

Traffic characterization and classification for web applications



Student: Chiu, Yen-Chun

Advisor: Prof. Lin, Po-Ching

National Chung Cheng University

Ming-Hsiung, Chiayi 621, Taiwan

shuaichiou@gmail.com

Abstract

Software defined network (SDN) is a next-generation concept that allows network administrator to manage network flows with ease. It is programmable, centrally managed, and being able to adapt to the change of topology. Meanwhile, these characteristics also leads to new security problems, which some have already been discussed and proven. And people also give out great protection mechanisms and suggestions. However, there are still more possibilities of attacks in different scenario left to be discovered. And as SDN evolves, it will definitely cause some more new issues. In this paper, we analsis several types of topology-related attacks, and create a method that is able to detect topology poisoning attacks by using the standard functionality of OpenFlow switch. We evaluate the effectiveness of our method under different conditions, and discuss the possibility of future works.

Contents

1	Introduction	1
2	Background and related work	5
2.1	Software-defined network	5
2.2	OpenFlow structure	7
2.3	SDN security	11
2.3.1	Controller	11
2.3.2	Host	11
2.3.3	Switch	11
2.3.4	Hardening strategies	11
2.4	16
3	Method of Web Traffic Classification	18
3.1	Data collection	19
3.1.1	Extract statistical signature	20

3.1.2	Feature definition	24
3.2	System Workflow	26
4	Traffic Characterization and Classification Evaluation	28
4.1	Scenarios of Packet Collection	28
4.2	Traffic Characterization of Web Applications	29
4.3	Feature analysis	31
4.3.1	Message size distribution	31
4.4	Classification results	34
4.4.1	Classification with Similar Interactions in Each Run . .	35
4.4.2	Classification with Diverse Interactions in Each Run . .	38
4.4.3	Classification with Interactions from Multiple Users . .	39
4.4.4	Early Classification	40
4.5	Practice and Limitation	44
5	Conclusion	46

List of Figures

3.1	System workflow of classification.	26
4.1	The message size distribution of each web application from two browsers.	32
4.2	Categorization of web applications.	33
4.3	The message size distribution for Dungeon Rampage.	42
4.4	The message size distribution for Google map.	43

List of Tables

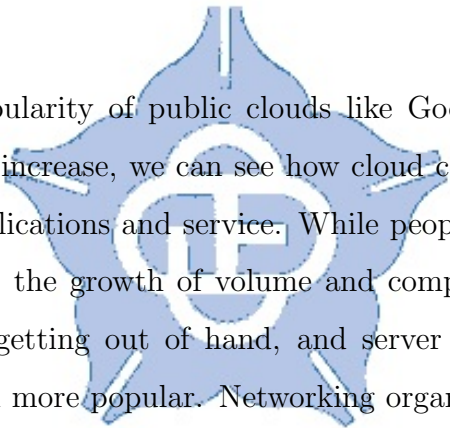
2.1	Pros and cons of each classification.	17
3.1	The scenario of each web application.	21
3.2	The judgment of the main connection.	23
3.4	A feature of editing by Google document.	25
3.5	A feature of downloading by Dailymotion.	25
3.6	A feature of downloading by Dropbox.	25
3.3	The contents of other connections	27
4.1	The average number of connections and the standard deviation (SD) for each web application.	30
4.2	True/false positive rates of classifying the interactive connec- tions.	35
4.3	True/false positive rates of classifying the interactive connec- tions with additional web applications.	36
4.4	True/false positive rates of classifying downloading connections.	37

4.5	True/false positive rates of classifying all connections.	39
4.6	True/false positive rates of classifying the interactive connections for multiple users.	40
4.7	Early classification true/false positive rate in all connections. .	43



Chapter 1

Introduction



Nowaday, as the popularity of public clouds like Google cloud, Microsoft Azure, Amazon EC2 increase, we can see how cloud computing offer a new way of deploying applications and service. While people try to move everything onto the cloud, the growth of volume and complexity of data center network (DCN) are getting out of hand, and server virtualization is also becomming more and more popular. Networking organization are under increasing pressure to be more efficient, agile, and maintainable than is possible with the traditional approach to networking.

In traditional network, most of the network functionality heavily rely on hardware because they are implemented in network devices. They are under the control of manufacturers. As a result, the evolution of those functionality will be limited. Furthurmore, implementing a network-wide policy requires configuring at the device-level. And similarly, tasks such as provisioning, change management, and de-provisioning are also very time-consuming, error-prone and require a lot of manwork. It will surely be a

formidable job for network administrators to manage a large scale network with traditional network. Scalability is also a problem, link oversubscription is no longer reliable enough to keep up with the growing demand on data centers [20].

Software defined network (SDN) is a dynamic, manageable, cost-effective, and adaptable network network structure. It gains great popularity among enterprises as well as academia recent years. One key concept of SDN is separating the control plane from data plane. It is also centrally contrrollable by software application. Comparing to lagacy network, it is more flexible, maintainable and agile. Nevertheless, new technology often comes with new security problems. In addition to switches and hosts, which of course are potential targets for hackers, SDN uses controllers to realize centralized control. If the applications in controllers are compromised, the whole network might fall into attacker's hand [6]. We will talk more about these issues in later chapters.

Another crucial element of SDN controller is the topology service. It is implemented in OpenFlow, a protocol for the control plane to communicate with the data plane. The topology-related services like host tracking services and link discovery service provides a manner to adjust to the modification of network topology automatically. Nevertheless, some parts of it is still not very mature and has proven to be insecure [5]. It may lead to Man In The Middle attack or Deny of Service in a similar way to how it could happen in regular network [1]. In this thesis, we will be focusing on those topology-related issues.

The motivation of our work is that, although some work has been done to deal with all those security problems in SDN [8,9], some aspect are missing.

4. Evaluate our method under different conditions.

The following chapters in this thesis will be: Chapter 2 gives detail background knowledge of the used technology and describes possible threats mentioned by previous work and discusses other possibilities of attacks. Also, it will mention some related work of countermeasurements. —————

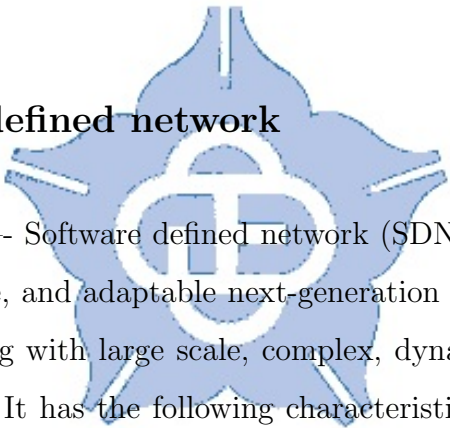
————— Chapter 3 tells about our threat model and the theory of our own detection method. Chapter 4 contains the experimental details including setup, considerations, simulations of attacks and evaluation methods. In Chapter 5, the proposed method will be evaluated under different conditions. Finally, the conclusion and future expectation of this work will be in Chapter 6.



Chapter 2

Background and related work

2.1 Software-defined network



————— Software defined network (SDN) is a dynamic, manageable, cost-effective, and adaptable next-generation network structure. It is designed for dealing with large scale, complex, dynamic data center network existing today. It has the following characteristics: (1) Software programmable: One can configure, manage, and optimize network resources very quickly via dynamic, automated SDN programs. (2) Agile: Abstracting control from forwarding lets administrators dynamically adjust network-wide traffic flow to meet changing needs. (3) Centralized Policy Control: Network intelligence is centralized in software-based SDN controllers that maintain a global view of the network (4) Manageability: Due to network programmability, companies can develop new applications that can further improve manageability, integration, communication and so on. (5) Open standards-based and vendor-neutral: SDN simplifies network design and operation because instructions are provided by SDN controllers instead of multiple, vendor-

specific devices and protocols [18]. To sum up, SDN provides greater visibility into network, reduces manual intervention, improves maintainability, and requires less hardware budget.

SDN Application (SDN App) SDN Applications are programs that explicitly, directly, and programmatically communicate their network requirements and desired network behavior to the SDN Controller via a northbound interface (NBI). In addition they may consume an abstracted view of the network for their internal decision making purposes. An SDN Application consists of one SDN Application Logic and one or more NBI Drivers. SDN Applications may themselves expose another layer of abstracted network control, thus offering one or more higher-level NBIs through respective NBI agents.

SDN Controller The SDN Controller is a logically centralized entity in charge of (i) translating the requirements from the SDN Application layer down to the SDN Datapaths and (ii) providing the SDN Applications with an abstract view of the network (which may include statistics and events). An SDN Controller consists of one or more NBI Agents, the SDN Control Logic, and the Control to Data-Plane Interface (CDPI) driver. Definition as a logically centralized entity neither prescribes nor precludes implementation details such as the federation of multiple controllers, the hierarchical connection of controllers, communication interfaces between controllers, nor virtualization or slicing of network resources.

