owns the most steady interval with only some peaks and the frequency of requests is at normal level with about 800 seconds between two requests. The requests are equally distributed during the running time.

- **Access rate:** Normal automatic software (e.g. updater and downloader) transmits a similar periodic pattern of traffic that has been generated within a short period of time. A suspicious software does not generate bulk data transfer [59, 75].

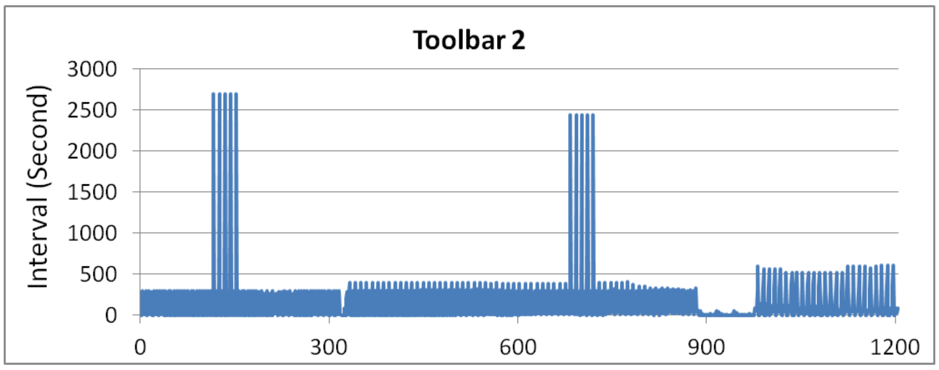
  The observation data of the feature are shown in Figure 2.2. In that X axis is the index of 1 hour time slot (about 47 slots, see Table I), Y axis represents the number of GET requests in a slot of time. The graphs in Fig. 4 illustrates the fluctuation of number of GET requests in each time slot for each selected auto-ware in attempt to keep communication to their servers.

  Once again, Toolbar 1 and Toolbar 2's graphs (respectively in Figure 2.2(a), Figure 2.2(b)) are looked similar with each other with the high fluctuation in the graphs. They send many requests in a short of time which shown as peaks. Normal fluctuation in diagram is represented in The Cloud Drive Sync Tool's graph (Figure 2.2(c)), however the number of requests in each slot always keep at high level around 80 requests. Can be conducted that three normal auto-ware keep the high variation access rate in communication with their servers. In contrast that, the graph of Zeus bot, in Figure 2.2(d), illuminates that in Zeus bot' s requests, there is almost no variation in requests number in each time segment just around 4-5 requests and active time is equally distribution in running time.

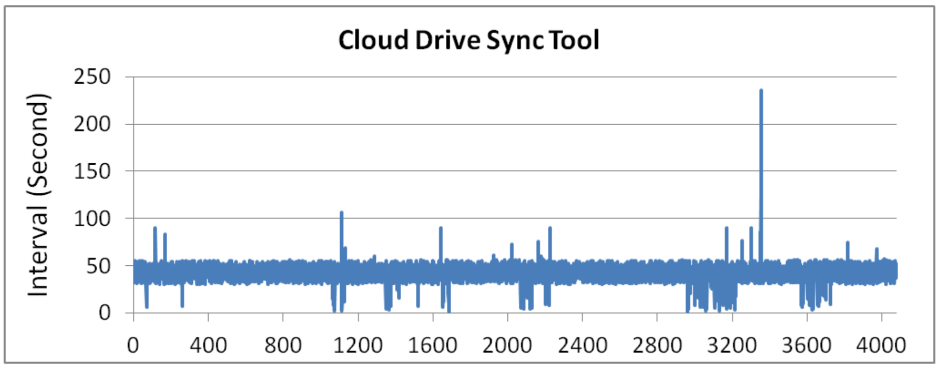
  In both observations of features, the negligible variation of suspicious auto-ware characteristics can be seen. If the total data collection is divided into N equivalent segments from t_1 to t_N with time duration $\triangle$, the distribu-



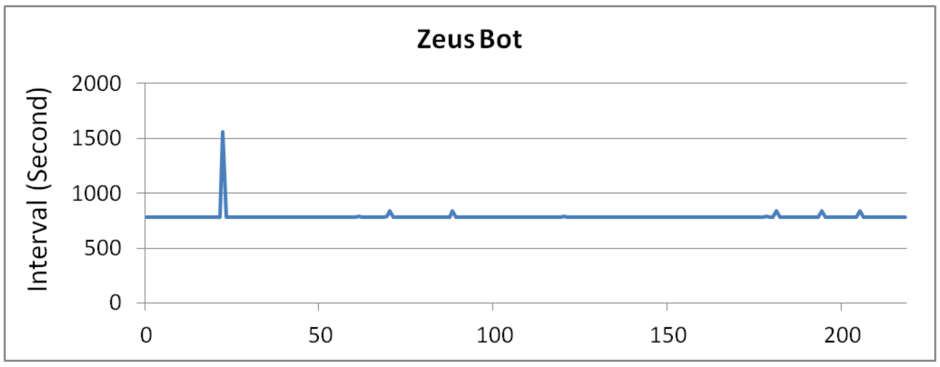
(a) Toolbar 1's periodic access data observation



(b) Toolbar 2's periodic access data observation

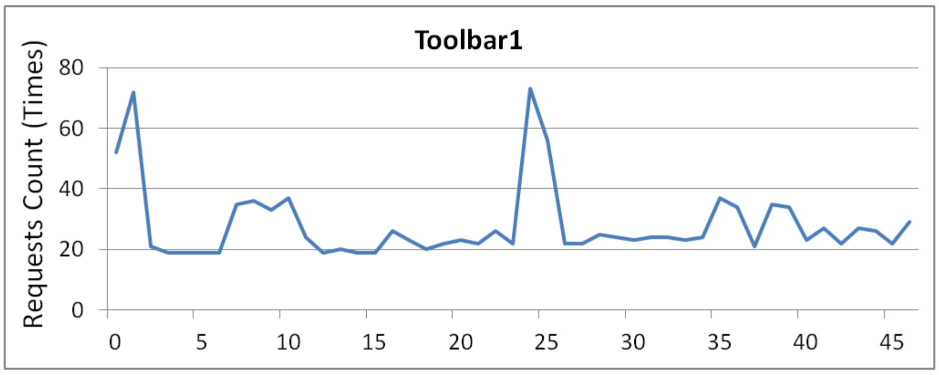


(c) Cloud Drive Sync Tool's periodic access data observation

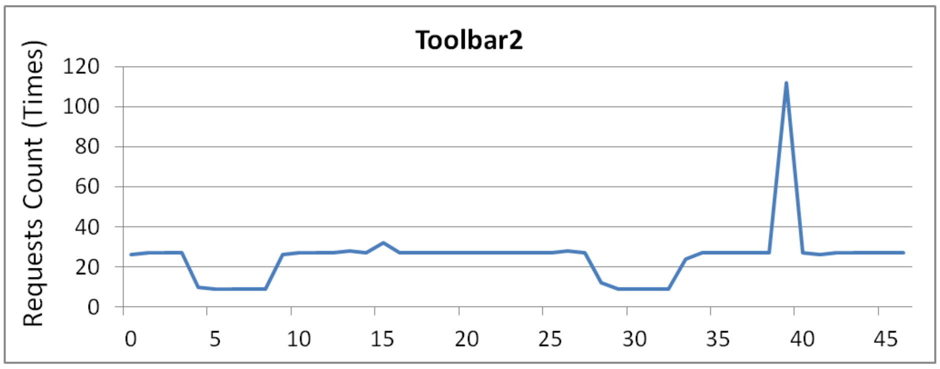


(d) Zeus bot's periodic access data observation

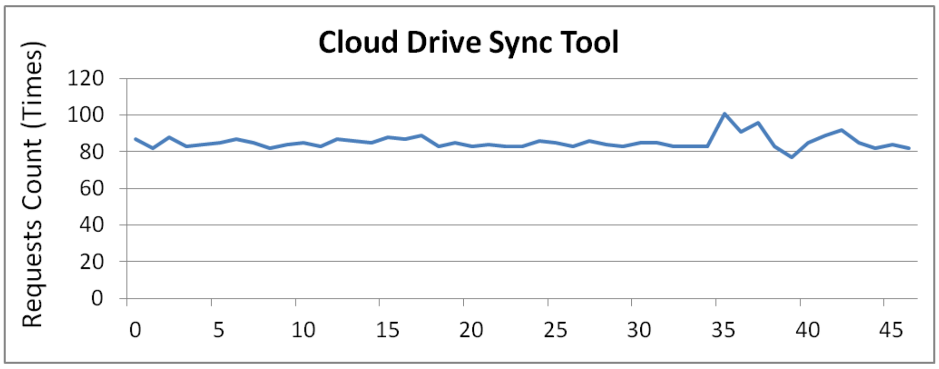
FIGURE 2.1: GET request time interval sequence graph of each observed application. X axis is the number of requests. Y axis represents the time(Interval) different in second of two requests side by side.



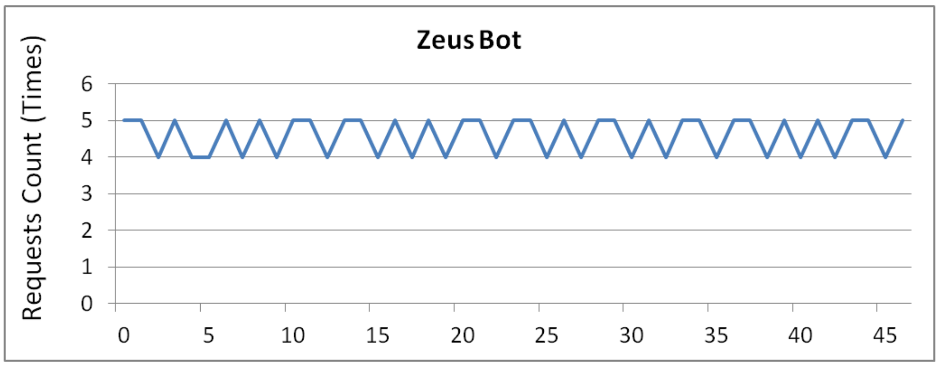
(a) Toolbar 1's access rate data observation



(b) Toolbar 2's access rate data observation



(c) Cloud Drive Sync Tool's access rate data observation



(d) Zeus bot's access rate data observation

FIGURE 2.2: The dispersion of GET requests number of each observed application. X axis is the index of each access time segment which is divided by △ (1 hour) , Y axis represents the number of GET requests (Requests Count) in a △ of time.

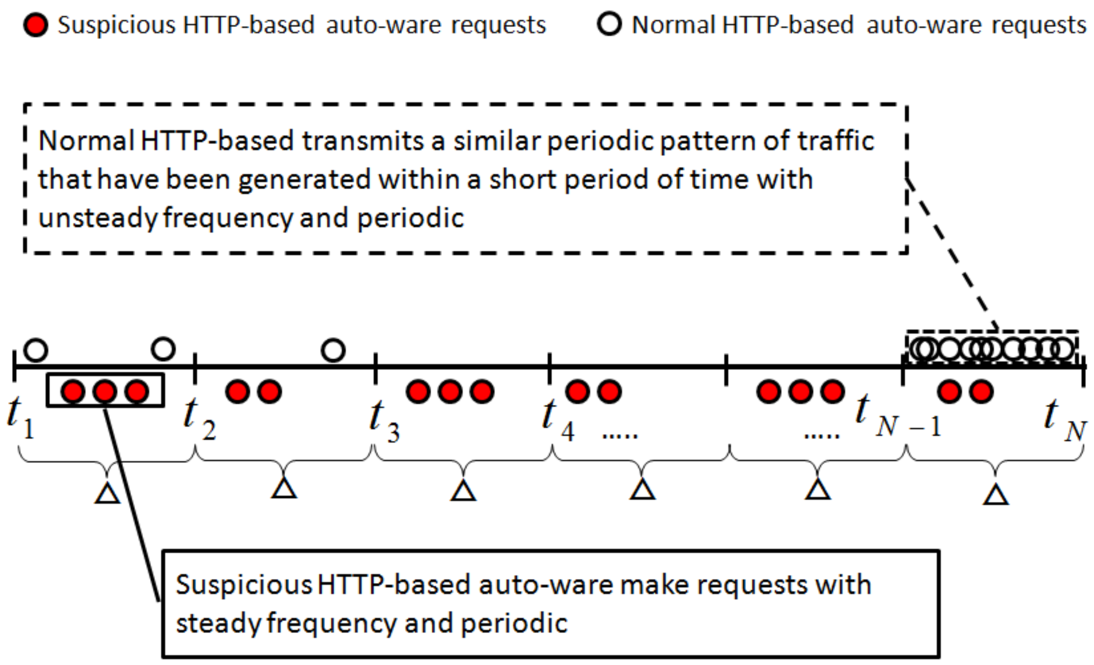

● Suspicious HTTP-based auto-ware requests
○ Normal HTTP-based auto-ware requests
Normal HTTP-based transmits a similar periodic pattern of traffic that have been generated within a short period of time with unsteady frequency and periodic
t_1
t_2
t_3
t_4
.....
.....
t_{N-1}
t_N
Suspicious HTTP-based auto-ware make requests with steady frequency and periodic

FIGURE 2.3: The distribution of HTTP-based normal and suspicious autoware.

tion of normal and suspicious auto-ware activities through two properties mentioned above can be illustrated as in Figure 2.3. Based on this observation, a method to detect suspicious HTTP-based auto-ware is proposed and expressed in next section.

2.1.2 Proposed method in suspicious HTTP based auto-ware detection

The method is proposed based on examining the variation of two features which are reviewed in section 2.1.1. Correspondingly, two parameters are introduced, $AutowareScore$ and $SuspiciousScore$, they are respectively shows the variation periodic access and access rate requests. Standard deviation is used usefully in measuring the amount of variation or dispersion from the average of sequences. For that meaning, the calculation approach of each parameter is using standard deviation σ formula [76]. σ of a vector $X = (x_1, x_2, ...x_n)$ is illustrated as in Equation (2.1.1):

$$\sigma(X) = \sqrt{\frac{\sum_{j=1}^{n}(\bar{x} - x_j)}{n-1}} \tag{2.1.1}$$

General imagination of method is show in Figure 2.4. In the remainder of the section, all parts of flow will be gone throw. For that purpose, assume that the total data collection is divided into N equivalent segments from t_1 to t_N with time duration $\triangle$.

Preprocessing step helps to reduce or filter out unnecessary information. There is many ways to do this, for example, if the number of URIs connects to a server is too small in a long period of time, that server is not a candidate for a suspicious communication (e.g. entire data collecting period) because botnet/-malicious software are designed to perform bigger tasks and much faster than humans, hence they do not generate brief traffic [59, 75]. Another simple way is using a good white list.

AutowareScore is determined by observing the periodic access of client to server with some following steps, a calculation sample of *AutowareScore* for a server S_i is illustrated in Figure 2.5.

- Access time to server S_i is divided into segments. Each time segment $\triangle_j$ from t_j to t_{j+1}, $(j = 1, 2, ...N - 1)$, k_j is defined by a number of requests in that segment, and $X_j = (x_{j1}, x_{j2}, ...x_{jn})$ is a vector which presents the periodic access graph between this time segment $\triangle_j$, the $\delta(t_j)$ is calculated as in Equation (2.1.2).

$$\delta(t_j) = \begin{cases} \sigma(X_j) & (k_j > 0) \\ \delta_{BIG} & (k_j = 0) \end{cases} \tag{2.1.2}$$

In the case of $k_j > 0$, $\delta(t_j)$ is standard deviation of vector X_j by using formula in Equation (2.1.1). The constant δ_{BIG} is a big number.

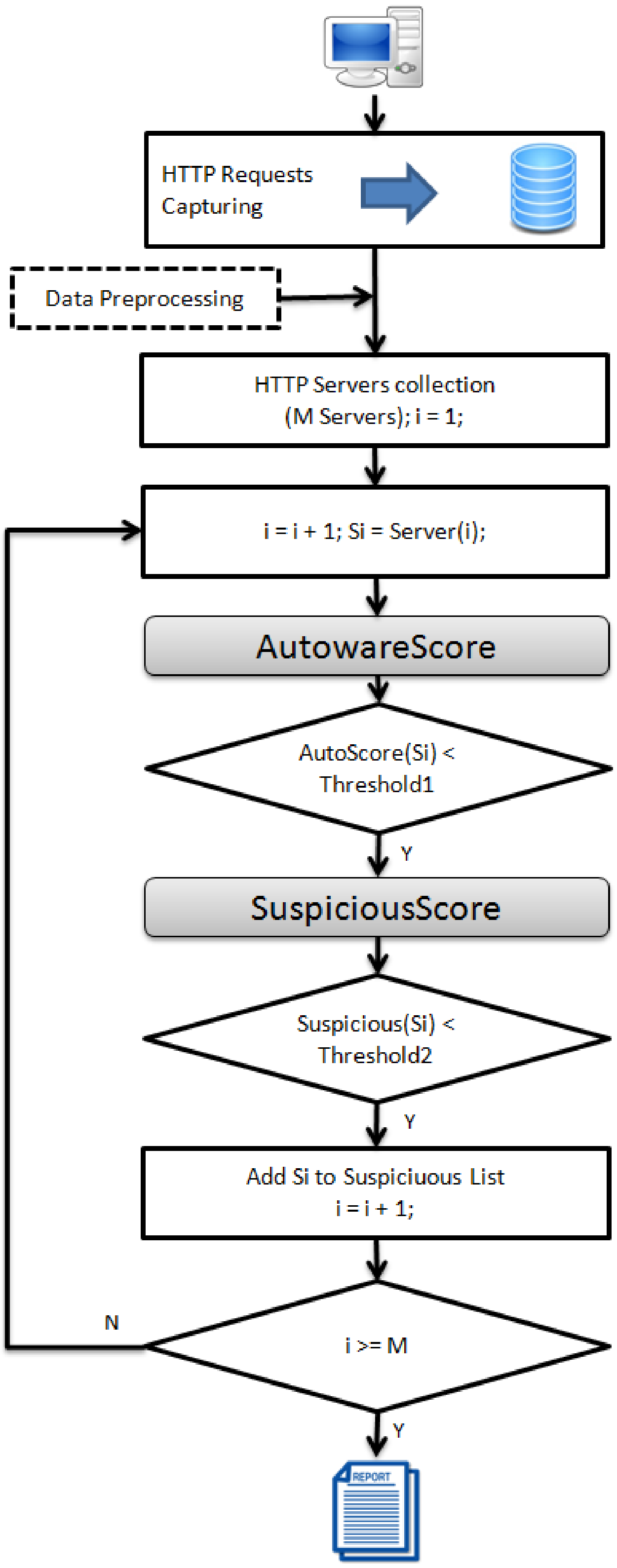
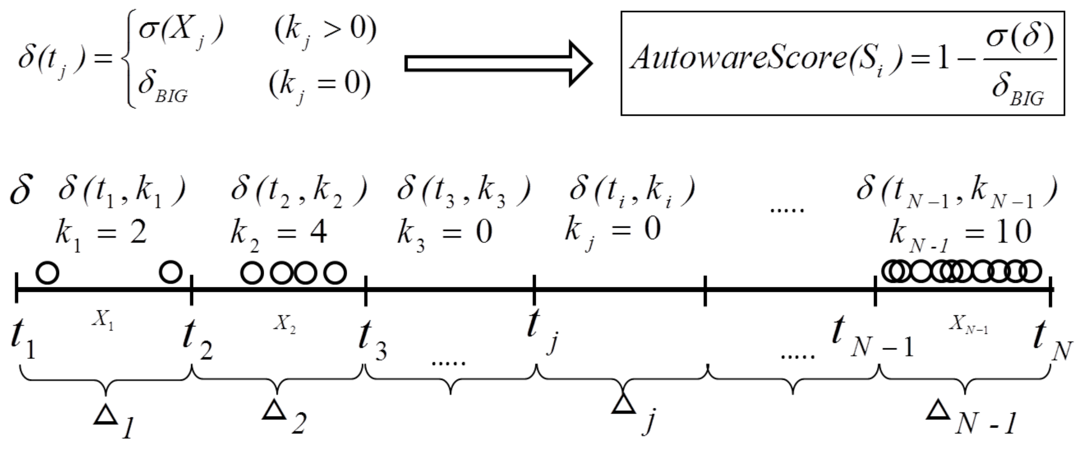

HTTP Requests Capturing
Data Preprocessing
HTTP Servers collection (M Servers); i = 1;
i = i + 1; Si = Server(i);
AutowareScore
AutoScore(Si) < Threshold1
Y
SuspiciousScore
Suspicious(Si) < Threshold2
Y
Add Si to Suspiciuous List i = i + 1;
N
i >= M
Y
REPORT

FIGURE 2.4: Overall flow of host-based proposed method.



