UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial : 7 août 2006

Présentée par

Maciej KORCZYŃSKI

Thèse dirigée par **Andrzej Duda**

préparée au sein de **UMR 5217 - LIG - Laboratoire d'Informatique de Grenoble**

et de **École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (EDMSTII)**

Classifying Application Flows and Intrusion Detection in Internet Traffic

Thèse soutenue publiquement le **26 Novembre 2012**,

devant le jury composé de :

Mr Jean-Marc Thiriet

Professeur, Université Joseph Fourier, Président

Mr Philippe Owezarski

Chargé de recherche au CNRS, Université de Toulouse, Rapporteur

Mr Guillaume Urvoy-Keller

Professeur, Université de Nice, Rapporteur

Mr Andrzej Pach

Professeur, AGH University of Science and Technology, Examinateur

Mr Andrzej Duda

Professeur, Grenoble INP - ENSIMAG, Directeur de thèse

Acknowledgments

I would like to thank most especially my supervisor and mentor Prof. Andrzej Duda. You taught me a great deal about how to do research. Thank you for your trust and freedom in exploring different research directions. I would like to express my gratitude for your contributions to this work including sleepless nights before deadlines and your invaluable support in my future projects.

I am also very grateful to Dr. Lucjan Janowski and Dr. Georgios Androulidakis for your guidance, patience, and encouragement at the early stage of my research. Thanks for all that I have learnt from you.

I would like to thank Marcin Jakubowski for sharing your network administrator experience and packet traces without which this research would not have been possible.

I am also thankful to my friends from the Drakkar team, especially to Bogdan, Ana, Isa, Nazim, Michał, my office mates Carina, Maru, Mustapha, and Martin as well as my friends from other teams, especially to Sofia, Azzeddine, and Reinaldo for sharing the ”legendary” and the more difficult moments of PhD students life.

To my girlfriend and best friend Audrey for your patience with my passion for research and continuous support even if it requires making difficult life decisions. Thank you so much!

Finally, my deepest gratitude goes to my parents and grandparents for your unconditional love, endless support throughout my entire life, and understanding of my decisions.

Abstract

The subject of traffic classification is of great importance for effective network planning, policy-based traffic management, application prioritization, and security control. Although it has received substantial attention in the research community there are still many unresolved issues, for example how to classify encrypted traffic flows. This thesis is composed of four parts. The first part presents some theoretical aspects related to traffic classification and intrusion detection, while in the following three parts we tackle specific classification problems and propose accurate solutions.

In the second part, we propose an accurate sampling scheme for detecting SYN flooding attacks as well as TCP portscan activity. The scheme examines TCP segments to find at least one of multiple ACK segments coming from the server. The method is simple and scalable, because it achieves a good detection with a False Positive Rate close to zero even for very low sampling rates. Our trace-based simulations show that the effectiveness of the proposed scheme only relies on the sampling rate regardless of the sampling method.

In the third part, we consider the problem of detecting Skype traffic and classifying Skype service flows such as voice calls, skypeOut, video conferences, chat, file upload and download. We propose a classification method for Skype encrypted traffic based on the Statistical Protocol IDentification (SPID) that analyzes statistical values of some traffic attributes. We have evaluated our method on a representative dataset to show excellent performance in terms of Precision and Recall.

The last part defines a framework based on two complementary methods for classifying application flows encrypted with TLS/SSL. The first one models TLS/SSL session states as a first-order homogeneous Markov chain. The parameters of the Markov models for each considered application differ a lot, which is the basis for accurate discrimination between applications. The second classifier considers the deviation between the timestamp in the TLS/SSL `Server Hello` message and the packet arrival time. It improves the accuracy of application classification and allows efficient identification of Skype flows. We combine the methods using a Naive Bayes Classifier (NBC). We validate the framework with experiments on three recent datasets—we apply our methods to the classification of seven popular applications that use TLS/SSL for security. The results show a very good performance.

Résumé

Le sujet de la classification de trafic réseau est d'une grande importance pour la planification de réseau efficace, la gestion de trafic à base de règles, la gestion de priorité d'applications et le contrôle de sécurité. Bien qu'il ait reçu une attention considérable dans le milieu de la recherche, ce thème laisse encore de nombreuses questions en suspens comme, par exemple, les méthodes de classification des flux de trafics chiffrés. Cette thèse est composée de quatre parties. La première présente quelques aspects théoriques liés à la classification de trafic et à la détection d'intrusion. Les trois parties suivantes traitent des problèmes spécifiques de classification et proposent des solutions précises.

Dans la deuxième partie, nous proposons une méthode d'échantillonnage précise pour détecter les attaques de type "SYN flooding" et "portscan". Le système examine les segments TCP pour trouver au moins un des multiples segments ACK provenant du serveur. La méthode est simple et évolutive, car elle permet d'obtenir une bonne détection avec un taux de faux positif proche de zéro, même pour des taux d'échantillonnage très faibles. Nos simulations basées sur des traces montrent que l'efficacité du système proposé repose uniquement sur le taux d'échantillonnage, indépendamment de la méthode d'échantillonnage.

Dans la troisième partie, nous considérons le problème de la détection et de la classification du trafic de Skype et de ses flux de services tels que les appels vocaux, SkypeOut, les vidéo-conférences, les messages instantanés ou le téléchargement de fichiers. Nous proposons une méthode de classification pour le trafic Skype chiffré basé sur le protocole d'identification statistique (SPID) qui analyse les valeurs statistiques de certains attributs du trafic réseau. Nous avons évalué notre méthode sur un ensemble de données montrant d'excellentes performances en termes de précision et de rappel. La dernière partie définit un cadre fondé sur deux méthodes complémentaires pour la classification des flux applicatifs chiffrés avec TLS/SSL. La première modélise des états de session TLS/SSL par une chaîne de Markov homogène d'ordre 1. Les paramètres du modèle de Markov pour chaque application considérée diffèrent beaucoup, ce qui est le fondement de la discrimination entre les applications. La seconde méthode de classification estime l'écart d'horodatage du message `Server Hello` du protocole TLS/SSL et l'instant d'arrivée du paquet. Elle améliore la précision de classification des applications et permet l'identification

efficace des flux Skype. Nous combinons les méthodes en utilisant une Classification Naive Bayésienne (NBC). Nous validons la proposition avec des expérimentations sur trois séries de données récentes. Nous appliquons nos méthodes à la classification de sept applications populaires utilisant TLS/SSL pour la sécurité. Les résultats montrent une très bonne performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivations

"Accurate identification and categorization of network traffic according to application type is an important element of many network management tasks such as flow prioritization, traffic shaping/policing, and diagnostic monitoring." [1]

"Classifying traffic flows according to the applications that generate them is an important task for (a) effective network planning and design, and (b) monitoring the trends of the applications in operational networks." [2]

"Accurate network traffic classification is fundamental to numerous network activities, from security monitoring to accounting, and from Quality of Service to providing operators with useful forecasts for long-term provisioning." [3]

"The subject of traffic classification has a crucial importance for effective network planning, policy-based traffic management, application prioritization, and security control." (cf. Abstract)

When reading numerous publications in the domain of traffic classification and intrusion detection many of them in the first place emphasize its importance for operators, Internet Service Providers (ISPs), and local network administrators. However, for the sake of completeness let us take a look at the problem from a different, user's perspective. Many of our every day activities are closely associated with and dependent on properly working Internet connections. Our daily habits consist of checking our emails (usually two accounts, i.e. professional and private), reading online news, etc. Other "crucial" activities are related to our Facebook and Twitter accounts, daily routine often includes Skype calls. While some of us like online shopping others prefer different kinds of entertainment such as online gaming or watching sport events in the pay-per-view (ppv) system, and many more. Many

of our daily, legal routines would not be possible without research and industry efforts in the domain of traffic classification. This is why this subject has received substantial attention in the research community and still continues to grow.

Traffic classification is, however, a challenging task due to a massive proliferation of new applications and new ways of spreading and infecting unaware, legitimate users with malicious software. Moreover, existing programs tend to use more sophisticated communication mechanisms to bypass security checks. As a result, we observe a race between illegal applications, such as streaming pirated videos, improving their obfuscation methods and operators searching for new solutions to filter unwanted traffic and prioritize remaining applications. Although we witness much research interest in the domain of traffic classification and intrusion detection, many issues still remain unsolved and even though research community finds appropriate methods, new countermeasures appear rapidly. For example, classification approaches proposed some years ago and based on identifying network flows according to corresponding ports or regular expressions in unencrypted packet header are not effective any longer due to port randomization and traffic encryption respectively.

Some of the problems presented in this thesis arose after extensive discussions with the administrator of the campus network at AGH University of Science and Technology in Cracow, whereas some others appeared from an in-depth analysis of the existing literature. We further attempt to address these problems by designing proof-of-concept classifiers. Where we possibly could, we have evaluated our classification methods on real-world datasets captured at edge routers to show excellent performance in terms of different criteria. Methods presented in the thesis deal with very different aspects of traffic classification, from network attack detection to the classification of encrypted applications.

1.2 Overview of the thesis

The presented work is divided into four parts composed of dense chapters. Each part of the thesis starts with a short introduction chapter with briefly described contributions and the list of corresponding publications (if applicable). Moreover, practical parts involve a complementary discussion on theoretical issues specific to the addressed problems. Finally, we briefly survey the related work relevant to each part of the thesis.

In Part I, we describe some traffic classification aspects. We present a taxonomy of network traffic classification as well as a short discussion on intrusion detection methods. Part II introduces an accurate method for SYN flooding attack

and portscan activity detection using sampling techniques to limit the volume of inspected data. In the third and fourth part of the thesis, we focus on encrypted traffic classification. In Part III, we develop a hybrid methodology based on flow and content features for identifying TCP Skype flows tunneled over the SSL protocol and classifying its service flows. In Part IV, we propose a framework for classifying TLS/SSL flows of various applications.

Part I

State of the Art

Chapter 2

Introduction

The importance of appropriate traffic classification methods continues to grow. They are essential for effective network planning, policy-based traffic management, application prioritization, and security control. However, traditional classification methods are becoming less efficient, because new applications begin to use sophisticated obfuscation mechanisms and an increased number of applications make use of encryption to avoid security checks. Moreover, applications are rapidly adapting to counteract attempts to identify certain types of traffic, creating new challenges for traffic classification schemes.

We use the expression *traffic classification* to refer to two areas of our interest according to specific goals, namely to *application classification* and *intrusion detection* as well as to *methods* of classifying traffic data sets based on *features* passively observed in the Internet traffic. In the following chapter, we discuss the above-mentioned aspects of traffic classification with respect to classification goals—we start with the formal definition of traffic classification, followed by a brief survey on the classification and intrusion detection methods. Finally, we introduce two essential concepts of the ground truth and the metrics of classification performance.

2.1 Contributions of Part I

The main contribution of this part is an extension to the payload-based taxonomy based on the research presented in the thesis. More specifically, we introduce a more general taxonomy of payload-based methods in comparison to existing ones. We propose to distinguish between the type of data to be analyzed rather than between verification or processing techniques. We make a distinction between message-based and header-based analysis and we separate the analysis of lower-layer protocol headers, in particular network and transport layers, from the application-layer protocol header. We argue that in some classification problems the analysis of lower-layer protocol fields is sufficient, while in other cases a more detailed application-layer protocol header analysis is required.

Chapter 3

Background on Traffic Classification

Contents

3.1 Definition of Traffic Classification

Traffic classification is a research area that helps us to understand the nature of the Internet traffic. It consist of examining IP packets to extract some specific features to answer some questions related to its origin, the carried content, or user intensions. Frequently, it deals with packet flows defined as sequences of packets uniquely identified by the same source IP address, source port, destination IP address, destination port, and transport layer protocol. However, packets might be grouped in any way according to classification needs.

Let us now formalize the *trafic classification* problem. A pattern p represents the object under analysis. Each pattern is described by a set of n features that have been derived from the analyzed traffic. Thus, it can be interpreted by the n-dimensional random variable X that corresponds to an accurate set of *features*: $p \rightarrow x = (x_1, x_2, x_3, \ldots, x_d)$.

In the *application classification* problem, where p could be represented by flows, we attempt to assign each of them to one of the given application classes defined by a random variable $Y : y = \{y_1, y_2, \ldots, y_c, y_{c+1}\}$. $Y = y_{c+1}$ means that the analyzed flow is not recognized as any of the given classes, i.e., it is unknown.

In the *intrusion detection* problem p could be represented by the aggregated traffic directed to the specific IP destination address. Thus, intrusion detection refers to a binary classification problem—we attempt to verify if the traffic to analyze corresponds to malicious behavior. Random variable Y takes values in the set $\{y_0, y_1\}$, where $Y = y_0$ means that the traffic conforms to legitimate behavior, whereas $Y = y_1$ indicates malicious activity.

In the presented thesis, solving the traffic classification problems corresponds to defining classifiers that categorize each pattern into one of $c => 2$ classes.

3.2 Classification Goals

Although the research area of traffic classification is rather specific, the motivations of research papers are not identical [4].

In Figure 3.1, we present typical classification objectives or, in other words, three different domains, where proposed methods operate. More precisely, some methods classify traffic according to its *category*, i.e., whether the traffic represents bulk-transfer, peer-to-peer (p2p) content sharing, games, multimedia, web, or attacks [3, 5, 6, 7, 2]. It is also referred to as the *coarse-grained classification* goal [8]. A number of methods aim at identifying the *application-level protocol* such as FTP, HTTP, SSH, Telnet [9, 1, 10, 11, 12], also referred to as the *finer-grained classification* goal [8]. The last group of methods classifies the traffic according to the exact *application* that generates traffic, such as Skype, Dropbox, eBay.

Unfortunately, users tend to confuse application classification with application - level protocol classification (cf. Figure 3.1) [13]. For instance, classification of Skype traffic illustrates the problem. It relies on a p2p infrastructure while its primary objective is Voice over IP (VoIP) service delivery. Moreover, for data transmission it uses its proprietary Skype protocol, but the HTTP or HTTPS (HTTP over SSL) protocols might be used as well. As a result, it might not be clear how to classify such traffic. Likewise, due to strict policies enforced by firewalls and restrictions

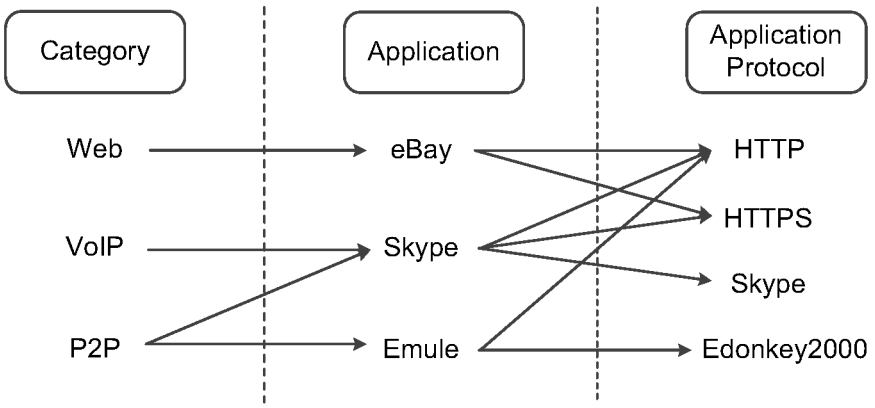

Figure 3.1: Classification goals.

for certain streaming and p2p applications, users commonly tunnel restricted traffic with Secure Shell (SSH) or SSL protocols. In both discussed examples, the classification results will be different according to classification goals. On the other hand, older applications defined their own communication solutions based on proprietary protocols such as BitTorrent or eDonkey2000, which makes the discussed problem trivial to solve. Nowadays, however, applications increasingly use other existing protocols such as HTTPS (cf. Figure 3.1) to tunnel their traffic and by doing so to bypass restrictions set by network configuration.

In the second part of the thesis, we propose a sampling detection scheme of SYN flooding attacks and portscan activity—we aim at classifying traffic according to its category. In Part III of the thesis, we attempt to solve even more detailed classification problem by proposing an accurate method for detecting Skype traffic and classifying its service flows such as voice calls, skypeOut, video conferencing, chat, file upload and download. In Part IV we propose a more general method to effectively classify application flows encrypted with TLS/SSL protocol, namely PayPal, MBank (an on-line bank service), Mozilla, Twitter, Opera, Gadu-Gadu (a popular Polish instant messenger), and Dropbox.

3.3 Classification Approaches

The selection of an appropriate approach used for traffic classification evolves with application development [4]. The variety of new Internet applications including services such as streaming, online gaming, p2p file sharing, or video/voice conferencing have intensified research efforts to discriminate against such applications. These, in turn, have inspired sophisticated obfuscation mechanisms. Figure 3.2 [4]

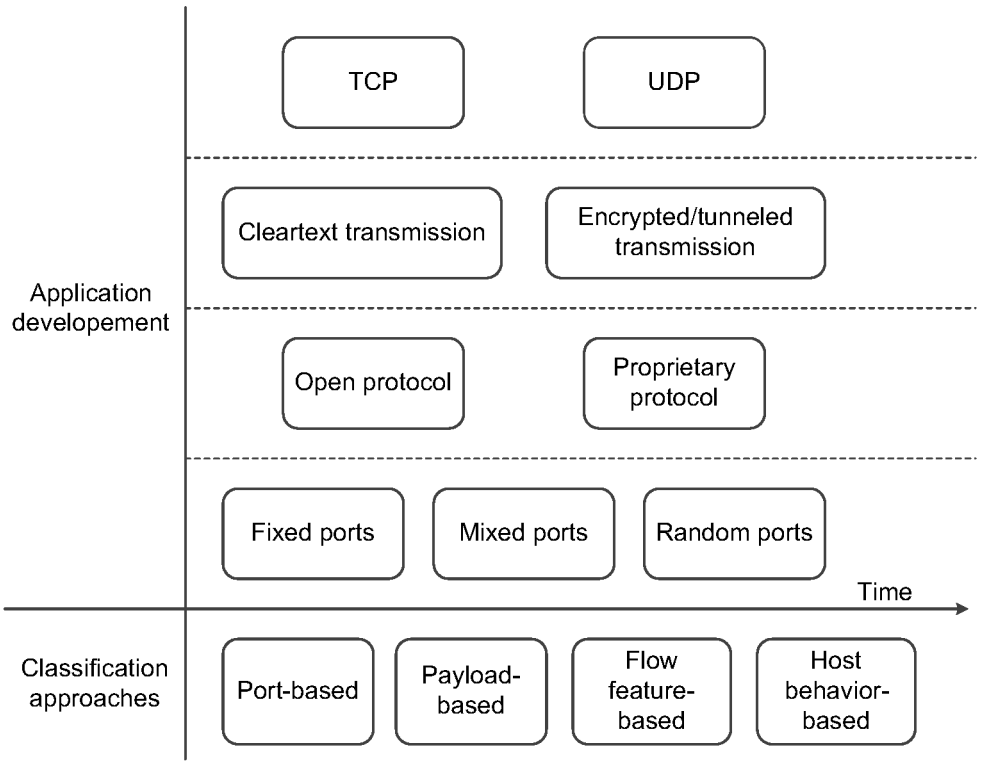

Figure 3.2: Trends in application development and classification approaches.