SDN Datapath The SDN Datapath is a logical network device that exposes visibility and uncontended control over its advertised forwarding and data processing capabilities. The logical representation may encompass all or a subset of the physical substrate resources. An SDN Datapath comprises a CDPI agent and a set of one or more traffic forwarding engines and zero or more traffic processing functions. These engines and functions may include simple forwarding between the datapath's external in-

interfaces or internal traffic processing or termination functions. One or more SDN Datapaths may be contained in a single (physical) network element, an integrated physical combination of communications resources, managed as a unit. An SDN Datapath may also be defined across multiple physical network elements. This logical definition neither prescribes nor precludes implementation details such as the logical to physical mapping, management of shared physical resources, virtualization or slicing of the SDN Datapath, interoperability with non-SDN networking, nor the data processing functionality, which can include L4-7 functions.

SDN Control to Data-Plane Interface (CDPI) The SDN CDPI is the interface defined between an SDN Controller and an SDN Datapath, which provides at least (i) programmatic control of all forwarding operations, (ii) capabilities advertisement, (iii) statistics reporting, and (iv) event notification. One value of SDN lies in the expectation that the CDPI is implemented in an open, vendor-neutral and interoperable way.

SDN Northbound Interfaces (NBI) SDN NBIs are interfaces between SDN Applications and SDN Controllers and typically provide abstract network views and enable direct expression of network behavior and requirements. This may occur at any level of abstraction (latitude) and across different sets of functionality (longitude). One value of SDN lies in the expectation that these interfaces are implemented in an open, vendor-neutral and interoperable way.

2.2 OpenFlow structure

It enables network controllers to determine the path of packets by managing forwarding tables in the switches [16]. In this paper, we will be using Open vSwitch, a software implementation of virtual multilayer network

switch that supports OpenFlow protocol.

Another expanding set of classification techniques are based on machine learning with statistical features of network traffic. The studies in [30, 31] classify the behavior of an application by the size and the direction of the first few packets of a TCP connection. Bernaille et al. [30] defines the behavior of an application as the size and direction of the first four packets it exchanges over a TCP connection. An et al. [31] uses the first five packets and converts the flow into a vector. The paper shows the possibility of classifying traffic using the payload size distribution of the packets; nevertheless, this method classifies all application traffic, except HTTP.

Some studies [32, 37, 38] extract the features from a complete flow. Common features are port numbers, average segment size, internal time of packets, and so on. [32] uses a Fast Correlation-Based Filter (FCBF) to choose the essential feature, which can reduce the processing time.

Furthermore, some studies [32–36] used clustering techniques in machine learning to classify traffic. [33, 34] are based on Naive Bayes algorithm, a supervised machine learning algorithm based on Bayesian theory. Artificial Immune System (AIS) algorithm is proven to be very versatile and low sensitivity to input parameters, so [32] used it as their base. The study implemented the features as hyper spheres within an 11-dimensional space, and if an example vector falls within the distance of the hyper sphere, then the example is classified to the same class as the feature belong to. The rest of studies are based on k-Nearest Neighbor (KNN) estimator [35] and on support vector machine (SVM) [36]. Machine-learning-based classification can automatically build a model from training data and reduce much artificial input; however, the classification will decrease because they only collect and

classify traffic in a limited range. The limitation is a challenge. It is difficult to identify all protocols by using only a machine learning model because most models are binary classification, which means one model just can classify one protocol.

As enterprises look to adopt Software Defined Networking (SDN), the top of mind issue is the concern for security. Enterprises want to know how SDN products will assure them that their applications, data and infrastructure will not be vulnerable. With the introduction of SDN, new strategies for securing the control plane traffic are needed. This article will review the attack vectors of SDN systems and share ways to secure the SDN-enabled virtualized network infrastructure. This article will then discuss the methods currently being considered to secure SDN deployments.

1. SDN Attack Vectors

Software-Defined Networking (SDN) is an approach to networking that separates the control plane from the forwarding plane to support virtualization. SDN is a new paradigm for network virtualization. Most SDN architecture models have three layers: a lower layer of SDN-capable network devices, a middle layer of SDN controller(s), and a higher layer that includes the applications and services that request or configure the SDN. Even though many SDN systems are relatively new and SDN is still in the realm of the early adopters, we can be sure, that as the technology matures and is more widely deployed, it will become a target for attackers.

We can anticipate several attack vectors on SDN systems. The more common SDN security concerns include attacks at the various SDN architecture layers. Lets look at the anticipated attacks that could occur at each of these layers. Following is a picture to illustrate a typical SDN architecture and

where attackers may be coming from. 1-1. Attacks at Data Plane Layer

Attackers could target the network elements from within the network itself. An attacker could theoretically gain unauthorized physical or virtual access to the network or compromise a host that is already connected to the SDN and then try to perform attacks to destabilize the network elements. This could be a type of Denial of Service (DoS) attack or it could be a type of fuzzing attack to try to attack the network elements.

There are numerous southbound APIs and protocols used for the controller to communicate with the network elements. These SDN southbound communications could use OpenFlow (OF), Open vSwitch Database Management Protocol (OVSDB), Path Computation Element Communication Protocol (PCEP), Interface to the Routing System (I2RS), BGP-LS, OpenStack Neutron, Open Management Infrastructure (OMI), Puppet, Chef, Diameter, Radius, NETCONF, Extensible Messaging and Presence Protocol (XMPP), Locator/ID Separation Protocol (LISP), Simple Network Management Protocol (SNMP), CLI, Embedded Event Manager (EEM), Cisco onePK, Application Centric Infrastructure (ACI), Opflex, among others. Each of these protocols has their own methods of securing the communications to network elements. However, many of these protocols are very new and implementers may not have set them up in the most secure way possible.

An attacker could also leverage these protocols and attempt to instantiate new flows into the devices flow-table. The attacker would want to try to spoof new flows to permit specific types of traffic that should be disallowed across the network. If an attacker could create a flow that bypasses the traffic steering that guides traffic through a firewall the attacker would have a decided advantage. If the attacker can steer traffic in their direction, they

may try to leverage that capability to sniff traffic and perform a Man in the Middle (MITM) attack. 1-2. Attacks at Controller Layer

It is obvious that the SDN controller is an attack target. An attacker would try to target the SDN controller for several purposes. The attacker would want to instantiate new flows by either spoofing northbound API messages or spoofing southbound messages toward the network devices. If an attacker can successfully spoof flows from the legitimate controller then the attacker would have the ability to allow traffic to flow across the SDN at their will and possibly bypass policies that may be relied on for security.

An attacker might try to perform a DoS of the controller or use another method to cause the controller to fail. The attacker might try to attempt some form of resource consumption attack on the controller to bog it down and cause it to respond extremely slowly to Packet_In events and make it slow to send Packet_Out messages.

2.3 SDN security

2.3.1 Controller

2.3.2 Host

2.3.3 Switch

2.3.4 Hardening strategies

Lastly, it would be bad if an attacker created their own controller and got network elements to believe flows from the rogue controller. The attacker

could then create entries in the flow tables of the network elements and the SDN engineers would not have visibility to those flows from the perspective of the production controller. In this case, the attacker would have complete control of the network.

1-3. Attacks at SDN Layer

Attacking the security of the northbound protocol would also be a likely vector. There are many northbound APIs that are used by SDN controllers. Northbound APIs could use Python, Java, C, REST, XML, JSON, among others. If the attacker could leverage the vulnerable northbound API, then the attacker would have control over the SDN network through the controller. If the controller lacked any form of security for the northbound API, then the attacker might be able to create their own SDN policies and thus gain control of the SDN environment.

Often times, there is a default password that is used for a REST API which is trivial to determine. If an SDN deployment didn't change this default password and the attacker could create packets toward the controllers management interface, then the attacker could query the configuration of the SDN environment and put in their own configuration.

2. Hardening an SDN System

With the introduction of SDN, a new method is needed for securing the control plane traffic. In traditional IP networks the control plane security came in the form of routing protocol security measures that involved using MD5 for EIGRP, IS-IS, or OSPFv2, IPsec AH in the case of OSPFv3, or GTSM/ACLs/passwords for MP-BGP. Some implementers do not even follow these simple techniques for traditional IP networks. If they approach deployment of an SDN with the same disregard for security, then they will

be exposing their organization to attacks. Lets look at how one can secure an SDN system based on hardening the three layers illustrated in the above architecture diagram.

2-1. Securing the Data Plane Layer

Typical SDN systems leverage X86 processors and use TLS (formerly SSL) for the security of the control plane. These long-lived HTTP sessions are susceptible to a wide range of attacks that could jeopardize the integrity of the data plane. This would bypass the multi-tenancy of these solutions and cause cloud-based services to be compromised. Organizations should prefer to use TLS to authenticate and encrypt traffic between network device agent and controller. Using TLS helps to authenticate controller and network devices/SDN agent and avoid eavesdropping and spoofed southbound communications.

Depending on the southbound protocol being used, there may be options to secure this communications. Some protocols may be used within TLS sessions as previously mentioned. Other protocols may use shared-secret passwords and/or nonce to prevent replay attacks. Protocols like SNMPv3 offer more security than SNMPv2c and SSH is far better than Telnet. Other proprietary southbound protocols may have their own methods to authenticate network device agents and controllers and encrypt data between themselves, thus thwarting the attackers eavesdropping and spoofing.

Similarly, depending on the Data Center Interconnect (DCI) protocol being used, there may be configurable options to authenticate tunnel endpoints and secure tunneled traffic. Again, passwords/shared-secrets might be an option. However, some DCI protocols may not have any option for security.