$\delta(t_j) = \begin{cases} \sigma(X_j) & (k_j > 0) \\ \delta_{BIG} & (k_j = 0) \end{cases}$
$AutowareScore(S_i) = 1 - \frac{\sigma(\delta)}{\delta_{BIG}}$
δ $\delta(t_1, k_1)$ $\delta(t_2, k_2)$ $\delta(t_3, k_3)$ $\delta(t_i, k_i)$ ..... $\delta(t_{N-1}, k_{N-1})$
$k_1 = 2$ $k_2 = 4$ $k_3 = 0$ $k_j = 0$ $k_{N-1} = 10$
X_1 X_2 X_{N-1}
t_1 t_2 t_3 ..... t_j ..... t_{N-1} t_N
Δ_1 Δ_2 Δ_j Δ_{N-1}

FIGURE 2.5: Description of *AutowareScore* calculation for a server S_i.

- A vector $\delta = (\delta(t_1), \delta(t_2), ...\delta(t_{N-1}))$ is accomplished after last step, $\sigma(\delta)$ is calculated based on Equation (2.1.1) and reform as in Equation (2.1.3) below:

$$\sigma(\delta) = \sqrt{\frac{\sum_{j=1}^{N-1}(\bar{\delta(t)} - \delta(t_j))}{N-2}} \tag{2.1.3}$$

- *AutowareScore* of server S_i is determined as in Equation (2.1.4), one more time Equation (2.1.1) is used.

$$AutowareScore(S_i) = 1 - \frac{\sigma(\delta)}{\sigma_{BIG}} \tag{2.1.4}$$

σ_{BIG} is chosen big enough which $\sigma_{BIG} \geq Max(\delta(\sigma))$ for standardization $0 \leq AutowareScore(S_i) \leq 1$.

SuspiciousScore is determined by observing the access rate of client to server. As is defined above, $k = (k_1, k_2, ...k_{N-1})$ is a vector which presents the number of requests in each time segment. The of *SuspiciousScore* server S_i is standard deviation of k which is formulated as in Equation (2.1.5).

$$SuspiciousScore(S_i) = \sigma(k) \tag{2.1.5}$$

$Threshold1$ and $Threshold2$ are can be set as configurable parameters, depending on typical traffic on a network.

2.2 Experimental Results

The experiment has been done by using the data which are described in Table 2.1 of section 2.1.1 (PC1) and web access data of 3 other clients (PC2, PC3, PC4), the information of experimental data is represented in Table 2.2, and experimental environment is shown in Figure 2.6, these data is after data preprocessing step, as in Figure 2.4.

Exception the applications which are installed in PC1 is already known as can be seen in Table 2.1, all the information about the kind of applications using in PC2, PC3 and PC4 are unknown before applying the proposed method.

An host-based Windows application is developed to apply the proposed approach with above data. All the threshold values has been chosen as description in Figure 2.4 and section 2.1.2, the time duration $\triangle$ will be 1 hour, δ_{BIG} has been automatically chosen as 30 after calculate by using Equation (2.1.3).

$Threshold1$ and $Threshold2$ are can be set as configurable parameters, depending on typical traffic on a network. In this implementation, $Threshold1 = 0.9$ and $Threshold2 = 0.5$ are chosen for experimentation purpose.

The experimental results have been summarized in Table 2.3. For evaluation the data of PC2, PC3, PC4 the application servers (column servers in Table 2.3) are referenced with users of these PCs.

The results show that all autoware are well detected by the parameter $AutowareScore$ of method combined with $Threshold1$, and Zeus bot is detected

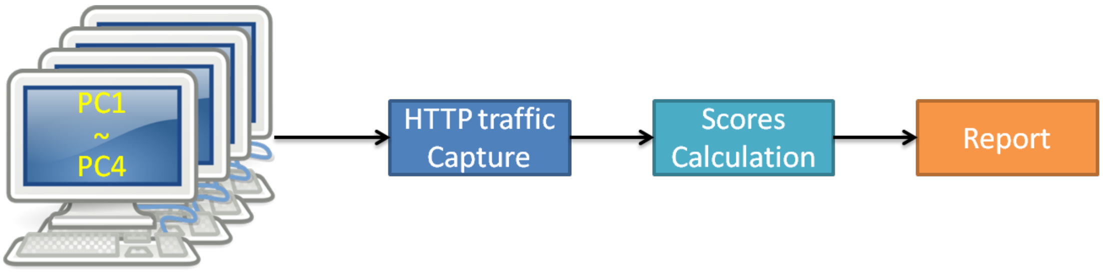


FIGURE 2.6: Experimental model and environment.

TABLE 2.2: Experimental Data Collection Information

PCs	Number requests	Data Length	Descriptions
PC1	60,439	2 days	Zeus Bot, 2 toolbars, 1 cloud drive sync tool are installed.
PC2	18,889	1 day	No information
PC3	10,094	1 day	No information
PC4	7,036	1 day	No information

TABLE 2.3: Experimental Results

PC	Servers	Autoware Score	Suspicious Score	Detection Results
PC1	Zeus Bot C & C server	0.994	0.49	Suspicious Auto-ware
	Toolbar1 server	0.989	3.84	Auto-ware
	Toolbar2 server	0.940	12.35	Auto-ware
	Cloud Drive Sync Tool server	0.997	14.89	Auto-ware
PC2	Anti-virus software Server	0.947	13.52	Auto-ware
PC3	Cloud Drive Sync Tool server	0.965	23.9	Auto-ware
	Anti-virus software Server	0.967	176.18	Auto-ware
PC4	All servers	$\leq$0.79	$\leq$35.03	No Auto-ware

as a suspicious autoware by *SuspiciousScore* in combined with *Threshold*2. Comparing to the methods are presented in [57, 58], the proposed method do not try to detect the periodic or interval of bot traffic which may face with the false alarm of other autoware traffic. Results in [57, 58] will be more effective if the proposed method is applied as a preprocessing before method of [57, 58].

2.3 Conclusions

This chapter has proposed an approach to detect suspicious HTTP based autoware merely through observing communication pattern features of natural HTTP traffic at host level. Compare to traditional using signature method, this method can detect new type of suspicious which behaving periodic communication to their server without contents investigation. The proposed methods are evaluated based on the detection of real data and its efficiency in the detection of a HTTP based botnet and other normal autoware. Proposed method also will be needed improvement for detection in various working models of autoware such as in network level. Furthermore, the thresholds, which are used in this method, depend on the network traffic condition. A method to calculate adaptive thresholds need to be considered for the future work.

Chapter 3

Clustering and Identifying HTTP Automated Communication

In this chapter, a network level method in classification of HTTP autoware is introduced. The method is based on the study of autoware Internet access behavior. In order to analysis HTTP autoware communication behavior, access graph is extracted from request based features. Accordingly, HTTP automated traffic are clustered into groups based on their behavior. All traffic will be identified as normal, suspicious or malicious. The method is tested with real HTTP traffic data collected through a proxy server of a private network.

3.1 Introduction

In a private network, due to security threats, all direct Transmission Control Protocol/Internet Protocol (TCP/IP) outbound/inbound connections should be banned. However, HTTP is an exception since, nowadays, application development is transferred more and more onto the web, and everything users need can be found through web services. Therefore, in some ways, if HTTP autoware can infect a user’ s PC or network, it might still transparently communicate with

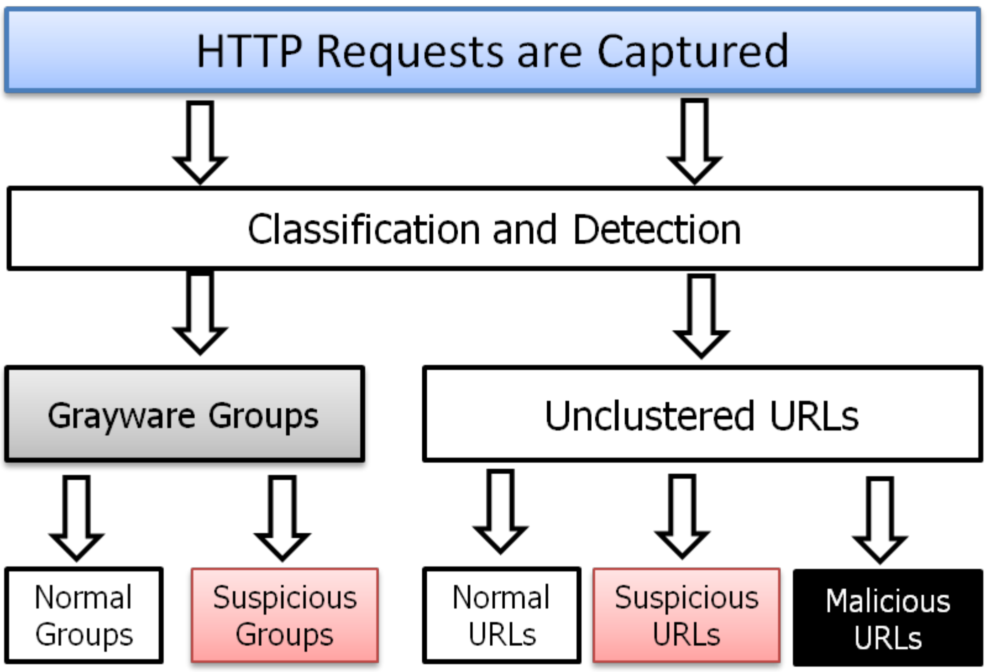


FIGURE 3.1: Main flows in proposed method in network level

its servers/sites. Furthermore, in a large private network, classification between types of HTTP autoware traffic is becoming more difficult when huge numbers of requests are generated each day.

To maintain communication, perform updates, or receive commands, all kinds of HTTP-based autoware have common characteristics in that they repetitively generate legal traffic and requests to their servers/domains. However, in details, there are some sophisticated differences in the communicating behavior of autoware with their sites. In this chapter, a method in clustering and identifying type of communication is proposed. Accordingly, HTTP automated traffic are not only classified but are also identified into three kinds of traffic (normal, suspicious and malicious) based on HTTP autoware access behavior as summarized in Figure 3.1

All output results are manually checked with the support of VirusTotal online system [84] and McAfee Web Gateway [85]. In this scope of research, some definition and evaluated steps are summarized as bellow:

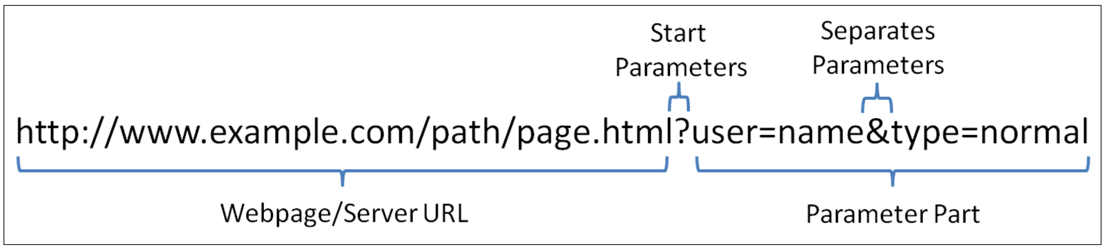


FIGURE 3.2: Main parts of URL.

- Grayware, which has not been detected as malicious in MacAfee and Virustotal, are unknown and unnoticed traffic with users.
- Grayware might include normal and suspicious traffic. In that, suspicious traffic are unrecommended or advertisement in MacAfee and VirusTotal.
- Other traffic will be checked manually by content investigation.

3.2 Methodology

3.2.1 Feature Extraction

HTTP traffic from a client consisted of many requests from that client to outside. At application layer, a request includes basic information: IP address of client, full URL, and request method. Full URL's parts contain webpage/server URL and parameter path, as shown in Figure 3.2. At network level, numerous features are extracted which are made from basic client requests information as follows

- Client IP: Source IP address of machine in network which generated requests
- Request Method: main methods of HTTP requests, POST/GET.
- Request date time: Date and time when a client sends request.
- Webpage/Server URL (shorten as URL): URL requested by a Client IP but without parameters part, as shown in Figure 3.2. Some normal web servers

are hacked and some of its resource paths are exploited as C&C servers. Additionally, parameter part are easily changed based on the specification of requests' content, but actually the functionality of that webpage/server URL, such as C&C server or advertise content update, are the same in each request. Therefore, non-parameter URL is used instead of domain or full URL to add more detail and accuracy in classification of autoware access behavior.

- Unique URL: Set of unique URLs requested by a Client.
- Request Interval: Break time between two consecutive requests to the same URLs.
- Request Count: Number of requests to a URL from a client in a period of observation data.
- URL Access Time: A period of time in second which a client accessed to a URL from the first request to the end request
- Client Access Time: a period of time in seconds which a client accessed to the Internet by HTTP protocol. It is determined by checking the difference in second between the first and the end request
- Access Graph: A graph is based on requests' intervals. It presents the access behavior of a client to a URL. Details are described in next slide

3.2.2 Access Graph

Access graph presents communication behavior of a client to a specific URL in a duration of time. It is formed on request interval which is extracted from HTTP traffic. Assuming that $R = \{r_1, r_2, ..., r_N\}$ is set of requests from a client to a webpage/server and all r_i have the same webpage/server URL, as described in Figure 3.2, then access graph G is a sequence included N-1 items, $G = \{g_1, g_2, ..., g_{N-1}\}$

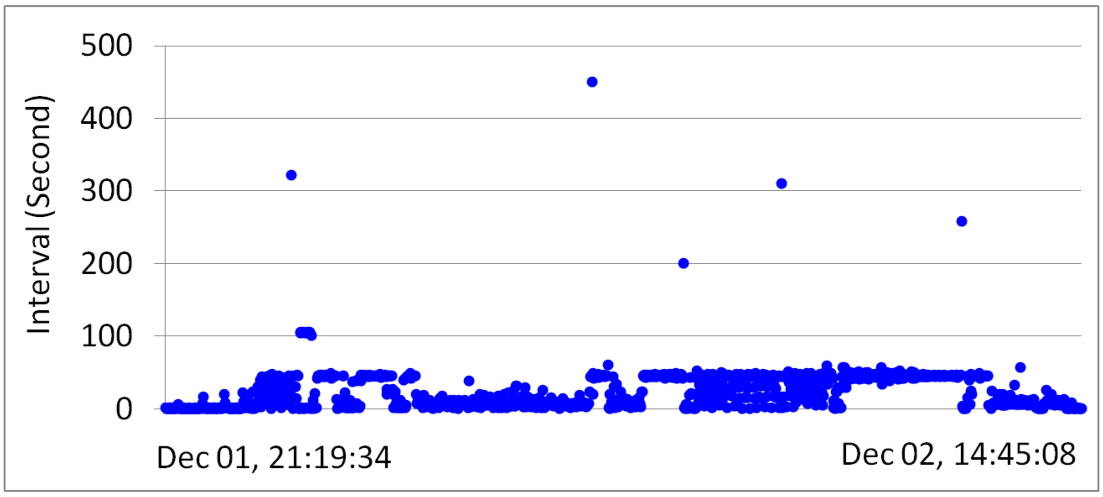


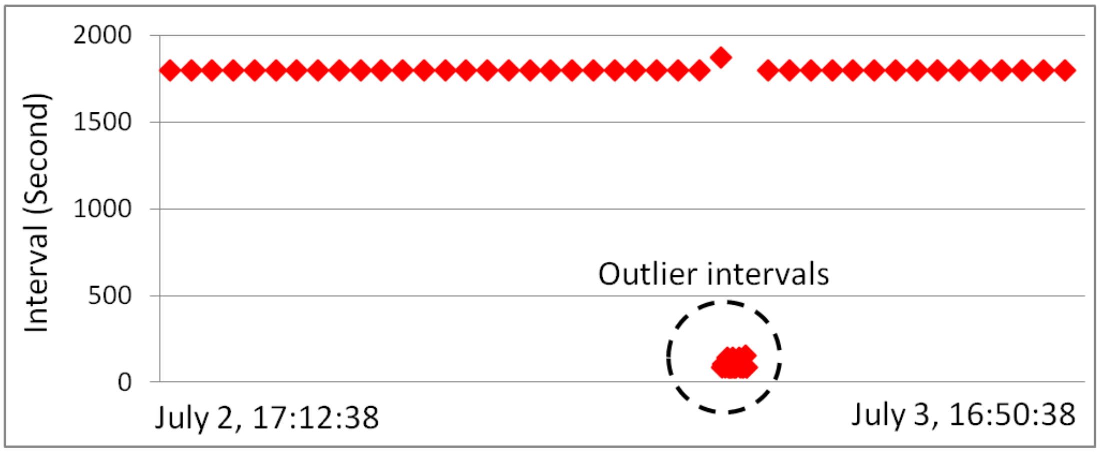
FIGURE 3.3: An access graph of a client request to a URL.

which is g_i is a pair of (t_i, d_i) which t_i is timing of request r_{i+1} and d_i is request interval between r_i and r_{i+1}. An access graph is shown as in Figure 3.3, in which, X axis is timing of request (except the first request), Y axis shows the request interval value in second. An installed autoware in a client will behave in different access graph to each URL which it requested to. For that, this graph can present the behavior in communication between an autoware to its webpage or server URLs.

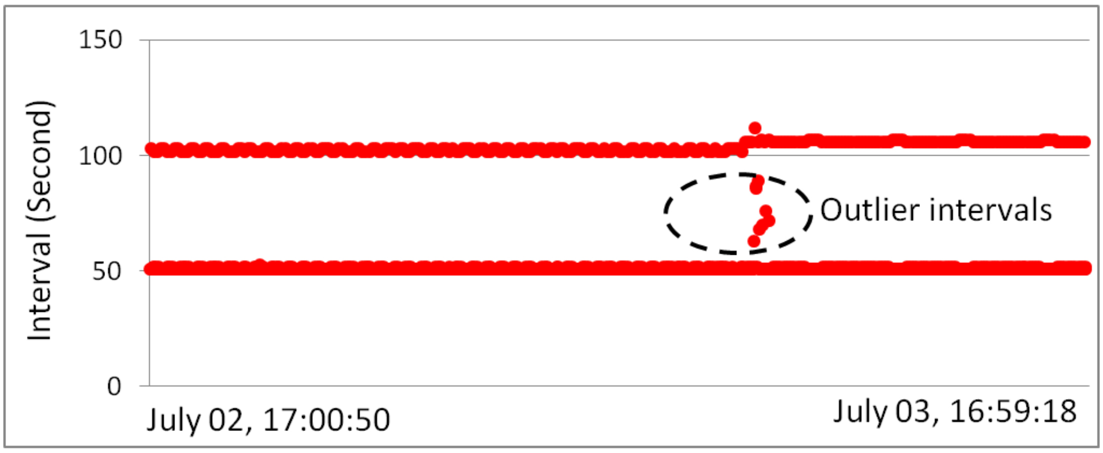
3.2.3 HTTP Access Behavior Analysis

In order to maintain the communication with its servers and because HTTP is a protocol which just defines individual requests and responses, and it does not perform interactive communication, HTTP autoware have common characteristics since they follow pull style and actively requests to their servers. However, in details, each type of them have sophisticated differences in their communication behavior.

Malicious HTTP-based bots always connect to their command and control server periodically in order to get the commands and updates. The number of requests from malicious bots are not high as that from normal autoware (e.g.