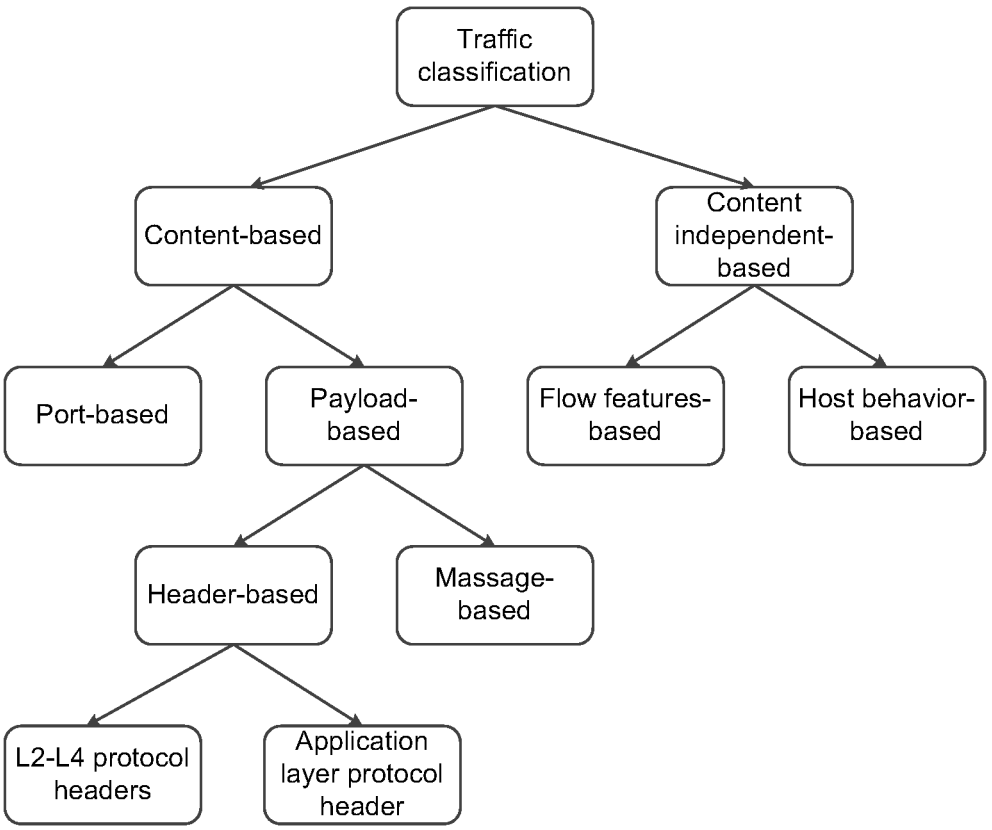

Figure 3.3: Approaches in traffic classification.

gives the first view of trends in application development over time with respect to four main classification approaches.

In the rest of the section, we present in detail a taxonomy for approaches in traffic classification. It encompasses four existing techniques developed in the traffic classification domain over almost two decades, namely two content-based methods consisting of port or payload inspection as well as flow feature-based and host behavior-based techniques (cf. Figure 3.3). Moreover, we propose some extensions to payload-based taxonomy based on the research presented in this thesis.

3.3.1 Port-based approach

Over 15 years ago, the ISPs and network administrators could accurately classify network traffic using UDP and TCP port numbers [14, 4]. Nowadays, new Internet applications tend to use unpredicted ports to evade traffic control (cf. Figure 3.2). A prominent example is Skype which puts significant efforts into bypassing restrictive

firewalls—it randomly selects ports and can switch to port 80 or 443 if it fails to establish a connection on dynamically chosen ports. As a result, simple inspection of port numbers is no longer a reliable classification mechanism [5, 15], especially when identifying applications.

Some recent studies critically revisit traffic classification including methods based on transport layer ports [16]. One of several insights of their studies is that ports still remain an important discriminator, particularly when combined with other features such as packets sizes, TCP flags and protocol information. However, their classification objectives are different from those presented in Parts III and IV. The methods evaluated in their studies aim at application protocol classification rather than in detailed application classification.

Maier et al. investigated the characteristics of residential broadband Internet traffic using packet-level traces augmented with the DSL session information [17]. Their most significant conclusion is that p2p is no longer dominant traffic in terms of bytes. HTTP once more seems to carry most of the traffic. Their classification method was based on a purely port-based approach, showing quite good results for their dataset. However, for a more detailed analysis aiming at the confirmation of the relevance of the port-based approach, they have examined the HTTP `Content-Type` header and the initial part of the HTTP body. Moreover, with this methodology, they only attempt to distinguish between p2p and HTTP. Finally, when analyzing application evolution, presented in Figure 3.2, we can observe that restricted applications tend to encrypt/tunnel their traffic through HTTPS or even through HTTP (e.g. Skype), which makes classification of p2p applications a particularly challenging task.

3.3.2 Payload-based approach

The second content-based approach involves inspecting the packet payload and for years, it was considered as the most accurate method. As soon as we can identify a unique payload-based signature, this technique can produce reliable classification results [5, 7]. Moreover, payload-based classifiers are often used to establish ground truth for other methods [16, 18]. Nevertheless, due to privacy issues and payload encryption other techniques have received more attention in the research community. We agree with the primary argument concerning users privacy, but we argue the common belief that payload-based methods always fail when traffic is encrypted [13, 4].

Risso et al. introduced a taxonomy of payload-based classification approaches regarding payload *verification* and *processing* methods. The former defines four degrees of verification. The first aims at locating some message signatures, the second

syntactical one, checks if the message is well formed, e.g., HTTP payload must contain HTTP headers. The third relates to protocol conformance, controlling client-server message exchange, while the semantic one verifies the type of object sent by the application layer protocol. The second taxonomy discusses payload-based processing methods, namely packet-based and message-based techniques, from a simple one that operates by checking some basic packet-header information to a sophisticated one that consists of inspecting and interpreting exactly what each application transmits.

In this thesis, we propose a more general taxonomy of payload-based classification concerning the type of data to be analyzed rather than verification or processing techniques. We propose to make a distinction between message-based and header-based analysis as shown in Figure 3.3. Moreover, we propose to separate the analysis of lower-layer protocol headers (until transport layer) from the application-layer protocol header since in some classification problems, the analysis based only on network and transport layer fields is sufficient, while in other cases, a more detailed application-layer protocol header analysis is required.

The method presented in Part II of this thesis illustrates an example of lower-layer protocol header analysis—we have proposed a scheme for detecting SYN flooding attacks and portscans that is based on identifying TCP SYN segments and corresponding ACKs. Furthermore, Sen et al. [7] presented an approach to identify the eDonkey protocol based on the application layer header analysis. More specifically, the authors discovered that signaling and downloading TCP packets have a particular eDonkey header on top of the TCP header. In the same paper, the authors propose a simple signature to reveal the Kazaa traffic based on the analysis of the HTTP protocol. Now, let us consider TLS/SSL encrypted traffic. Indeed, both message and ”old” application layer protocol header are encrypted so simple pattern verification methods based on signatures will fail. In Part III and IV of this thesis, we adopt a payload-based approach to demonstrate that it is still possible to effectively reveal and classify encrypted flows by inspecting ”new” application layer protocol, namely TLS/SSL. Moreover, Bonfiglio et al. have investigated the Skype traffic transported by the UDP protocol [18]. They concluded that the encrypted Skype UDP messages can be identified by examining the initial portion of the payload the so-called Start of Message (SoM) located on top of the header.

3.3.3 Host behavior-based approach

Host behavior-based approaches [2, 19] can potentially address some limitations of content-based methods. The approach is based on the analysis of the social behavior of network hosts and can be observable even when payload is encrypted.

More specifically, social interactions between communicating hosts are represented by graphs that visualize the "who-talks-to-whom" relationship. The classification consists of matching previously observed graphs with graphs resulting from the behavior of a host under examination [2].

BLINC, for example, proposes an interesting method based on observing and recognizing models of host behavior and then classifying its flows according to the models [2]. It analyzes patterns at three levels: (i) social level—it inspects the interaction with other hosts, (ii) functional level—it checks whether a host acts as a consumer or a provider of the service (or both), (iii) application level—it records the transport layer ports to identify the origin of the application.

Iliofotou et al. introduce the idea of Traffic Dispersion Graphs (TDGs) as a promising monitoring and classification tool [19]. Their work on TDGs represents a natural extension of the previous approach. More precisely, they propose a different way of looking at network traffic—they focus on network-wide interactions of hosts instead of modeling single host behavior. The same authors extended their previous work and developed a proof of the concept to detect p2p traffic [20]. Their application classification framework, evaluated on real-world backbone traces, can identify 90% of p2p flows with 95% precision.

3.3.4 Flow feature-based approach

The second fundamentally different group of content independent methods uses flow features such as average packet sizes, packet inter-arrival times, or flow durations (cf. Section 3.5). Features are computed over multiple packets grouped in flows and further used in the training process that associates sets of features with known traffic classes. The classification consists of a statistical comparison of unknown traffic with previously learned rules [21]. Flow feature-based approaches mainly include data mining techniques and machine learning algorithms. We do not, however, describe these techniques when discussing approaches based on the analysis of flow features because more and more other approaches including content-based ones use machine learning in classification purposes. Instead, we briefly discuss classification methods in Section 3.4.2.

For example, Bonfiglio et al. [18] presented a framework based on two complementary techniques to reveal Skype traffic. The second approach is based on a stochastic characterization of Skype traffic in terms of the packet arrival rate and packet length, which are used as features of a decision process based on a Naive Bayesian Classifier (NBC).

Moore et al. [3] proposed a statistical approach to classify traffic into different types of services based on a combination of flow features such as flow length, time

between consecutive flows, or inter-arrival times. The classification process using a bayesian classifier combined with a kernel density estimation method leads to an accuracy of up to 95%.

3.4 Methods

While machine learning algorithms using flow features for traffic classification received substantial attention, the content-based approaches mainly relied a simple pattern matching [13]. In recent studies, however, several hybrid solutions based on machine learning methods and taking into account content features were proposed [22, 23, 18, 24, 25, 12]. In this section, we present a brief survey of some popular methods (cf. Figure 3.4) used in classification approaches discussed in the previous section.

3.4.1 Pattern Matching

A few years ago, simple pattern matching combined with content-based approaches was one of the most accurate classification methods. However, pattern matching based on identifying the application level signatures is less effective (if possible) in the case of encrypted traffic. In one of the most interesting papers considering the pattern matching problem in recent years [7], the authors provide an efficient method for identifying five popular p2p applications through application level signatures. All of the proposed signatures, however, become useless once traffic encryption or tunneling methods are applied. Risso et al. [13], argue that content-based approaches are mainly based on pattern verification, thus they always fail in the case of encrypted traffic and often in the case of tunneled traffic. We argue, however, that a key challenge in encrypted traffic classification is to replace traditional pattern verification with more sophisticated methods based on statistical fingerprints.

3.4.2 Machine Learning

In the last few years, machine learning algorithms using flow features for traffic classification has received substantial attention [26, 27, 16, 3]. More recently, several authors have investigated the use of machine learning techniques with payload information [22, 23, 18, 25]. In general, machine learning algorithms are categorized into supervised learning and unsupervised learning (cf. Figure 3.4). Supervised learning requires some labeled data to generate models of applications of interest, whereas unsupervised learning clusters flows with similar characteristics. Since our

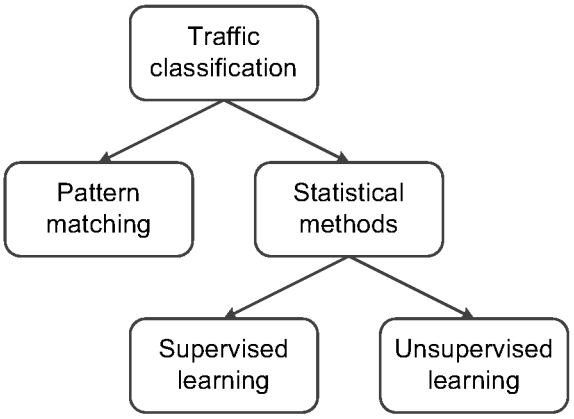

Figure 3.4: Methodologies in traffic classification.

main focus is traffic classification based on pre-labeled data rather than detecting new applications or flow clustering, in our studies, we concentrate on supervised algorithms. We briefly describe the algorithms we use in this thesis.

- **Kullback-Leibler Divergence** [28] is a measure of the difference between two probability distributions. The first one represents the distribution of a particular set of features corresponding to the flow to be analyzed and the second is the distribution corresponding to a known traffic model. Classification consists of comparing the distribution of the classified flow with all known traffic models and selecting the label corresponding to the smallest average divergence threshold.

- **First-order Homogeneous Markov Chain** [29] is characterized by a set of states S and the transition probabilities between the states. They are studied at discrete instants and each of them is dependent only on the preceding event. In the presented thesis, we use Markov chains to build stochastic models reflecting TLS/SSL protocol session states associated with some selected applications. Classification consists of comparing the Markov chain of a classified flow with all known Markov models and selecting the closest one.

- **Naive Bayes Classifier** [30] is a simple probabilistic classifier that applies the Bayes theorem with an assumption of the independence of input features describing an object. It analyzes the relationship between each feature and the application class to derive the conditional probabilities used later in flow classification.

This section provides only some insight into methods used in the thesis. The formal definitions are presented in relevant chapters.

3.5 Feature Selection

As it was highlighted in the previous section, the majority of traffic classification methods use some form of machine learning techniques to build traffic models from observed data. They share the general idea of measuring the distance of new objects from the learned models that represent particular traffic classes. In practice, however, the effectiveness of traffic classification frameworks strongly depends on the choice of traffic *attributes* or *features*. Two families of features have recently been used for traffic analysis. The first one consists of the in-depth analysis of the packet content, whereas the second one relies on flow-level statistics.

Various packet content features have been applied to traffic classification, such as an IP address [31, 32], a transport layer protocol [33], a packet (payload) size [33, 34, 35, 36, 37, 38], or particular values in TCP and UDP headers [31, 18], for example, a port number [5, 3, 39, 40, 41] and TCP flags [42, 43, 44, 45, 32, 41]. While packet header based features have proved to be effective in traffic classification and against some network attacks, other classification problems require more advanced payload processing techniques. Thus, Deep Packet Inspection (DPI) methods have been proposed to create some payload-based signatures based on an in-depth analysis of application layer data [23, 18, 22, 5, 2, 46, 18]. Moreover, features based on the relationship between various metrics have been applied in many classification problems [23, 22, 2]. For example, the detection methods based on matching TCP control segments such as SYN and FIN (or RST) pairs have been proposed in intrusion detection [42]. Furthermore, the approach based on the relationship between the number of destination IP addresses and ports for specific applications per source IP has been proposed in traffic classification [2].

On the other hand, researchers propose the use of flow-level features, such as flow duration [33, 3, 27, 47], a number of packets per flow [33, 31, 32], a variance and/or an average, minimum, maximum value of inter-arrival time [33, 31, 34, 18, 3, 26, 48, 27, 47, 49, 50], packet (or payload) sizes per flow (or per few first packets of the flow) [31, 51, 18, 2, 3, 26, 52, 27, 49, 50], a bit rate [51, 35], a round-trip time [35], a flow size [3, 27], or time between consecutive flows [3]. Moreover, joint application of some metrics have been proposed, for example, distribution of flow duration and number of packets transferred [53], or direction and packet size distribution [22].

In the second part of this thesis, we propose a scheme that relies on the simple and robust packet header feature based on matching TCP SYN segments to at least

one of multiple ACK segments coming from the server side. In Part III, we present a hybrid method that combines traffic flow features with complex DPI elements to identify Skype service flows (cf. Section 9.1.2). Finally, the last part defines a framework based on two complementary methods applying payload features for classifying application flows encrypted with TLS/SSL. The first proposed Markov Classifier takes into account message types in a TLS/SSL session as a classification feature while the second classifier considers the deviation between the timestamp in the TLS/SSL `Server Hello` message and the packet arrival time.

A common problem in the domain of traffic classification is to decide among different features to be used. The feature selection can be done manually, but a better strategy is to have a learning algorithm that decides which set of features is the best. The problem of automatic feature selection has been well studied in the context of traffic classification [54, 55, 56, 57] and anomaly detection [58]. In the third part of this thesis, we face the problem of selecting an appropriate subset of features called *attribute meters*. We applied a method called *forward selection* based on the Analysis of Variance (ANOVA) [59]. It consists of starting with an initial attribute in the model trying attributes one by one, and adopting them, if they improve the classification performance.

3.6 Intrusion Detection

In this section we focus on a special case of traffic classification, namely on intrusion detection. At first sight, the main difference between intrusion detection and, for instance, application identification is the number of traffic classes considered in the classification process (cf. Section 3.1). Moreover, in terms of traffic classification goals discussed in Section 3.2 the objective of intrusion detection is to categorize traffic as either intrusive or legitimate. Nevertheless, the crucial importance of network security resulted in decoupling intrusion detection from traffic classification.