Organizations may believe that a private network has certain inherent security. As organizations extend their virtual networks and SDNs to cloud services and to remote data centers, verifying the physical path may not be so easy. Preventing unauthorized access is easier when an organization controls the physical access, but as networks virtualize, the actual physical path gets a little murky. It is difficult to secure what you cant see.

2-2. Securing the Controller Layer

The controller is a key attack target and therefore, it must be hardened. Hardening the security posture of the controller and the network elements typical comes down to host OS hardening. All the best practices for hardening public-facing Linux servers are applicable here. Still, organizations will want to closely monitor their controllers for any suspicious activity.

Organizations will also want to prevent unauthorized access to SDN control network. SDN systems should allow for configuration of secure and authenticated administrator access to controller. Even Role-Based Access Control (RBAC) policies may be required for controller administrators. Logging and audit trails could be useful for checking for unauthorized changes by administrators or attackers alike.

If there is a DoS attack of the controller, then it is beneficial to have a High-Availability (HA) controller architecture. SDNs that use redundant controllers could suffer the loss of a controller and continue to function. This would raise the bar for an attacker trying to DoS all the controllers in the system. Besides, that attack would not be particularly stealthy and further the attackers aims of remaining undetected.

2-3. Securing the SDN Layer

Another protection measure is to use an Out-of-Band (OOB) network for control traffic. It is easier and less costly to construct an OOB network in a data center than across an enterprise WAN. Using an OOB network for the northbound and southbound communications could help secure the protocols for controller management.

Using TLS or SSH or another method to secure northbound communications and secure controller management would be considered a best practice. The communications from the applications and services requesting services or data from the controller should be secured using authentication and encryption methods.

Secure coding practices for all northbound applications requesting SDN resources should be a best practice. Not only are secure coding practices beneficial to the security of public-facing Internet web applications, but they are also applicable to northbound SDN connections.

There are some SDN systems that have the ability to validate flows in network device tables against controller policy. This type of checking (similar to FlowChecker) of the flows in the network devices against the policy in the policy could help identify discrepancies that are the result of an attack.

3. Summary

We can only try to anticipate what the attackers may try to target with SDNs. The deployments are new, the protocols are new, the controller software is new, and the history of past SDN attacks is unknown. Based on the SDN architecture, we can predict where an attacker may be likely to strike. If we put ourselves in the attackers shoes, we might be able to spot a weakness to exploit. Then we can harden that weakness ahead of time.

Before an organization embarks on an SDN deployment project, they should consider how they will secure the system during the early design stage. Dont leave security until the final clean-up phase. If an organization waits until it is working, then hardening the northbound and southbound control messages may cause service-affecting problems. Like most things, setting it up right from the start will save organizations many problems down the road. [19]

2.4

The main idea behind fine-grained traffic classification is how to categorize application traffic into different traffic groups, and it has some advantages to improve the network security and the QoS management. There are some instances below:

S. Valenti et al. presented [39], which use the Abacus behavioral classification engine and SVM to categorize the P2P traffic. They apply the traces of the six P2P applications, ranging from file-sharing to VoIP and live-streaming services, such as SopCast, Skype and BitTorrent. The research use Abacus classifier by only relying on the count of packets and bytes peers exchange during fixed-length time-windows, but the accuracy would be decreased when the vantage points moves from the edge.

In [40], the authors select P2P applications for verification since the various behavior can strongly represents the complexity of Internet applications. P2P applications provide many functions: web browsing, searching, downloading, messenger and commercial advertisement. This paper uses Fileguri and BitTorrent to generate the flows, and creates the signatures by fine-

grained classifier and LASER. They collected 450Gbytes from their campus network, then transform each packet into a vector, and exploited Jaccard for clustering.

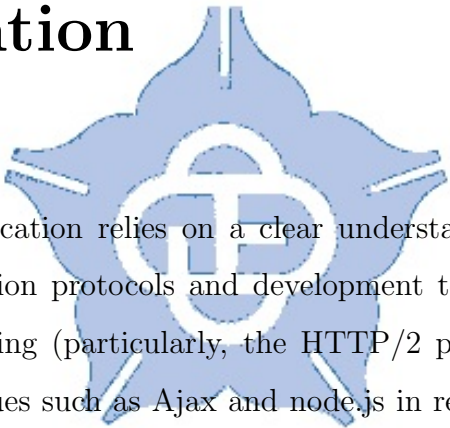
Table 2.1: Pros and cons of each classification.

Classification	Features	Pros and Cons	Used in studies
Port-based	well-known port numbers	simply and fast but can't be used in the tunnelling traffic	[25]
Payload-based	string-based matching, regular expression matching and IP protocol	accurate classification capability and easy development but slow and can't be used in encrypted traffic	[27] , 17-filter
flow-based	packet size, packet inter-arrival time, flow duration, packet numbers,etc.	can detect the class of yet unknown applications but cannot deal with multiple transport transport protocols simultaneously	[30], [31], [32], [37], [38], [32–36]

Chapter 3

Method of Web Traffic

Classification



Precise traffic classification relies on a clear understanding of web traffic. However, as application protocols and development techniques of web applications keep evolving (particularly, the HTTP/2 protocol [49] and new development techniques such as Ajax and node.js in recent years), the traffic characterization of web applications in prior studies may not reflect the state-of-the-art. Therefore, it is necessary to re-examine the traffic characteristics from typical web applications, and find out effective features for precise classification. Typical features in traffic characterization usually include application payloads, packet size, connection length and duration, packet inter-arrival time, and so on. We argued in [24] that the features may be unstable due to the variations of Internet traffic. In this work, we focus primarily on the application-level features that are not subject to the conditions in underlying networks. We study the traffic characteristics in popular web applications, including office applications (Google document), maps (Google

maps), file sharing (Google drive and dropbox), video streaming (Youtube, Dailymotion, Tudou), and online games (Tetris battle and Dungeon Rampage on Facebook), in terms of the features on various browsers. We then extract the features from the main connection described in next section, so we do not need to compute the average and the standard deviation of the request/response lengths like [24].

3.1 Data collection

A web application usually involves highly user interactions between the browser and the web server. Developers often use techniques such as Ajax to improve user experiences, e.g., by actively pushing web content to the browser before the user requests it. Moreover, new application protocols, particularly SPDY (www.chromium.org/spdy) primarily developed at Google and HTTP/2 based on SPDY, support features to reduce the latency of loading web pages for efficient web browsing. Major browsers such as IE, Firefox and Chrome have supported SPDY and HTTP/2, and most of them have enabled the support by default at the time of this thesis writing. The web traffic from the Google services covered in this work is all sent over SPDY. However, the traffic from the other web applications (see Table 3.1) also use SPDY, except that from Facebook, Dailymotion and Tudou. Thus, the traffic studied in this work reflect the latest status in the development of web applications.

Since a browser may keep prior web content in a local cache to speed up web browsing, we use the guest mode of a browser (in which the web content in prior browsing will not be preserved) when interacting with a web application to ensure a complete set of packets during the interaction can be

collected. We browse only a specific web application at a time to ensure the web traffic is all from that application, and use Wireshark (www.wireshark.org) to collect the packets.

In this work, we consider several typical scenarios of using web applications, and capture the web traffic from them. This work covers totally five types of 9 web applications, as listed in Table 3.1. We implore users to operate each web application on either Chrome or Firefox in the scenarios described in this table. A user is requested to interact with the applications for a period from one to two minutes as usual. We do not use Internet Explorer for several reasons. First, the browser is not supported on many operating systems, e.g., Mac OS and Android [50]. Second, the usage of IE is reported to be dropped to only 7.1% [51]. Third, Windows 10 no longer supports IE.

3.1.1 Extract statistical signature

A web application server may offer two or more services, so identifying the application based solely on the server's IP address is unreliable. For example, Google offers all its services on the same back-end server infrastructure (e.g., Google document and Google map can be provided by the same IP address 74.125.23.102 in our observation.), so users can reuse existing TCP connections to Google servers to access the other services [52]. Thus, classification with statistical features to distinguish the traffic from various web applications is necessary.

There are several requirements for the training packet traces to acquire precise application-level statistical features:

1. The collected packet traces must be generated only from the targeted

Table 3.1: The scenario of each web application.

no.	type	application	scenario
1	Document	Googledoc	arbitrarily typing and editing
2	Map	Googlemap	typing a location name, arbitrarily browsing the map and zooming-in and zooming-out
3-5	Video	Youtube/ Tudou/ Dailymotion	typing a video name and arbitrarily moving to a specific time position during watching
6-7	File transfer	Google drive/ Dropbox	up/downloading
8-9	Gaming	Facebook : Tetris Battle/ Dungeon Rampage	arbitrary operation

web application. We set the filter of Wireshark to capture the packets from or to port 80 and 443, and ask the user runs only a web application on the browser for every time of packet collection.

2. The packets in the beginning of connections should be preserved for TCP state tracking, which is necessary for packet reassembly to recover the application-level features.
3. The collected traffic should be sufficient and diverse because it may affect the accuracy and reliability. The collected traffic must contain the packet traces from the user interactions from the targeted web application. Moreover, we performed various interactions on a same

web application for every collection to ensure the diversity of packet traces. For example, we changed the frequency of typing extremely every time when we collected the traffic from Google document.