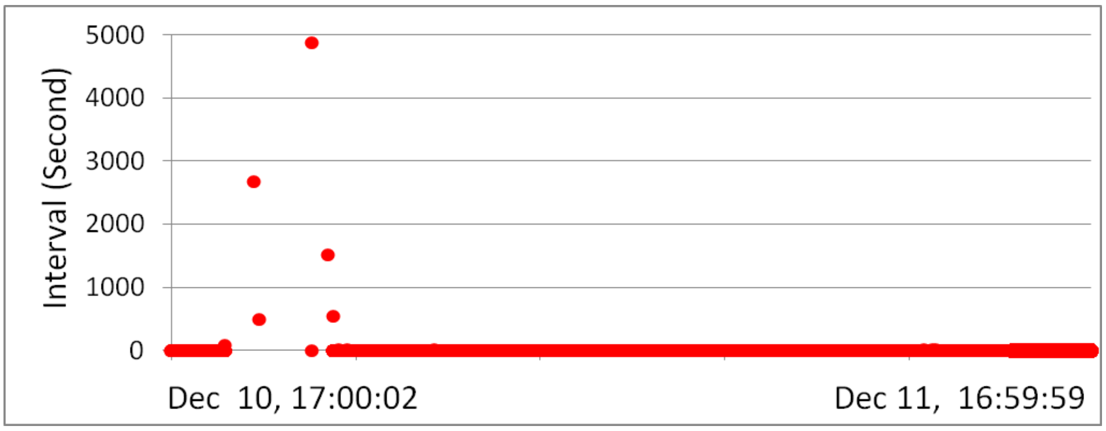
(a) Access graph from a malicious bot to a C&C server



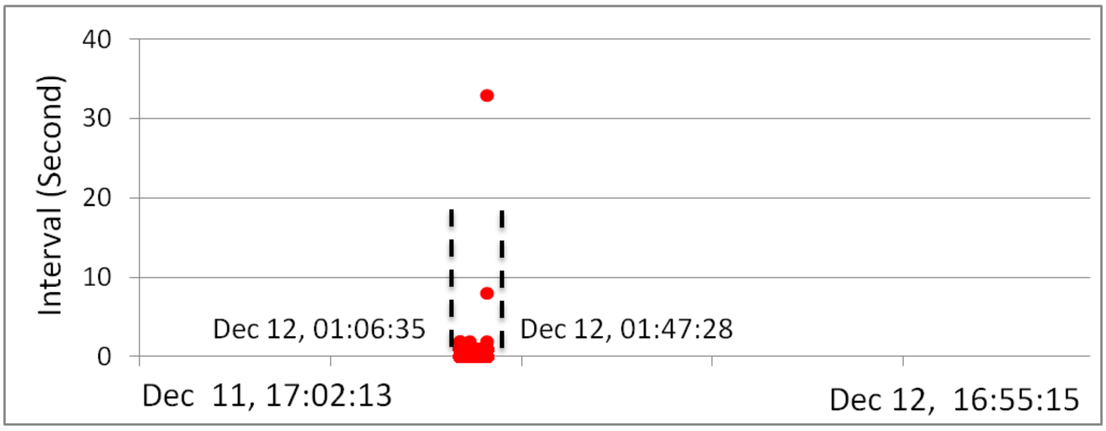
(b) Access graph from a malicious bot to a C&C server

FIGURE 3.4: Almost no variation in access graphs of two malicious bots to their C&C servers.

updater and downloader) which just generate requests with a long interval than unusual malicious bots [61]. Because intervals in communication between a malicious bot to their C&C server is stable, there is almost no variation in their access graph, Figure 3.4 shows the access graph of two malicious bots' communication. As can be seen there, even some outlier intervals are found but the main intervals of these two malicious bots are durable. Figure 3.4(a) shows access graph of a malicious bot which has a main interval is 1800 seconds, and the bot Figure 3.4(b) owns two main intervals of around 50 seconds and 100 seconds. Malicious access can be recognized by scoring its access speed . If they are requested with extremely high speed they will be detected as malicious. As in Figure 3.5(a).



(a) Access graph of URL which is accessed at high speed. It is accessed continuously in 23.9 hours from Dec 10, 17:00:02 to Dec 11, 16:59:59, which is the same with client's web access time. This constitutes 71,994 requests with speed at 0.82 times per second.



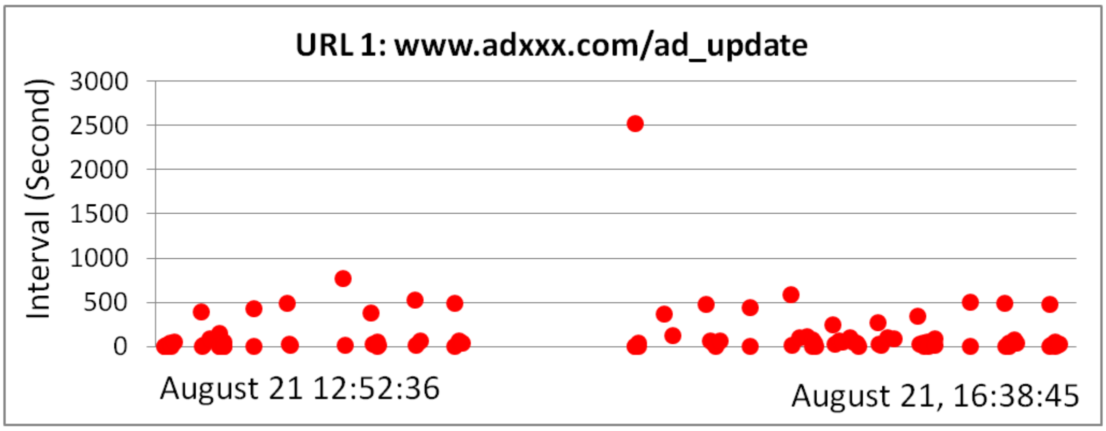
(b) Access graph of URL which is accessed at extremely high speed. This URL is accessed continuously in 0.68 hours from Dec 12, 01:06:35 to Dec 12, 01:47:28, which is much smaller than the client's web access time in 23.88 hours from Dec 11, 17:02:13 to Dec 12, 16:55:15. This constitutes 80,903 requests with speed at 32.98 times per second.

FIGURE 3.5: Access graph of 2 URLs which are accessed with very high speed.

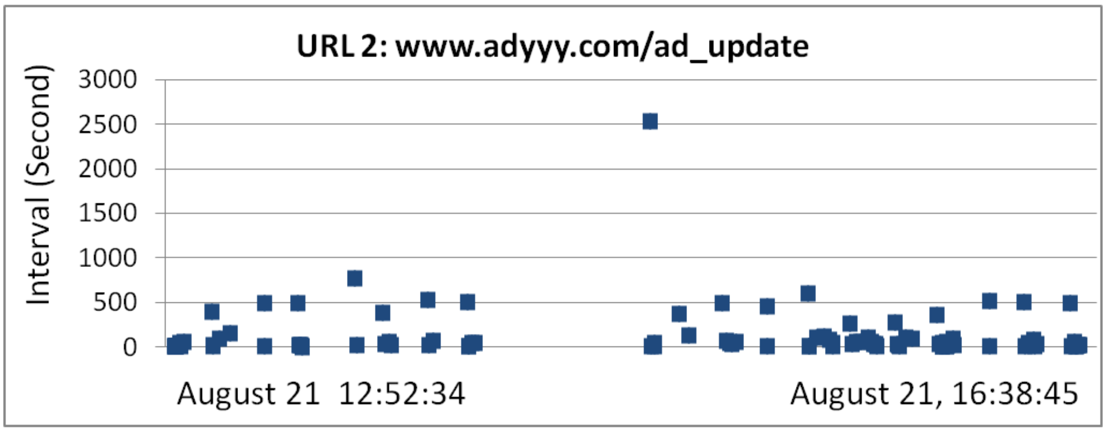
This graph shows that a malicious URL's access graph is requested 71,99 times in 23.9 hours (0.82 time per second).

Malicious bots often connect to one control domain and to a specific server resource during a given period of time [63]. Different from that, unwanted HTTP applications, or grayware, such as annoying adware or spyware, often report back to or request new information from many external resources [2]. Therefore, they keep communicating to their numerous advertising sites or URLs to update pop-up or advertisement and commercial content areas.

Autoware will behave with the same communication pattern to its Different



(a) Access graph from an adware to URL1



(b) Access graph from an adware to URL2

FIGURE 3.6: Access graphs from an adware of a client IP to two different URLs are similar

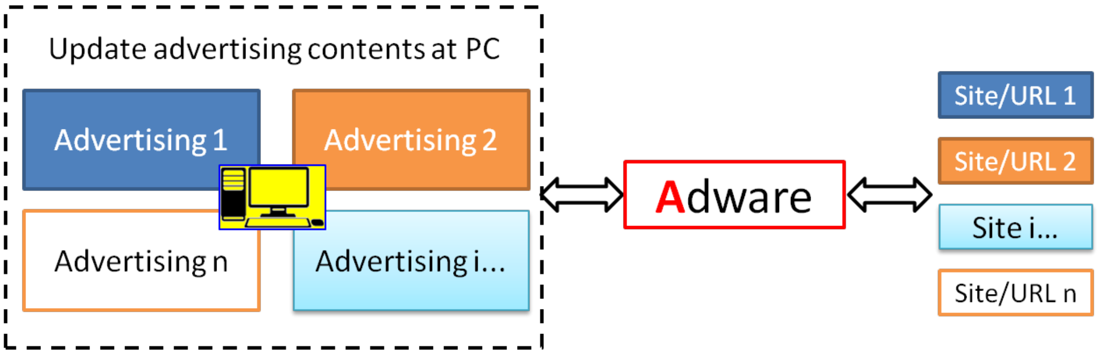


FIGURE 3.7: Autoware have trend access to many URLs at the same timing.

URLs as shown in Figure 3.7. If they are requested at the same or approximately equivalent timing, access graph of URLs from a specified autoware are presented similarly. Illustrated in Figure 3.6, a sample of two similar access graphs present

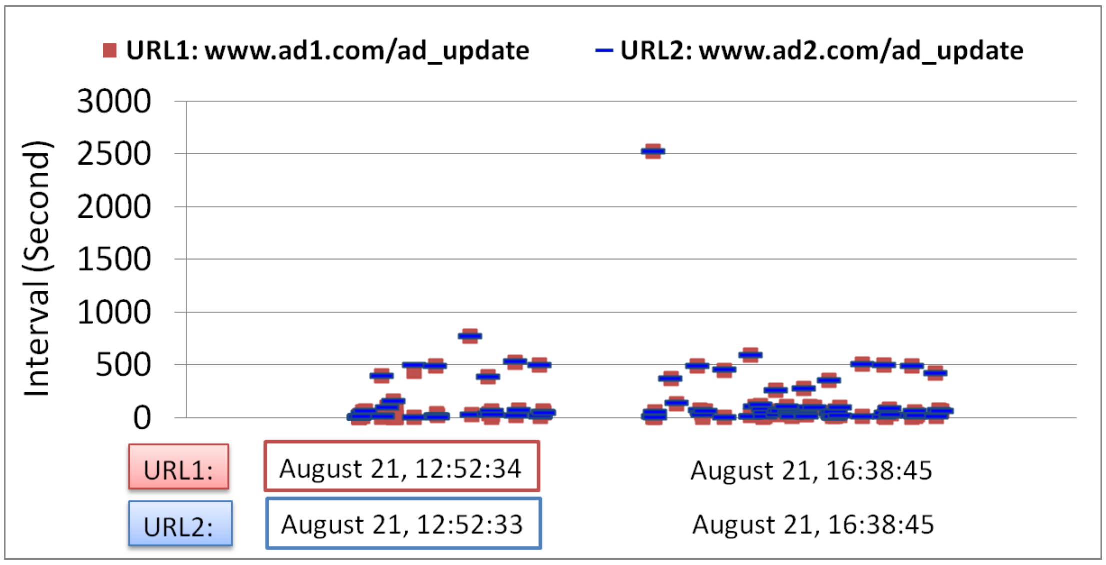


FIGURE 3.8: Access time to two URLs from an Autoware are nearly equivalent. The (URL_1 is requested from August 21, 12:52:34 (URL_2 is accessed from August 21, 12:52:33) to Dec 09, 16:38:45.

the communication from one adware to two different URLs. In order to measure the similarity between any access graphs, a graph distance is proposed in section 3.3. In addition, autoware requested to many URLs with the same timing, so the access duration to these URLs is approximately equal. It means that the first and the last request's timing to these URLs are almost the same with each other. As can be seen on Figure 3.8, beside of the similarity of access graph, the first and the last request's moment of them are nearly equivalent.

Since suspicious autoware - adware, for instance- update contents, like other autoware, they access to many URLs. However, they will collect data from many URLs of multiple sites which own various domain names. This is not alike with normal autoware, such as an electronic newspaper, because it self-refreshes the contents of presenting page by accessing to many URLs but with only one domain name. As can be seen in Table 3.1, a suspicious adware access to multiple URLs from 10 various domains, but news update requests also access to many URLs but just from one domain.

A suspicious autoware starts with the time of human computer start. There-

Table 3.1: Suspicious autoware have trend access to many URLs with difference domain

No	Description	Access Samples
1	Suspicious autoware ,ex. adware, access to many URLs from various domains	http://domain1/a/h/sUK48UTFUtM2xUGD6uq5_qdnVgRzoU_ZYbtGp4ZBCeA=
		http://domain2/x/brs1024
		http://domain3/a/h/H9HzuwJ17XDi4ZEalNIJgcQTzfLdWpgZJrXem
		http://domain3/a/h/VlXJI07mKkppFeZ7norZRqmL43NzxlE+LnJIHOG8Wy0=
		http://domain4/publisher/VASTGenerator.xml
		http://domain5/crossdomain.xml
		http://domain6/lr/bid.php
		http://domain7/crossdomain.xml
		http://domain8/crossdomain.xml
		http://domain8/6019&vid_id=YOUR_VIDEO_ID&vid_title=YOUR_VIDEO_TITLE
		http://domain9/server/bid/liverail.bid
		http://domain10/lg.php
		http://domain10/
2	Normal autoware ,ex. news update, accesses to many URLs but from the same domain	http://domain/_layouts/TNO.tn.MainPageSection/SdkIntegration.swf
		http://domain/_layouts/Images/img-tn/yahoo.gif
		http://domain/PublishingImages/img-tn/Null.gif;pv74463024071c66a5
		http://domain/PublishingImages/img-tn/banner.jpg
		http://domain/PublishingImages/img-tn/input-search.jpg
		http://domain/PublishingImages/img-tn/bg_tag.gif
		http://domain/PublishingImages/img-tn/arrow-breakcum.png
		http://domain/PublishingImages/img-tn/zingmeicon.gif;pvead4c6fe82dfdf0d
		http://domain/PublishingImages/img-tn/menu-top.gif

fore, it is expected that the access duration of a suspicious autoware might be similar with user computer interaction.

On contrary with autoware, there are no interval or periodic pattern in users' web access. However, in recent years, many sites (e.g. shopping online site or social media webpage) append advertisement path to their sites and use JavaScript or Flash as auto-aware part to automatically collect the advertising content. Therefore parts of users access sites can generate HTTP traffic which acts as autoware communication.

3.3 Access Graph Similarity

As analysis in section 3.2.3, autoware has trend to access to multiple URLs, so they will behave with similar access graph as can be seen in Figure 3.6. In order to find similar access behavior among URLs, an access graph similarity is considered to be used. There are many kind of distance or similarity measures are presented in [69] and [70] such as Minkowski family including Euclidean L_2, City block L_1 or Inner Product family including Cosine, Harmonic mean. Accordingly, the number of items in an element needs to be equal to be compared with each other. However, the number of requests from autoware to each URLs are not always the same with each other. Therefore, number of points in its graph are also not equal. In addition, autoware access graph to an URL might be difference each time, so can not use the normal distance such as Euclidean distance to score their similarity. To solve the problem, Modified Hausdorff distance is suggested, the distance is presented in [79] which based on the Hausdorff distance [80] and summarized in next sub section.

3.3.1 Modified Hausdorff Distance

This section presents about Modified Hausdorff distance which is extracted from [79, 80]. The Hausdorff distance measures the extent to which each point of a ”model” set lines near some point of an ”image” set vice versa. Given tow finite point sets $A = (a_1, a_2, ..., a_p)$ and $B = (b_1, b_2, ..., b_q)$, the the Hausdorff distance is defined as:

$$H(A, B) = max(h(A, B), h(B, A)) \tag{3.3.1}$$

where

$$h(A, B) = max_{a \in A} h(a, B) \tag{3.3.2}$$

and

$$h(a, B) = min_{b \in B} ||a - b|| \tag{3.3.3}$$

and $|| \cdot ||$ is some underlying norm on the points of A and B, it might be L_2 or Euclidean norm.

The function $h(A, B)$ is called the *directed* Hausdorff distance from A to B. It identifies the point $a \in A$. That is the furthest from any point of B and measures the distance from a to its nearest neighbor in B (using the given norm $|| \cdot ||$), that is, $h(A, B)$ in effect ranks each point of A based on its distance to the nearest point of B and then uses the largest ranked such point as the distance (the most mismatched point of A). Intuitively, if $h(A, B) = d$, then each point of A must be within distance d of some point of B, and there is also some point of A that is exactly distance d from the nearest point of B (the most mismatched point).

The Hausdorff distance $H(A, B)$ is the maximum of $h(A, B)$ and $h(B, A)$. Thus, it measures the degree of mismatch between two sets by measuring the distance of the point of A that is furthest from any point of B and vice versa. Intuitively, if the Hausdorff distance is d, then every point of A must be within a distance d of some point of B and vice versa. Thus, the notion of resemblance encoded by this distance is that each member of A be near some member of B and vice versa. Unlike most methods of comparing shapes, there is no explicit pairing of points of A with points of B (for example, many points of A might be close to the *same* point of B). The function $H(A, B)$, can be trivially computed in time $O(pq)$ for two point sets of size p and q, respectively.

The Modified Hausdorff distance (MHD) in [79] compare the many functions of $h(A, B)$ using with $H(A, B)$, in which $h(A, B)$ could be as bellow, where in that Equation 3.3.8 is similar with Equation 3.3.2.

$$h_1(A, B) = min_{a \in A} h(a, B) \tag{3.3.4}$$

$$h_2(A, B) =^{50} K^{th}_{a \in A} h(a, B) \tag{3.3.5}$$

$$h_3(A, B) =^{75} K^{th}_{a \in A} h(a, B) \tag{3.3.6}$$

$$h_4(A, B) =^{90} K^{th}_{a \in A} h(a, B) \tag{3.3.7}$$

$$h_5(A,B) = max_{a \in A} h(a,B) \tag{3.3.8}$$

$$h_6(A,B) = \frac{1}{N} \sum_{a \in A} h(a,B) \tag{3.3.9}$$

where $xK^{th}_{a \in A}$ represents the K^{th} ranked distance such that $K/p = x\%$. For example, ${}^{50}K^{th}_{a \in A}$ corresponds to the median of the distances $d(a,B), \forall a \in A$.

The directed distance $h(A,B)$ and $h(B,A)$ between two point sets A and B can be combined in the following four ways to define an undirected distance measures $H(A,B)$, where Equation 3.3.1 is similar with Equation 3.3.11.

$$H_1(A,B) = min(h(A,B), h(B,A)) \tag{3.3.10}$$

$$H_2(A,B) = max(h(A,B), h(B,A)) \tag{3.3.11}$$

$$H_3(A,B) = \frac{h(A,B) + h(B,A)}{2} \tag{3.3.12}$$

$$H_4(A,B) = \frac{p \times h(A,B) + q \times h(B,A)}{p+q} \tag{3.3.13}$$

where $h(A,B)$ is any 3.3.4 to 3.3.9 and summarized in Table 3.2, in that $h_5(A,B)$ and $H_2 A, B$ are similar3.3.1 and 3.3.2 respectively, so $D18$ is original definition of Hausdorff distance in [80].

After comparing the ways of usage which are summarized in Table 3.2 with experiment, [79] determined that, D_{22} is best for matching two objects based on their edge points. Therefore, the Equation 3.3.9 and 3.3.11 will be used to score the similarity between access graphs in this thesis.

TABLE 3.2: 24 distance measures between two point sets

Directed Distance	Function			
	H_1	H_2	H_3	H_4
h_1	D_1	D_2	D_3	D_4
h_2	D_5	D_6	D_7	D_8
h_3	D_9	D_{10}	D_{11}	D_{12}
h_4	D_{13}	D_{14}	D_{15}	D_{16}
h_5	D_{17}	D_{18}	D_{19}	D_{20}
h_6	D_{21}	D_{22}	D_{23}	D_{24}

3.3.2 Access Graph Similarity

In an assumption that there are two access graphs: $A = (a_1, a_2, ..., a_N)$ and $B = (b_1, b_2, ..., b_M)$. In order to score the similarity between any two access graphs A and B, a distance is proposed. It is based on the modified Hausdorff (MH) distance which is presented in previous section 3.3.1.

Firstly, the distance between two points a_i and b_j is defined, which is calculated as a Euclidean distance $d(a_i, b_j) = ||a_i - b_j||$ based on the Equation 3.3.3.

After that, based on the Equation 3.3.9 the distance between point a_i and graph B is defined as $d(a_i, B) = min_{b_j \in B}||a_i - b_j||$.