Two detection approaches have received substantial attention in the research community, namely *signature-based* and *anomaly-based* detection. Although, they are opposite in nature, they share a common drawback—they require an in-depth knowledge of network traffic to be effective. Therefore, they are collectively referred to as knowledge-based detection approaches [60]. A relatively new research area in intrusion detection that can potentially overcome the limitations of knowledge-based approaches relies on unsupervised anomaly detection. In the rest of this section, we briefly discuss three approaches summarized in Figure 3.5

The signature-based approach [61] requires an extensive knowledge of security

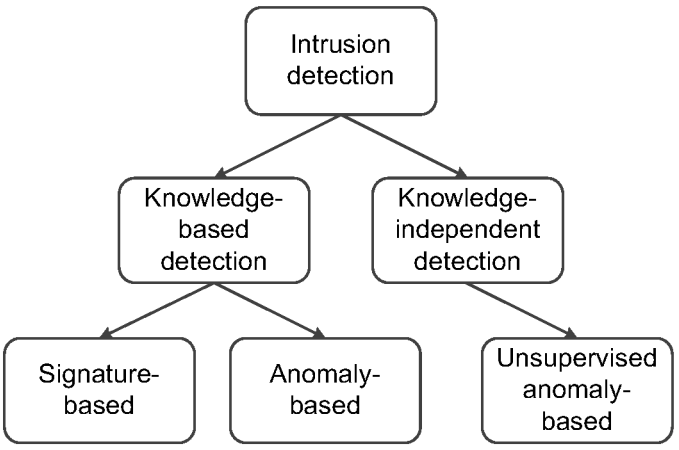

Figure 3.5: Methodologies in intrusion detection.

experts in order to identify some particular features of malicious traffic, so-called *signatures*. Once we have created a unique signature of a network attack, this technique will be highly effective in the detection of *well-known attacks* with a low False Positive Rate. It cannot, however, detect new intrusions and it requires frequent signature updating of previously seen attacks.

The anomaly-based approach [62, 63, 64, 65] attempts to estimate the normal-operational behavior of the observed system and to generate an alarm whenever it detects the deviation between the observed and the normal-operational traffic profile. This technique can potentially detect all intrusions, but has the limitation of a higher false alarm rate. Moreover, it requires a training phase to build profiles that correspond to legitimate traffic. This is particularly difficult, because some malicious activity can be hidden even in pre-labeled traffic that is assumed to be attack-free.

To overcome the problems related to knowledge-based approaches, unsupervised anomaly-detection methods have been proposed [66, 67, 68, 60]. They do not rely on pre-determined attack signatures or pre-labeled normal-operational traffic. Instead, using unsupervised learning machine algorithms, they automatically extract traffic models and detect outliers, i.e. structures that are detached from the main traffic patterns. As a result, they are capable of revealing previously unseen intrusion attempts.

Despite extensive research in the domain of intrusion detection, we consider it as a subclass of the traffic classification problem and we limit our discussion to a general description of detection approaches. Moreover, previously discussed issues related to traffic classification apply to the intrusion detection as well. In

Section 5, we focus our attention on more specific problems we had to face when proposing a scalable anomaly-detection *sampling* scheme of high-volume malicious traffic composed of SYN flooding attacks and low-volume portscan activity.

3.7 Ground Truth

The appropriate set of pre-labeled packet traces containing the so-called *ground-truth* information is one of the key aspects in any classification problem. The two most popular approaches employ the following procedures. The first one consists of the manual generation of Internet traffic by running a broad pool of applications on many machines. Nevertheless, such a dataset might not cover realistic application instances and traffic characteristics due to the lack of live, human interactions. The second approach assumes assigning traffic labels to all flows by means of DPI methods after packet capturing. However, simple pattern-matching techniques are not reliable anymore due to many obfuscation mechanisms and traffic encryption. Moreover, most of the existing methods deal with ground-truth flow labeling in the protocol domain [1, 16]. A recent ground-truth classifier that could potentially fulfill our requirements is based on a pre-installed client tool to supervise a kernel of each monitored host [69]. Even if the presented results are very promising, we need to find a number of users who consent to be monitored with the classifier. Finally, the same authors compare some previous methods based on the joint port analysis and payload inspection [70]. The experimental results demonstrate that, in many cases, the ground-truth data provided is incorrect. In this thesis, we do not rely on any of the systematic solutions presented above. Instead, we try to develop ad hoc methods that meet our current classification needs.

The second dataset used in the evaluation process in Part II has been collected on the link connecting an operational university campus network at the AGH University of Science and Technology in Cracow to the Internet. We have manually generated some network attacks by the use of common attacking programs available in BackTrack linux security distribution [71] against servers set up especially for this purpose. The traffic has been captured on the border router that monitors packets generated in the controlled part of the network. The methodology is reliable in obtaining the ground-truth data, but, in addition, it enables us to take advantage of rich background traffic generated by students including recent p2p applications as well as standard services like web, ftp, or mail.

In Part III, we had to face even more challenging tasks, namely, generating and labeling Skype service flows. Manual generation of each service separately in a closed laboratory environment enabled us to effectively obtain the ground-truth

information. It was particularly difficult due to the p2p nature of Skype resulting in spreading services on several TCP connections. Another challenge was to manually distinguish between service flows and the corresponding signaling Skype traffic.

To establish the ground truth in the third part of this thesis, we have developed a simple Domain Name System Classifier (DNSC) to extract encrypted application flows according to their domain names. More specifically, DNSC matches hostnames to signatures of well known applications, such as *opera* in case of Opera or *twitter*, *twttr* in case of Twitter. The solution presents the number of constraints like a rather limited classification scope. For example, we cannot label Skype flows, because in general, we are not able to convert IP addresses to domain names. On the other hand, if the mapping between an IP address and a domain name is possible, the method can classify flows with a very high confidence level.

3.8 Criteria for Classification Performance

To evaluate any classification method we need to define criteria for classification performance. In this section we discuss the metrics we use to quantify the performance of our classifiers, namely the False Positive Rate (FPR), the True Positive Rate (TPR), which is also known as Recall, and Precision. They are defined as follows:

$$FPR = \frac{FP}{FP + TN}, \tag{3.1}$$

$$TPR = Recall = \frac{TP}{TP + FN}, \tag{3.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

The following metrics are built upon the concept of True Positives (TPs), True Negatives (TNs), False Positives, (FPs), and False Negatives (FNs). These notions are often used in anomaly detection and traffic classification where each object is placed into one of several classes.

To give the reader intuition about the statistical metrics to be used in this thesis, let us make an analogy. Suppose we want to make a blind test of beer recognition to test the knowledge of beer according to the brewing process. We classify them into two categories, i.e., either lager or ale. We select a set of 100 beers—60 of these are lagers, whereas 40 represent ale. Let us assume that you have classified 70 beers as being lagers. Actually, 50 of these are lagers, which correspond to the

number of True Positives, but 20 are ales, which represent the number of False Positives. Moreover, you categorize 30 beers as ales. Of these, 10 represent in fact the lager type, whereas 20 of them are indeed ales. Let us focus on the lager type. True Positive Rate (or Recall) is the number of beers correctly categorized as lager divided by the total number of beers that are actually lagers—you have $TPR = Recall = 50/(50+10) = 0.833$. Moreover, False Positive Rate is the number of falsely classified beers as the lager type to the total number of non-lager beers, $FPR = 20/(20 + 20) = 0.5$. A complementary measure to Recall is Precision, that is the number of correctly classified lager beers to all beers classified as the lager type, thus, $Precision = 50/(50 + 20) = 0.714$.

To assess the performance of the proposed classification methods in the second and fourth part of the thesis we use True Positive and False Positive Rates as classification metrics, whereas in Part III we use Precision, Recall, and $F\text{-}Measure$ as classification metrics. $F\text{-}Measure$, which combines Precision and Recall, is defined as:

$$F\text{-}Measure = \frac{2 * Precision * Recall}{Precision + Recall}, \tag{3.4}$$

For more specific examples, please refer to the respective sections describing the criteria for classification performance of each proposed method.

3.9 Summary

So far, we have introduced some background information on traffic classification important to understand methodologies proposed in this thesis. In Table 6.1, we present the summary of the discussed issues according to the further described solutions. In the remaining parts of the thesis, we first propose an accurate sampling scheme for defeating SYN flooding attacks and TCP portscans, while in the two following parts we propose two frameworks for classifying application flows in encrypted traffic.

Table 3.1: Summary of traffic classification issues according to proposed solutions

Research area	Traffic classification		
Sub-domain	Application classification		Intrusion detection
Proposed solutions	Classifying service flows in the encrypted Skype traffic	Classifying TLS/SSL encrypted application flows	A sampling scheme for detecting SYN flooding attacks and portscans
Classification goals	Application	Application	Category
Classification approaches	Hybrid: header- (L7 protocol header) & flow feature-based	Header-based (L7 protocol header)	Header-based (L2-L4 protocol headers)
Methods	Supervised ML: SPID algorithm based on Kullback-Leibler Divergence	Supervised ML: K-L Div. & Markov Chain & Naive Bayes	Anomaly-based: Rate limiting method
Features	Packet size, direction, byte frequencies, byte pairs reoccurring, etc.	Message types and timestamps from a TLS/SSL session header	Rate of TCP SYN to ACK segments
Ground truth/ datasets	Manually generated flows in a closed laboratory environment	University campus datasets pre-labeled with DNSC classifier	University campus datasets with manually generated attacks
Performance Metrics	True Positive and False Positive Rates	Precision, Recall, and F-Measure	True Positive and False Positive Rates

Part II

An Accurate Sampling Scheme for Detecting SYN Flooding Attacks and Portscans

Chapter 4

Introduction

Contents

Distributed Denial-of-Service (DDoS) attacks and portscan activity strongly influence Internet security. According to a NANOG report [72], the major cause of denial of service attacks is TCP SYN flooding that consists of sending many SYN segments from a large number of compromised computers. It prevents victim machines or even whole subnetworks from offering a service to their legitimate users. A portscan activity is usually a precursor for an intrusion attempt—a compromised computer sends multiple SYN segments to probe other hosts for open ports to gain control over more computers that become potential attackers. SYN flooding and portscans differ in terms of intensity, behavior, and security threats so usually they are handled independently. However, both types of traffic exploit the inherent asymmetry in the TCP three-way handshake mechanism and the fact that the victim cannot authenticate TCP SYN segments it receives. As a result, malicious packets can easily reach the victim without its approval.

Among various defense mechanisms, SYN flooding detection mechanisms placed in border routers have received much attention in recent literature [42, 43, 44, 73, 74]. All these methods take advantage of the relationship between TCP control segments responsible for connection establishment and release. However, they all may fail when routers sample traffic by inspecting only some packets. Efficient traffic monitoring requires advanced sampling techniques to limit the volume of inspected data. Sampling consists of partial observation of the network traffic and drawing conclusions about the whole behavior of the system. Detecting DDoS attacks and portscans becomes more difficult when routers sample packets.

4.1 Contributions of Part II

In this part, we propose a novel and scalable sampling detection scheme of high-volume malicious traffic composed of SYN flooding attacks and low-volume portscan activity. The scheme examines TCP segments to find at least one of multiple ACK segments coming from the server. In this case, it concludes that the connection was successfully established so its opening SYN segment was not a part of a SYN flooding attack or portscan activity. This principle is particularly suitable when routers sample packets with very low rates. We combine the proposed method with a rate limiting scheme that controls traffic rates and compare with three other representative detection methods. We show that our method achieves a high attack detection rate (True Positive Rate). In comparison with existing methods, we significantly reduce the False Positive Rate, i.e., when legitimate packets are classified as malicious ones.

We also study the impact of three basic packet sampling techniques proposed by PSAMP IETF working group [75] on our detection scheme. The results reveal that even the simplest and the most commonly used sampling technique—*systematic* sampling also known as *deterministic* sampling [76], performs fairly well under low sampling rates when combined with our detection and rate limiting method. Unlike some other proposals that used network simulations or experiments on obsolete data sets with outdated background traffic, we validate our scheme on two recent data sets of network traces captured during real network attacks.

4.2 Relevant Publications for Part II

[45] Maciej Korczyński, Lucjan Janowski, and Andrzej Duda. An Accurate Sampling Scheme for Detecting SYN Flooding Attacks and Portscans. *2011 IEEE International Conference on Communications (ICC'11)*, pages 1–5, June 2011

[77] Maciej Korczyński and Lucjan Janowski. Implementation of The Algorithm to Detect and Prevent Network Attacks Based on Rate Limiting Method. *Conference on Next Generation Services and Networks - the Technical Aspects, Application and Market*, November 2010

[78] Gilles Berger-Sabbatel, Maciej Korczyński, and Andrzej Duda. Architecture of a Platform for Malware Analysis and Confinement. *3rd INDECT/IEEE International Conference on Multimedia Communications*, May 2010

[79] Karol Adamski, Maciej Korczyński, and Lucjan Janowski. Trace2Flow. *3rd NMRG Workshop on Netflow/IPFIX Usage in Network Managament*, 2010

Chapter 5

Issues in Traffic Analysis

Contents

5.1 Analyzing TCP Connections

Analyzing TCP connections is one of the most important issues to address in the case of SYN flooding attacks and portscans. To open a connection, a client sends an initial SYN segment. Upon its reception, the server allocates some resources in the backlog queue and replies with a SYN/ACK segment. Finally, the client returns an ACK segment (further called *Client ACK*) to complete the three-way handshake. Then, communication goes on until the client or the server sends a segment with the FIN flag set, a RST segment, or the connection times out. The potential for exploiting this behavior for denial of service lies in the early allocation of the server resources. During a TCP SYN flooding attack, the attacker generates multiple SYN requests without sending the Client ACK to complete the connection establishment. The requests can quickly exhaust the server memory so it cannot accept more incoming connection requests.

SYN scanning is fairly similar to TCP SYN flooding attacks: an attacking computer tries to identify vulnerable hosts by sending multiple TCP SYN segments. If a port is open, the server responds with a SYN-ACK segment, the port scanner completes the three-way handshake and immediately closes the connection with a RST segment.

Several authors proposed interesting detection methods that can operate in border routers to detect attacks and block them near their sources [42, 43, 44, 73, 74]. They take advantage of the relationships between the TCP control segments: the appearance of a SYN segment implies further SYN/ACK, Client ACK, and FIN or RST segments. However, if we want to apply sampling at border routers of Intranets or parts of operator networks for improved monitoring efficiency, considering only

a small part of packets may degrade the detection capacity of all existing methods, because low probability of sampling essential control segments is fairly low. For instance, without sampling, the ratio between SYN/ACK and Client ACK segments should be around 1 for regular traffic while it may be different when routers use high sampling rates. Because of this sampling effect, the existing methods result in poor detection performance especially with respect to the False Positive Rate.

5.2 Sampling Techniques

We consider three basic and most commonly used count-based sampling techniques: systematic, random 1-out-of-N, and uniform probabilistic sampling (cf. Figure 5.1) proposed by the PSAMP IETF working group [75] and thoroughly investigated in the literature [80]. They present the advantage of simple implementation with low CPU and memory requirements.

Systematic sampling takes every *N-th* packet, whereas random 1-out-of-N sampling randomly chooses one packet in every bucket of size *N*. Finally, uniform probabilistic sampling analyzes every packet with the same small probability. Systematic sampling, also known as deterministic sampling, is usually used in current network devices, one example being the Cisco Netflow protocol [76].

Some previous work addressed the problem of how sampling techniques influence the anomaly detection process [81]. The authors focused on portscan anomalies and evaluated some representative anomaly detection techniques. Later, they extended this work and examined various kinds of sampling methods with respect to volume and scanning anomalies [82]. They concluded that packet sampling can introduce a fundamental bias by changing traffic features and they pointed out the need for better measurement techniques. Other authors considered the impact of sampling methods on various detection metrics examined on traces with TCP SYN flooding attacks [80]. Their results reveal that systematic sampling does not perform well under low sampling rates when the detection process depends on specific packet characteristics like TCP flags. Our detection scheme also overcomes the limitation of systematic sampling and it becomes as appropriate method as other more enhanced sampling techniques.

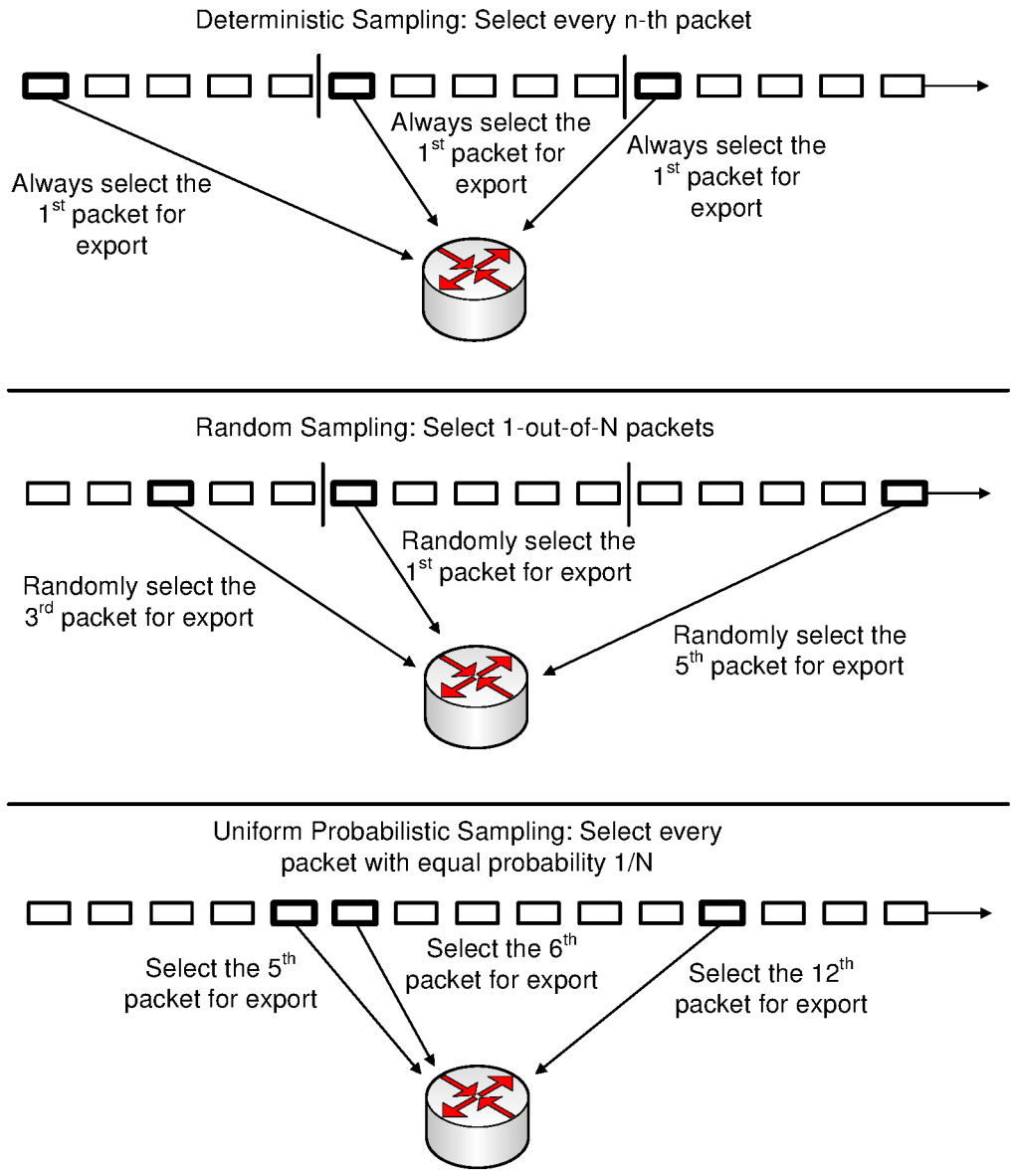

Figure 5.1: Principle of three sampling methods.