We set the target IP address and ports and follow the scenarios listed in Table 3.1 to run the web applications when collecting the training packet traces. The quantity of collected traffic is important to reflect practical user behavior precisely. We take advantage of this to observe whether diverse user behaviors will impact on the classification or the characterization. Not only the user interactions, but also the environment will influence the results. For example, some browsers support the SPDY protocol, but Mozilla Firefox did not until 2011. The difference will influence the patterns of some flows when the packets are carried over different protocols.

After collecting a new set of packet traces for a web application in the training set, we find the main connection by using the developer tool affiliated to the browser (e.g., Firebug on Firefox), which allows developers to view information about the transmitted messages. Hence we confirm which connection is the most representative of the user interactions by referring to the parameters in the HTTP messages summarized in Table 3.2. Nevertheless, many simultaneous connections may load web information in Table 3.3 to provide smooth user experiences, so we choose the longest one as the main connection. The only exception is the main connection for Tetris Battle, which transfers the elements related to the game platform instead of the game interactions. Thus, our mechanism takes advantage of this property to decide the main connection when a set of packet traces are analyzed. We can effectively segregate the noise, such as embedded advertising, with the method. This work uses the `libnids` library (libnids.sourceforge.net),

which offers IP defragmentation, TCP stream reassembly and TCP port scan detection, to reconstruct the request-response streams of all the connections. The tool needs to track the states of TCP connections, so the beginning time of packet capturing is crucial to ensure important state transitions, e.g., 3-way handshaking, are not missing. We extract the source/destination IP addresses and source/destination ports to recognize each flow.

Table 3.2: The judgment of the main connection.

	Referred parameter	Judgment
Document	bundles	contain some arrays to store the characters we typed
Map	content-type	(1)image/png : comparing whether the preview picture and the map showed on the website are the same or not (2)application/vnd.google.octet-stream-compressible;charset=x-user-defined : this message continuously appear as we arbitrarily send actions to the map
Video	content-type	For example, audio/mp4, video/webm and video/f4v.
File transfer	content-disposition content-type	(1)checking the filename in the content-disposition match with the file we transfer (2)comparing the content-type match with the file we transfer
Game	content-type	the content-type is application/x-shockwave-flash game applications always use .swf to transfer files

3.1.2 Feature definition

We assume that f_m is the main connection when a user interacts with a certain web application, and extract all n messages to be analyzed (after processed by `libnids`). Let s_i be the i -th message size in the request connection, where $i = 1 \dots n$. We do not adopt the features in the bi-direction because including the features from the responses will introduce more ambiguity between some web applications. For example, the features generated from file download applications are sometimes similar to those from games and maps, and will decrease accuracy by 6% according to our preliminary experiment (not shown in Chapter 4). This work counts the occurrence frequency of each message size, and takes the top five frequent sizes to be represented as a vector v_i for every main connection.

Table 3.4 presents an example, in which we arbitrarily typed in a Google document on Chrome for around two minutes. Thus, the feature to characterize this interaction is (38, 34, 220, 224, 116). However, the number of different message sizes may be less than five to form a five-tuple vector. We take two ways to solve this problem. Table 3.5 is generated by watching a video clip on Dailymotion with Chrome. It describes that there is a vacancy within the vector. We can fill up the missing tuple with the mode (i.e, the most common value) of the other message sizes [53], and the vector becomes (636, 665, 658, 589, 636). Table 3.6 is generated by downloading a file from Dropbox on Firefox. The occurrence time of all message sizes are the same in this table. We choose the first one to fill up the vacancies, and the vector is (753, 202, 162, 753, 753). The message sizes mentioned in these three tables are sorted by the occurrence times in the decreasing order. The features are stable and the testing results are showed in Section 4.4.

Table 3.4: A feature of editing by Google document.

	s_1	s_2	s_3	s_4	s_5	s_6	...	s_n
Message size (bytes)	38	34	220	224	116	199	...	m_n
Count	177	177	37	27	13	12	...	c_n

The vector is (38, 34, 220, 224, 116).

Table 3.5: A feature of downloading by Dailymotion.

	s_1	s_2	s_3	s_4	s_5
Message size (bytes)	636	665	658	589	
Count	4	3	2	1	

The vector is (636, 665, 658, 589, 636).

Table 3.6: A feature of downloading by Dropbox.

	s_1	s_2	s_3	s_4	s_5
Message size (bytes)	753	202	162		
Count	1	1	1		

The vector is (753, 202, 172, 753, 753).

3.2 System Workflow

Figure 3.1 illustrates the workflow of the classification process. A user may use either Mozilla Firefox or Google Chrome to run a web application as mentioned above. Simultaneously, the capture filter for the web application traffic is set to port 80 and 443 on the PC. The developer tool and Wireshark is employed to find the main connection that reflects user interactions most. Next, `libnids` will reassemble the captured packets to extract application-level messages. In the last step, these features extracted from the messages are fed into Weka (<http://www.cs.waikato.ac.nz/ml/weka>) for traffic classification with four machine learning methods: NBTree, Random Forest, J48graft and Naive Bayes.

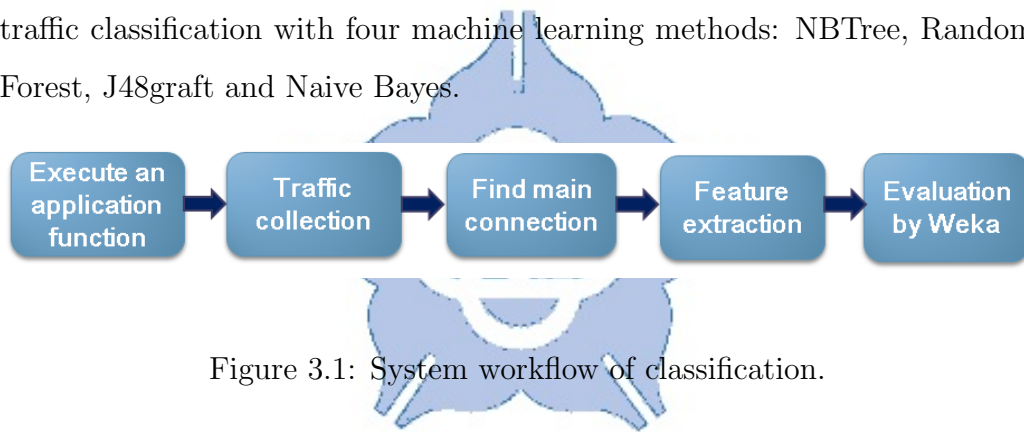


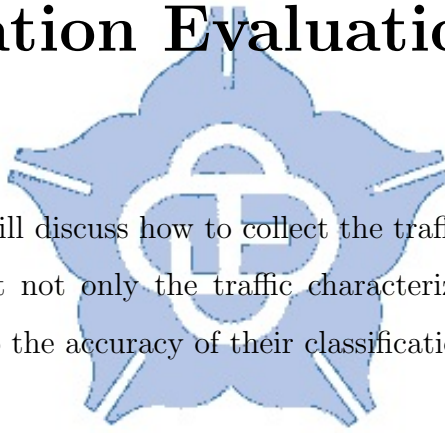
Figure 3.1: System workflow of classification.

Table 3.3: The contents of other connections

Label	Web application	Content-Type
Document	Google doc	application/json, text/html, text/javascript, font/woff, image/png, image/gif, etc.
Map	Google map	application/json, text/javascript, image/gif, image/png, etc.
Game	Dungeon Rampage	text/html, application/x-javascript, image/png, application/x-shockwave-flash, etc.
	Tetris Battle	image/gif, image/png, application/xml, application/x-shockwave-flash, audio/mpeg, etc.
File Transfer	Google Drive	text/javascript, application/json, text/css, text/html, image/gif, etc.
	Dropbox	application/x-javascript, image/gif, image/png, application/octet-stream, text/css, etc.
Video Stream	Youtube	audio/mp4, video/mp4, text/javascript, text/css, application/x-shockwave-flash, image/gif, etc.
	Dailymotion	application/x-shockwave-flash, text/css, text/xml, image/jpeg, application/x-javascript, etc.
	Tudou	application/x-shockwave-flash, image/jpeg, application/octet-stream, text/xml, image/gif, etc.

Chapter 4

Traffic Characterization and Classification Evaluation



In this chapter, we will discuss how to collect the traffic of web applications in detail and present not only the traffic characterization of various web applications, but also the accuracy of their classification.

4.1 Scenarios of Packet Collection

The network traffic is generated from user interactions on individual applications. We choose the following web applications for analysis: (1) Google docs for the office application, (2) Google map for the map application, (3) Youtube, Tudou, Dailymotion for the video streaming applications, (4) Google drive and Dropbox for the file transfer applications, and (5) Tetris Battle and Dungeon Rampage on Facebook for the game applications. In this work, we repeatedly performed the following actions ten times on either

Chrome (version 41.0.2272.89) or Firefox (version 36.0.1) for a period from one to two minutes each time.