Finally, based on Equation (3.3.1), the modified Hausdorff (MH) distance [79] between graph A and B is

$$S(A, B) = max(d(A, B), d(B, A)) \tag{3.3.14}$$

The smaller the $S(A, B)$ between graphs A and B, the more graphs A and B are similar to each other. Two more type of distances which are based on $S(A, B)$ in Equation (3.3.14) are proposed. They are max and average distance between any access graph of a group URLs, and are defined as bellow:

$$MaxS(G) = max(S(U_i, U_j)_{\forall U_i, U_j \in G}) \tag{3.3.15}$$

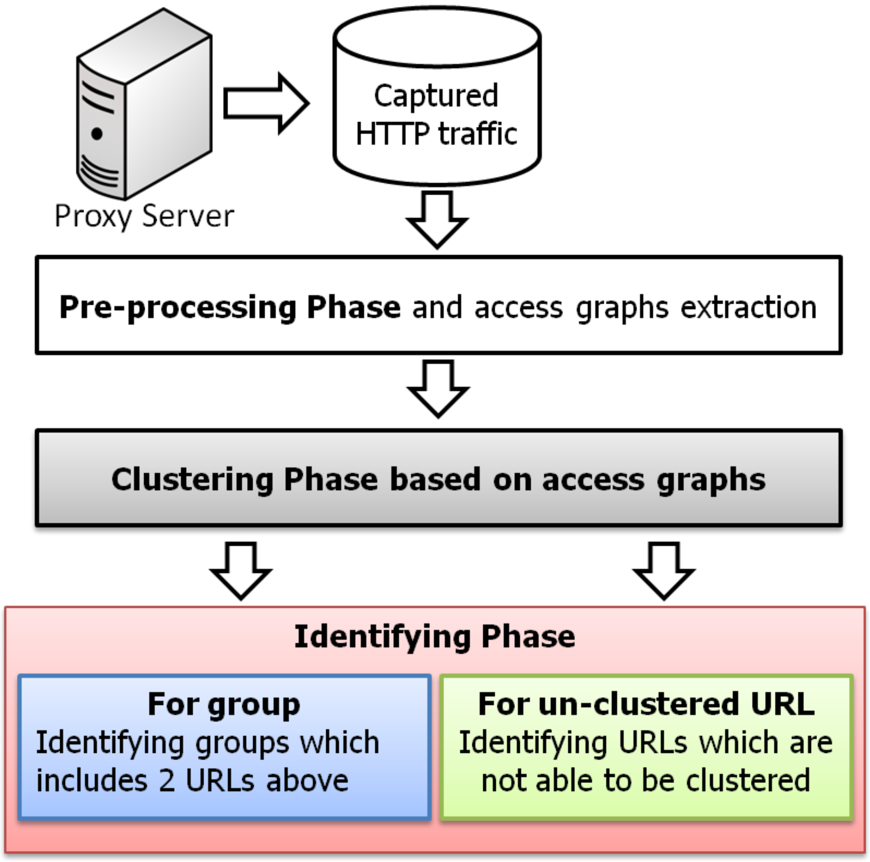

Proxy Server
Captured HTTP traffic
Pre-processing Phase and access graphs extraction
Clustering Phase based on access graphs
Identifying Phase
For group
Identifying groups which includes 2 URLs above
For un-clustered URL
Identifying URLs which are not able to be clustered

FIGURE 3.9: Main flows of network level proposed method.

$$AvgS(G) = average(S(U_i, U_j))_{\forall U_i, U_j \in G}) \quad (3.3.16)$$

In Equation (3.3.15) and (3.3.16), G is group of URLs, and U_i and U_j are any access graphs of URLs in G.

3.4 Proposed Method

Based on the autoware analyzed communication behavior and many suggested parameters, a clustering and identifying automated communication method in HTTP environment, including main three phrases, is proposed, as outline in Figure 3.9 and 3.10. Three phases are pre-processing, clustering and identifying phases. Accordingly, bellow steps will be processed

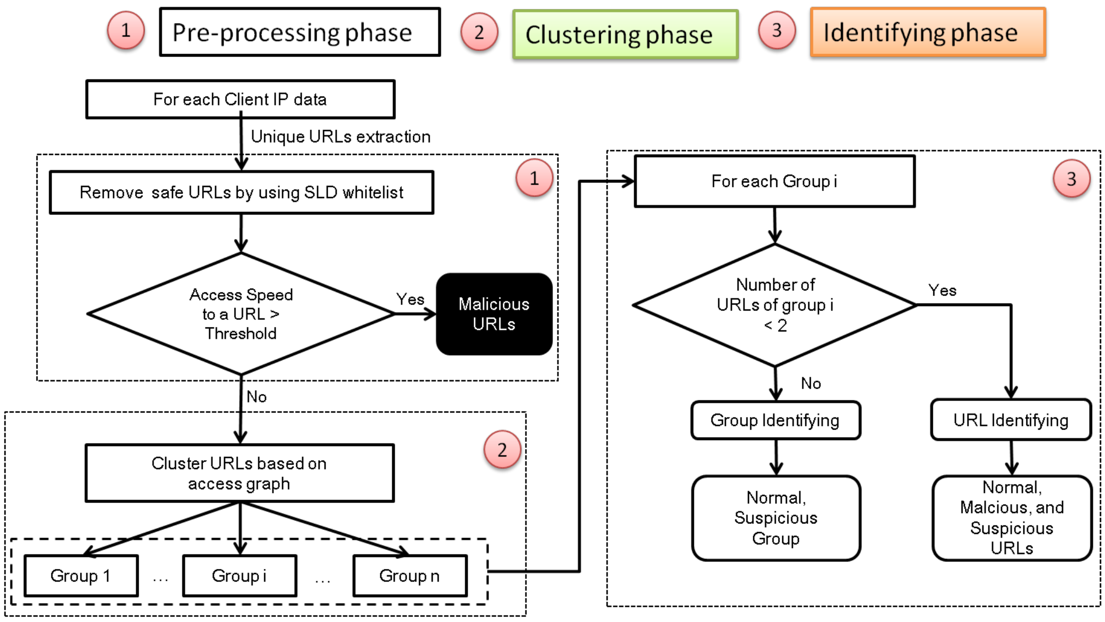


FIGURE 3.10: Phases of network level proposed method.

- The traffic will be captured from network, for example from a proxy.
- These traffic will be preprocessed in order to reduce the unnecessary processed data by checking the legitimate via a second level domain. Detail of preprocessing phase will be presented in next section 3.5.
- Next, remain URLs will be group by an unique URLs set without parameter. This set will be clustered in clustering phase based on an algorithm which presented in section 3.6.
- After that, in the clustering phase, unique URLs will be classified into two group clustered URLs and unclustered URLs. All of them will be identified in identifying phase which is presented in section 3.7.

3.5 Preprocessing Phase

This preprocessing phase is objective to eliminate unnecessary processed data. For each client IP, the one day HTTP traffic features are extracted and prepro-

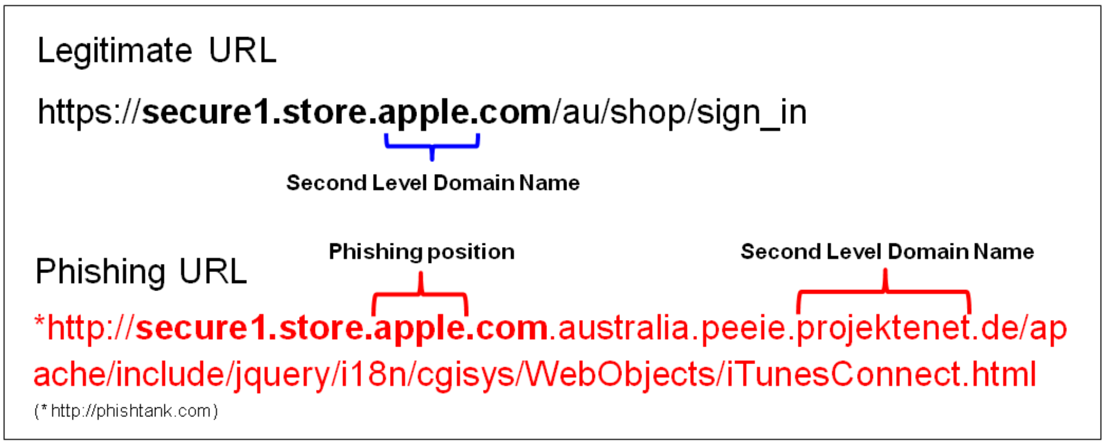

Legitimate URL
https://secure1.store.apple.com/au/shop/sign_in
Second Level Domain Name
Phishing URL
Phishing position
Second Level Domain Name
*http://secure1.store.apple.com.australia.peeie.projektenet.de/apache/include/jquery/i18n/cgisys/WebObjects/iTunesConnect.html
(* http://phishtank.com)

FIGURE 3.11: Suspicious websites are less consistent with their content when compared with those of legitimate websites.

cessed, in order for this phase to be processed, two methods are applied:

- The first one is to filter URLs requests from Client IP through a white list of second level domain name (SLDN). This filter method is described in [81], according to which, the tokens in the URLs of phishing websites are less consistent with their content when being compared with those of legal websites. An example is illustrated in Figure 3.11. In this example, the legitimate website contains the brand names *apple* in the SLDN. Even though the phishing website also contains the brand name *apple* in the URL, it is not in the SLDN. Therefore, a domain name which contains a second level domain name is defined in SLDN white list is marked as benign.

- The second method is based on the number of requests to a URL from a client IP. Based on the number of observed requests from autoware to a URL, it can be seen that suspicious autoware are accessed many times to a URL in a duration of time. Therefore, if the number of requests to a URL is too small, it seem not to be requested by an autoware.

Also in this phase, URLs which requested with extremely fast speed in a certain duration of time will pose a malicious autoware communication. Access speed is defined as in Equation (3.5.17).

$$AccessSpeed(URL_i) = \frac{RequestCount(URL_i)}{AccessTime(URL_i)} \tag{3.5.17}$$

In that, Access Time and Request Count features are described in section 3.2.1.

3.6 Clustering Phase

After pre-processing phase, in clustering phase, remaining URLs will be clustered into number of groups based on their characteristics. As presented in [70], there are many types of clustering algorithms or methods such as supervised and unsupervised methods. In supervised method, which k-means algorithm is a famous one, the number of classes needs to be determined as its input parameter. However, in this research, URLs are clustered based on their access behavior with undetermined number of classes because the access graph from an autoware can change by its setting or their timely update. In this section propose an unsupervised clustering method is proposed with only minor input parameter which is just URLs access graph. The proposed algorithm is distance/similarity based algorithm, the input data is just access graph. The number of classes do not need to be determined as a parameter. In which, two URLs are the same group (requested by the same autoware from a client) if they match one of follow conditions:

- The first and the last request timing to two URLs are approximately the same.
- Based on the similarity of its access graphs, by checking through adaptive threshold if their similarity score is small enough, they will be marked as in the same group.

Accordingly, adaptive thresholds are proposed to score the similarity of access

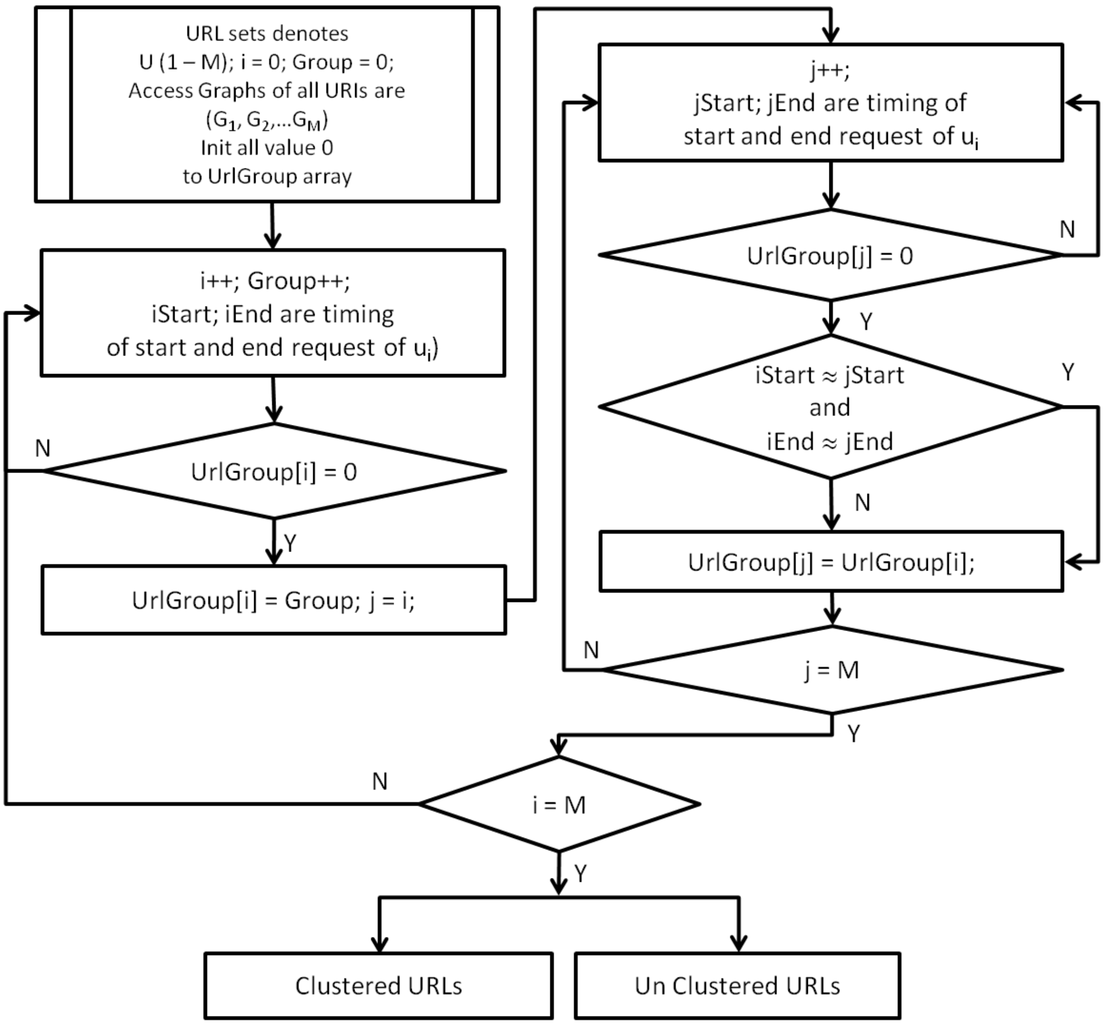


Figure 3.12: The first step of clustering algorithm

graph of each URL to determine the group.

In this chapter, in order to improve the clustering phase, a new algorithm is proposed in Table 3.3. Accordingly, there are three main steps by using the characteristics of access time and access graph similarity.

- The first step is to determine group of URLs based on their access time, two URLs are marked as in the same group if they have approximately equivalent access time. This step is illustrated in Figure 3.12.
- The second step will collect the group from result of step 1 and calculate a threshold δ which is average similarity of any two in URLs. This δ will

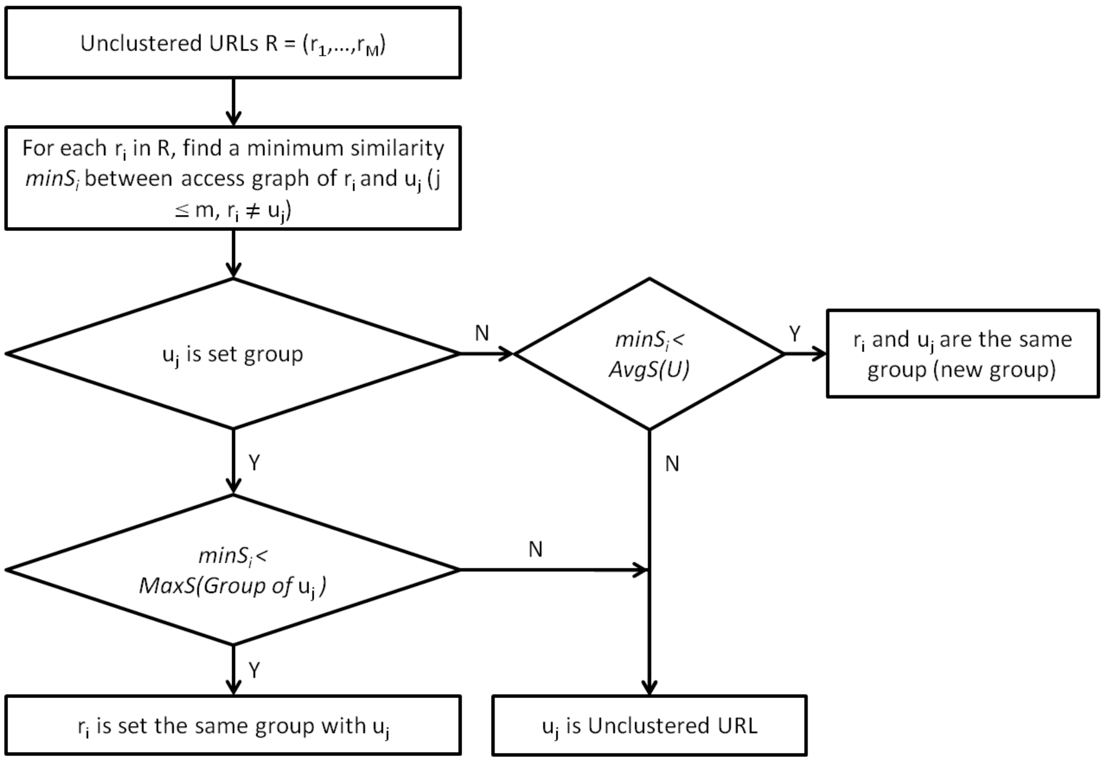

Unclustered URLs R = (r1,...,rM)
For each ri in R, find a minimum similarity minSi between access graph of ri and uj (j ≤ m, ri ≠ uj)
uj is set group
N
minSi< AvgS(U)
Y
ri and uj are the same group (new group)
Y
N
minSi< MaxS(Group of uj)
N
Y
ri is set the same group with uj
uj is Unclustered URL

FIGURE 3.13: The third step of clustering algorithm

TABLE 3.3: Steps of Clustering Algorithm

Step	Description
1	For each pair (u_i, u_j) in set U of unique URLs $(1 - M)$. Denoted that (i_{Start}, i_{End}) and (j_{Start}, j_{End}) are timing of start and end request of ui and uj respectively.
	If $(i_{Start} \cong j_{Start})$ and $(i_{End} \cong j_{End})$ then u_i and u_j, are in the same group.
2	After the first step, part of URLs are clustered, each group owns at-least two URLs. A threshold $\delta = AvgS(U)$ as in Equation (3.3.16).
3	For each URL u_i which is not set group in previous step 1, find a u_j which has minimum similarity to u_i denote that value is $minS_i$
	u_i and u_j are in the same class if one of two bellow conditions is matched: - if $minS_i < \delta$ in the case u_j is still not set to any group yet. - if $minS_i < MaxS$(Group of u_j) (as Equation (3.3.15)) in the case u_j was already set to a group.

help cluster remaining URLs based on their access graph similarity.