Chapter 6

Design and Evaluation

Contents

6.1 Principles of the Detection Scheme

To overcome the limitations of methods that match pairs of TCP control segments, we propose a novel method not limited to the analysis of the three-way handshake or connection termination. To make it insensitive to sampling, we propose to find at least one of multiple ACK segments coming from the server instead of looking for a single control segment like SYN/ACK, Client ACK, FIN, or RST. In other words, to detect legitimate established connections, we take advantage of the fact that all segments originated from the server with the ACK flag set on and the SYN flag set off indicate a successfully established connection. Obviously, when we sample packets, the probability that the sampled packet contains one of multiple ACK segments coming from the server is much more greater than when we try to detect a SYN-SYN/ACK pair. This approach decreases the False Positive Rate

and does not influence the True Positive Rate, because in the case of SYN flooding attacks as well as portscans, there are almost no corresponding ACK segments coming from the server. Finally, it is impossible for the attacker to avoid detection by spoofing control segments.

The proposed scheme is placed in a border router that monitors packets generated in the controlled part of the network (e.g. an Intranet or an enterprise LAN) to confine the possible malicious activity close to the source of an attack. It is composed of three modules: the first one validates outgoing TCP segments, the second one processes corresponding control segments, while the third one changes the packet filter list if needed.

We combine the method with a rate limiting scheme. If the traffic rate is less than or equal to a predefined rate for a given IP address, it is allowed to pass the filter of outgoing traffic, whereas traffic that exceeds the rate is dropped or delayed.

We provide a detailed description of the proposed defense scheme below.

6.1.1 TCP History Check

For each sampled packet, we extract its source and destination IP address and place them in the Source IP List (SIPL) and the Destination IP List (DIPL), respectively. We also extract other information such as timestamps, sequence numbers, and ACK sequence numbers. When the router samples any outgoing TCP SYN segment, the module checks if a timeout has elapsed. Depending on the result, it either resets the source and destination IP lists and allows the segment to pass or it increases request counter R_{src} corresponding to the particular source IP address by a positive integer. If there are more unacknowledged SYN segments originating from the specific source IP address and $R_{src} > R_{src}^{max}$, then this module decides that the segments are parts of portscan activity and inserts the source IP address in the filter blacklist. Moreover, the module increases request counter R_{dst} by a positive integer for a particular destination IP address. If $R_{dst} > R_{dst}^{max}$, it means that there is an excessive number of connections to the destination address. Then, as this behavior may indicate host scan activity or a SYN flooding attack, the module updates the filter blacklist to block packets that follow.

6.1.2 TCP Validation Check

The goal of this module is to overcome the problem of losing some useful information because of sampling. It analyzes TCP control segments to determine whether the three-way handshake was successfully completed. Any incoming segment from the server side with the ACK flag set and SYN flag disabled indicates

Table 6.1: Summary of packet traces

Trace	Attack ratio in packets/sec (%)
SYN flooding	4000 packets/sec (20%)
Host scan	120 packets/sec (9%)
Network scan	80 packets/sec (5%)
Clear1	0 packets/sec
Clear2	0 packets/sec

that the particular connection has been successfully established. In this case, the module decreases the R_{src} (R_{dst}) counter, because the connection becomes legitimate. Consequently, the requirement $R_{src} > R_{src}^{max}$ ($R_{dst} > R_{dst}^{max}$) might not be valid any more, so the module will eventually update the packet filter blacklist to permit further outgoing TCP requests from/to the specified IP address.

6.1.3 Filtering

This module applies all changes to the Access Control List (ACL) in the border router so that it will discard all malicious segments.

6.2 Evaluation Results

To evaluate the method and compare it with the previous work, we have developed a prototype in the Matlab environment. We use the open source TracesPlay program [83] to read traces and to directly put the required data into Matlab.

6.2.1 Dataset Description

We have validated our scheme by means of trace-driven simulations on two data sets: the first traces were gathered on an operational university campus network at the National Technical University of Athens (NTUA) with an average traffic of 70-80 Mbits/sec and 20000 packets/sec. It contains a Distributed Denial of Service attack (TCP SYN flooding attack) captured on May 21, 2003 against a single host inside the NTUA campus. The second set has been collected on the link connecting an operational university campus network at the AGH University of Science and Technology in Cracow with a limit of 45 Mbits/s for incoming and 22 Mbits/s for outgoing traffic. In the evaluation presented in this thesis, we have used set of four traces collected on March 24, 2010 containing host scans and port scans originated from the campus network as well as packet traces without malicious activity (cf.

Table 6.1). Moreover, traces contain rich background traffic including recent p2p applications as well as standard services like web, ftp, or mail. Note that all traces are bidirectional. We consider them as a meaningful set of traces—if the data set is not recent, we cannot trust evaluation results especially when we consider the False Positive Rate because of unrealistic background traffic (new applications have different traffic characteristics from old ones). Some previous work proposed other detection algorithms [84, 85], unfortunately the evaluation process uses outdated data sets.

6.2.2 Criteria for Detection Performance

We consider two meaningful metrics to evaluate the performance of detection methods: the True Positive Rate (TPR) and the False Positive Rate (FPR) (cf. Eq. 3.1 and 3.2). Such rates are usually presented as the Receiver Operating Characteristics (ROC) curve by plotting TPR as a function of FPR. As attack detection is a Boolean action, the ROC curve is useful for network operators, because it indicates how to find the right tradeoff between the False Positive and True Positive Rates. However, in our evaluation, we have separated both values and presented them as a function of the sampling rate, because the evaluation is also based on traces that do not contain malicious activity.

6.2.3 Comparing with Existing Detection Schemes

To evaluate our scheme, we have compared it with other three representative detection schemes that leverage TCP relationships: SYN-SYN/ACK, SYN-FIN, and SYN-Client ACK. The key point of schemes based on matching SYN-SYN/ACK and SYN-Client ACK pairs is the need of finding the corresponding SYN/ACK or Client ACK segment after the first SYN segment. The time interval between them is the RTT (Round Trip Time), usually less than 500ms for more than 90% of connections. Therefore, the methods have to inspect all control segments during at least this interval to conclude that the connection was successfully established. The detection methods based on matching SYN-FIN (or RST) pairs, simply waits for the corresponding FIN (or RST) segment.

6.2.4 Calibration Process

We had to face the problem of setting the right rate limiting thresholds, i.e., maximum values of the request counters corresponding to a regular traffic pattern. We have calibrated them for every examined trace in order to achieve a high TPR with no FPs regardless of the detection method when we analyze all packets.

Table 6.2: Rate limiting thresholds obtained during the calibration process for particular traces.

Trace	R_{src}^{max} (packets)	R_{dst}^{max} (packets)
SYN flooding	200	1300
Host scan	300	100
Network scan	400	240
Clear1	50	170
Clear2	120	90

The calibrated values (cf. Table 6.2) reflect the relation between outgoing SYNs per destination and per source and corresponding control packets (SYN/ACK, Client ACK, FIN). In long-term regular conditions, the TCP semantics requires a one-to-one match between TCP requests and control segments. Nevertheless, in reality there is always quite huge difference between the number of SYNs and SYN/ACK, Client ACK or FIN packets. Nowadays, the major cause of this difference is the legitimate p2p traffic that initiates TCP connections to unreachable seeds. We have empirically found that the rate limiting thresholds expressed in packets are directly proportional to the sampling rate, which alleviates the problem of losing potentially useful data during the sampling process.

6.2.5 Influence of the Sampling Process on Different Detection Schemes

In our experiments, we have evaluated the influence of uniform probabilistic sampling on the proposed method and compared it with other three schemes. We have decided to choose this particular sampling method, because it is claimed to be more effective in the process of packet selection compared to systematic sampling [80].

We have repeated all simulations to obtain 95% confidence intervals computed according to the bootstrap method [86].

As shown in Figure 6.1, all methods present approximately the same high TPR and very low FPR in case of TCP SYN flooding attacks. Similar results for all four methods are due to setting high rate thresholds corresponding to regular traffic for this particular trace. As we can observe, TPR curves of all detecting schemes are similar until 0.008% when sampling process increases randomness in the results. As far as FPR is concerned, we can see that SYN-SYN/ACK, SYN-FIN, SYN-Client ACK methods deviate from our scheme, but differences are insignificant. For

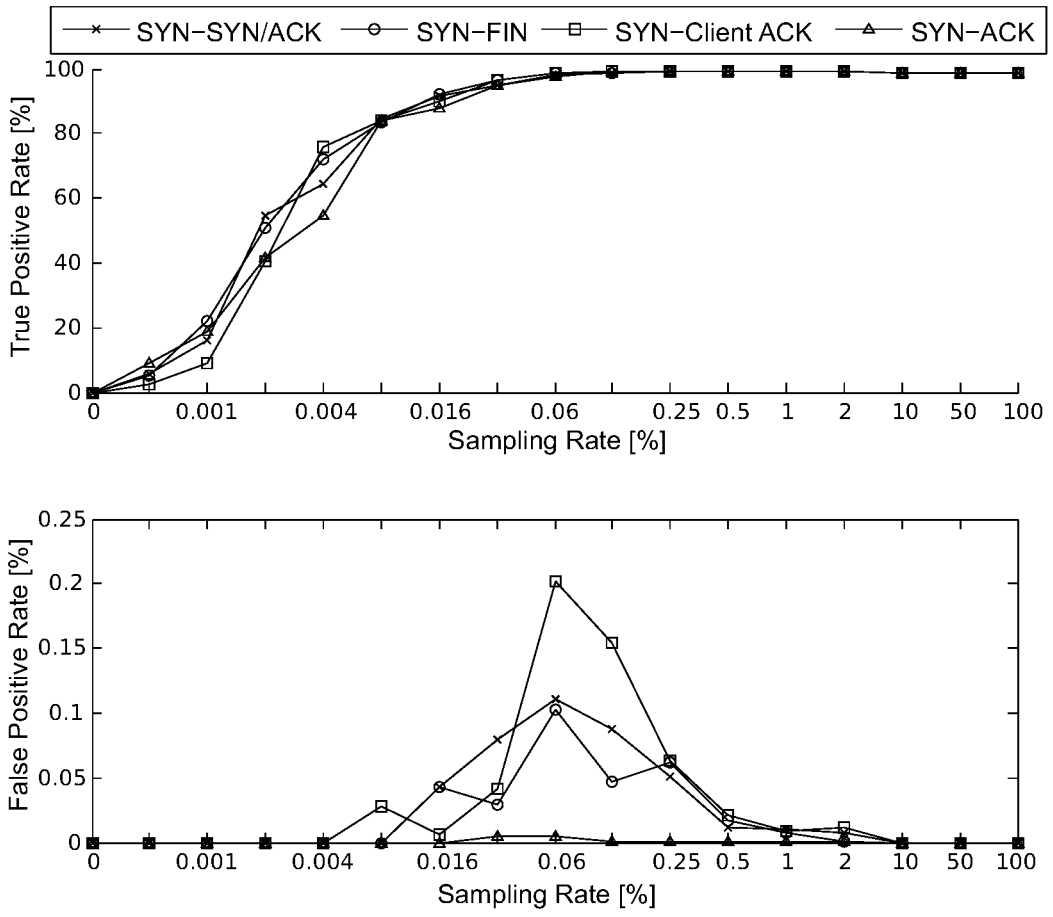

Figure 6.1: Comparison of TPR and FPR for schemes based on analyzing SYN-SYN/ACK, SYN-FIN, SYN-Client ACK, and SYN-ACK segments in the case of TCP SYN flooding.

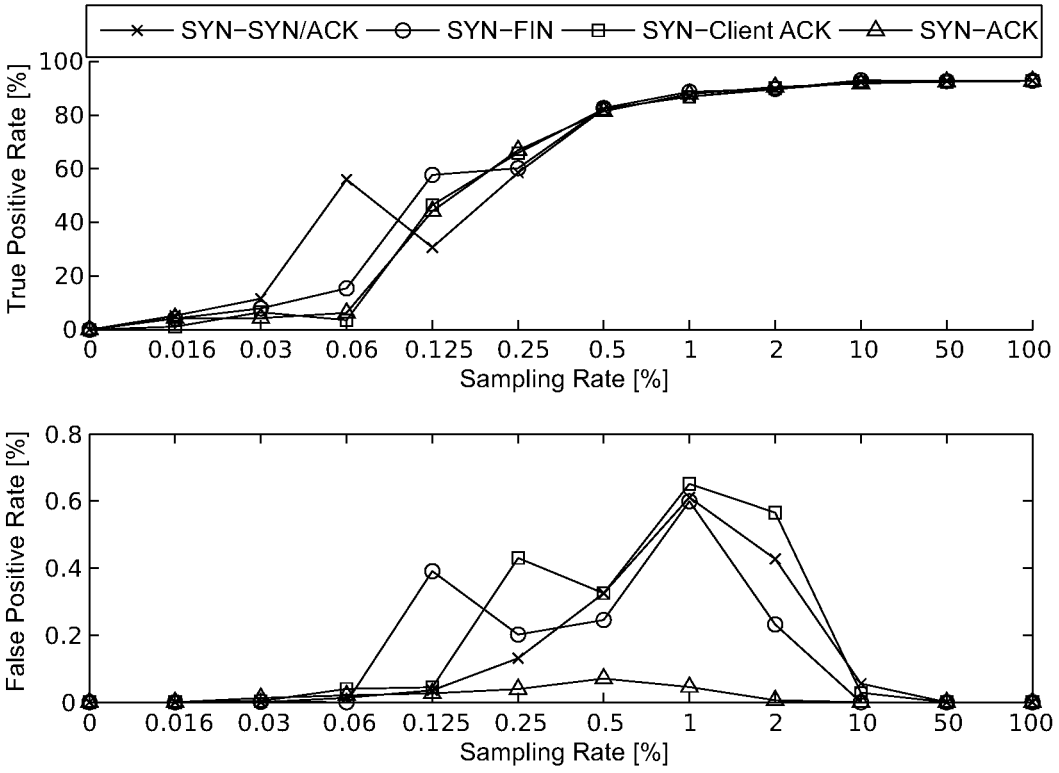

Figure 6.2: Comparison of TPR and FPR for schemes based on analyzing SYN-SYN/ACK, SYN-FIN, SYN-Client ACK, and SYN-ACK segments in the case of host scan activity.

sampling rates lower than 0.008%, FPs are completely eliminated by the sampling process itself.

Our experiments considered host scans (cf. Figure 6.2) as well as network scans (cf. Figure 6.3). The results, however, demonstrate similar behavior with respect to the attacking rate (cf. Table 6.1). Therefore, we only discuss the corresponding results for host scan activity (cf. Figure 6.2). Again, all four detecting schemes show nearly the same behavior until 0.5% when the sampling process introduces randomness in the results. It is slightly higher in comparison with the case of the TCP SYN flooding attack due to a lower attacking ratio of host scans (cf. Table 6.1). Here, we can observe that SYN-SYN/ACK, SYN-FIN, SYN-Client ACK methods result in some FPs that do not, however, exceed 0.8%.

Figure 6.4 presents the corresponding results for the packet trace without malicious activity marked in Table 6.1 as Clear1. Obviously, only FPR is presented and the results differ from the previous ones. In the following trace, rate limiting thresholds corresponding to legitimate traffic were established lower than in the previous

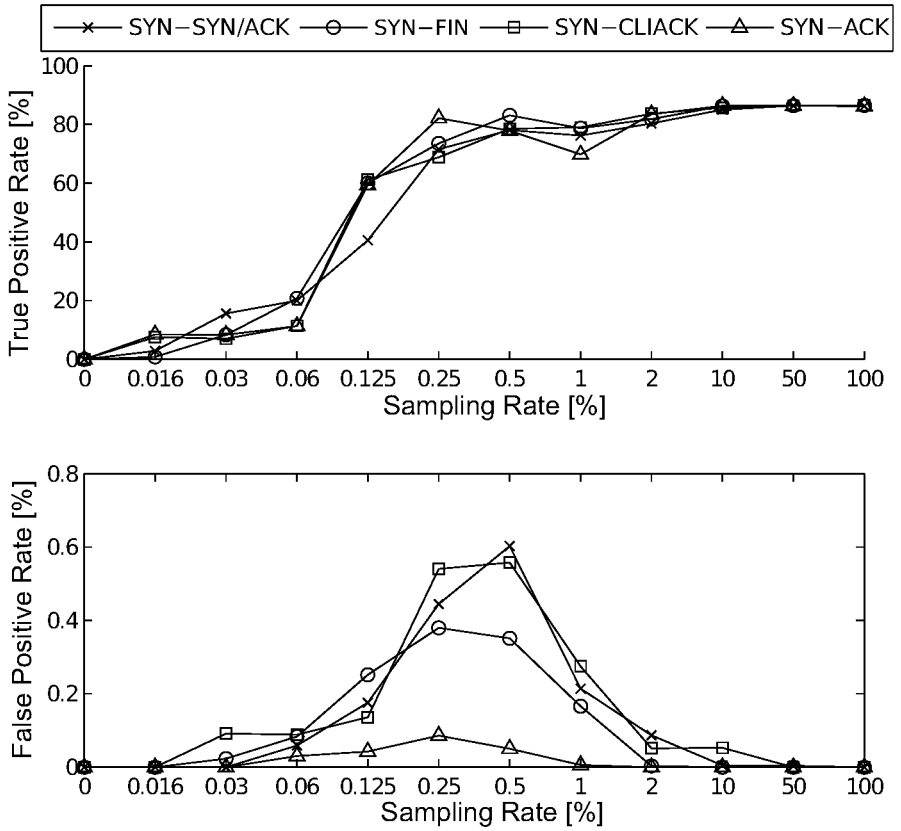

Figure 6.3: Comparison of TPR and FPR for schemes based on analyzing SYN-SYN/ACK, SYN-FIN, SYN-Client ACK and SYN-ACK pairs in the case of the network scan activity.