The interactions in the office application involve arbitrarily typing characters and changing the colors. The interactions in the map application involve typing an arbitrary location name to be searched in the map, zooming-in and zooming-out the map, and browsing the map. The interactions in the game application involve (1) arbitrarily moving a role and fighting with the other roles in *Dungron Rampage* and (2) moving and rotating the blocks in *Tetris Battle*. We also consider the download functions in the video streaming and the file sharing applications. In the former application, we typed an arbitrary video name and moved to a specific time position when watching a video or just silently watched a video until the end. In the latter application, we just downloaded an arbitrary file and waited silently until the completion of the file up/downloading.

4.2 Traffic Characterization of Web Applications

A typical web session may involve several parallel connections to speed up fetching the requested content, and may also contain irrelevant connections such as those for advertisements. Thus, we manually inspected the content of requests and responses with the developer tools affiliated with the browsers when collecting the related packet traces, and focused only on the connections over which the user interactions are carried. To determine the connections associated with the user interactions, we watched the ongoing connection activities in the developer tool when interacting with a web application, and selected the longest one of these active connections as the main

connection. Table 4.1 presents the average number of connections and the standard deviation in each web application. The values demonstrate that Chrome creates more connections than Firefox on average to transfer the data for the same web applications.

Table 4.1: The average number of connections and the standard deviation (SD) for each web application.

		Chrome		Firefox	
Label	Web application	Avg.	SD	Avg.	SD
Document	Google doc	37.30	4.69	33.70	2.71
Map	Google map	27.60	2.50	19.40	1.65
Game	Dungeon Rampage	125.40	10.01	108.30	7.45
	Tetris Battle	218.30	14.77	174.70	8.65
File	Google Drive	37.50	3.17	31.70	0.95
Transfer	Dropbox	68.00	11.26	61.30	3.30
Video Stream	Youtube	52.3	6.02	23.70	2.63
	Dailymotion	380	118.08	130.50	8.67
	Tudou	150.80	16.60	196.30	25.19

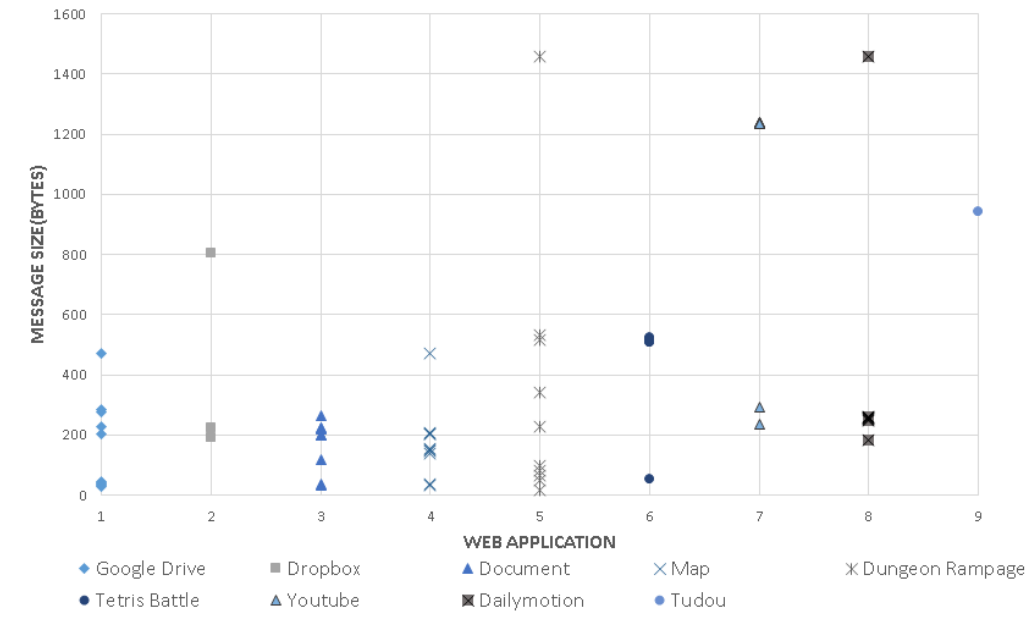
We analyzed the traffic from the web applications with the developer tools and Wireshark. The traffic characteristics are summarized as follows. In the office application, the characters from user typing are sent in short messages in the same connection from both Chrome and Firefox, and the length of the short message size for each browser is either 34 bytes or 46 bytes. When we enter the website of the map application, it will display the map of the user's location according to the source IP address. Furthermore, we search a new location, the application will transfer all the data associated with the

region to the browser at a time, so no further packets are transferred when we slightly zoom-in and zoom-out the map. When we play Tetris Battle, we find that around 17.28% of the packets in our each collection contain the TCP PSH flags to “push” the packets from the receive buffer to the server application because this game is highly interactive. In the file sharing applications, the client keeps transferring short messages during downloading a file on Google drive. In contrast, there are few messages transferred from the client during file downloading on Dropbox. In video applications except Youtube, the connection that downloads the video clip will be reset when the user changes the time position.

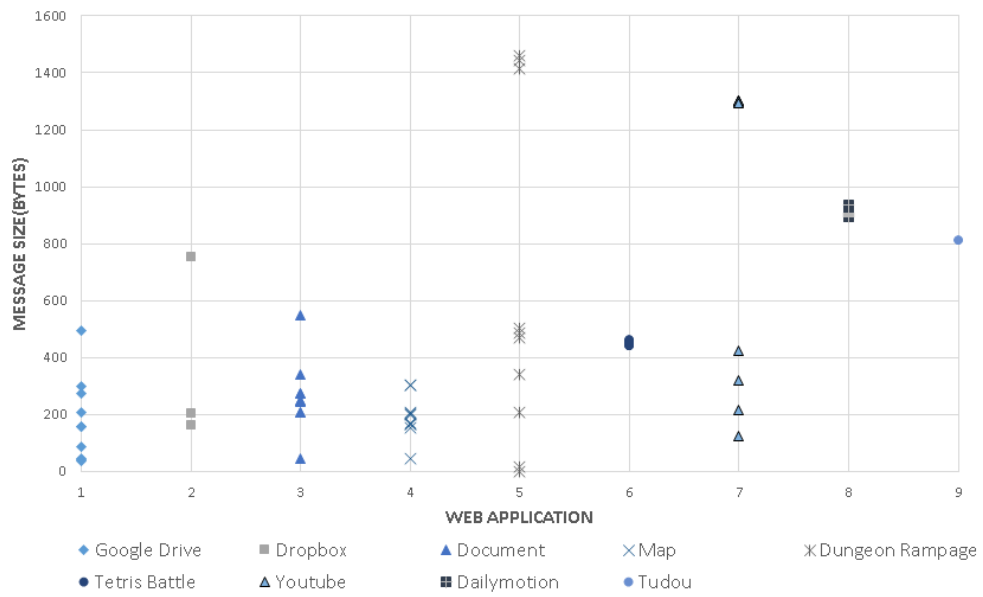
4.3 Feature analysis

4.3.1 Message size distribution

Figure 4.1 presents the top 10 frequent message sizes from the two browsers in each scenario. The messages are reassembled packet content in the main connections by `libnids`, and their sizes are ordered by the occurrence frequency. This figure demonstrates that the message sizes in different web applications vary significantly, and the occurrence frequencies of each message size also vary with the applications. The significant variations in different applications imply that the feature of message sizes should be effective for classification. Moreover, the features in the same web applications on different browsers are similar, meaning that the classification is expected to label the applications correctly even though they are running on different browsers.



(a) Chrome



(b) Firefox

Figure 4.1: The message size distribution of each web application from two browsers.

Performing the actions on similar web applications will result in different features. For example, the message size distribution of video streaming applications (Youtube, Dailymotion, Tudou) differ obviously according to Figure 4.1. However, we do not need to classify individual applications in practice because the management policies (e.g., bandwidth management or QoS) for similar applications are likely to be the same. Thus, we argue that it suffices to classify similar applications into the same category.

Figure 4.2 shows the categorization procedure. We divide the web applications into five categories: file sharing, office, map, game and video streaming. A category consists of one or multiple similar applications. If an application is correctly classified, it is also classified into the correct category. Even though the features are occasionally ambiguous between the applications in the same category (e.g., Dailymotion and Tudou), the applications can be still classified into the category of video streaming applications.

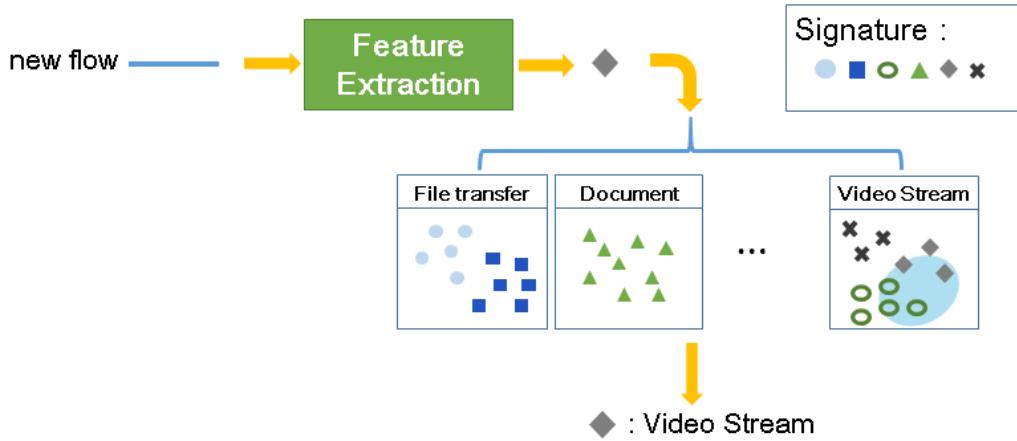


Figure 4.2: Categorization of web applications.