- The last step is to continuously cluster remaining URLs after step 1 by checking the similarity of their access graph. In that, two adaptive thresholds are determined for an unclustered URL to be a member of clustered

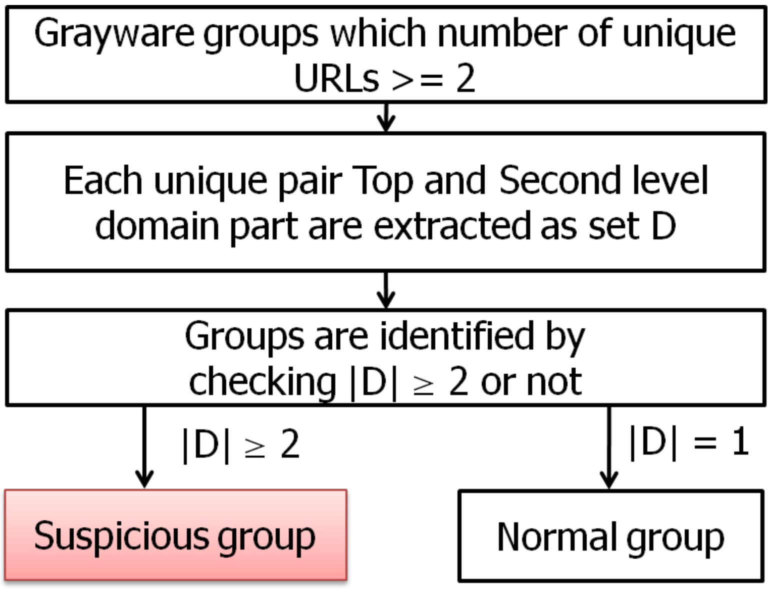

Grayware groups which number of unique URLs >= 2
Each unique pair Top and Second level domain part are extracted as set D
Groups are identified by checking |D| ≥ 2 or not
|D| ≥ 2
|D| = 1
Suspicious group
Normal group

FIGURE 3.14: Group identifying phase

group which is determined in step 1 or to be paired with other unclustered URL to become a new group. This step is illustrated in Figure 3.13

3.7 Identifying Phase

3.7.1 Group Identifying Phase

After clustering phase, groups which contain from more than 2 URLs will be identified into two type of group normal or adware. The steps for this phase is shown in Figure 3.14. This phase is based on the analysis in section 3.2.3, in which, adware tends to access to many different domain name to maintain advertising contents. Therefore, in this phase, by counting the unique domain names in each group, if it is greater or equal to 2, this group will be marked as suspicious group. Vice versa, group will be marked as normal one.

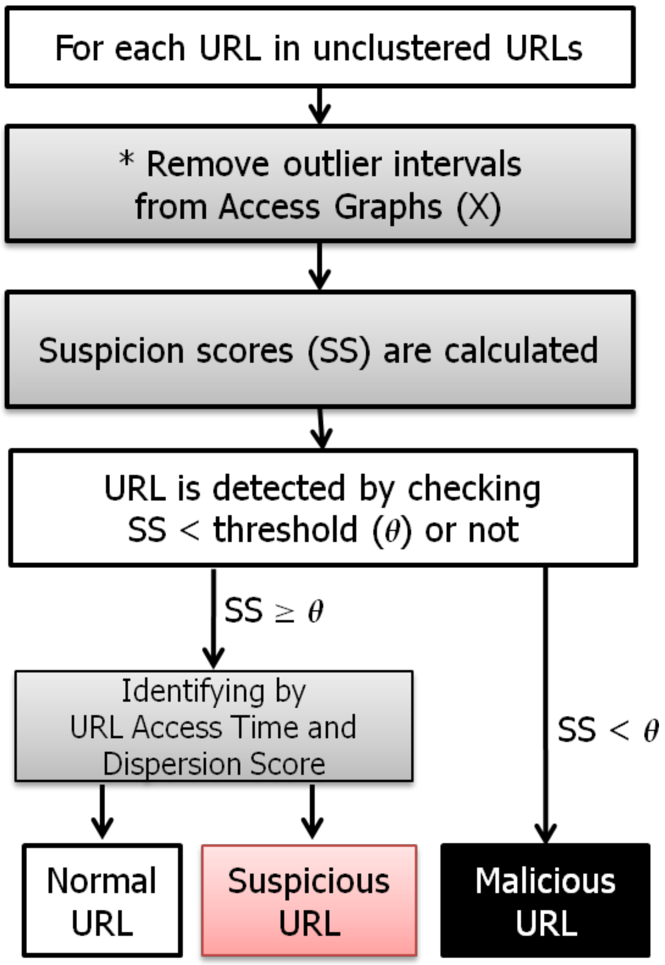

For each URL in unclustered URLs
* Remove outlier intervals from Access Graphs (X)
Suspicion scores (SS) are calculated
URL is detected by checking SS < threshold (θ) or not
SS ≥ θ
SS < θ
Identifying by URL Access Time and Dispersion Score
Normal URL
Suspicious URL
Malicious URL

FIGURE 3.15: Unclustered URL identifying phase

3.7.2 URL Identification Phase

For URLs are not clustered, a suspicion score based on their access graphs is presented to recognize a URL is malicious or normal. Malicious bot always communicate to a specific URL or resource by automatically generating requests with a stable interval (section 3.2.3), therefore, malicious bot access graph almost has no variation. In this phase, the method do not try to detect the interval of malicious requests but in target to score the variation of access graph. However, the interval of malicious bot requests are changed some times, these are outlier intervals, but the main interval is steady. As can be seen in Figure 3.4(a), some intervals are uncertain but stable interval is around 1800 seconds. The number of points in access graph outlier intervals in access graph is minor in comparison with main stable intervals. In order to detect outlier intervals from access graph X, a density-based algorithm (DBSCAN) in [86] is suggested. DBSCAN will help cluster all the similarity intervals in access graph X, from there, outlier intervals are clustered in group with smaller number of members. Details of DBSCAN algorithm is in [86], a basic contents of this algorithm is presented in next section 3.7.2.

The flow for this phase with three main steps are shown in Figure 3.15:

- First, intervals in access graph of URL are clustered into groups by DBSCAN algorithm. Because malicious bot will communicate to their server by main intervals, so groups containing main intervals will own much more members than other groups. Outlier intervals account for the few points in access graph, and they are belong groups which they contain smallest number of members, for that matter. These groups will be removed from access graph. As can be seen in Figure 3.4, intervals in the dashed circles will be detected by DBSCAN algorithm and remove from access graph before to be processed in next step.

- Next, in order to identify a URL is malicious or not, a suspicion score is calculated based on its access graph (outlier intervals are omitted). This score is described in section 3.7.2. If suspicion score of a URL is less than a threshold, this URL will be recognized as being accessed by malicious communication.

- Remaining URLs are identified by checking the access time and dispersion score. Suspicious autoware working along with human computer, therefore, if access time to a URL is similar with user computer interaction, URL will marked as accessing from suspicious. Different with malicious bot, communication which is generated from suspicious autoware such as adware, is alway with variation intervals. Therefore, URL own high dispersion score (above 0.5 in this experiment) in access graph is also marked as suspicious

Table 3.4: Density-based Algorithm Discovering Clusters

```
DBSCAN (SetOfPoints, Eps, MinPts)
//SetOfPoints is UNCLASSIFIED
  ClusterId := nextId(NOISE);
  FOR i FROM 1 TO SetOfPoints.size DO
    Point := SetOfPoints.get(i);
    IF Point.ClId = UNCLASSIFIED THEN
      IF ExpandCluster(SetOfPoints, Point, ClusterId, Eps, MinPts)
        THEN
        ClusterId := nextId(ClusterId)
      END IF
    END IF
  END FOR
END; // DBSCAN
```

one. Dispersion score is described in Sub-section 3.7.2.

Density-based algorithm discovering clusters

A basic version of DBSCAN omitting details of data types and generation of additional information about clusters is in Table 3.4 [86]. In that, $SetOfPoints$ is either the whole database or a discovered cluster from a previous run. Eps and $MinPts$ are the global density parameters determined either manually. The function $SetOfPoints.get(i)$ return the i-th element of $SetOfPoints$. The most important function used by DBSCAN is *ExpandCluster* which is presented as in Table 3.5. In that, A call of $SetOfPoints.regionQuery(Point, Eps)$ returns the $Eps - Neighborhood$ Point in $SetOfPoints$ as a list of points. The $ClId(clusterId)$ of points which have been marked to be NOISE may be changed later, if they are density-reachable from some other point of the database. This happens for border points of a cluster. Those points are not added to the seeds-list because we already know that a point with a $ClId$ of NOISE is not a core point. Adding those points to seeds would only result in additional region queries which would yield no new answers.

Table 3.5: ExpandCluster Function of DBSCAN

```
ExpandCluster(SetOfPoints, Point, ClId, Eps, MinPts) : Boolean;
  seeds := SetOfPoints.regionQuery(Point, Eps);
  IF seeds.size <MinPts THEN //no core point
    SetOfPoint.changeClId(Point, NOISE);
    RETURN False;
  ELSE //all points in seeds are density-reachable from Point
    SetOfpoints.changeClIds(seeds,ClId);
    seeds.delete(Point);
    WHILE seeds $\neq$ Empty DO
      currentP := seeds.first();
      result := setofPoints.regionQuery(currentP,Eps);
      IF result.size $\geq$ MinPts THEN
        FOR i FROM 1 TO result.size DO
          resultP := result.get(i);
          IF resultP.ClId IN (UNCLASSIFIED, NOISE} THEN
            IF resultP.CiId = UNCLASSIFIED THEN
              seeds.append(resultP);
            END IF;
            SetOfPoints.changeCiId(resultP,CiId);
          END IF; // UNCLASSIFIED or NOISE
        END FOR ;
      END IF; // result.size >= MinPts
      seeds.delete(currentP);
    END WHILE; //seeds <>Empty
    RETURN True ;
  END IF
END; //ExpandCluster
```

Suspicion Score

After removing the outlier intervals by DBSCAN algorithm. A score is proposed to measure the variation of a access graph, from which it shows suspicion of communication between client to its URL. Assuming that the access graph after removing outlier intervals of a URL U is specified and denoted as $X = (x_1, ..., x_N)$. Most of the HTTP malware intent to use a constant time interval or a random interval time within a constant value between two request periods. For example in Figure 3.4(b), it shows that the malicious bot have random intervals switching between 50 and 100 seconds. Therefore, in order to detect the periodic characteristics occurred in C&C communication, average time interval vector $X_{Avg} = (\bar{x}_1, ..., \bar{x}_M)$ is are established, by calculating the average interval between consecutive two subsequent flows in X. It means that $\bar{x}_1$ is average of $(x_{i1}, ..., x_{ik})$. A suspicion score will be defined as *coefficient of variation* of X_{Avg} as bellow equation.

$$SuspiciousScore(X_{Avg}) = \frac{\sigma(X_{Avg})}{\mu(X_{Avg})} \tag{3.7.18}$$

In that $\sigma(X_{Avg})$ and $\mu(X_{Avg})$ are standard deviation and mean of X_{Avg} respectively. The smaller suspicious score shows that URL is more suspicious.

Dispersion Score

Dispersion score of a URL is to measure the fragment degree of intervals in its access graph. The score is determined by proportion between number clusters of access graph by DBSCAN algorithm and number of requests to a URL. Assuming that N is number of requests to a URL from a client and C is number of groups which are clustered by DBSCAN. The dispersion score is determined as bellow.

$$DispersionScore(X) = \frac{C}{N} \tag{3.7.19}$$

3.8 Experimental Results and Discussion

From the experiment purpose, outbound HTTP traffic from a university network are captured through a proxy server in separated files and stored in a proxy storage. The experimental model is shown as in Figure . For experiment purpose, 70 HTTP log data are selected from 430 GB real traffic imported in our experimental system. Statistic of experimental data are summarized in Table 3.6. Data are collected from various range of time which not just in one day. As in Table 3.6, log data is from 150 minutes to 5 days, after pre-processing phase 58.85% of URLs need to be continuously clustered and identified in next phases. All output

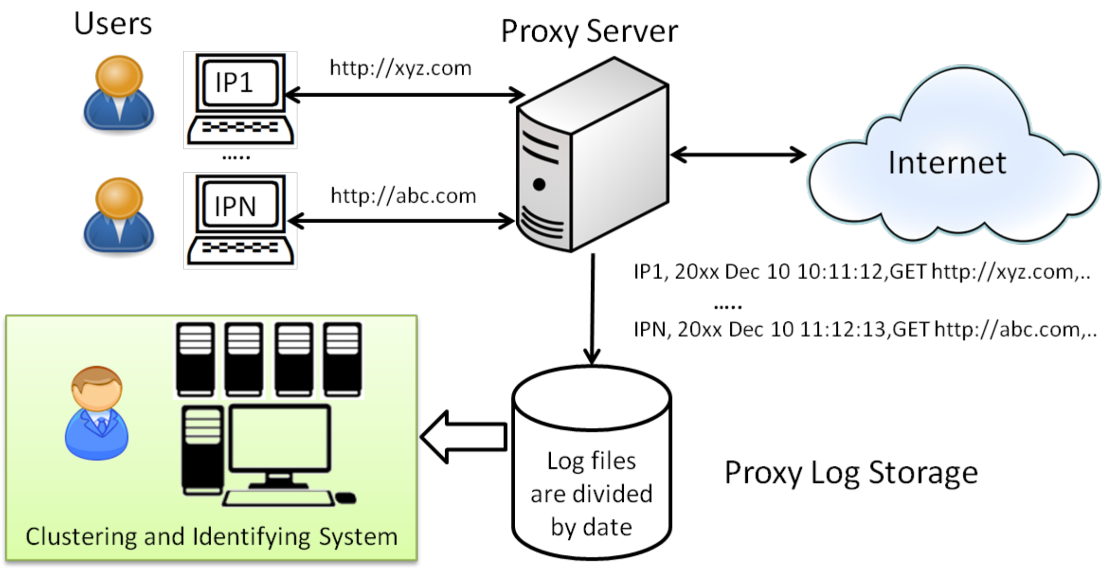

Users
IP1
.....
IPN
http://xyz.com
http://abc.com
Proxy Server
Internet
IP1, 20xx Dec 10 10:11:12,GET http://xyz.com,..
.....
IPN, 20xx Dec 10 11:12:13,GET http://abc.com,..
Clustering and Identifying System
Log files are divided by date
Proxy Log Storage

FIGURE 3.16: Experimental environment for network level proposed method.

TABLE 3.6: Experimental Data Statistic

No	Item	Values
1	Number of log data	70
2	Total number of requests	16,211,257
3	Max requests in a log data	3,030,216
4	Min requests in a log data	2,110
5	Min access time in a log data	150 minutes
6	Max access time in a log data	5 days
7	Requests after pre-processing phase. (58.85% of total requests remaining)	9,540,608 (58.85%)
8	Number of Unique URLs after pre-processing phase	10,942

results are manually check with the support of VirusTotal online system [84] and McAfee Web Gateway [85] which is installed in experiment network.

After pre-processing phase, remaining 10,942 unique URLs are as input of clustering phase. Results of this phase are summarized in Table 3.7. In that, 92.30% URLs (10,100 URLs) are clustered in group. These results indicate that mostly autoware communicate to their servers with the similar behavior since just about 7.70% URLs are unclustered which are requests with specified behavior. As analysis in section 3.2.3, these unclustered URLs are considered as malicious communication. As can be seen in Figure 3.17, it depends on the characteristics

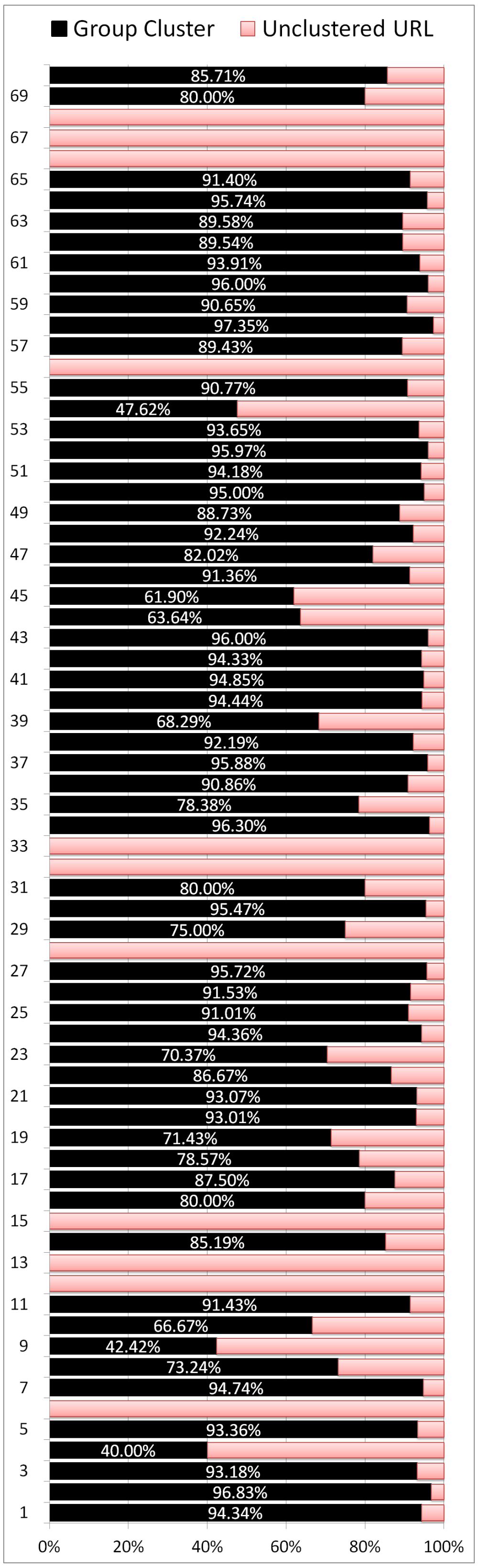


Figure 3.17: Proportion of clustered and unclustered URLs of clustering phase

Table 3.7: Clustering Phase Results

No	Item		Values	Percent
1	Unique URLs after pre-processing		10,942	100%
2	Clustered	Groups (| URLs| ≥ 2)	1,591	
		Number of URLs	10,100	92.30%
3	Unclustred	Number of URLs	842	7.70%

Table 3.8: Group Identifying Results

No	Group Type	Groups	Detect Results		True		False	
			URLs	Percent	URLs	Percent	URLs	Percent
1	Normal	627	2,476	24.51%	2,325	93.90%	151	6.10%
2	Suspicious	964	7,624	75.49%	7,021	92.09%	603	7.91%
3	Total	1,591	10,100	100%	9,346	92.53%	754	7.47%

Table 3.9: URL Identifying Results

No	URL Type	Detect Results		True		False	
		URLs	Percent	URLs	Percent	URLs	Percent
1	Normal	399	47.39%	316	79.20%	83	20.80%
2	Suspicious	389	46.20%	264	67.87%	125	32.13%
3	Malicious	54	6.41%	51	94.44%	3	5.56%
4	Total	842	100%	631	74.94%	211	25.06%

Table 3.10: Overall Experimental Results

No	Result	Number of URLs	Percent
1	True	9,977	91.18%
2	False	965	8.82%
3	Total	10,942	100%

of each client, the clustered proportion between clustered and unclustered URLs are different. Most of log data contain suspect automated access which having burst requests since the proportion around 50% and above (black bar in Figure 3.17), and 11 log data own the access to separate URLs with difference behavior.

As in Table 3.7, 1,591 groups of 10,100 clustered URLs are as input data for group identifying phase which is described in Figure 3.14. The identified results are details in Table 3.8. There are two kind of groups are detected, normal and suspicious and evaluated. Normal groups mostly include all URLs which are requested for news or analytic updates. Vice versa, suspicious groups contain URLs which accessed for unwanted action such as advertised purposes or suspicious download. Even still exist some false detection rate, accuracy in this step is 93.90% and 92.09% for normal and suspicious group identifying respectively, and total accuracy reach 92.53% since the total of false detection rate is 7.47%.

In the final step, 842 unclustered URLs (Table 3.7) will be identified as described in Figure 3.15. In this step, three kinds of URLs normal, suspicious and malicious are identified. The results are summary in Table 3.9. In that, malicious URLs are detected with highest accuracy at 94.44% in that 11 URLs are recognized as accessed at very high speed, one of the access graph is as in Figure 3.5. Based on the characters of malicious autoware which infected into client IP, the access speed and also communication behavior to these URLs are determined. For example, as can be seen in Figure 3.5(b), just in only 0.6 hour, this URL is requested 80,903 times so it owns highest access speed at 32.98 requests per second. Vise versa, with URL in Figure 3.5(a), it is requested with lowest speed at 0.82 request per second, 71,004 times in 24 hours, however it is still higher than access speed to other URLs in experimental data. The next highest true identifying rate is for normal URLs detection with 79.20% and lowest is of suspicious since it reaches 67.87%. The false negative rate of normal identifying are 20.80% because detected normal URLs are suspicious and vi-versa, the false negative rate of suspicious identifying are 32.13% because detected suspicious URLs are normal. These results show that identifying between normal and suspicious unclustered URLs is really tougher since behavior of the communication to them are similar with each other. This challenge needs to be solved to increase the effectiveness of current achievement. For that, in the future work, an additional features need to be considered in order to reduce these false negative.

The overall results for whole phases are concluded in Table 3.10. In that 100% URLs are clustering and identifying with the accuracy reaches at 91.18% and error rate constitutes 8.82%. In that, there are malicious communication to 5 URLs which are not detected or updated by [85] but they are identified by our method. Comparing to [57–63], the proposed method can detect and identify not just malicious but also suspicious traffic with very minor features at application level.