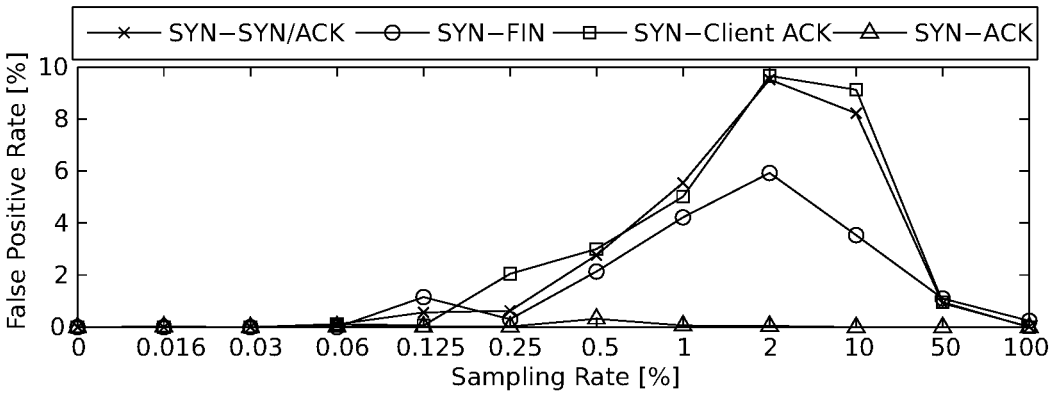

Figure 6.4: Comparison of FPR for schemes based on analyzing SYN-SYN/ACK, SYN-FIN, SYN-Client ACK, and SYN-ACK segments in the case of the Clear1 trace without malicious activity.

cases. The results reveal that only our detection method based on sampling ACK segments coming from the server instead of looking for a single control segment to detect legitimate established connections works well. When we reduce sampling rate to 2%, we can observe that detection schemes based on analyzing control segments (SYN/ACK, Client ACK and FIN) result in up to 10% of the False Positive Rate while our detection method still presents almost the same behavior as the unsampled one.

Figure 6.5 corresponds to the results for the second trace without malicious activity marked in Table 6.1 as Clear2. Again, our method becomes the best choice under the sampling process. We have observed the minimal False Positive Rate not exceeding 0.1% in case of our detection scheme. Other three methods present much worse performance giving FPR exceeding 2% up to 7% at 2% sampling rate.

To conclude, the evaluation shows that our method can significantly reduce FPR in comparison with other three methods and still accurately detect both high-volume DDoS attacks as well as low-volume scanning activity even if the method does not analyze all packets. Good performance comes from two features: first, at the instant of an attack, there are almost no ACK segments coming from the victim side so we obtain the high True Positive Rate. Second, the method benefits from the novel approach to identify legitimate TCP connection request by sampling one of many ACK segments sent by the server to a legitimate client.

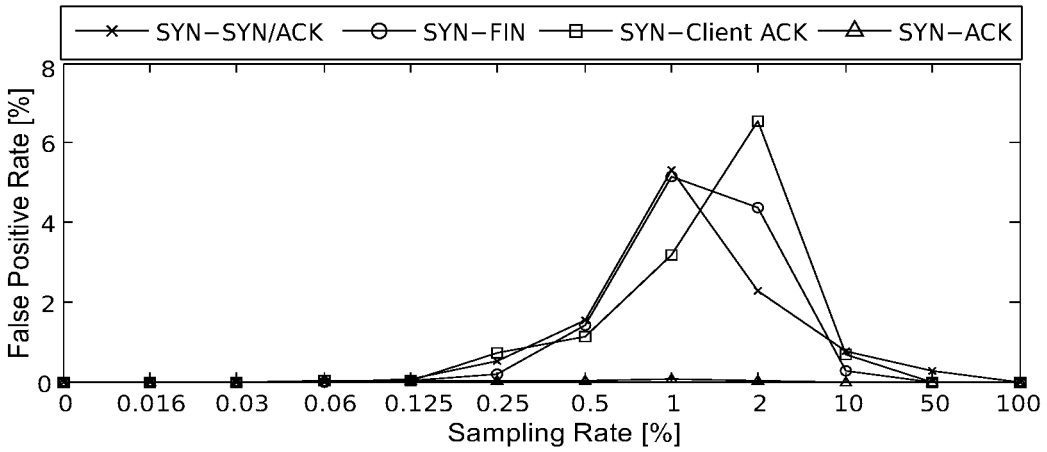

Figure 6.5: Comparison of FPR for schemes based on analyzing SYN-SYN/ACK, SYN-FIN, SYN-Client ACK, and SYN-ACK segments in the case of the Clear2 trace without malicious activity.

6.2.6 Impact of Sampling Techniques on the Proposed Scheme

Let us now examine the detection performance of our scheme under three representative sampling techniques described in Section 5.2.

The results presented in Figure 6.6 correspond to a high-volume TCP SYN flood attack under the sampling process. We can observe that all three curves are similar regardless of the sampling method even at the sampling rate as low as 0.008%.

The results corresponding to low-volume portscan activity like host scans (cf. Figure 6.7) and network scans (cf. Figure 6.8) show similar behavior. Again, our detecting scheme shows nearly the same performance for all examined sampling methods until 0.125% in the case of host scans and 2% in the case of network scans when the sampling process itself introduces randomness in the results.

For two other packet traces related to the traffic without network attacks and marked in Table 6.1 as Clear1 and Clear2, FPR does not exceed 0.4% in the worst case at the sampling rate 0.5% irrespective of the sampling method (cf. Figure 6.9 and Figure 6.10).

To sum up, the evaluation process indicates fairly similar behavior regardless of the sampling technique when we benefit from our novel detection scheme combined with the rate limiting scheme. This is due to the fact that this scheme is based on detecting segments with the ACK flag set. Such segments seem to appear evenly in the traffic thus resulting in good performance under sampling even if systematic sampling is used.

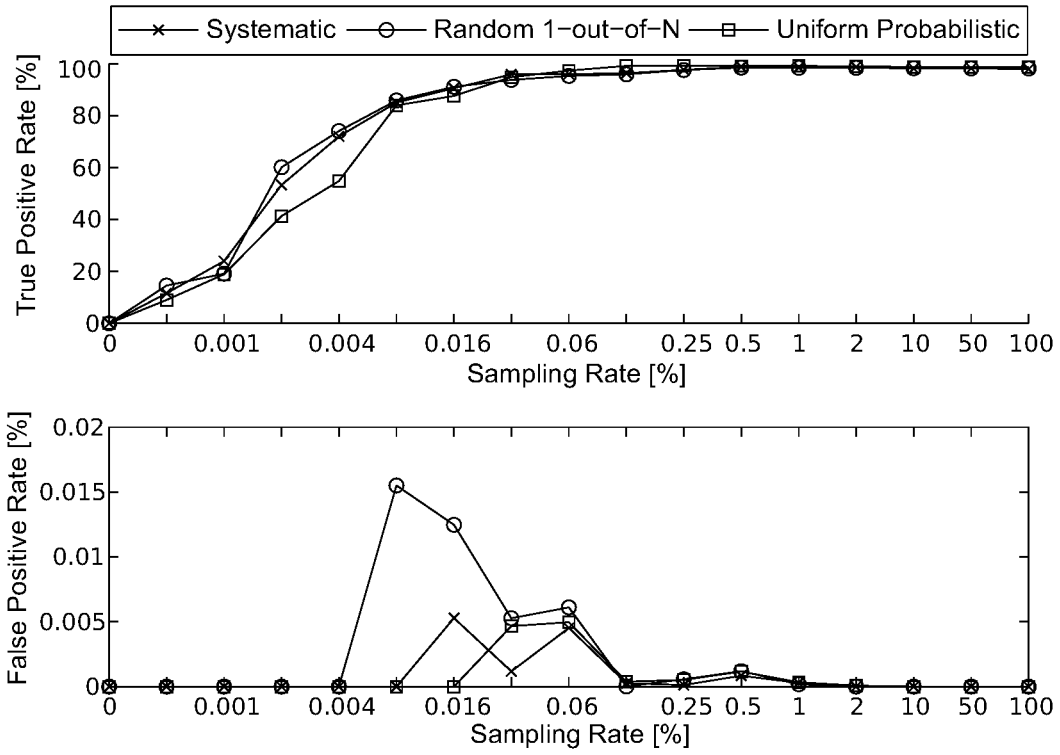

Figure 6.6: TPR and FPR—the influence of three sampling methods on the proposed scheme based on analyzing SYN-ACK segments in the case of the TCP SYN flooding.

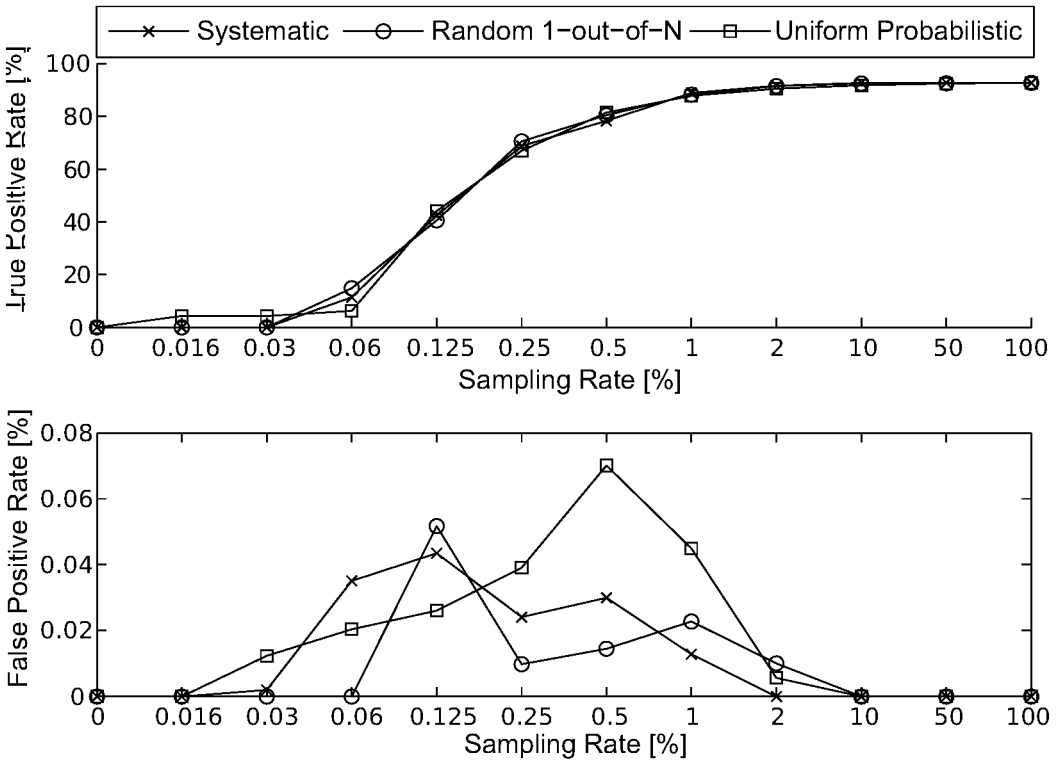

Figure 6.7: TPR and FPR—the influence of three sampling methods on the proposed scheme based on analyzing SYN-ACK segments in the case of host scan activity.

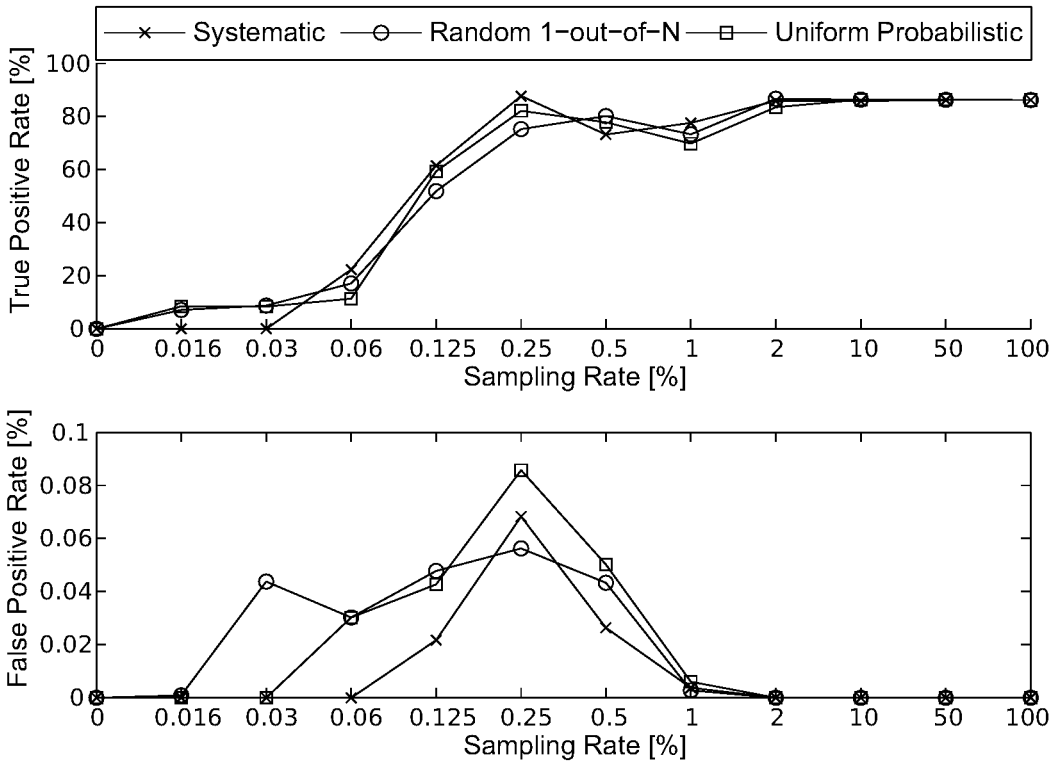

Figure 6.8: TPR and FPR—the influence of three sampling methods on the proposed scheme based on analyzing SYN-ACK segments in the case of network scan activity.

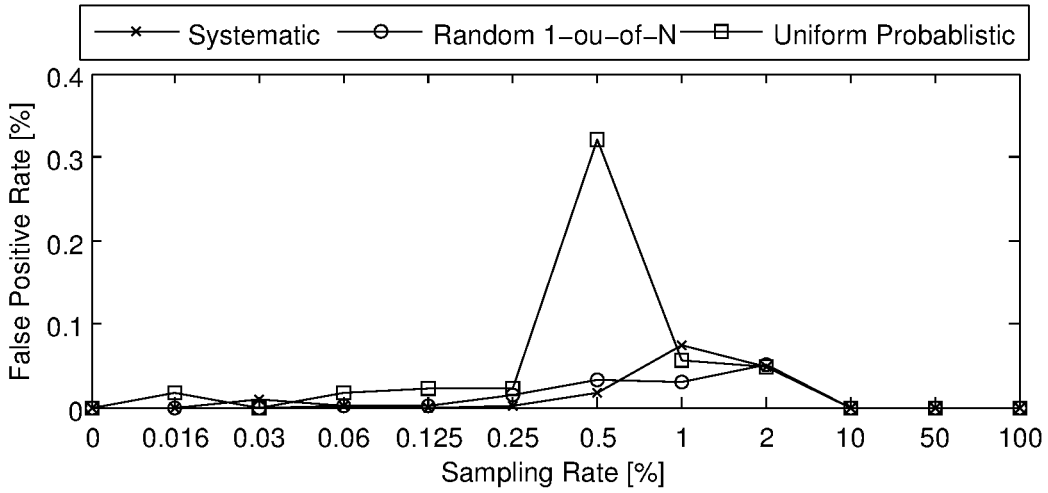

Figure 6.9: FPR—the influence of three sampling methods on the proposed scheme based on analyzing SYN-ACK segments in the case of the Clear1 trace without malicious activity.

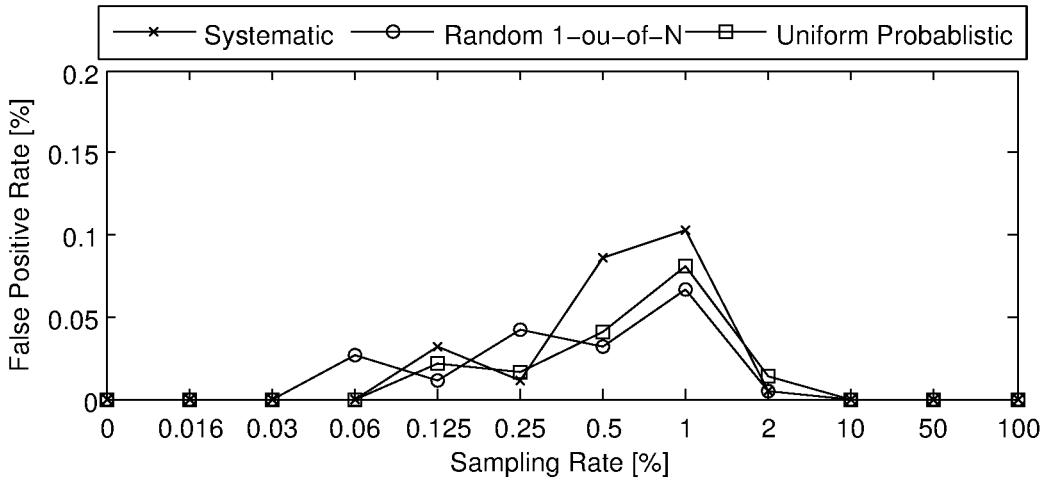

Figure 6.10: FPR—the influence of three sampling methods on the proposed scheme based on analyzing SYN-ACK segments in the case of the Clear2 trace without malicious activity.