4.4 Classification results

After we collect sufficient data from the browsers, we employ Weka to evaluate whether the feature is effective or not. Weka supports a collection of machine learning algorithms for data mining, among which we choose NBtree, Random Forest, J48graft and Naive Bayes to compare the classification accuracy by using different algorithms. We use 10-fold cross-validation to ensure the reliability. When testing the performance, we divide the web applications into two groups, i.e., interaction applications (document, map, game) and file transfer applications (video streaming, file transfer), since we want to compare the result with the previous method, which tests the accuracy of interaction applications and file transfer applications separately. In addition, the possibility of early classification is also evaluated and it will be described in detail on Section 4.4.4.

We choose the previous study to compare with our mechanism. Lin et al. presented the method based on fine-grained classification and used the average and the standard deviation of the request/response lengths as the features for classification. The difference between our targets is that we only want to classify the kinds of web applications; however, they want to distinguish what action a user does. In the work, they subdivide the flows into search keystrokes, editing, action and download. To dislodge the unnecessary message, they artificially input the first request time or the first request bytes. Nevertheless, that method is more suitable for offline analysis. The reason we choose this study as our comparison is that the way of features extraction, which is independent of the network condition under the application layer, is similar to our mechanism.

4.4.1 Classification with Similar Interactions in Each Run

Table 4.2 shows the classification accuracy of each web application. The accuracy of all classifiers reach 95% or above with our method, but the values still are slightly lower than previous work. There are some reasons may affect the result. For instance, our targets are different and we do not divide search keystroke messages from the flows we extract, since search packets and interaction packets may be transferred in the same connection. The true-positive rate in Google Map does not achieve 1 is because a few instances of Google Map are misidentified as those of Google document.

Table 4.2: True/false positive rates of classifying the interactive connections.

	Correctly classified (%)		Document TP/FP		Map TP/FP		Game TP/FP	
	Prior	New	Prior	New	Prior	New	Prior	New
NBtree	99.80	97.50	0.993/ 0.001	1/ 0.033	0.999/ 0.005	0.90/0	1/0	1/0
RandomForest	99.89	98.75	1/ 0.001	1/ 0.017	0.999/ 0	0.95/0	1/0	1/0
J48graft	99.28	97.50	0.993/ 0.004	1/ 0.033	0.996/ 0.019	0.90/0	0.95/0	1/0
NaiveBayes	99.49	95.00	0.987/ 0	1/ 0.067	0.996/ 0.009	0.80/0	1/0	1/0

Prior: Previous work / New : Our method

We also chose three additional web applications to disturb classification:

Google sheets (type: document), Bing map (type: map) and Dungeon Blitz (type: game). We followed the scenarios described in Table 3.1, and operated each web applications ten times on either Chrome or Firefox. We gathered the features generated from these additional applications and the original training set to verify the accuracy after the disturbance. The result is showed in Table 4.3 and the accuracy after 10-fold cross-validation can be still up to 97.14% for NBtree. In this work, we add additional applications only to the group of interaction applications to verify the accuracy because the variation of operations for file transfer applications are usually low.

Table 4.3: True/false positive rates of classifying the interactive connections with additional web applications.

	Correctly classified (%)		Document TP/FP		Map TP/FP		Game TP/FP	
	Orig.	Add.	Orig.	Add.	Orig.	Add.	Orig.	Add.
NBtree	97.50	97.14	1/ 0.033	0.975/ 0.03	0.90/0	0.925/ 0.01	1/0	1/0
RandomRorest	98.75	95.71	1/ 0.017	0.95/ 0.04	0.95/0	0.90/ 0.02	1/0	1/0
J48graft	97.50	92.14	1/ 0.033	0.90/ 0.06	0.90/0	0.85/ 0.04	1/0	0.983/ 0.013
NaiveBayes	95.00	95.71	1/ 0.067	0.975/ 0.05	0.80/0	0.875/ 0.01	1/0	1/0

Orig.: Original training set / Add.: With additional web applications

We also differentiate between downloading of video stream and general

file transfer. We typed a video name and arbitrarily switched to multiple time positions when watching a video or just silently watched a video until the end on Youtube, Dailymotion and Tudou on the two browsers, but we up/downloaded files by Google drive and Dropbox. In summary, we watched 60 videos and transferred 40 files. After extracting the features, we used 10-fold cross-validation to evaluate the classification. It is presented in Table 4.4 that the classification accuracy can be up to 97% for all the classifiers, even better than previous work.

Table 4.4: True/false positive rates of classifying downloading connections.

	Correctly classified (%)		Video Streaming TP/FP		Files Transfer TP/FP	
	Prior	New	Prior	New	Prior	New
NBtree	92.72	97.00	0.907/0	0.967/0.025	1/0.093	0.975/0.033
RandomForest	92.33	97.00	0.902/0	0.967/0.025	1/0.098	0.975/0.033
J48graft	91.57	97.00	0.893/0	0.967/0.025	1/0.107	0.975/0.033
NaiveBayes	92.72	97.00	0.907/0	0.967/0.025	1/0.093	0.975/0.033

Prior : Previous work / New : Our method

4.4.2 Classification with Diverse Interactions in Each Run

In this subsection, instead of separating the web applications into interaction function and download function, we classify the traffic into five categories: document, map, game, video stream and file transfer. In the prior study of [24], the authors evaluated classification separately because the features for interaction function and download function are different in that study, so we also separate the evaluation as well in the last subsection to compare with the prior study fairly. However, we use the same feature, i.e., top-5 most frequent message sizes from the requests, for all the web applications, so we classify all the categories for evaluation in this subsection. We also operated each web application ten times on either Chrome or Firefox, but deliberately performed an extremely different action in the same scenario. For example, we may just continuously type or intermittently edit a document for each time we collected in document function. The classification accuracy after 10-fold cross-validation also can be up to 93.89% for Random Forest, meaning that the feature is stable and can be used for the classification with high accuracy.

Table 4.5: True/false positive rates of classifying all connections.

	Correctly classified (%)	Document TP/FP	Map TP/FP	Game TP/FP	Video Streaming TP/FP	Files Transfer TP/FP
NBtree	92.22	1/ 0.05	0.80/ 0.006	0.95/ 0.007	0.917/ 0	0.925/ 0.029
RandomForest	93.89	0.90/ 0.013	0.90/ 0.019	0.975/ 0.021	0.933/ 0.017	0.95/ 0.007
J48graft	92.22	0.95/ 0.031	0.80/ 0.013	0.975/ 0.021	0.917/ 0.008	0.925/ 0.021
NaiveBayes	92.22	0.95/ 0.025	0.95/ 0.013	0.95/ 0.029	0.917/ 0.017	0.875/ 0.014

4.4.3 Classification with Interactions from Multiple Users

To ensure the accuracy does not too depend on specific users, we invited eight users to generate the traffic for the classification. In this subsection, we compare only interactive functions because the their actions are more diverse than download functions. The users were asked to perform similar actions to those from a single user and repeated three times for each web applications on Chrome and Firefox. The total number of packet traces is 192: 48 for Google document, 48 for Google map, 48 for Dungeon Rampage and 48 for Tetris Battle.

Table 4.6 shows the accuracy for each algorithm. We also used 10-fold cross-validation for this classification. The accuracy of our method for Naive Bayes is even higher than the results in Section 4.4, and can be up to 97.92%

with random forest. Compared the result with the training set shown in last subsection, the accuracy is degraded just slightly, meaning that the feature is stable.

Table 4.6: True/false positive rates of classifying the interactive connections for multiple users.

	Correctly classified (%)		Document TP/FP		Map TP/FP		Game TP/FP	
	Prior	New	Prior	New	Prior	New	Prior	New
NBtree	96.63	96.35	0.939/ 0.025	0.938/ 0.028	0.971/ 0.044	0.938/ 0.021	0.993/ 0	0.99/ 0
RandomForest	98.28	97.92	0.954/ 0.009	1/ 0.007	0.99/ 0.032	0.979/ 0.014	1/ 0	0.969/ 0.01
J48graft	97.65	93.75	0.939/ 0.011	0.917/ 0.035	0.986/ 0.044	0.917/ 0.035	0.993/ 0.001	0.958/ 0.021
NaiveBayes	95.74	95.83	0.87/ 0.017	0.938/ 0.028	0.98/ 0.091	0.938/ 0.028	1/ 0.001	0.979/ 0

Pre: Previous work / New : Our method

4.4.4 Early Classification

The classification method in this work can be applied to manage various web applications, even though the web traffic is encrypted. The specifics of classification may vary in practice, depending on the purpose of management. If the purpose is accounting the usage of web applications in a network, it would suffice to classify the web applications offline based on the features

from *the entire packet traces* that have been observed. However, if the purpose is access control, classification after extracting the features from the entire packet traces will be too late to block the traffic in time. Thus, we also evaluate the effectiveness of early classification with the features from the partial messages in the beginning of the connections. In early classification, a connection can be blocked as soon as the web application is recognized based on the features from the partial messages.