3.9 Conclusions

In this chapter, a new novelty method in clustering and identifying HTTP automated communication is proposed. The method is improved from result [90, 91], accordingly, URLs are not just classified into group but also identified and detected by their access purposes. This findings assist network and system administrator clarify the HTTP automated traffic which are almost unknown to users, from there the internal threats caused by HTTP autoware might be inspected early. The huge benefit of the method is being independent from payload signatures which enables the identification of many kinds of automated communication with obfuscated and encrypted message contents.

The method works well on any network which just allow HTTP in outbound traffic to prevent threats from TCP/IP direct connection. The accuracy of the approach is equally high, and achieve 91.18% for overall and 94.44% for malicious URLs detection by using only a few features at application level. In that some URLs which are detected by our method in while they are bypassed in [85]. The evaluation also proved that the results are not dependency period of captured data.

Chapter 4

Application Model Proposal

In this chapter two application models are introduced for applied the research into the real life. They are proposed as in Fig.4.1 and Fig.4.2. In which, HASCD is abbreviation of "HTTP Automated Software Classification and Detection".

4.1 Host based application model proposal

In host based app, in each client computer will installed a tiny software running the method in chapter 2. The software will detect and sent results to a database. The report should include client ip, traffic, detection results. From there, an application will extract and support existed network defense system.

The results from clients will be unstructured and huge each day, so Mark-Logic NoSQL (Not only Structure Query Language) database is recommended. As described in [77, 78, 83], MarkLogic is an enterprise NoSQL database which supports a very flexible and convenient XQuery when working with structured and also unstructured data. In addition, it also has had ACID transactions (ACID stands for Atomicity, Consistency, Isolation, and Durability). In a transactional application ACID' s properties are necessary so that reads and writes are durably logged to disk and strongly isolated from other transactions. Without

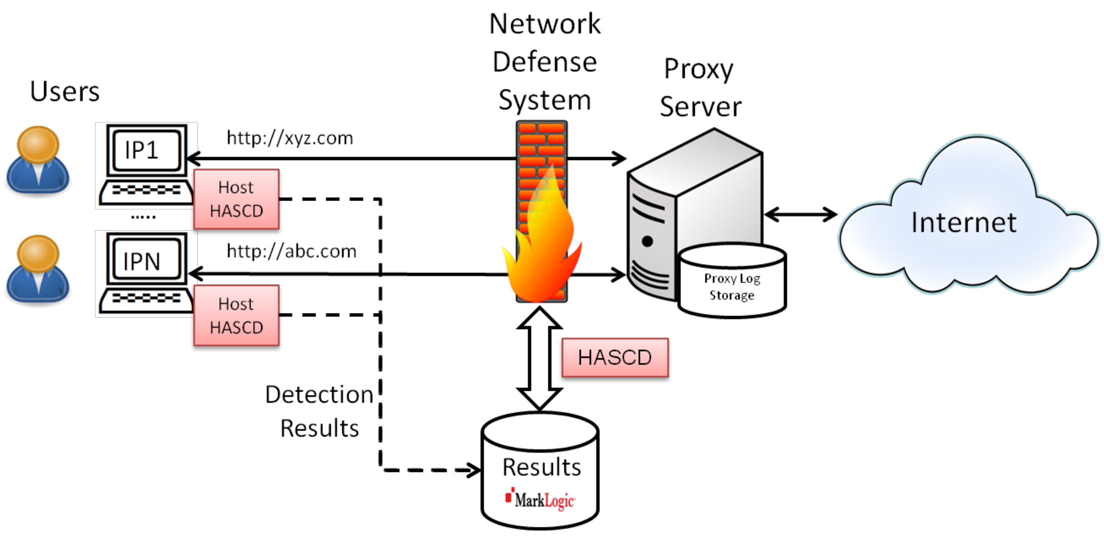


FIGURE 4.1: Proposal application model of host based method

this feature, users run the risk of encountering data corruption, stale reads, and inconsistent data. In addition, MarkLogc is really easy for setting of replication, backing up data and it supports loading and indexing data ”as is”. That means format of data can be stored with its original format.

4.2 Network level application model proposal

In network based app, network based HASCD will executed the method proposed in chapter 3. The network based HASCD will detect and extract suitable results to support existed network defense system.

The network based method in chapter 3 includes three phase: preprocessing, clustering and identifying phases. Difference with host based app, all HTTP traffic will be centralized processed. As consequently, the processed data will be huge, especially the clustering phase since the number of traffic and groups are increasingly everyday. Therefore, a big data application which it is combination of MarkLogic data base and MapReduce of Hadoop and summarized in Figure 4.3

Hadoop is a great tool to help database application developers and organiza-

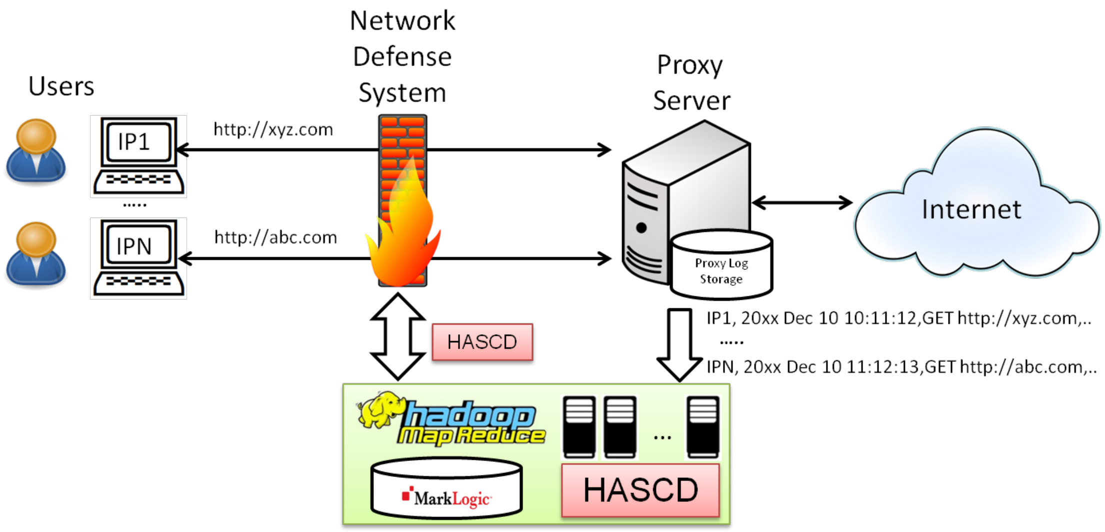


FIGURE 4.2: Proposal application model of network based method

tions to store and analyze massive amounts of structured and unstructured data from disparate data sources, which data are too massive to manage effectively with traditional relational databases. Hadoop has become popular because it is designed to cheaply store data in the Hadoop Distributed File System (HDFS) and run large-scale MapReduce jobs for batch analysis. MapReduce is a processing framework that uses a divide-and-conquer paradigm that takes a huge task and breaks it into small parts (Map) and then aggregates the resulting outputs from each part (Reduce). Any large task that can be broken into smaller pieces is a candidate for use with Hadoop [87].

The combination between MarkLogic database and MapReduce of Hadoop in this framework is described in Figure 4.3. Whereby, a cluster of MarkLogic is set, and due to optimizing performance in query to database, three XML Database Connector (XDBC) application servers, Data Collection, Clustering and Data Analysis, are configured along with a number of forests. There are three modules are working independently for each phase, details are expressed as bellow:

- Data Manipulation Module (DMM) which will read raw log files, convert to XML and text format, and do the preprocessing before stored into MarkLogic database via Data Collection Application Server. Pre-processing

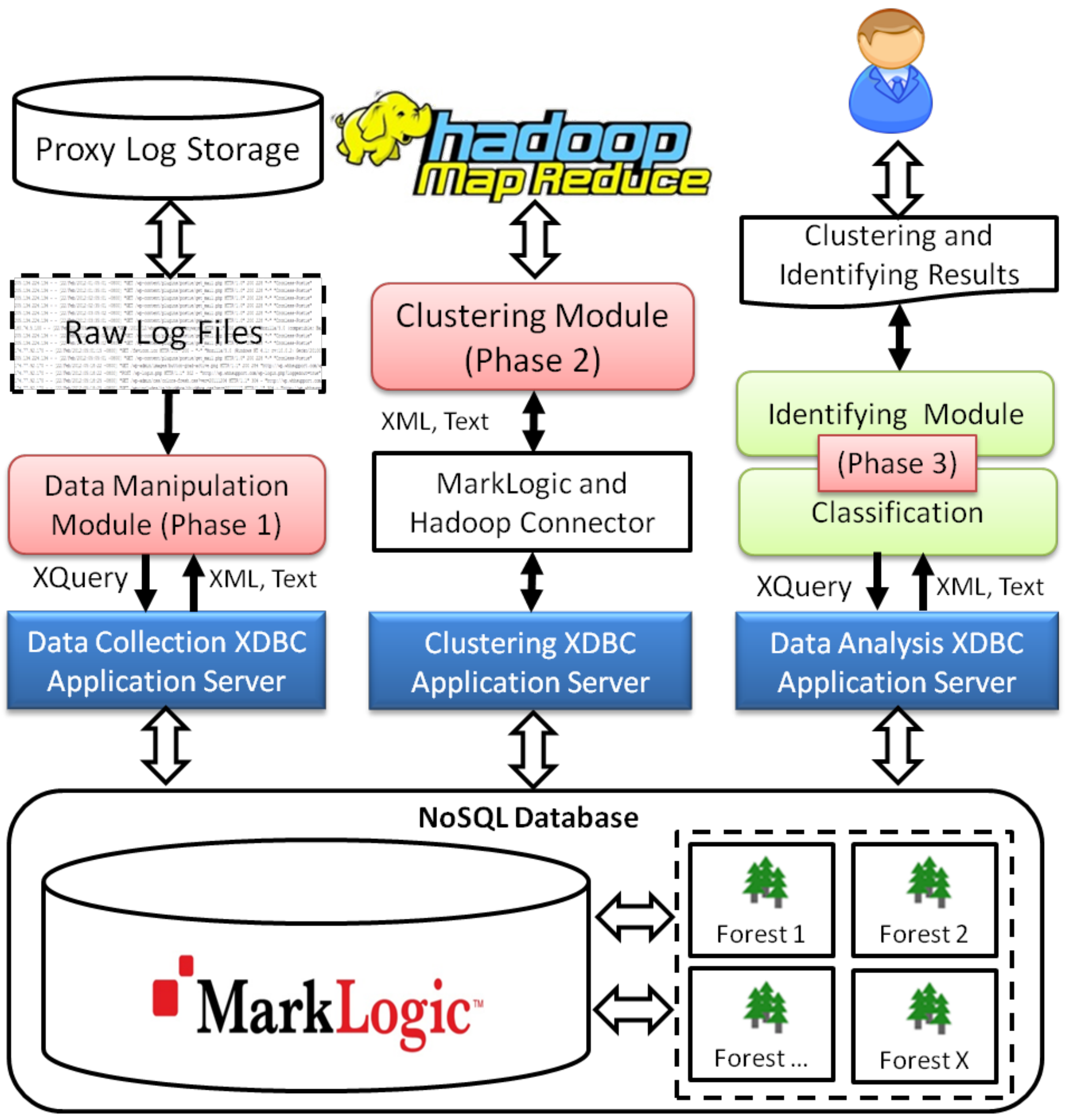

Proxy Log Storage
hadoop Map Reduce
Clustering and Identifying Results
Raw Log Files
Clustering Module (Phase 2)
XML, Text
Identifying Module
(Phase 3)
Classification
MarkLogic and Hadoop Connector
Data Manipulation Module (Phase 1)
XQuery
XML, Text
XQuery
XML, Text
Data Collection XDBC Application Server
Clustering XDBC Application Server
Data Analysis XDBC Application Server
NoSQL Database
MarkLogic
Forest 1
Forest 2
Forest ...
Forest X

FIGURE 4.3: Big data application for observation, clustering and identifying framework. Phase 1, 2, 3 are pre-processing, clustering and identifying phase respectively

phase (phase 1) is included as a part in this module. Details of this phase is presented in section 3.5 of Chapter 3.

- Core and heavy functions are deployed in the middle part between MarkLogic database and MapReduce of Hadoop, Clustering Module (CM). This module will archive results from Pre-processing phase, and URLs are clustered in MapReduce by the distributed processing paradigm. This is clustering phase (phase 2). Algorithm of this phase is presented in section 3.6

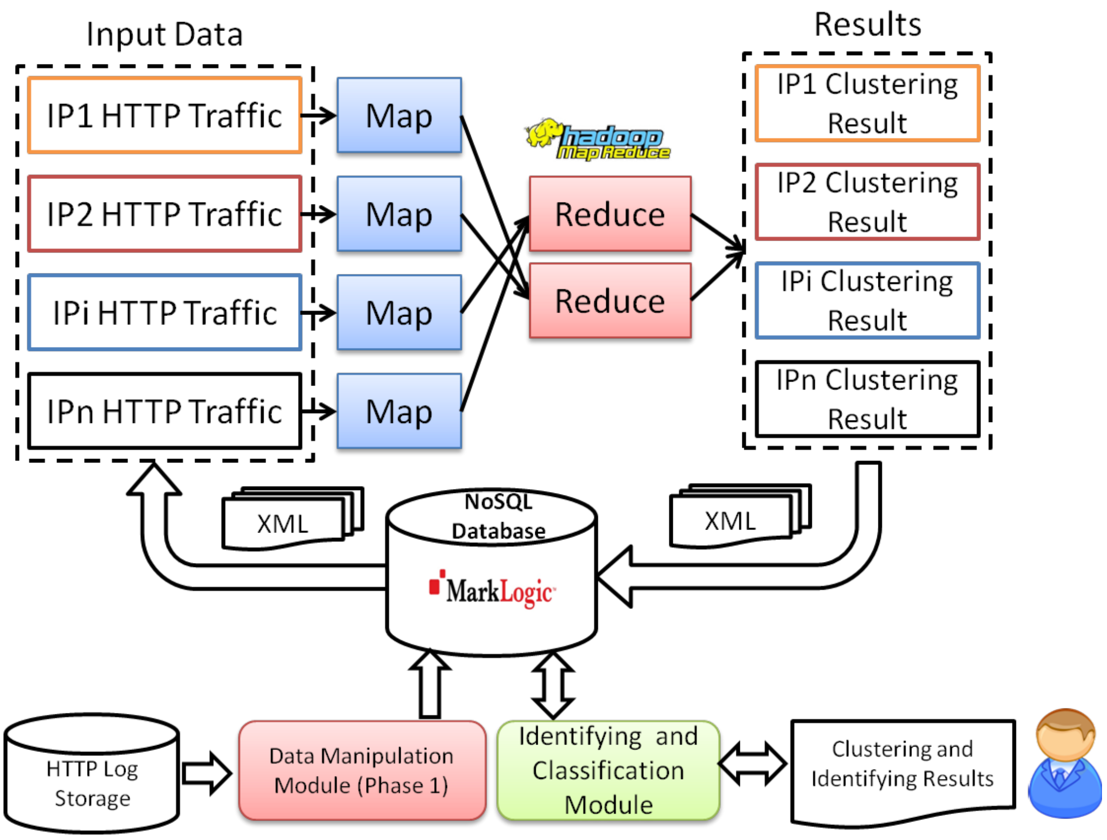


FIGURE 4.4: Process flow of Clustering phase on MapReduce of Hadoop

of Chapter 3. Finally, results of Phase 2 will be returned to MarkLogic database through Clustering XDBC application server. The data exchange between MarkLogic and MapReduce of Hadoop will be undertaken by a connector [88]. Detail process flow of this phase is described in Figure 4.4.

- Detection and Identification Module (DIM) is implemented for identifying phase (phase 3). This phase process is described in section 3.7 of Chapter 3. It will process the result which archived from Phase 2, and work with database through Data Analysis Application Server, after that give out processed results.

Each modules DMM, CM and DIM in framework are implemented as each pre-processing, clustering, and identifying phase respectively. Phases' logic or algorithm will be described detail in chapter 3. In the experimental environment, a free developer licenses of MarkLogic version 8.0.1 and Hadoop 2.6.0 are used

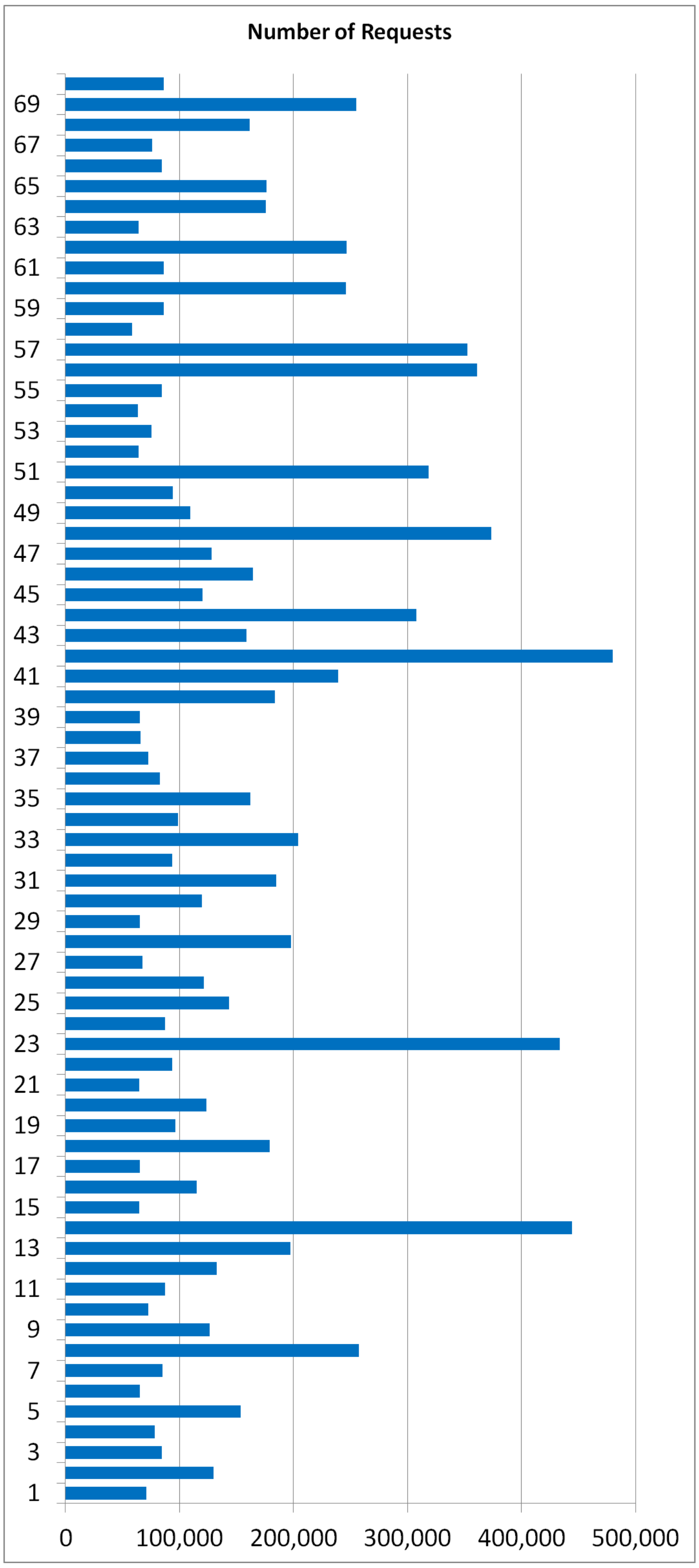


FIGURE 4.5: Number of requests in each log data of one IP. X axis shows the number of requets and Y axis show the client's log data index from 1 to 70.