6.3 Related Work

We can observe that our results overcome the problems considered in the literature [80] in which the authors evaluated the impact of the same three sampling methods on anomaly detection techniques. They conclude that methods that rely on systematic sampling are the worst choice for the detection of attacks based on certain TCP control flags like SYN or FIN, because such segments are not evenly distributed across traffic. Our scheme alleviates the problem of sampling SYN-SYN/ACK, SYN-FIN and SYN-Client ACK pairs and proposes a novel solution based on considering ACK segments.

To detect and mitigate scanning activity, SYN flooding, and DDoS attacks, several authors proposed various methods [87, 88, 89, 90]. The end-host method based on SYN cookies is the most commonly used technique to protect against SYN flooding attacks [91]. Nevertheless, SYN cookies are not able to encode all TCP options, in particular the window scale and selective acknowledgements that are widely supported and serve to significantly improve TCP performance. Moreover, the method does not overcome the problem of bandwidth consumption in case of high-volume TCP SYN attacks. Consequently, we have focused on SYN flood methods located in border routers [42, 43, 44, 73, 74] and we designed a novel scheme insensitive to sampling.

6.4 Conclusion

We have proposed a novel scheme for detecting TCP SYN flooding attacks and portscans that offers good performance in the case of sampling. The scheme considers TCP connections as legitimate if it samples one of multiple ACK segments (with disabled SYN flag) coming from the server. This differs from existing methods based on pair matching of control segments SYN/ACK, FIN (RST) or Client ACK etc. Our trace-based simulations show that unlike other techniques, the proposed method significantly decreases the False Positive Rate under a sampling process. Moreover, the results reveal that our method alleviates the problem of losing some information when systematic sampling is used. The effectiveness of the presented method only relies on the sampling rate and not on the type of a sampling method.

Part III

Classifying Service Flows in the Encrypted Skype Traffic

Chapter 7

Introduction

Contents

Accurate traffic identification and classification are essential for proper network configuration and security monitoring. Application-layer encryption can however bypass restrictions set by network configuration and security checks. Several application protocols adopted the Secure Socket Layer (SSL) encryption to protect the confidentiality of communications, which raises new challenges with respect to traffic classification and malware detection.

In this chapter, we focus on Skype as an interesting example of encrypted traffic and provide a method for identifying different encrypted TCP Skype flows tunneled over SSL—we want to discriminate between voice calls, video conferencing, skypeOut calls, chat, and file sharing. Previous papers on Skype concentrated on its architecture and the authentication phase [33, 31, 34], on the mechanisms for firewall and NAT traversal [92] as well as on characterizing traffic streams generated by VoIP calls and Skype signaling [51, 35]. Bonfiglio et al. proposed identification methods for encrypted UDP Skype traffic [18], but no work has tackled the problem of how to classify encrypted TCP flows generated by all Skype services.

Skype exemplifies the problem of identifying encrypted flows, because it multiplexes several services using the same ports: VoIP calls, video conferencing, instant messaging, or file transfer. A network administrator may assign a higher priority to VoIP calls, but other flows may also benefit in an illegitimate way from a higher priority if we cannot distinguish them from VoIP calls.

7.1 Contributions of Part III

We propose a classification method for Skype encrypted traffic based on the Statistical Protocol IDentification (SPID) [22] that analyzes statistical values of flow and application layer data. We propose an appropriate set of attribute meters to de-

tect encrypted TCP Skype traffic and identify its service flows. We consider a very special case of Skype traffic that is, in addition to proprietary encryption, tunneled over SSL. Our method involves three phases with progressive identification: the first classification phase early reveals Skype traffic, while the second one provides the preliminary Skype flows identification: the distinction between voice/video communication, chat, voice calls towards phones using skypeOut, and file sharing. The final phase identifies Skype flows in detail: voice calls, video and voice communication (denoted later as just video), chat service, skypeOut calls, file upload and download. To select the right attribute meters for each phase, we applied a method called *forward selection* [55] that evaluates how a given attribute meter improves classification performance and promotes it to the traffic model if its influence is significant. *Forward selection* uses the Analysis of Variance (ANOVA) [59]. We have evaluated our classification method on a representative dataset to show excellent performance in terms of Precision and Recall.

7.2 Relevant Publications for Part III

[23] Maciej Korczyński and Andrzej Duda. Classifying Service Flows in the Encrypted Skype Traffic. *2012 IEEE International Conference on Communications (ICC'12)*, pages 1–5, June 2012.

Chapter 8

Issues in the Analysis of Skype Traffic

Skype traffic presents a major challenge for detection and classification, because of proprietary software, several internal obfuscation mechanisms, and a complex connection protocol designed for bypassing firewalls and establishing communication regardless of network policies.

Skype differs from other VoIP applications, because it relies on a p2p infrastructure while other applications use the traditional client-server model. Skype nodes include clients (ordinary nodes), supernodes, and servers for updates and authentication. An ordinary node with a public IP address, sufficient computing resources and network bandwidth may become a supernode. Supernodes maintain an overlay network, while ordinary nodes establish connections with a small number of supernodes. Authentication servers store the user account information. A Skype client communicates with the authentication server and another ordinary node in an indirect way via supernodes that relay packets. Skype can multiplex different service flows such as voice calls to another Skype node, skypeOut calls to phones, video conferencing, chat, file upload and download. Our goal is to detect and classify the service flows in Skype traffic.

Even if the maintenance of the supernode list is possible through some active and passive methods [31], the associated information may only be useful in revealing Skype traffic and not in detecting Skype service flows. We cannot use traditional port-based flow identification methods, because Skype randomly selects ports and switches to port 80 (HTTP) or 443 (HTTP over SSL) if it fails to establish a connection on chosen ports.

Another feature of the Skype design is the possibility of using both TCP and UDP as a transport protocol. Skype uses TCP to establish a connection and then it can switch to UDP for both signaling and regular communication. Once it makes the initial connection, it can interchangeably use TCP or UDP depending on network restrictions.

Skype encrypts its traffic with the proprietary encryption technologies to protect

communications exchanged between its clients and severs. It uses several encryption algorithms [93], which makes traffic classification a challenging task. Its servers use the strong 256-bit Advanced Encryption Standard (AES), the supernodes and clients use three different types of Rivest Cipher 4 (RC4) encryption. Finally, the clients also use AES-256 on top of RC4 algorithm to protect from potential eavesdropping. Skype entirely encrypts TCP traffic, but some information in the UDP payload is not encrypted so a part of the Skype messages encapsulated in UDP can be obtained and used for identification [18].

We propose an accurate method for classification of encrypted TCP Skype service flows tunneled over SSL. It is a hybrid method combining traffic flow metering with Deep Packet Inspection (DPI) elements.

Chapter 9

Design and Evaluation

Contents

9.1 Classification Method

We need to apply a new methodology to classify encrypted TCP Skype flows. We propose to consider various statistical flow and application layer data features.

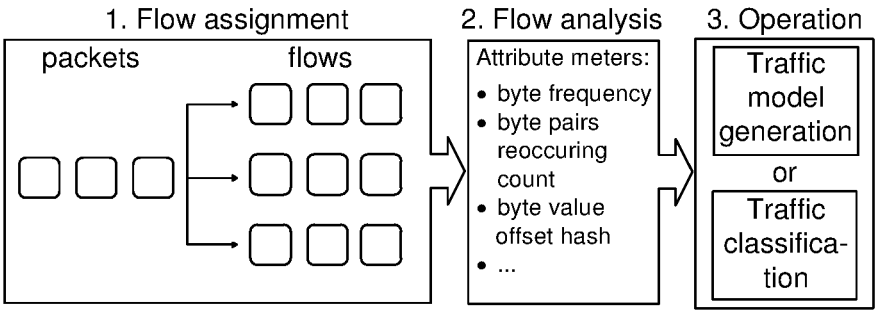

Figure 9.1: Three steps of SPID.

9.1.1 Classification Based on SPID

We build our method upon SPID (Statistical Protocol IDentification) [22] (cf. Figure 9.1). It is based on *traffic models* that contain a set of *attribute fingerprints* represented as probability distributions. They are created through frequency analysis of traffic properties called *attribute meters* of application layer data or flow features. An example of such an attribute meter is *byte frequency* that measures the frequency at which all of the possible 256 values occur in a packet. Other attribute meters defined later in Table 9.1 and 9.2 include for instance byte offset, byte re-occurring, direction change, and packet size.

As illustrated in Figure 9.1, SPID operates in three steps. First, packets are classified into bi-directional flows. All connections are represented as 5-tuples according to the source IP address, source port, destination IP address, destination port, and transport layer protocol. However, only packets carrying data are significant, because the analysis is based on both the application layer data and flow features. Then, each flow is analyzed in terms of attribute meters to obtain a collection of attribute fingerprints. Finally, the obtained attribute fingerprints are used either in traffic model generation or in traffic classification.

To illustrate the process of fingerprint creation, consider an example of the *byte frequency* attribute meter computed on the first 5 bytes of the SSL `Server Hello` packet, a part of the SSL handshake protocol. The first 3 bytes refer to the message type (0x16) and the SSL version (0x03 01), while the last two bytes correspond to the size of the remaining part of the SSL record (0x00 4a). Each time we observe a particular value, its counter is incremented. In the example, all five counters referring to the five values will be incremented. Then, SPID maintains a probability vector—the normalized counter vector with all elements summing up to one.

At the initial training phase, the method creates *traffic models*—attribute fingerprints representative for the traffic we want to detect. During the classification phase, the method computes attribute fingerprints on the flows to classify and compares them with traffic models by means of the Kullback-Leibler (K-L) divergence [28]:

$$D(P||Q) = K\text{-}L(P,Q) = \sum_{x \in X} P(x) log_2 \frac{P(x)}{Q(x)}. \tag{9.1}$$

The K-L divergence is a measure of the difference between two probability distributions $P(x)$ and $Q(x)$. $P(x)$ represents the distribution of a particular attribute of an observed flow and $Q(x)$ is the distribution corresponding to a known traffic model. Classification consists of comparing $P(x)$ with all known traffic models and

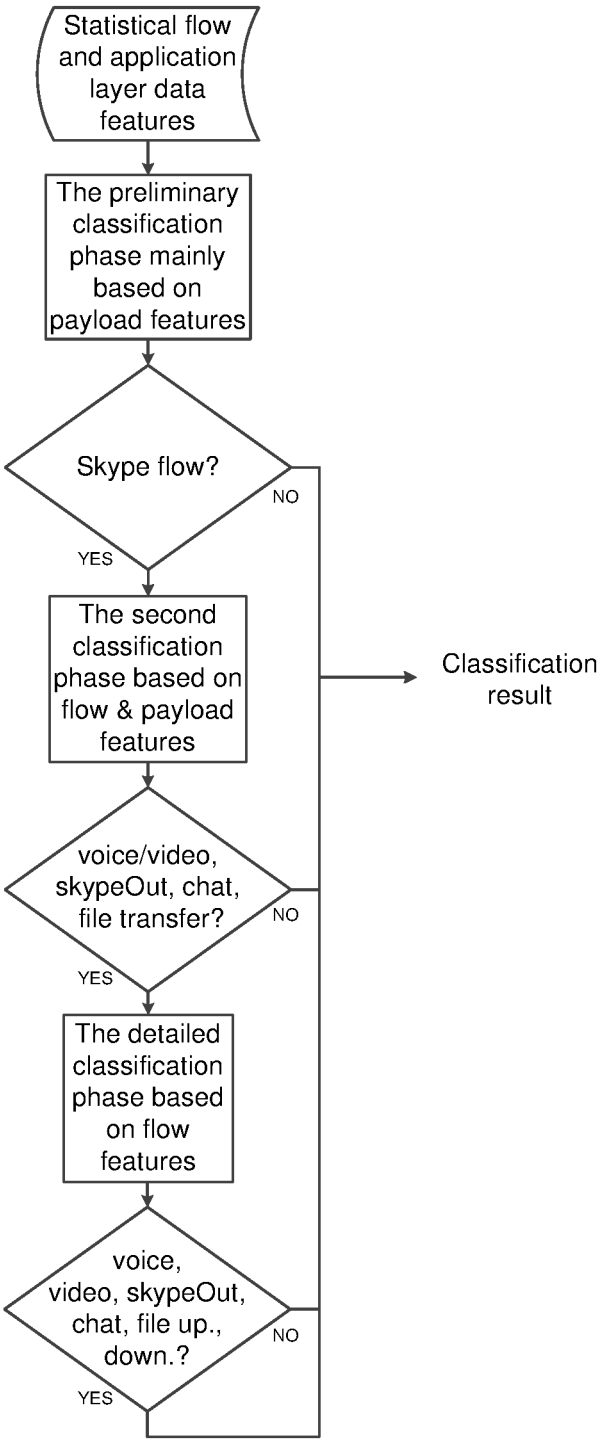

Figure 9.2: Simplified classification process.

selecting the protocol with the smallest average divergence $D(P||Q)$ and lower than a given threshold. We need to correctly set the divergence threshold to decrease the False Positive Rate for known traffic models—we only take into consideration the K-L divergence average values below the threshold.

Figure 9.2 presents a simplified process of the proposed classification method. In the first phase, it detects Skype traffic after a TCP three-way handshake based on the first five packets of the connection by considering attribute meters, the majority of which reflects application level data. Then, it changes the set of attribute meters to both packet independent and application level data features to detect service flows in the Skype traffic: voice/video, skypeOut, chat, and file transfer. This phase requires a larger number of packets to analyze to be effective: our calibration sets this value to 450 packets. Finally, the method considers more packets (the threshold is set to 760) to further distinguish between voice and video flows, and between file upload and download.

9.1.2 Attribute Meters for Skype

In this subsection, we present the set of attribute meters defined for classifying Skype traffic (cf. Table 9.1 and 9.2) with notation presented in Table 9.3.

- **byte frequency**: in each packet it measures and returns the frequency of individual bytes in the payload. Encrypted data seems to have equally distributed byte frequencies, whereas the plain text may exhibit different distributions. The SSL protocol, in the first bytes of the transmitted packets, tends to provide some unencrypted information related to the session, such as the SSL version, message type, compression method selected by the server, etc.

- **action-reaction of first bytes**: it creates hash values based on the first 3 bytes of each packet that was sent in a different direction than the previous one. It is sometimes better to analyze packets sent alternately in different directions instead of looking at all packets, because we can easily analyze the request-response phase between a client and a server.

- **byte value offset hash**: it combines individual byte values in each packet with the offset at which the bytes are positioned. The meter considers up to 32 bytes of the 4 first packets. The SSL is one of the protocols that use several positions in particular packets (e.g. in `Client Hello` or `Server Hello` messages). As a result, the combination of bytes with their positions provides some additional information with respect to the byte frequency.

Table 9.1: Definition of attribute meters used in classification

Attribute meter	Definition
byte frequency	$\mathcal{M}_1 : \{(k, p_k)\},\ k = 0, 1, ..., 255; p_k = \frac{m_k}{\sum m_k}\ ,\ m_k = \sum_{i=1}^{8} \sum_{j=1}^{100} \delta_{x_j^i}$
action-reaction of first 3 bytes	$\mathcal{M}_2 : \{(h^i, p_{h^i}), \forall_{i \in (1,3)}\},\ h : (y_{3\Delta}^i, z_{3\Delta}^i) \rightarrow h(y_{3\Delta}^i, z_{3\Delta}^i),$ $p_{h^i} = \frac{m_{h^i}}{\sum m_{h^i}},\ m_{h^i} = \delta_{h(y_{3\Delta}^i, z_{3\Delta}^i)}$
byte value offset hash	$\mathcal{M}_3 : \{(h, p_h)\},\ h : (j, x_j^i) \rightarrow h(j, x_j^i),\ p_h = \frac{m_h}{\sum m_h},$ $m_h = \sum_{i=1}^{4} \sum_{j=1}^{32} \delta_{h(j, x_j^i)}$
first 4 packets byte reoccurring distance with byte	$\mathcal{M}_4 : \{(h, p_h)\},\ \forall_{d<=16} : h : (x_j^i, d) \rightarrow h(x_j^i, d),\ p_h = \frac{m_h}{\sum m_h},$ $m_h = \sum_{i=1}^{4} \sum_{j=1}^{32} \delta_{h(x_j^i, d)})$
first 4 packets first 16 byte pairs	$\mathcal{M}_5 : \{(h, p_h)\},\ h : (x_j^i, x_{j+1}^i) \rightarrow h(x_j^i, x_{j+1}^i),\ p_h = \frac{m_h}{\sum m_h},$ $m_h = \sum_{i=1}^{4} \sum_{j=1}^{16} \delta_{h(x_j^i, x_{j+1}^i)}$
first 4 ordered direction packet size	$\mathcal{M}_6 : \{(f, p_f)\},\ f : (i, s(x^i), dir(x^i)) \rightarrow f(i, s(x^i), dir(x^i)),$ $p_f = \frac{m_f}{\sum m_f},\ m_f = \sum_{i=1}^{4} \delta_{f(i, s(x^i), dir(x^i))}$
first packet per direction first N byte nibbles	$\mathcal{M}_7 : \{(f, p_f)\},\ \forall_{x^1 \in \{z^1, y^1\}} : f : (nib(x_j^1), j, dir(x^1))$ $\rightarrow f(nib(x_j^1), j, dir(x^1)),$ $p_f = \frac{m_f}{\sum m_f},\ m_f = \sum_{j=1}^{8} \delta_{f(nib(x_j^1), j, dir(x^1))}$

Table 9.2: Definition of attribute meters used in classification - cont.

Attribute meter	Definition
direction packet size distribution	$\mathcal{M}_8 : \{(f, p_f)\}$, $f : (s(x^i), dir(x^i)) \rightarrow f(s(x^i), dir(x^i))$, $p_f = \frac{m_f}{\sum m_f}$, $m_f = \sum_{i=1}^{s(x)} \delta_{f(s(x^i),dir(x^i))}$
byte pairs reoccurring count	$\mathcal{M}_9 : \{(f, p_f)\}$, $\forall_{x_j^i = x_j^{i+1}} : f : (x_j^i, dir(x_j^i), dir(x_j^{i+1})) \rightarrow f(x_j^i, dir(x_j^i), dir(x_j^{i+1}))$, $p_f = \frac{m_f}{\sum m_f}$, $m_f = \sum_{i=1}^{s(x)} \sum_{j=1}^{32} \delta_{f(x_j^i,dir(x_j^i),dir(x_j^{i+1}))}$

- **first 4 packets byte reoccurring distance with byte**: it creates a short hash value (usually a 4-bit representation) and combines it with the distance between the two occurrences. The measurement detects the bytes that occurred more than once within 16 previous bytes. Originally, it was created to identify banners in plain text packets like e.g. `TT` in HTTP GET and POST messages, but it also applies to the case of the encrypted or the tunneled SSL content.