The packet traces for evaluating early classification are same as those described in the last subsection. We extracted the first k messages from each main connection ($k = 15, 20, 25, 30, 50, 100$) and used the size distribution of only these messages for the evaluation. Like the features from the entire packet traces, we also extracted the top five frequent message sizes in terms of occurrence frequency.

The previous analysis show that the true-positive rate of almost all kinds of functions are at least up to 0.90; however, the true-positive rate for the map application is decreased to 0.80 because its features are not so stable as others. We choose Dungeon Rampage on Chrome as example for stable one shown as Figure 4.3 and Google map on Chrome as example for unstable one shown as Figure 4.4. The line chart is created by entire messages within a main connection and the scatter chart is created by the first k messages in each figure.

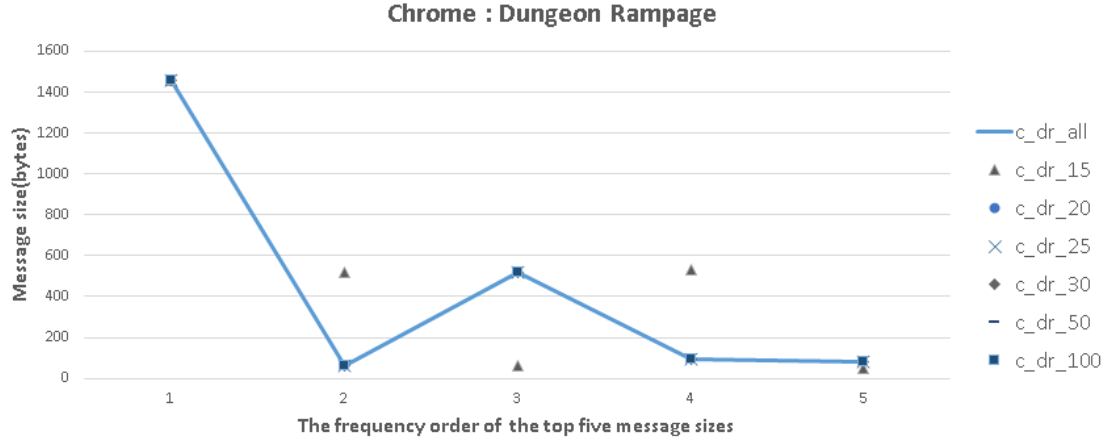


Figure 4.3: The message size distribution for Dungeon Rampage.

Figure 4.3 depicts that the scatter charts are similar with the trend of line chart when we extract more than the first 20 messages. Figure 4.4 depicts that the scatter charts are almost similar with the trend of line chart when we extract more than the first 30 messages. So we finally extracted the first 30 messages from each web applications as our feature for early classification. The classification accuracy after 10-fold cross-validation can be at least 86.67% and even can be up to 93.89% for Random Forest.

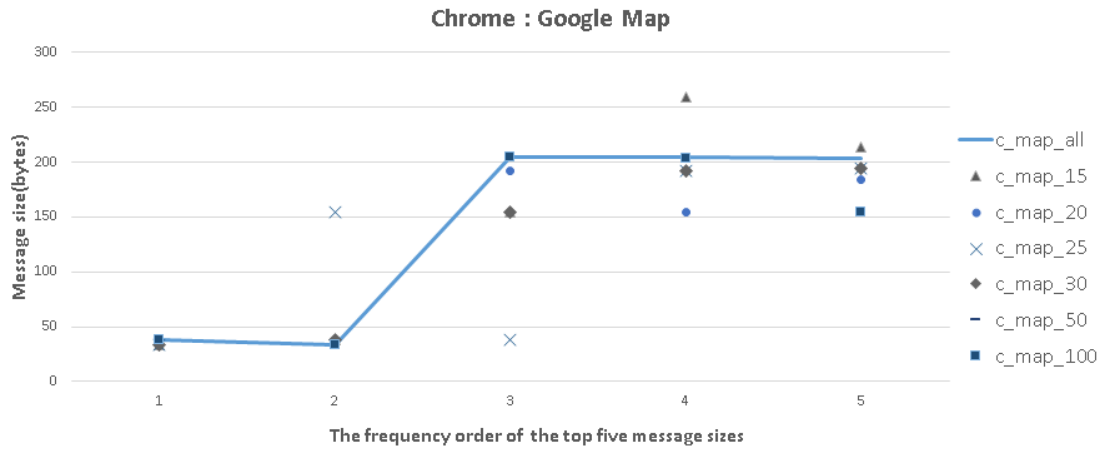


Figure 4.4: The message size distribution for Google map.

Table 4.7: Early classification true/false positive rate in all connections.

	Correctly classified (%)	Document TP/FP	Map TP/FP	Game TP/FP	Video Stream TP/FP	File Transfer TP/FP
NBtree	86.67	0.70/ 0.038	0.95/ 0.019	0.925/ 0.007	0.933/ 0.025	0.75/ 0.079
RandomForest	93.89	0.85/ 0.013	1/0	1/ 0.014	0.933/ 0.017	0.90/ 0.036
J48graft	88.33	0.85/ 0.019	1/ 0.019	0.95/ 0.029	0.85/ 0.042	0.825/ 0.043
NaveBayes	87.22	0.85/ 0.063	0.95 /0.013	0.925/ 0.021	0.95/ 0.017	0.675/ 0.043

4.5 Practice and Limitation

Even though the classification accuracy is high for extracted packet traces from real user interactions with web applications, there are still some issues that should be addressed to deploy this classification in practice.

First, although one IP address may be associated with more than one web application, as we demonstrated earlier in this work, we can still record the mapping between the IP addresses and their associated applications from earlier classification in a list to speed up classification. Since the mapping may be one-to-many, if a remote IP address can be found in the list and it is mapped to only one application, we can leverage the result of earlier classification to label the traffic to that application directly; otherwise, we can follow the procedure described in Chapter 3 to classify the web application. The result from earlier classification can at least help reduce the scope of possible labels.

Second, the traffic for each packet trace is just generated from a specific web application in this work, but in reality, we will need to analyze multiple applications at the same time and extracting the main connection is a problem. To solve this issue, we will observe the traffic density and quantity of each IP address during a period time and if the values both reach the thresholds, we will extract it as the main connection for classification. After classification, if the device that employs the design determines to block the main connection, the connections having the same pair of source/destination IP addresses as the main connection and happening around it will be blocked as well. Furthermore, users may use a web application not in the training set. Thus, after traffic classification, we will have to compute the distance between the feature vectors of the analyzed traffic and the application(s) that

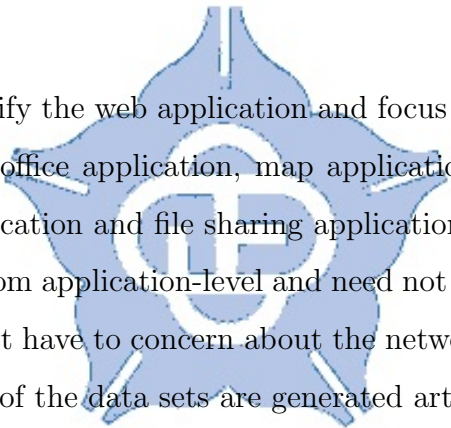
the analyzed traffic is supposed to be. If the distance is too long, than the analyzed traffic will be labeled as unknown.

Third, if the main connection is identified after the web application has been executed for a period of time, it may be too late to block an application for access control since the function is likely to have already completed its execution when the flows are collected and analyzed. The early classification described in Section 4.4.4 can address this issue. Moreover, the accuracy may be decreased if the execution time of an web application function is too short to extract meaningful feature.



Chapter 5

Conclusion



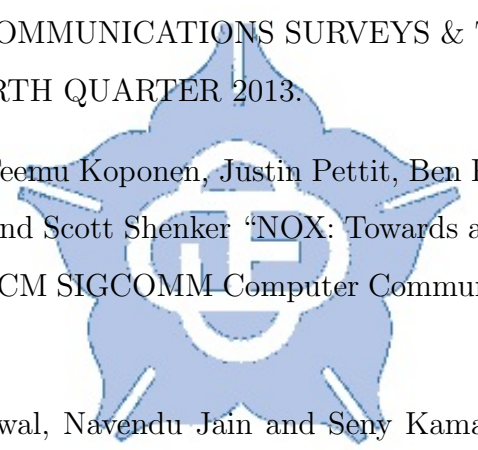
In this work, we classify the web application and focus on the five categories of web applications: office application, map application, game application, video streaming application and file sharing application. The statistical features are extracted from application-level and need not to analyze the packet payloads, so we do not have to concern about the network condition and the encryption issue. All of the data sets are generated artificially, so the condition is close to reality. This work uses Weka and 10-fold cross-validation for evaluating the classification accuracy.

The experimental results demonstrate that the accuracy can be as high as 98.75% and the false-positive rate is no more than 0.017 using random forest for the interaction functions. For the download functions, the accuracy can be up to 97.00% and the false-positive rate can be 0.025 for each algorithm. For the five functions, the accuracy can be up to 93.89% and the false-positive rate can be 0.007 using random forest. For the early classification, the accuracy can be up to 93.89% and the false-positive rate is no more

than 0.036 using random forest. Finally, the results show that the feature can effectively classify traffic of various web applications, and even in early classification.