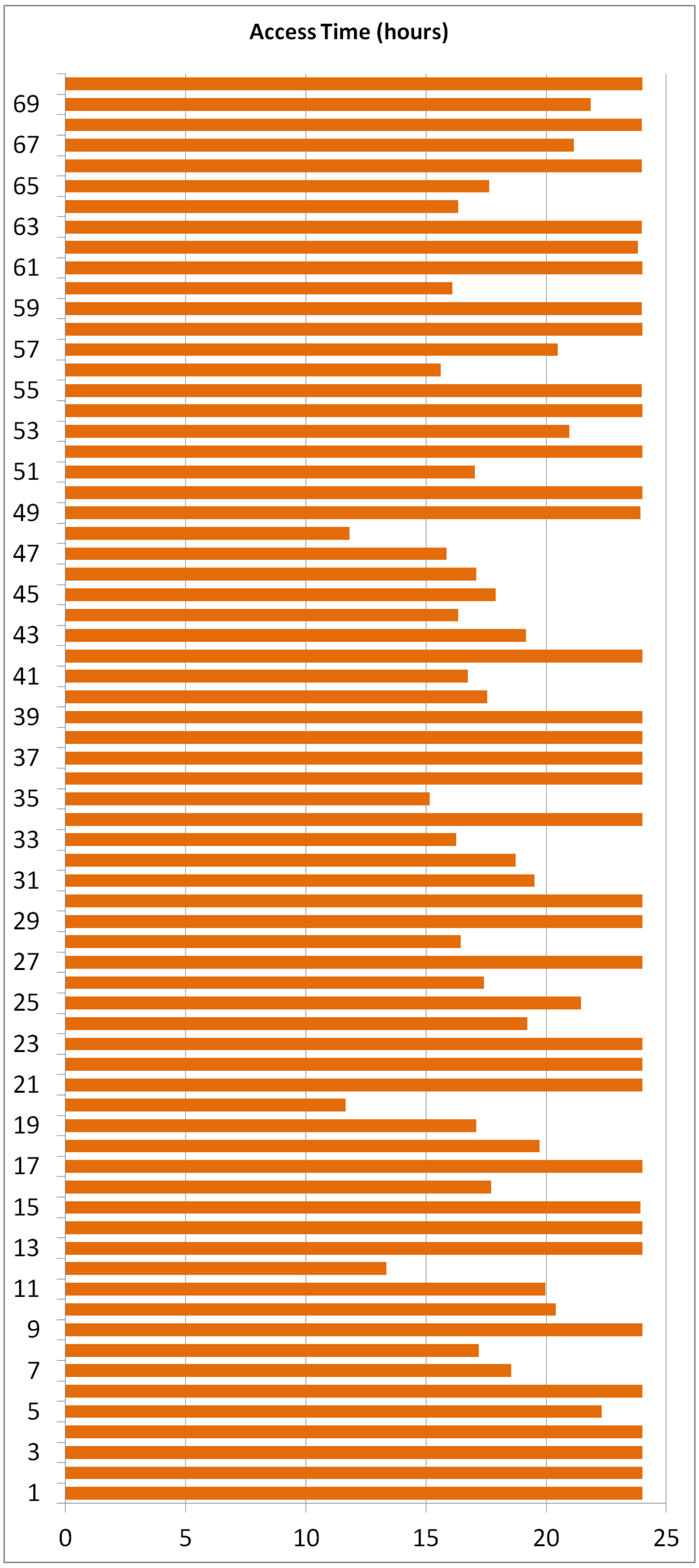


FIGURE 4.6: Access time from client IP to the Internet. X axis shows the access time in Hours and Y axis show the client's log data index from 1 to 70.

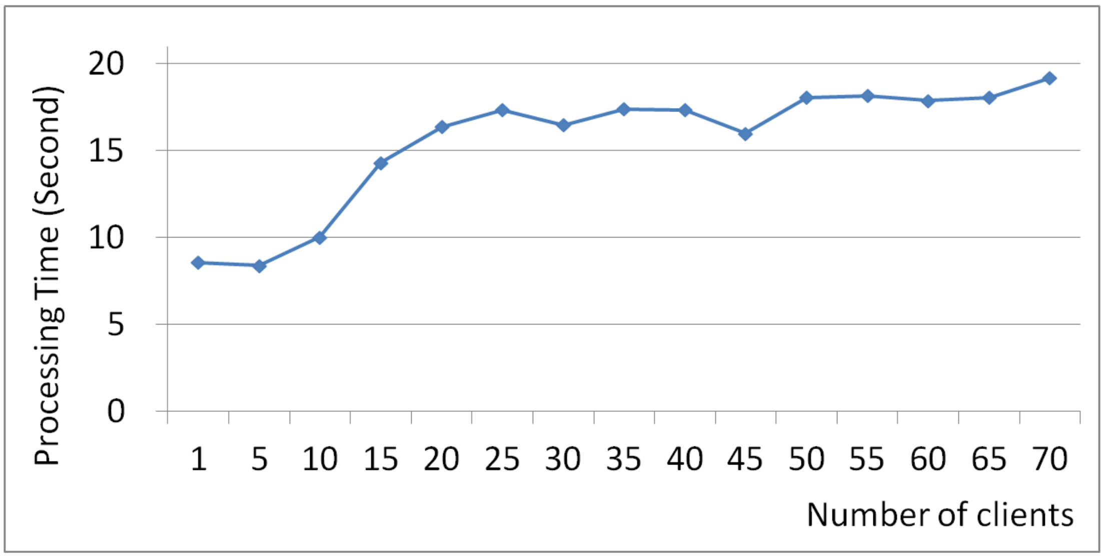


FIGURE 4.7: Processing time of experimental implementation

TABLE 4.1: Experimental System Specification

No	Items	Description
1	MarkLogic	A Cluster is installed in two PCs, PC1 is exchange data to Mapreduce of Hadoop (phase 2), PC2 is implemented as phase 1 and phase 3
		PC1 specification - CPU: Intel Core 2 Quad CPU Q6600 - CPU Cores: 4 - Processor base operating frequency: 2.4 GHz - RAM: 4GB
		PC2 specification - CPU: Intel Core i7 965 - CPU Cores: 4 - Processor base operating frequency: 3.2 GHz - RAM: 8GB
2	Mapreduce of Hadoop	Execution on a sever with below Specification - CPU: Intel Xeon Processor E5-2430 v2 (15M Cache, 2.50 GHz) - CPUs: 2 - CPU Cores: 6 - Threads number: 12 - Processor base operating frequency: 2.5 GHz - RAM: 128GB
3	Experiment System Programming	Programming Language are C# on Microsoft Dotnet Framework and Java.

[89]. Outbound HTTP traffic from a university network is captured through a proxy server in separated files and stored in a proxy storage.

Three phases in Figure 4.3 are implemented in experiment environment as described in Table 4.1. In order to have balance input data to evaluate the processing time, experimental data of clients Internet access are captured in one day, around 24 hours. Details of access time and request number are expressed in Figure 4.5 and 4.6. In that, number of requests in each log are from 58,606 to 479,751 times. Processing time of application (excluding processing time of phase 1) is shown in Figure 4.7. In that, data of clients will be grouped into 5 clients and executed increasingly from 1, 5, 10 and so on until 70 clients. As can be seen that, the processing time fast increased since process data of 1 to 25 clients, after that, the processing time is stable around 18 seconds since the number of clients are in the range from 30 to 70. The processing time takes logarithmic complexity. Overall, minimum and maximum of processing time are respectively 8.56 and 19.18 seconds, average processing time is 15.58 seconds.

Chapter 5

Conclusion and Future work

5.1 Research Results Summary

HTTP autoware is very diverse and sophisticated. Beside the good one, suspicious and malicious application are widely deployed on the public network. The traffic from these application are almost unknown and unnoticed to users. Detecting and Identifying these unknown HTTP automated traffic are really great challenge because they are merge transparently with normal and users' requests. Unconsciously, there are high risks of PCs in any network are not strictly controlled by users. These exposed to be infected by unwanted autoware and become internal threat. In this research, host and also network based methods are proposed in classification and detection of autoware based on their access communication behavior. In that, access behavior of multiple autoware is analyzed with minor features.

In the host base, the behavior of autoware is observed and analyzed through two parameters periodic access and access rate. The observation shows that these two features of suspicious autoware are stable than others normal autoware. In another word, almost no variation in the graph of periodic access and access rate in a period of time. The advantage of host-based application is that the application can be applied for single PC (without interconnected network) with limitation of memory and resource. In addition, the design in host space within interconnected network will help to reduce the risk the whole network since the traffic are possible generated from malware infection can be ban from the source. It is self-surveillance for users. The application will help users monitor or watch their traffic to identify suspicious one and do the filtering before they are allowed joining the network from their PC.

In the network level, multiple traffic from many clients are observed and behavior of each kind of autoware are recognized. In that, malicious HTTP-based bots always connect to their command and control server periodically in order to get the commands and updates. The number of requests from malicious bots are not high as normal autoware (e.g. updater and downloader) which just generate requests with a long interval than unusual malicious bots. Malicious access is recognized by scoring the its access speed. Malicious bots often connect to one control domain and to a specific server resource during a given period of time. Difference with that, unwanted HTTP applications, or grayware, such as annoying adware or spyware, often report back to or request new information from many external resources. Therefore, they keep communicating to their numerous advertising sites or URLs to update pop-up or advertisement and commercial content areas. Autoware will behave the same communication pattern to its difference URLs. if they are requested at the same or approximately equivalent timing, so access graph of URLs from a specified autoware are presented similar. In addition, autoware requested to many URLs with the same timing, so the access duration to these URLs is approximately equal. It means that the first and the last requests timing to these URLs are almost the same with others. Suspicious autoware, for instance adware, since they update contents, like other autoware they access to many URLs however they will collect data from many URLs of multiple sites which own various domain names. This is not alike with normal autoware, such as a electronic newspaper, since it self-refresh the contents of presenting page by accessing to many URLs but with only one domain name. A suspicious autoware starts with the time of human computer start. Therefore, it expected that the access duration of a suspicious autoware might be similar with user computer interaction.

The results in network level are not focused for a specific URLs but for group of URLs for a specific purpose. The findings assist network and system administrator clarify the HTTP automated traffic which are almost unknown to users, from there the internal threats caused by HTTP autoware might be inspected early. Besides that, a framework using combination of MarkLogic NoSql and Map Reduce Hadoop database is implemented to help dealing with huge captured traffic.

All methods are experimented by using real traffic and give a good results since mixture traffic of all kind autoware are still identified and detected. The proposed methods are just based on analysis of access graph and other minor features which can be simply captured at application level, therefore, they are also easily implemented in real environment. Network level proposed method is able to processed distributed, these will help reducing the processing time. In that, the experimental implementation for network level method in a big data system, which are combined MarkLogic NoSQL Database with Map Reduce of Hadoop, proves that it shows good performance since it has to process with huge traffic.

5.2 Future Work

All experiments show good results, however false alarm still contributes, especially in network level method. In that, the result shows that the false detection between normal and suspicious unclustered are slightly high. The main necessary improvement in the future work is to reduce these false alarm. In that, new features are considered to be added, that are related to the URLs properties to improve the accuracy in suspicious and normal identifying for unclustered URLs in network based method. In host based method, a mechanism need to be considered to calculate adaptive thresholds all feature scores.

Along with that, deep machine learning method is also considered in order to improve the achievement of current result. In addition, suspicious and malicious software always effort to spread out them-self on other clients in the same network. A proposal method in order to cluster of clients based on their automated communication is considered, in which, system or network administrator can score the activity of malware and also early detect the threats from internal. In addition, malicious HTTPS detection need to be considered since all the communication has trend to establish in secure channel.

Acknowledgements

The contents in this thesis were made possible by many people. Firstly, I would like to express my sincere gratitude to my advisor Professor Yasuhiro Nakamura for being excellent mentor and for his guidance, feedback, motivation, immense knowledge and patience throughout my PhD study. I could not have imagined having a better advisor and mentor for my PhD study. Besides my advisor, special thanks to my PhD Assessment Committee members, Professor Hiroshi Yoshiura of The University of Electro-Communications and Associate Professor Hidema Tanaka of National Defense Academy for their insightful comments, valuable discussions and encouragement, but also for the hard questions which invented me to widen my research from various perspectives.

I would like to thank Associate Professor Munetoshi Iwakiri and my fellow lab mates of Software Engineering Laboratory for the stimulating discussions, for the wonderful time we were working together, and for all the fun we have had in the last three years. I am grateful to all the staff members in the Software Laboratory, Electronics and Information Engineering Computer and also the support staffs at National Defense Academy for encouraging and supporting my study through grants and course planning.

Last, but not least, I would like to thank my family and friends for their support. Special thanks to my parents, my parents-in-law, my brothers, my wife (Trang) and my children (Minh and Nam) for supporting me spiritually, allowing me to concentrate throughout producing this thesis and my life in general.

Bibliography

[1] Sarah Gordon, “Fighting Spyware and Adware in the Enterprise,” in Information Security Journal: A Global Perspective, vol. 14, no. 3, pp. 14–17, 2005.

[2] Bartlett, G.; Heidemann, J.; Papadopoulos, C., “Low-rate, flow-level periodicity detection,” in Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, pp.804–809, April 2011.

[3] G. Macia-Fernandez, J. E. Dıaz-Verdejo, P. Garcıa-Teodoro, and F. de Toro-Negro, “LoRDAS: A low-rate DoS attack against application servers,” in Critical Information Infrastructures Security: Second International Workshop, CRITIS 2007, ser. Lecture Notes in Computer Science, J. Lopez and B. M. Hammerli, Eds. Berlin/Heidelberg, Germany: Springer-Verlag, vol. 5141, pp. 197–209, 2008.

[4] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang, “On the Analysis of the Zeus Botnet Crimeware Toolkit,” in Proc. of International Conference on Privacy, Security and Trust, pp. 31–38, August 17-19, 2010.

[5] Nicolas Falliere and Eric Chien, “Zeus: King of the bots,” Symantec Security Response, 2009 Retrived from https://www.symantec.com/content/en/us/enterprise/media/security response/whitepapers/zeus_king_of_bots.pdf, last accessed: November 2016.

[6] P. Royal, M. Halpin, D. Dagon, R. Edmonds, and W. Lee. Polyunpack, “Automating the hidden-code extraction of unpack-executing malware,” In Computer Security Applications Conference, 2006. ACSAC’ 06. 22nd Annual, pages 289–300. IEEE, 2006.

[7] J. Oberheide, E. Cooke, and F. Jahanian., “Cloudav: N-version Antivirus in the Network Cloud,” In Proceedings of the 17th conference on Security symposium, pages 91–106. USENIX Association, 2008.

[8] M. A. Rajab, L. Ballard, N. Lutz, P. Mavrommatis, and N. Provos, “CAMP: Content-agnostic malware protection,” in Network and Distributed Systems Security Symposium (NDSS), Internet Society, 2013.

[9] Silvio Valenti, Dario Rossi, Alberto Dainotti, Antonio Pescapè, Alessandro Finamore, Marco Mellia, “Reviewing Traffic Classification,” in Chapter Data Traffic Monitoring and Analysis, Lecture Notes in Computer Science, Springer Berlin Heidelberg, vol. 7754, pp. 123–147, 2013.

[10] Bermolen, P., Mellia, M., Meo, M., Rossi, D., Valenti, S., “Abacus: Accurate behavioral classification of P2P-TV traffic,” Elsevier Computer Networks, vol. 55, issue 6, pp. 1394-−1411, 25 April 2011.

[11] Laurent Bernaille, Renata Teixeira, and Kave Salamatian, “Early application identification,” in Proceedings of the ACM CoNEXT conference (CoNEXT ’06). ACM, New York, NY, USA, article 6, 12 pages, 2006.

[12] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli, “Traffic classification through simple statistical fingerprinting,” in ACM SIGCOMM Computer Communication Review, vol. 37, issue 1, pp. 5-16, January 2007.

[13] Alberto Dainotti, Antonio Pescapé and Carlo Sansone, “Early Classification of Network Traffic through Multi-classification,” in Chapter Traffic Monitoring and Analysis, Lecture Notes in Computer Science, Springer Berlin Heidelberg, vol. 6613, pp. 122–135, 2011.

[14] Alessandro Finamore, Marco Mellia, Michela Meo and Dario Rossi, “KISS: Stochastic Packet Inspection Classifier for UDP Traffic,” in IEEE/ACM Transactions on Networking, vol. 18, no. 5, pp. 1505–1515, Oct. 2010.

[15] Tom Z. J. Fu, Yan Hu, Xingang Shi, Dah Ming Chiu and John C. S. Lui, “PBS: Periodic Behavioral Spectrum of P2P Applications”, in Chapter Passive and Active Network Measurement, Lecture Notes in Computer Science, Springer Berlin Heidelberg, vol. 5448, pp. 155–164, 2009.

[16] Patrick Haffner, Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang, “ACAS: automated construction of application signatures,” in Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data (MineNet ’05). ACM, New York, NY, USA, pp. 197–202, 2005.

[17] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. “Transport layer identification of P2P traffic,” in Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC ’04). ACM, New York, NY, USA, pp. 121-134, 2004.

[18] Amir R. Khakpour and Alex X. Liu, “High-Speed Flow Nature Identification,” Distributed Computing Systems, 2009. ICDCS ’09. 29th IEEE International Conference on, Montreal, QC, pp. 510–517, 2009.

[19] Hyunchul Kim, KC Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee, “Internet traffic classification demystified: myths, caveats, and the best practices,” In Proceedings of the 2008 ACM CoNEXT Conference (CoNEXT ’08). ACM, New York, NY, USA, , Article 11 , 12 pages, 2008.

[20] Justin Ma, Kirill Levchenko, Christian Kreibich, Stefan Savage, and Geoffrey M. Voelker, “Unexpected means of protocol inference,” in Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (IMC ’06). ACM, New York, NY, USA, pp.313–326, 2006.

[21] Anthony McGregor, Mark Hall, Perry Lorier and James Brunskill, “Flow Clustering Using Machine Learning Techniques,” in Chapter Passive and Ac-

tive Network Measurement, Lecture Notes in Computer Science, Springer Berlin Heidelberg, vol. 3015, pp. 205–214, 2004.

[22] David Moore, Ken Keys, Ryan Koga, Edouard Lagache, and K. C. Claffy, “The CoralReef Software Suite as a Tool for System and Network Administrators,” in Proceedings of the 15th USENIX conference on System administration (LISA ’01). USENIX Association, Berkeley, CA, USA, pp. 133–144, 2001.

[23] Andrew W. Moore and Konstantina Papagiannaki, “Toward the accurate identification of network applications,” in Proceedings of the 6th international conference on Passive and Active Network Measurement (PAM’05), Constantinos Dovrolis (Ed.). Springer-Verlag, Berlin, Heidelberg, pp. 41–54, 2005.

[24] Thuy T.T. Nguyen and Grenville Armitage, “A survey of techniques for internet traffic classification using machine learning,” in IEEE Communications Surveys & Tutorials, vol. 10, no. 4, pp. 56–76, Fourth Quarter 2008.

[25] Vern Paxson, “Bro: a system for detecting network intruders in real-time,” Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 31, issue 23-24, pp. 2435–2463, December 1999.

[26] Matthew Roughan, Subhabrata Sen, Oliver Spatscheck, and Nick Duffield, “Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification,” in Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC ’04). ACM, New York, NY, USA, pp. 135–148, 2004.

[27] Subhabrata Sen, Oliver Spatscheck, and Dongmei Wang, “Accurate, scalable in-network identification of p2p traffic using application signatures,” in Proceedings of the 13th international conference on World Wide Web (WWW ’04). ACM, New York, NY, USA, pp. 512–521, 2004.

[28] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya, "Profiling internet backbone traffic: behavior models and applications," in Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '05). ACM, New York, NY, USA, pp. 169–180, 2005.

[29] IANA, List of assigned port numbers. http://www.iana.org/assignments/port-numbers, last accessed: November 2016.

[30] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy and M. Faloutsos, "Is P2P dying or just hiding? [P2P traffic measurement]," in Proceedings of Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE, vol. 3, pp. 1532–1538, 2004.

[31] Tstat, http://tstat.tlc.polito.it, last accessed: November 2016.

[32] l7filter, Application layer packet classifier for Linux, http://l7-filter.clearfoundation.com/, last accessed: November 2016.

[33] G. Aceto, A. Dainotti, W. de Donato and A. Pescape, "PortLoad: Taking the Best of Two Worlds in Traffic Classification," in Proceedings of INFOCOM IEEE Conference on Computer Communications Workshops, San Diego, CA, pp. 1–5, 2010.

[34] F. Risso, M. Baldi, O. Morandi, A. Baldini and P. Monclus, "Lightweight, Payload-Based Traffic Classification: An Experimental Evaluation," in Proceedings of IEEE International Conference on Communications, Beijing, pp. 5869–5875, 2008.

[35] Wm. A. Wulf and Sally A. McKee, "Hitting the memory wall: implications of the obvious," ACM SIGARCH Computer Architecture News, vol. 23, issue. 1, pp. 20–24, March 1995.

[36] Sailesh Kumar, Sarang Dharmapurikar, Fang Yu, Patrick Crowley, and Jonathan Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," in Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '06). ACM, New York, NY, USA, pp. 339–350, 2006.