- **first 4 packets first 16 byte pairs**: it combines neighboring bytes in a 16-bit value and converts to a 8 bit hash value (the size is determined by the fingerprint length). It analyzes only application layer data regardless of the flow information, i.e. packet size, directions, or inter-arrival times. The meter indicates that there are some specific, not random two-byte combinations like e.g. list compression methods supported by the client in the SSL `Client Hello` message sent to the server.

- **first 4 ordered direction packet size**: the meter returns the compressed version of the packet size—it represents a range in which the packet lies instead of the exact value. Measurements are separately done for each of four first packets in connection and the returned value is associated with the packet direction and the order number. It is a flow based attribute created for early traffic recognition.

Table 9.3: Notation

$\mathcal{M} : \{(k, p_k)\}$ – attribute meter
$k = 0, 1, 2, \ldots, 255$ – random variable of an attribute meter
m_k – attribute meter counter
p_k – probability distribution of an attribute meter (corresponds to $Q(x)$ in traffic model generation and $P(x)$ in traffic classification)
δ – indicator function; $\delta : X \to \{0, 1\}, \delta_{x_j^i} = \begin{cases} 1 & \text{if } X = x_j^i \\ 0 & \text{if } X \neq x_j^i \end{cases}$
h – hash function, $h = 0, 1, 2, \ldots, 255$
f – compressing function, $f = 0, 1, 2, \ldots, 255$
x^i – packet i
x_j^i – byte j in packet i
$x_{j(m)}^i$ – bit m in byte j in packet i
$\sum_i x^i \leftrightarrow x$ – all packets in a TCP session
y^i – packet i, z^i – packet sent in a different direction than y^i
$x_{\Delta j}^i$ – first j bytes in packet i
d – distance between two identical bytes; if $x_j^i = x_{j-d}^i \Rightarrow d, 0 < d < j$
$s(x)$ – size of x; amount of packets in a TCP session
$s(x^i)$ – size of packet x^i in bytes
dir – packet direction
nib: $x_j^i \leftrightarrow x_{j(m\in(1\ldots8))}^i$; $x_{j(m\in(1\ldots4))}^i \; XOR \; x_{j(m\in(5\ldots8)}^i \Rightarrow nib(x_j^i)$

- **first packet per direction first N byte nibbles**: it analyzes the first packet in each direction and inspects its first few bytes depending on the fingerprint length (8 bytes for a fingerprint length of 256). It provides a measure combining the packet direction, byte offset, and a compact representation of the byte value so-called *nibble*, (it divides a byte into two 4-bit groups, performs an XOR calculation, and returns the resulting 4-bit value). The first packet in each direction and the first few bytes corresponding to these packets say a lot about the application layer protocol and might also provide some hidden information of the underlying service.

- **direction packet size distribution**: this attribute is very similar to the first 4 ordered direction packet size meter. The only difference is that it inspects all packets in a connection and does not mark each measurement with the order number of the packet in a connection. It is an example of a flow based attribute especially suitable for detailed Skype classification: it is able to classify flows in which packet sizes per direction are different, which

enables to distinguish, for example, file upload from download.

- **byte pairs reoccurring count**: it detects bytes that reoccur in the same position in two consecutive packets. In addition, it takes into account the direction of a given packet and its predecessor.

9.1.3 Methodology for Attribute Meter Selection

Our classification process is based on three phases and each of them requires a proper set of attribute meters. We applied a method called *forward selection* for choosing attribute meters. It consists of starting with an initial attribute in the model, trying attributes out one by one, and adopting them, if they improve the classification performance. The selection terminates when adding an attribute does not improve the performance.

We consider a set of n attribute meters $x_1, ..., x_n \in X$ and a set of m Skype services. We begin with a model that includes the most significant attribute in the initial analysis. More precisely, we compute $Precision$, $Recall$, and $F\text{-}Measure$ (cf. Eq. 3.2 - 3.4) for a particular Skype service and for each individual attribute meter. The True Positive (TP) term refers to all Skype flows that are correctly identified, False Positives (FPs) refer to all flows that were incorrectly identified as Skype traffic. Finally, False Negatives (FNs) represent all flows of Skype traffic that were incorrectly identified as other traffic.

We select attribute $x_i \in X$ with the largest average $F\text{-}Measure$ defined as:

$$\max_{x \in X} \frac{1}{m} \sum_{a \in (1,m)} FM_a^x, \tag{9.2}$$

where FM_a^x denotes a^{th} observation of $F\text{-}Measure$ value corresponding to x^{th} attribute meter.

In the next step, each of the remaining attributes $x_1, ...x_{i-1}, x_{i+1}, ...x_n \in X$ is tested for inclusion in the model. We run several *F-tests* (explained below) that compare the variance of $F\text{-}Measure$ values obtained in the preliminary selection, i.e. $FM_a^{x_i}$, where $a \in (1, m)$, with the corresponding values obtained after including each attribute meter separately.

Let us focus on a particular *F-test* [59] that compares the influence of attribute meter $x_j \in x_1, ...x_{i-1}, x_{i+1}, ...x_n \in X$ with the first model based on $x_i \in X$. We examine two groups of $F\text{-}Measure$ values $FM_a^{x_i}$ and $FM_a^{x_{ij}}$ that respectively correspond to attribute x_i and to the set of two attribute meters, i.e. x_i and x_j. We test the null hypothesis that two means of the discussed population are equal. If we fail to reject it, the additional attribute meter does not improve the classification

performance and we need to exclude it from further consideration. To examine these two groups, we use the one-way Analysis of Variance (ANOVA) *F-test* [59] that compares the variance between the groups to the variance within the groups. The *between–groups* variance is given by:

$$S_{bet} = m * \sum_{x} \frac{(\overline{FM}^{x} - \overline{FM})^2}{(k-1)}, \tag{9.3}$$

where $\overline{FM}^{x}$ denotes the mean of FM_a^x values, $\overline{FM}$ denotes the overall mean of $F\text{-}Measure$ observations, i.e. $FM_a^{x_i}$ and $FM_a^{x_{ij}}$, m is the number of $F\text{-}Measure$ values for Skype services and k is the number of groups (in the discussed case equal to 2). The *within–group* variance is given by:

$$S_{wit} = \sum_{x,a} \frac{(FM_a^x - \overline{FM}^{x})^2}{k * (m-1)}, \tag{9.4}$$

where FM_a^x denotes a^{th} observation corresponding to each x^{th} classification (in the discussed case to the classification based on x_i and the classification based on the set of two attributes x_i and x_j).

The *F-statistics* is computed as:

$$F = \frac{S_{bet}}{S_{wit}}, \tag{9.5}$$

and it follows the *F-distribution* with $k-1$, $k*(m-1)$ degrees of freedom under the null hypothesis. If the null hypothesis is rejected and the average $F\text{-}Measure$ value corresponding to x_i is lower than $F\text{-}Measure$ related to the set of two attribute meters, i.e. x_i and x_j, then attribute x_j is considered as a candidate to be included in the model.

For each of the attribute meters, the method computes *F-statistics* that reflects the contribution of attributes to the model. The most significant attribute is added to the model, if *F-statistics* is above a predefined level set to 0.1. Moreover, if *F-statistics* is above 1, it is included in the model and considered as a significant attribute meter. The *forward selection* method then computes *F-statistics* again for the attribute meters still remaining outside the model and the evaluation process repeats. Therefore, attributes are added one by one to the model until no remaining attribute results in significant *F-statistics*.

9.2 Evaluation Results

We start with the description of the datasets used for training the method and evaluating its performance. We then present the criteria for classification performance and we discuss the evaluation results of the proposed method. We also explain how we calibrate the parameters of the method (the choice of the number of packets to analyze during each step, the number of flows used in the training process, and the selection of the right K-L threshold).

9.2.1 Dataset Description

The appropriate selection of packet traces containing ground-truth information is one of the key aspects in the training and evaluation process. It should be as extensive as possible and should cover various environments. We have generated TCP Skype traffic in the following conditions:

- various operating systems: Linux, MacOS, Windows,
- wireless and wired networks,
- connections within one LAN as well as WAN connections between LANs located in France and Poland,
- different versions of Skype (2, 3, and 5)

To force Skype to generate desired flows, we have used firewall rules to block UDP so that all communications use TCP and allowed only well-known TCP ports so that Skype switches to port 443.

We have used Wireshark [94] to collect packet traces and to distinguish Skype flows from other network traffic. We have tested all Skype services separately to simplify the extraction of the desired flows and captured flows containing voice, video, skypeOut, chat, file upload and download. We have observed the use of the G.729 codec for skypeOut calls and SILK_V3 for Skype-to-Skype voice communication. Skype adopts VP7.1 codec for video communication. Overall, we gathered 479 Skype flow traces taking more than 770 MB.

Therefore, we have divided the collected set of flows into several groups according to operating systems, network access technologies, and Skype versions. For the traffic model generation purpose we have selected a group of traces generated by MacOS over a WAN connection between wireless LANs located in France and Poland. We have used the remaining datasets to evaluate the classification mechanism. Our fingerprint database with 6 Skype service flow models has the size of 1.78MB in the XML format.

Furthermore, we have gathered a separate set of traces without Skype traffic to test the discrimination of our method. It contains various types of traffic: SSL, SSH, HTTP, SCP, SFTP, VoIP, BitTorrent, and standard services like streaming, video conferencing, chat service, mail, file sharing. The traces contain 18945 flows of around 3GB and were gathered between December 2010 and March 2011.

9.2.2 Criteria for Classification Performance

We use three metrics to quantify the performance of classification: Precision, Recall, and *F-Measure* (cf. Eq. 3.3 - 3.4). *F-Measure* is an evenly weighted combination between Precision and Recall, which means that if the system can for instance identify skypeOut traffic with Precision 100% (no False Positives) and Recall is 96.6% then the *F-Measure* is 98.2%.

9.2.3 Performance of Classification

To evaluate the proposed method, we have extended the version 0.4.6 of SPID [95].

Our method depends on three parameters: the amount of packets required for reliable traffic and flow identification during each of the three steps, the K-L divergence threshold, and the number of flows used in the training process. We first present the classification results for the number of packets in each phase set to 5, 450, and 760 packets, respectively, the K-L divergence threshold of 1.9, and 15 training flows (we evaluate the impact of parameters further on and explain how we have chosen their values, cf. Section 9.3).

After each classification step, the classifier decides if there are any instances of Skype flow for further analysis. If the identification result is positive, then it continues with more detailed classification of Skype flows with a different set of attribute meters. Otherwise, it finishes as no Skype flows were recognized.

The objective of the first classification phase is to early detect encrypted TCP Skype flows tunneled over the SSL protocol. The most significant attribute meter chosen in the selection process is $\mathcal{M}_5$ (cf. Table 9.1). Two other important attributes are $\mathcal{M}_7$ and $\mathcal{M}_6$ while $\mathcal{M}_3$, $\mathcal{M}_4$, and $\mathcal{M}_1$ are less meaningful. In addition to payload inspection attributes ($\mathcal{M}_5$, $\mathcal{M}_7$, $\mathcal{M}_3$, $\mathcal{M}_4$, and $\mathcal{M}_1$), we have chosen one typical flow based attribute that combines features like size, direction, and packet order number ($\mathcal{M}_6$). Such selection indicates that the first SSL packets contain some characteristic values that differ from the headers of other services that use SSL (cf. Section 12.2).

Our experiments show that inspecting only the first five packets containing the

Table 9.4: Performance of Phase 1, Early Recognition of Skype Traffic

Traffic	Precision %	Recall %	F-M. %
Skype	100	100	100
No Skype	100	100	100

payload is sufficient to reveal Skype traffic with Precision, Recall, and F-Measure equal to 100% (cf. Table 9.4).

Once the method detects Skype traffic, it classifies the underlying type of service, i.e. voice/video communication, skypeOut calls, chat, file sharing. In the second phase, the method uses another set of attribute meters ($\mathcal{M}_8$ as the most important, $\mathcal{M}_7$ as a significant one, and $\mathcal{M}_9$, $\mathcal{M}_2$, and $\mathcal{M}_5$ as additional ones). The selected set of attributes is composed of payload independent *direction packet size distribution* attribute meter ($\mathcal{M}_8$) with DPI attributes ($\mathcal{M}_7$, $\mathcal{M}_9$, $\mathcal{M}_2$, and $\mathcal{M}_5$).

Table 9.5: Performance of Phase 2, Classification of Skype Flows

Skype Service	Precision %	Recall %	F-M. %
voice/video	99.1	95.7	97.4
skypeOut	100	96.6	98.2
chat	86.4	100	92.7
file sharing	100	98.6	99.3

Table 9.5 shows very good results of classification after inspecting 450 packets. However, this phase cannot distinguish between voice communications and voice/video calls due to similar traffic characteristics. Nevertheless, from the Quality of Service (QoS) perspective, network administrators may already give priority to Skype voice/video traffic and limit Skype file sharing flows regardless of the traffic direction.

The objective of Phase 3 is to further refine the classification of voice and video flows as well as file sharing. We have applied $\mathcal{M}_8$ as the most important flow based attribute meter and DPI based $\mathcal{M}_7$ as an additional one. Table 9.6 presents the final results obtained after analyzing 760 packets. We can observe that the results are very good for most of Skype flows. We can easily distinguish between file upload and download based on the flow attribute combining the direction with the packet size distribution (cf. attribute $\mathcal{M}_8$ in Table 9.2). The classification is based on the fact that the sizes of packets sent from the client significantly differs from the sizes of packets sent in the opposite direction.

Classification of voice and video flows performs slightly worse, because our

Table 9.6: Performance of Phase 3, Detailed Classification of Skype Flows

Skype Service	Precision %	Recall %	F–M. %
voice	72.9	57.4	64.2
video	60.3	73.2	66.1
skypeOut	100	96.6	98.2
chat	90.2	97.4	93.7
file upload	100	96.9	98.4
file download	100	97.5	98.7

method does not capture some characteristics of the Skype behavior (it is meant to be applied to other classification problems as well). We have observed that in the case of Skype calls (both voice and video), the Skype client sends traffic simultaneously through several nodes depending on network conditions. In other words, the Skype voice or video traffic may spread on several TCP connections, which we cannot capture, because our method considers each TCP flow separately.

In contrast to voice/video communication and file sharing, we have noticed that chat messages and skypeOut calls seem to be sent through a single node. Considering chat messages, we have observed that when an intermediary node goes down, communication switches to another one without any interference for the users. This is not surprising if we take into account a small amount of data to send. For skypeOut calls, however, we have observed that the whole communication goes through a single intermediary node and the range of relay addresses is limited. This may come from higher requirements for bandwidth and computing resources to support high quality of calls. To sum up, in this classification step it was easier to identify these two type of services, because the whole traffic was sent over single flows.

9.3 Calibration of the Method

In this section, we consider the choice of the right values for the mathod parameters:

- the K-L divergence threshold,
- the number of inspected packets per flow in each classification phase,
- the number of flows used in the training process.

Figure 9.3 shows the F-Measure for 15 training flows and for three classification steps (analysis after 5, 450, and 760 packets) depending on the K-L divergence

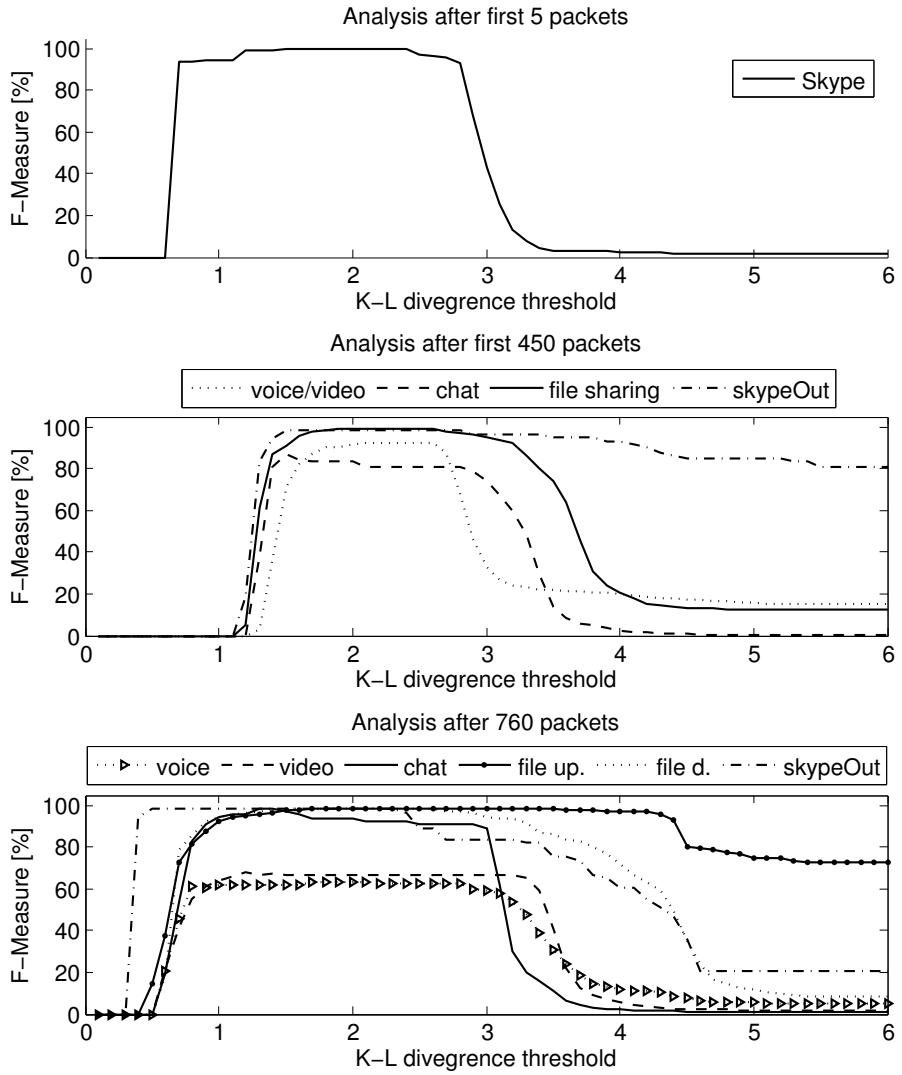

Figure 9.3: F-Measure depending on the K-L divergence threshold for three classification phases.