Bibliography

- 
- [1] Adnan Nadeem and Michael P. Howarth, “A Survey of MANET Intrusion Detection & Prevention Approaches for Network Layer Attacks,” , In Proc. of IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 15, NO. 4, FOURTH QUARTER 2013.
 - [2] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martn Casado, Nick McKeown and Scott Shenker “NOX: Towards an Operating System for Networks”, ACM SIGCOMM Computer Communication Review 38.3 (2008): 105-110.
 - [3] Anurag Khandelwal, Navendu Jain and Seny Kamara “Attacking Data Center Networks from the Inside”,
 - [4] Markku Antikainen, Tuomas Aura, and Mikko Srel, “Spook in Your Network: Attacking an SDN with a Compromised OpenFlow Switch”, In proc. of Springer International Publishing Switzerland 2014
 - [5] TIEN THANH BUI, “Analysis of Topology Poisoning Attacks in Software-Defined Networking”, Degree project in security and mobile computing, Second Level Stockholm, Sweden 2015
 - [6] Kreutz, Diego, Fernando Ramos, and Paulo Verissimo, “Towards secure and dependable software-defined networks.”, Proceedings of the second

- ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, 2013.
- [7] Farzaneh Pakzad, Marius Portmann, Wee Lum Tan and Jadwiga Indulska, “Efficient Topology Discovery in Software Defined Networks”, In Signal Processing and Communication Systems (ICSPCS), 2014 8th International Conference on, pp. 1-8. IEEE, 2014.
- [8] Po-Wen Chi, Chien-Ting Kuo, Jing-Wei Guo and Chin-Laung Lei, “How to Detect a Compromised SDN Switch”, In Network Softwarization (Net-Soft), 2015 1st IEEE Conference on (pp. 1-6). IEEE.
- [9] Sungmin Hong, Lei Xu, Haopei Wang and Guofei Gu, “Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures”, In NDSS. 2015.
- [10] Alharbi, Talal, Marius Portmann, and Farzaneh Pakzad, “The (In) Security of Topology Discovery in Software Defined Networks.”, In Proc. of Local Computer Networks (LCN), 2015 IEEE 40th Conference on. IEEE, 2015.
- [11] “”
- [12] “”
- [13] “”
- [14] “”
- [15] “”
- [16] <https://en.wikipedia.org/wiki/OpenFlow>
- [17] <http://groups.geni.net/geni/wiki/OpenFlowDiscoveryProtocol>.

- [18] https://en.wikipedia.org/wiki/Software-defined_networking.
- [19] <http://www.networkworld.com/article/2840273/sdn/sdn-security-attack-vectors-and-sdn-hardening.html>
- [20] <http://www.wipro.com/documents/insights/the-thinking-network.pdf> —————
- [21] T. T. Nguyen and G. Armitage, “A Survey Of Techniques for Internet Traffic Classification Using Machine Learning,” In Proc. of IEEE Comm. Surveys Tutorials, vol. 10, no. 4, pp. 56-76, Oct.-Dec.2008.
- [22] J. S. Park, S. H. Yoon and M. S. Kim, “Performance Improvement of Payload Signature-based Traffic Classification System Using Application Traffic Temporal Locality,” In Proc. of the 15th Network Operations and Management Symposium (APNOMS), Sept. 2013.
- [23] M. Roughan, S. Sen, O. Spatscheck and N. Duffield, “Class-of-servicemapping for QoS: a statistical signature-based approach to IP traffic classification,” In Proc. of ACM SIGCOMM Conference on Internet Measurement, Oct. 2004.
- [24] P. C. Lin, S. Y. Chen and C. H. Lin, “Towards Fine-grained Traffic Classification for Web Applications,” In Proc. of Australasian Telecommunication Networks and Applications Conference (ATNAC), Nov. 2014.
- [25] P. Schneider, “TCP/IP traffic Classification Based on port numbers,” Diploma Thesis, Division of Applied Science, Harvard University, 29 Oxford Street, Cambridge, MA 02138, USA, 1-6

- [26] Y. Xue, D. Wang and L. Zhang, "Traffic Classification: Issues and Challenges," In Proc. of the International Conference on Computing, Networking and Communications (ICNC), Jan. 2013.
- [27] <http://www.ntop.org/products/deep-packet-inspection/ndpi/>
- [28] <http://17-filter.clearos.com/>
- [29] B. Hullar, S. Laki and A. Gyorgy, "Efficient Methods for Early Protocol Identification," IEEE Journal on Selected Areas in Communications, vol. 32, issue 10, Oct. 2014.
- [30] L. Bernaille, R. Teixeira and K. Salamatian, "Early application identification," In Proc. of the Conference on Future Networking Technologies, 2006.
- [31] H. M. An, M. S. Kim and J. H. Ham, "Application traffic classification using statistic signature," In Proc. of the 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), Sept. 2013.
- [32] B. Schmidt, D. Kountanis and A. Al-Fuqaha, "Artificial Immune System Inspired Algorithm for Flow-Based Internet Traffic Classification," In Proc. of the IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), Dec. 2014.
- [33] S. Tapaswi and A. S. Gupta, "Flow-Based P2P Network Traffic Classification Using Machine Learning," In Proc. of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Oct. 2013.
- [34] F. Dehghani, N. Movahhedinia, M. R. Khayyambashi and S. Kianian, "Real-Time Traffic Classification Based on Statistical and Payload Con-

- tent Features,” In Proc. of the 2nd International Workshop on Intelligent Systems and Applications (ISA), May 2010.
- [35] S. Huang, K. Chen, C. Liu, A. Liang and H. Guan, “A statistical-feature-based approach to internet traffic classification using Machine Learning,” In Proc. of the International Conference on Ultra Modern Telecommunications and Workshops (ICUMT), Oct. 2009.
- [36] M. Kohara, T. Hori, K. Sakurai, H. Lee and J. C. Ryou, “Flow Traffic Classification with Support Vector Machine by Using Payload Length,” In Proc. of the 2nd International Conference on Computer Science and its Applications (CSA), Dec. 2009.
- [37] M. Roughan, S. Sen, O. Spatscheck and N. Duffield, “Class-of-Service Mapping for Qos: A Statistical Signature-based Approach to IP Traffic Classification,” In Proc. of the Internet Measurement Conference (IMC), Oct. 2004.
- [38] J. Zhang, C. Chen, Y. Xiang and W. Zhou, “Classification of Correlated Internet Traffic Flows,” In Proc. of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), June 2012.
- [39] S. Valenti and D. Rossi, “Fine-grained behavioral classification in the core: the issue of flow sampling,” In Proc. of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC), July 2011.
- [40] B. Park, J. W. Hong and Y. J. Won, “Toward Fine-Grained Traffic Classification,” IEEE Communication Magazine, vol. 49, issue. 7, pp. 104-111, July 2011.

- [41] B. Augustin and A. Mellouk, "On Traffic Patterns of HTTP Applications," In Proc. of the IEEE Global Communications Conference (GLOBECOM), Dec. 2011.
- [42] R. Archibald, Y. Liu, C. Corbett and D. Ghosal, "Disambiguating HTTP: Classifying web Applications," In Proc. of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC), July 2011.
- [43] P. Casas and P. Fiadino, "Mini-IPC: A minimalist approach for HTTP traffic classification using IP addresses," In Proc. of the 9th International Wireless Communications and Mobile Computing Conference (IWCMC), July 2013.
- [44] <http://libnids.sourceforge.net>.
- [45] M. Lee, R. R. Kompella and S. Singh, "Ajaxtracker: active measurement system for high-fidelity characterization of Ajax applications," In Proc. of the 2010 USENIX conference on Web application development (WebApps '10), 2010.
- [46] S. Veres and D. Ionescu, "Measurement-based traffic characterization for Web 2.0 applications," In Proc. of the IEEE Instrumentation and Measurement Technology Conference (I2MTC), May 2009.
- [47] L. Shuai, G. Xie and J. Yang, "Characterization of HTTP behavior on access networks in Web 2.0," In Proc. of the International Conference on Telecommunications (ICT), June 2008.
- [48] S. Lin, Z. Gao and K. Xu, "Web 2.0 traffic measurement: analysis on online map applications," In Proc. of the 18th International Workshop

on Network and Operating Systems Support for digital audio and video, ser. NOSSDAV 09, 2009.

- [49] M. Belshe, BitGo, R. Peon and M. Thomson, Ed., “Hypertext Transfer Protocol Version 2 (HTTP/2),” RFC 7540, May 2015.
- [50] <http://internet-browser-review.toptenreviews.com/>.
- [51] http://www.w3schools.com/browsers/browsers_stats.asp.
- [52] F. Schneider, S. Agarwal, T. Alpcan and A. Feldmann, “The New Web: Characterizing AJAX Traffic,” In Proc. of the Passive and Active Measurement Conference, 2008.
- [53] “The Data Mining and Knowledge Discovery Handbook,” In Oded Maimon and Lior Rokach Tel-Aviv (eds.), University Israel, pp. 40-41.