[37] Niccolo' Cascarano, Pierluigi Rolando, Fulvio Risso, and Riccardo Sisto, "iNFAnt: NFA pattern matching on GPGPU devices," ACM SIGCOMM Computer Communication Review, vol. 40, issue. 5, pp. 20–26, October 2010.

[38] Muhammad Asim Jamshed, Jihyung Lee, Sangwoo Moon, Insu Yun, Deokjin Kim, Sungryoul Lee, Yung Yi, and KyoungSoo Park, "Kargus: a highly-scalable software-based intrusion detection system," in Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12). ACM, New York, NY, USA, pp. 317–328, 2012.

[39] Y. Liu, D. Xu, L. Sun and D. Liu, "Accurate Traffic Classification with Multi-threaded Processors," in Proceedings of Knowledge Acquisition and Modeling Workshop, 2008. KAM Workshop 2008. IEEE International Symposium on, Wuhan, pp. 478–481, 2008.

[40] G. Szabó, I. Gödor, A. Veres, S. Malomsoky, and S. Molnar, "Traffic classification over gbit speed with commodity hardware," IEEE J. Communications Software and Systems, vol. 5, 2010.

[41] Giorgos Vasiliadis, Michalis Polychronakis, and Sotiris Ioannidis, "MIDeA: a multi-parallel intrusion detection architecture," in Proceedings of the 18th ACM conference on Computer and communications security (CCS '11). ACM, New York, NY, USA, pp. 297–308, 2011.

[42] Yuan Zu, Ming Yang, Zhonghu Xu, Lin Wang, Xin Tian, Kunyang Peng, and Qunfeng Dong, "GPU-based NFA implementation for memory efficient high speed regular expression matching. In Proceedings of the 17th ACM

SIGPLAN symposium on Principles and Practice of Parallel Programming (PPoPP '12). ACM, New York, NY, USA, pp. 129–140, 2012.

[43] A. R. Khakpour and A. X. Liu, "High-Speed Flow Nature Identification," Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on, Montreal, QC, pp. 510–517, 2009.

[44] Valentín Carela-Español, Pere Barlet-Ros, Marc Solé-Simó, Alberto Dainotti, Walter de Donato, Antonio Pescapé, "K-Dimensional Trees for Continuous Traffic Classification," in Traffic Monitoring and Analysis, Lecture Notes in Computer Science, vol. 6003, pp. 141–154, 2010.

[45] A. Dainotti, W. de Donato, A. Pescape and P. Salvo Rossi, "Classification of Network Traffic via Packet-Level Hidden Markov Models," IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference, New Orleans, LO, pp. 1–5, 2008.

[46] A. Dainotti, A. Pescape and H. c. Kim, "Traffic Classification through Joint Distributions of Packet-Level Statistics," in Proceedings of Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, Houston, TX, USA, 2011, pp. 1–6, 2011.

[47] Andrew Moore and Denis Zuev, "Internet traffic classification using bayesian analysis techniques," in Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS '05). ACM, New York, NY, USA, pp. 50–60, 2005.

[48] Nigel Williams, Sebastian Zander, and Grenville Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," ACM SIGCOMM Computer Communication Review, vol. 36, issue. 5, pp. 5–16, October 2006.

[49] Andrew Moore , Michael Crogan , Queen Mary , Denis Zuev , and Michael L. Crogan, "Discriminators for use in flow-based classification," in Technical report RR-05-13, University of Cambridge, 2005.

[50] Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), https://tools.ietf.org/html/rfc3954, October 2004, last accessed: November 2016.

[51] Thomas Karagiannis, Konstantina Papagiannaki, Nina Taft, and Michalis Faloutsos, "Profiling the end host," in Proceedings of the 8th international conference on Passive and active network measurement (PAM'07), Steve Uhlig, Konstantina Papagiannaki, and Olivier Bonaventure (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 186–196, 2007.

[52] Marios Iliofotou, Prashanth Pappu, Michalis Faloutsos, Michael Mitzenmacher, Sumeet Singh, and George Varghese, "Network monitoring using traffic dispersion graphs (tdgs)," in Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07). ACM, New York, NY, USA, pp. 315–320, 2007.

[53] Yu Jin, Nick Duffield, Patrick Haffner, Subhabrata Sen, and Zhi-Li Zhang, "Inferring applications at the network layer using collective traffic statistics," in ACM SIGMETRICS Performance Evaluation Review, vol. 38, issue. 1, pp. 351–352, June 2010.

[54] A. Moser, C. Kruegel and E. Kirda, "Exploring Multiple Execution Paths for Malware Analysis," 2007 IEEE Symposium on Security and Privacy (SP '07), Berkeley, CA, pp. 231–245, 2007.

[55] Probst, Christian W.and Hansen, René Rydhof and Nielson, Flemming, "Where Can an Insider Attack?," Formal Aspects in Security and Trust: Fourth International Workshop, FAST 2006, Hamilton, Ontario, Canada, Springer Berlin Heidelberg, pp 127–142, August 26–27, 2006.

[56] A. Averbuch, M. Kiperberg, and N. Zaidenberg. “An efficient vm-based software protection,” In Network and System Security (NSS), 2011 5th International Conference on, pages 121–128. IEEE, 2011.

[57] Daryl Ashley, “An algorithm for http bot detection,” University of Texas at Austin - Information Security Office, 2011.

[58] J. S. Lee, H. Jeong, J. H. Park, M. Kim and B. N. Noh, “The Activity Analysis of Malicious HTTP-Based Botnets Using Degree of Periodic Repeatability,” Security Technology, 2008. SECTECH ’08. International Conference on, Hainan Island, pp. 83–86, 2008.

[59] Meisam Eslahi, Habibah Hashim and Noorita Tahir, “An Efficient False Alarm Reduction Approach in HTTP-based Botnet Detection,” in Proc. IEEE Symposium on Computers & Informatics, pp. 201–205, April 2013.

[60] Wei Lu, Mahbod Tavallaee, and Ali Akbar Ghorbani, “Automatic discovery of botnet communities on large-scale communication networks,” in Proc. the 4th International Symposium on Information, Computer, and Communications Security, Sydney: Australia, pp. 1–10, 2009.

[61] Basil AsSadhan, Jose M.F. Moura, “An efficient method to detect periodic behavior in botnet traffic by analyzing control plane traffic”, Journal of Advanced Research, vol. 5, issue. 4, pp. 435–448, 2014.

[62] Christian J. Dietrich, Christian Rossow, Norbert Pohlmann, “CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis,” Computer Networks, vol 57, issue 2, pp. 475–486, 4 February 2013.

[63] Sung-Jin Kim, Sungryoul Lee and Byungchul Bae, “HAS-Analyzer: Detecting HTTP-based C&C based on the Analysis of HTTP Activity Sets,” KSII Transactions on Internet and Information Systems, vol. 8, no. 5, May, 2014.

[64] Seungwon Shin, Zhaoyan Xu, and Guofei Gu, "EFFORT: A new host-network cooperated framework for efficient and effective bot malware detection", Computer Networks: The International Journal of Computer and Telecommunications Networking, vol 57 issue 13, pp. 2628–2642, September, 2013.

[65] A. Blum, B. Wardman, T. Solorio, and G. Warner., "Lexical feature based phishing url detection using online learning," InAISec ' 10 Proceedings of the 3rd ACM workshop on Artificial intelligence and security, pp 54−-60, 2010.

[66] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker., "Beyond blacklists: learning to detect malicious web sites from suspicious urls," InProceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 1245-−1254. ACM, 2009.

[67] Teh-Chung Chen, Scott Dick, and James Miller, "Detecting visually similar Web pages: Application to phishing detection," ACM Transaction on Internet Technology, vol. 10, issue 2, article 5, pp. 5:1–5:38, June 2010.

[68] Dilip Singh Sisodia and Shrish Verma, "Web Usage Pattern Analysis Through Web Logs: A Review," in Proc. of the International Joint Conference on Computer Science and Software Engineering (JCSSE), pp.49–53, 2012.

[69] Sung-Hyuk Cha, "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions," International Journal of Mathematical Models and Methods in Applied Sciences, Vol. 1, no. 4, pp.300–307, 2007.

[70] Rui Xu and Donald Wunsch II, "Survey of Clustering Algorithms," IEEE Transactions on Neural Networks, Vol.16, No.3, pp. 645–678, 2005.

[71] Chris Biemann, "Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems," in Proc. of the

First Workshop on Graph Based Methods for Natural Language Processing, Stroudsburg, PA, USA, Association for Computational Linguistics, pp.73–80, 2006.

[72] Amblard, “Which ties to choose? A survey of social networks models for agent-based social simulations,” In Proc. of the SCS International Conference On Artificial Intelligence, Simulation and Planning in High Autonomy Systems, Lisbon, Portugal, pp.253–258, 2002.

[73] G. Gu, R. Perdisci, J. Zhang and W.Lee, “BotMiner: Clustering Analysis of Network Traffic for Protocol and Structure Independent Botnet Detection,” in Proc. the 17th Conference on Security Symposium, San Jose: USA, pp. 139-154, 2008.

[74] K. Tung-Ming, C. Hung-Chang and W. Guo-Quan, “Construction P2P firewall HTTP-Botnet defense mechanism,” in Proc. the IEEE International Conference on Computer Science and Automation Engineering (CSAE), Zhangjiajie: China, pp. 33-39, 2011.

[75] W. T. Strayer, R. Walsh, C. Livadas and D. Lapsley, “Detecting Botnets with Tight Command and Control,” in Proc. the 31st IEEE Conference on Local Computer Networks, Tampa, Florida: USA, pp. 195-202, 2006.

[76] Erwin Kreyszig, “Advanced Engineering Mathematics,” in E-book, 10th ed, Shannon Corliss, Ed. John Wiley & Sons, Inc, ch. 24, pp. 1011–1015, 2011.

[77] MarkLogic database, “What is Marklogic”, Retrieved online from “http://www.marklogic.com/what-is-marklogic/”, last accessed: November 2015.

[78] MarkLogic 8 Product Documentation, Retrieved online from “https://docs.mark logic.com/”, last accessed: October 2015.

[79] Dubuisson, M.-P.; Jain, A.K., “A modified Hausdorff distance for object matching,” in Proc. of the 12th IAPR International Conference, vol.1, pp.566-568, 9–13 Oct 1994.

[80] D. P. Huttenlocher, G. A. Klanderman, and W. J.Rucklidge, “Comparing images using the Hausdorff distance”, IEEE Transaction PAMI, vol. 15, pp. 850–863, 1993.

[81] Yi-Shin Chen; Yi-Hsuan Yu; Huei-Sin Liu; Pang-Chieh Wang, “Detect phishing by checking content consistency,” in Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on, pp.109-119, Aug. 2014.

[82] The PhishTank website, Retrieved online from “https://www.phishtank.com/”, last accessed: June 2016.

[83] Charlie Brooks, “Enterprise NoSQL for Dummies,” in John Wiley & Sons, Inc, Hoboken, 2014.

[84] VirusTotal, Retrieved online from “http://virustotal.com/”, last accessed: October 2016.

[85] McAfee Web Gateway, Retrieved online from “http://www.mcafee.com/us/products/web-gateway.aspx”, last accessed: October 2016.

[86] Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in Proc. of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pp. 226–231, 1996.

[87] MapReduce Tutorial, Apache Hadoop, Retrieved online from “https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html”, last accessed: November 2016.

[88] MarkLogic Connector for Hadoop, Retrieved online from “https://docs.marklogic.com/guide/mapreduce/quickstart”, last accessed: November 2016.

[89] MarkLogic Developer License, “Enterprise NoSQL Power for Developers”, 2008 Retrived online from “https://developer.marklogic.com/free-developer”, last accessed: October 2015.

[90] Manh Cong Tran and Yasuhiro Nakamura, ”Communication Behaviour-based Big Data Application to Classify and Detect HTTP Automated Software,” Journal of Electrical and Computer Engineering (Emerging Sources Citation Index), volume 2016, issue: Innovations in Communications Security, Article ID 2017373, pp.1–11, 2016.

[91] Manh Cong Tran and Yasuhiro Nakamura, “Classification of HTTP Automated Software Communication Behavior Using a NoSQL Database,” IEIE Transactions on Smart Processing & Computing, vol. 5, no. 2, pp. 94–99, April 2016.

[92] Manh Cong Tran and Yasuhiro Nakamura, “In-Host Communication Pattern Observed for Suspicious HTTP-Based Auto-Ware Detection,” International Journal of Computer and Communication Engineering vol. 4, no. 6, pp. 379–389, 2015.

List of Publications

Journal

[1] **Manh Cong Tran** and Yasuhiro Nakamura, ”Communication Behaviour-based Big Data Application to Classify and Detect HTTP Automated Software,” Journal of Electrical and Computer Engineering (Emerging Sources Citation Index), volume 2016, issue: Innovations in Communications Security, Article ID 2017373, pp.1–11, 2016.

[2] **Manh Cong Tran** and Yasuhiro Nakamura, “Classification of HTTP Automated Software Communication Behavior Using a NoSQL Database,” IEIE Transactions on Smart Processing & Computing, vol. 5, no. 2, pp. 94–99, April 2016.

[3] **Manh Cong Tran** and Yasuhiro Nakamura, “In-Host Communication Pattern Observed for Suspicious HTTP-Based Auto-Ware Detection,” International Journal of Computer and Communication Engineering vol. 4, no. 6, pp. 379–389, 2015.

[4] **Manh Cong Tran** and Yasuhiro Nakamura, “A Supplementary Method for Malicious Detection Based on HTTP-Activity Similarity Features,” Journal of Communications, vol. 9, no. 12, pp. 923–929, 2014.

International Conferences

[1] **Manh Cong Tran**, Hai Nam Nguyen, Minh Hieu Nguyen, and Yasuhiro Nakamura, “A Method for Clustering and Identifying HTTP Automated Software Communication,” International Conference on Advances in Information and Communication Technology, Thai Nguyen, Vietnam, December 12–13, 2016, volume 538 of the series Advances in Intelligent Systems and Computing, pp. 53–62, 2016.

[2] Hung Dao Tuan, Hieu Minh Nguyen, **Cong Manh Tran**, Hai Nam Nguyen, Minh Hieu Nguyen, and Moldovyan Nikolay Adreevich, “Integrating Multisignature Scheme into the Group Signature Protocol,” International Conference on Advances in Information and Communication Technology, Thai Nguyen, Vietnam, December 12–13, 2016, volume 538 of the series Advances in Intelligent Systems and Computing, pp. 294–301, 2016.

[3] Sei Kudo, **Manh Cong Tran** and Yasuhiro Nakamura, “Traffic Visualization for Specific Behavior Detection,” The 13th IEEE Transdisciplinary-Oriented Workshop for Emerging Researchers(TOWERS) International Conference on Computing and Communication Technologies, Tokyo, Japan, December 03, 2016.

[4] **Manh Cong Tran**, Sei Kudo and Yasuhiro Nakamura, “Malicious HTTP Communication Detection Based on Access Graph Analysis,” The 12th IEEE-RIVF International Conference on Computing and Communication Technologies, Hanoi, Vietnam, November 07–09, 2016.

[5] **Manh Cong Tran** and Yasuhiro Nakamura, “Non-Human Traffic in HTTP Communication Environment,” The 21th International Defense Seminar, National Defense Academy, Japan, July 04–07, 2016.

[6] **Manh Cong Tran** and Yasuhiro Nakamura, “Behaviour Similarity Based to Cluster Automated HTTP Communication,” The 6th IEEE International

Conference on Communications and Electronics, Halong, Vietnam, pp. 19–24, July 27-29, 2016.

[7] **Manh Cong Tran** and Yasuhiro Nakamura, "Classification of HTTP Automated Software Communication Behaviour Using NoSql Database," The 15th International Conference on Electronics, Information and Communication (ICEIC2016), IEIE&IEEE, Da Nang, Vietnam, pp. 1–4, January 27-30, 2016.

[8] **Manh Cong Tran** and Yasuhiro Nakamura, "Web Access Behaviour Model for Filtering Out HTTP Automated Software Accessed Domain", The 10th ACM International Conference on Ubiquitous Information Management and Communication (ACM IMCOM 2016), Da Nang, Vietnam, January 04-06, 2016.

[9] **Manh Cong Tran** and Yasuhiro Nakamura, "An Approach in Suspicious Domain Mining Based on HTTP Auto-ware Communication Behavior", The Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic, and Digital Forensic (CyberSec 2015), Jakarta, Indonesia, Sampoerna University, pp.1–5, October 29-31, 2015.

[10] **Manh Cong Tran** and Yasuhiro Nakamura, "Suspicious Domain Filtering Based on Auto-ware Communication Features," The 30th International Technical Conference on Circuits/Systems, Computers and Communications, IEIE&IEICE, pp. 620-622, Seoul, Korea, June 29-July 02, 2015.

[11] **Manh Cong Tran** and Yasuhiro Nakamura, "In-Host Communication Pattern Observed for Suspicious HTTP-Based Auto-Ware Detection," The 7th International Conference on Computer Research and Development, Ho Chi Minh city, Vietnam, February 6-7, 2015.

[12] **Manh Cong Tran** and Yasuhiro Nakamura, "A Supplementary Method for Malicious Detection Based on HTTP-Activity Similarity Features,", The 5th

International Conference on Networking and Information Technology ICNIT 2014, Singapore, November 21-23, 2014.

[13] **Manh Cong Tran** and Yasuhiro Nakamura, "Abnormal Web Traffic Detection Using Connection Graph," The First International Symposium on Computing and Networking, IPSJ&IEICE, NCSS Workshop Poster, Matsuyama, Ehime, Japan, December 4-6, 2013, BNCSS vol. 3, no. 1, pp. 57–62, January 2014.

Domestic Conferences

[1] 工藤　聖，マン・トラン・コン，中村康弘，"HTTP リクエストシーケンスに注目した Drive-By Download 検知手法," 情報処理学会第 78 回全国大会 2016, 慶應義塾大学矢上キャンパス２０１６年０３月１０日～１２日.

[2] **Manh Cong Tran** and Yasuhiro Nakamura, "A HTTP Auto-ware Communication Behavior Based Approach to Mine Suspicious Domains," コンピュータセキュリティシンポジウム 2015, IWSEC of IPSJ&IEICE , ポスター, 長崎ブリックホール２０１５年１０月２１日～２３日.

[3] 工藤　聖,　マン・トラン・コン, 中村　康弘, "HTTP リクエストシーケンスに注目した不正リダイレクトの検出,"　コンピュータセキュリティシンポジウム 2015, pp. 221-225, 長崎ブリックホール２０１５年１０月２１日～２３日.

[4] **Manh Cong Tran** and Yasuhiro Nakamura, "A Study of Malicious HTTP-based Auto-ware Identification Using Traffic Features," 情報処理学会第 77 回全国大会, 5E-01, pp.3-445-3-446, 京都大学　吉田キャンパス, ２０１５年３月１７日～１９日.

[5] **Manh Cong Tran**, Chan Sambathratanak and Yasuhiro Nakamura, "Access Similarity in Analyzing Non-Human HTTP Traffic," 日本セキュリティ・マ

ネジメント学会 (JSSM), 第２８回全国大会発表要旨, 東京工科大学　八王子キャンパス, pp. 135-138, ２０１４年６月２１日.

[6] Chan Sambathratanak, **Manh Cong Tran** and Yasuhiro Nakamura, “Exploring Cross-Site Scripting Attacks and Possible Defense Methods,” 日本セキュリティ・マネジメント学会 (JSSM), 第２８回全国大会発表要旨, 東京工科大学　八王子キャンパス, pp. 129-133, ２０１４年０６月２１日.

[7] 後藤　洋一, マン・トラン・コン,　中村　康弘, “ダークネット観測結果に基づく攻撃元アドレスの傾向分析手法の提案,” 2014 年　暗号と情報セキュリティシンポジウム, 2C2-1, 鹿児島県鹿児島市, ２０１４年０１月２１日～２４日.

Others

[1] **Manh Cong Tran** and Yasuhiro Nakamura, “A Study of Application Layer Slow-rate Denial of Service Attacks,” A commentary on 日本セキュリティ・マネジメント学会誌 (JSSM), ISSN 134-6619, vol. 29, no1, pp. 33-37, ２０１５年０５月.

Adward

[1] Best Oral Presentation, the 7th International Conference on Computer Research and Development, Ho Chi Minh city, Vietnam, February 6-7, 2015.