threshold. Choosing an appropriate value of K-L divergence threshold is important because a too low value results in an increased number of False Negatives, i.e. Skype flows are incorrectly identified as unknown traffic, which decreases the F-Measure. If the threshold is too high, then it may lead to multiple False Positives, i.e. other protocols are incorrectly identified as Skype. As shown in Figure 9.3, a large value of the threshold significantly affects the F-Measure. The results suggest that the

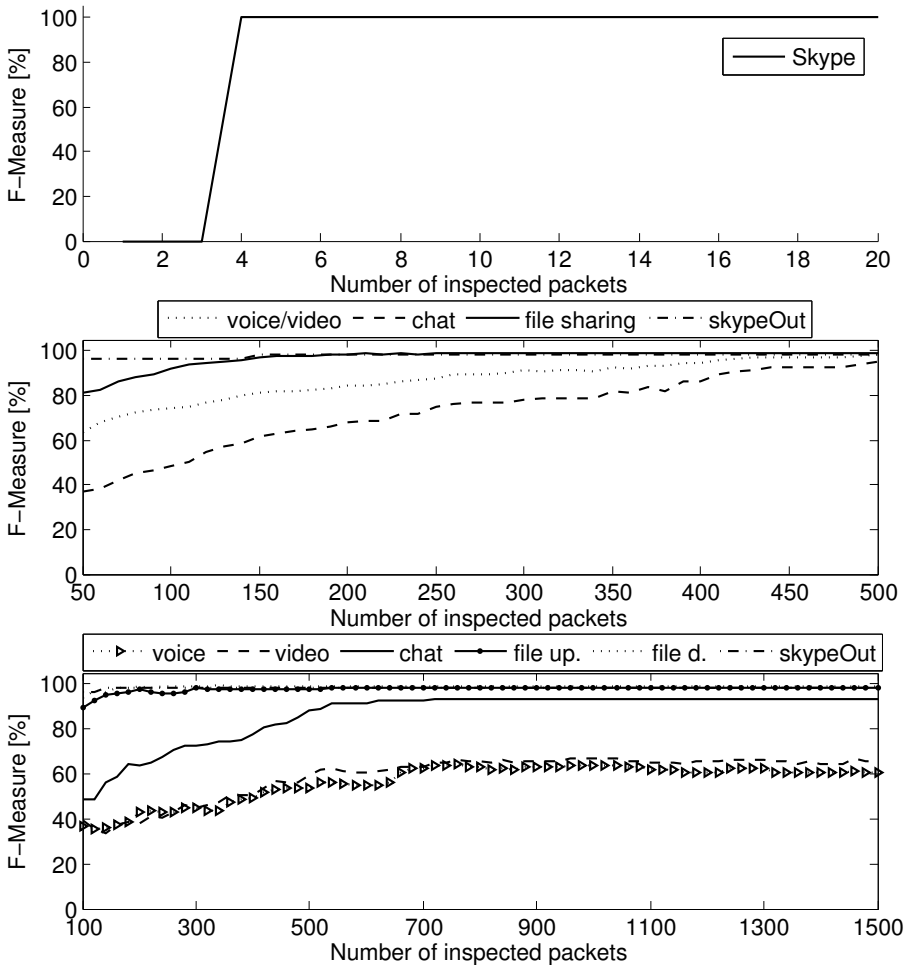

Figure 9.4: F-Measure depending on the number of inspected packets for three classification phases.

optimal value for all three classification phases is 1.9.

Figure 9.4 shows the F-Measure depending on the number of inspected packets containing payload for three classification phases with the K-L divergence threshold equal to 1.9 and 15 training flows per traffic model. As we can see in the figure, the first classification phase requires only 4 packets to achieve the F–Measure equal to 100%. In the second classification phase, the distinction between Skype services, i.e. voice/video, skypeOut, chat, and file sharing, is very clear (the average F-Measure close to 97%) after 450 packets containing payload. A slightly lower Precision for the chat service when the number of inspected packets is less than 450 packets is

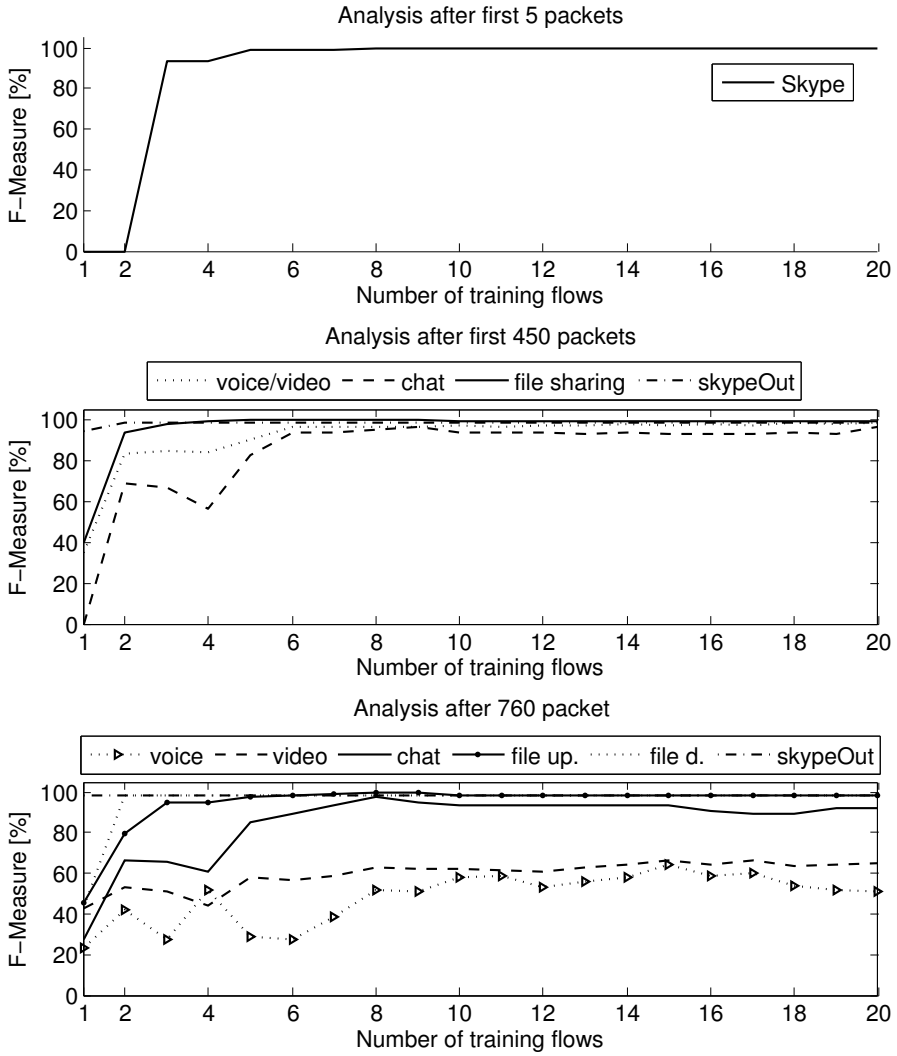

Figure 9.5: F-Measure depending on the number of training flows for three classification phases.

probably due to a limited number of observations during the construction of the chat traffic model. Despite the same number of training flows set up to 15 for all traffic models, i.e. voice, video, chat, skypeOut, traffic upload and traffic download, the amount of data available during the creation of the chat traffic model was three times lower than in other cases. Therefore, the lower Precision for chat means that other Skype services were incorrectly identified as chat, which results in a

lower Recall for voice/video as well as for file sharing, while the performance for skypeOut remains almost unaffected due to different traffic characteristics.

In the third classification phase, the F-Measure significantly raises with the number of inspected flows up to around 700 packets. Choosing the number of 760 results in the F–Measure for voice and video traffic models around 65%, whereas the accuracy of the classification is nearly ideal for other types of Skype traffic.

Finally, Figure 9.5 presents the influence of the number of training flows on the F–Measure value for three classification phases. We can see that the critical amount of training flows essential for identification of the encrypted Skype traffic is equal to 3. However, to improve classification performance, we have used 15 training sessions for each traffic model, because the training phase is done off line, so it does not influence the speed of classification.

9.4 Related Work

Much research has concerned the domain of traffic classification during recent years [5, 2, 3, 26, 96], however only a few authors focused on encrypted traffic [52, 25, 33, 31] or on the classification of encrypted flows [97, 18, 51, 35]. Some of these methods were applied to the problem of Skype classification and cannot be easily extended to other identification problems of encrypted traffic.

Teixeira et al. [52] extended their previous work [26] and proposed a method based on the size of the first few packets of an encrypted connection, which enables an early application protocol recognition with the accuracy of more that 85%. In our work, we make a step forward by proposing an accurate method for detecting service flows in encrypted Skype traffic based on various traffic flow and payload attributes.

A recent approach focused on the detection of Skype flows especially voice service traffic [31]. Even if the method results in high accuracy, it is not applicable to other classification problems, because it makes use of flow features and node information obtained from some passive and active measurements within the Skype p2p network. The solution proposed by Alshammari et al. [33] is based on a machine learning algorithm using flow features without taking into account IP addresses, port numbers, and the payload. It is a fairly general methodology and like for us, Skype is a test case for the classification of encrypted traffic. However, it is not sure that the method can classify Skype flow services. Another approach tries to address the problem of identifying encrypted application layer protocols by means of a hybrid method that combines signature-based and statistical analysis methods [25]. The work is closely related to ours, but their objective is limited to the classifi-

cation of encrypted application layer protocols, while we focus more on an in-depth analysis of particular Skype services. Wright et al. tackled the challenging task of identifying the language of conversations by using the length of encrypted VoIP packets [97]. Even if their work differs from our study in terms of classification objectives, we believe that the conclusions may remain the same, i.e. when using certain traffic characteristics it is possible to extract some meaningful information even from encrypted traffic.

Bonfiglio et al. investigated the characteristics of traffic streams generated by voice and video communications as well as the signaling traffic generated by Skype [51]. Chen et al. concentrated on the QoS level provided to Skype users [35]. Due to simple classification criteria, they cannot however distinguish between voice, video, and skypeOut calls.

Finally, Bonfiglio et al. proposed a framework based on two complementary techniques [18]. The first one detects Skype traffic fingerprints and the second one is based on flow characteristics (the packet arrival rate and the packet size). The authors evaluate two classifiers to reveal Skype traffic. The first Chi-Square Classifier checks the characteristics of the message content after cyphering. With this methodology, they can only distinguish between voice and skypeOut flows transported over UDP based on some deterministic unencrypted bytes in Skype messages. The second Naive Bayes Classifier checks the resemblance of the measured traffic with expected stochastic characteristics. Despite the fact that the framework is inspiring and innovative, it is limited to only classifying some classes of Skype traffic and depends on the deterministic byte values in the unencrypted UDP payload. In our studies, we have focused on TCP traffic where the whole content of a Skype message is encrypted and tunneled over the SSL protocol.

To summarize, the research described above focused either on limited classification of Skype traffic depending on particular unencrypted payload bytes or on some typical behavior of the Skype protocol. We believe that the problem of the detailed classification of encrypted traffic, in particular, the identification of service flows in the encrypted Skype traffic has not received sufficient attention yet. Our hybrid method provides a step forward in this direction.

9.5 Conclusions

In this paper, we have considered the problem of detecting encrypted TCP Skype traffic tunneled over SSL and classifying Skype service flows. Our three-phase hybrid classification method is based on SPID and combines traditional statistical flow features with DPI elements. In each phase, we select a subset of relevant

attribute meters through *forward selection* based on ANOVA. The performance of the method on a representative dataset is very promising—it achieves high Precision and Recall for most Skype service flows, whereas distinguishing between voice and video flows in the final classification phase is more challenging due to spreading traffic on several TCP connections.

Part IV

Classifying TLS/SSL Encrypted Application Flows

Chapter 10

Introduction

Contents

In this thesis, we have already emphasized the importance of adequate traffic classification methods for effective network planning, policy-based traffic management, application prioritization, and security control. Traditional port-based [14, 17, 98] and payload-based [7, 5, 99] classification methods, however, become less efficient, because new applications begin to use sophisticated obfuscation mechanisms and an increased number of applications make use of encryption, e.g., Tor [100], I2P [101], Bittorrent [102], IMule [103], Skype [93]. Applications can hide their nature by dynamically assigning ports, by using tunneling, or by applying proprietary payload encryption methods. This situation has led to the development of new flow feature-based [3, 52] and host behavior-based [2, 19] classification methods. In opposition to these approaches, we propose a classification framework that uses two statistical, payload-based methods to accurately classify traffic encrypted with the Transport Layer Security/Secure Sockets Layer (TLS/SSL) protocols.

TLS/SSL is a fundamental cryptographic protocol suite that supports secure communication over the Internet [104] by encapsulating and encrypting application layer data. Many WWW portals and servers, especially those providing commercial services, use TLS/SSL for guaranteeing security of all operations. In addition to security, TLS/SSL tunnels are increasingly used as tools for defeating security control and bypassing restrictions set by network configuration and security checks. Enforcing control over TLS/SSL encrypted flow is difficult, because the protocol was specifically designed to prevent eavesdropping and data tampering. Thus, the side-effect of its powerful mechanisms for supporting security is the lost capability of monitoring and controlling traffic.

The past research on traffic analysis and classification showed that once we are able to generate a unique signature based on the packet or message payload (e.g., HTTP request headers), we can classify applications with high accuracy [7, 16].

Unfortunately, such approaches fail in case of encrypted traffic [13], which spawned the development of flow-based and host behavior-based approaches. Nevertheless, recent research has also shown that it is still possible to create some statistical signatures despite traffic encryption [18, 23]. We follow this approach by defining a framework for classifying TLS/SSL encrypted applications based on inspecting the packet content of the application layer. We have found that the use of the TLS/SSL protocol strongly depends on service and application needs so it can reflect some traffic features, which allows us to discriminate between applications. In other words, we extract some indirect information from the TLS/SSL layer and use it to classify underlying applications.

10.1 Contributions of Part IV

Following these principles, we define a framework based on two complementary methods for classifying applications. In the first method, we use a first-order homogeneous Markov chain to build a stochastic model reflecting TLS/SSL session states. We call this method a *Markov Classifier (MC)*. The session states represent the TLS/SSL protocol and message types in single-directional traffic flows (client to server and server to client). In this way, we obtain a TLS/SSL session model per flow direction associated with each application. To the best of our knowledge, such a method is applied for the first time to the classification of encrypted traffic. The second method called a *Timestamp Classifier (TC)* considers the deviation between the timestamp in the TLS/SSL `Server Hello` message and the packet arrival time.

We validate the framework with experiments on three recent datasets gathered on two edge routers. They serve in a training phase to build application models. To obtain the ground truth, we use a simple *Domain Name System Classifier (DNSC)* that extracts application flows based on the corresponding host names. We only keep the flows for which we can find the domain names of the chosen portals and services so we are sure that training flows correspond to the considered applications. Then, we use the proposed methods to classify chosen applications and evaluate the amount of true positives and false positives. The chosen applications are representative of TLS/SSL encrypted traffic: PayPal, MBank (an on-line bank service), Mozilla, Twitter, Opera, Gadu-Gadu (a popular Polish instant messenger), and Dropbox.

We test the proposed classifiers separately or jointly on different datasets and we evaluate the contribution of each method to the final classification result. In the case of the most heterogenous datasets used for training and in the testing phase (the conditions favorable for the methods), we achieve very good accuracy with

more than 92% of the True Positive Rate and less than 1.5% of the False Positive Rate. Under adverse conditions for the methods, in the case of a less representative training dataset, the methods obtain slightly less accurate results with the True Positive Rate ranging from 77.7% to 80.1% and the False Positive Rate between 2.4% to 3.8%.
Our key contributions are:

- we successfully apply stochastic modeling in terms of a first-order homogeneous Markov chain to the classification of application flows encrypted with TLS/SSL;
- we propose a simple discrimination method based on the deviation between the timestamp in the TLS/SSL `Server Hello` message and the packet arrival time. The method improves the accuracy of application classification and allows efficient identification of Skype flows;
- our experimental results show very good classification performance on recent datasets reflecting different network environments and conditions.

10.2 Relevant Publications for Part IV

Maciej Korczyński and Andrzej Duda. Classifying TLS/SSL Encrypted Application Flows. *to be submitted*.

Chapter 11

Background on Encrypted Traffic Classification

Contents

11.1 Related Work

A lot of research effort concerned traffic classification [7, 5, 99, 3, 52, 2, 19, 16, 13, 25, 96]. Recent attention has turned to the problem of revealing and identifying encrypted applications and their underlying flows [18, 23]. We briefly review this recent work.

As new Internet applications started to use obfuscation methods (port masquerading, tunneling, and encryption) to evade traffic control and restrictions, simple inspection of port numbers is no longer a reliable classification mechanism [5, 15] (cf. Section 3.3.1). Moreover, payload encryption easily thwarts traditional payload-based classification based on pattern matching. Host behavior-based approaches [2, 19] (cf. Section 3.3.3) can potentially address the inefficiency of content-based methods. BLINC for example, proposes an interesting method based on observing and recognizing models of host behavior and then classifying its flows according to the models [2]. However, the method might be less effective when only a small part of behavioral information on individual hosts is available.

The second fundamentally different group of payload-independent approaches use flow-based features such as average packet sizes, packets inter-arrival times, or flow durations [3, 52, 25] (cf. Section 3.3.4). A recent hybrid method tries to identify TLS/SSL encrypted application layer protocols with a combination of a signature-based and a flow-based statistical analysis scheme [25]. The method is closely related to our proposal, however its objective is limited to the classification

of encrypted application layer protocols, while we concentrate more on an in-depth analysis of the TLS/SSL protocol and revealing application flows.

In our work, we adopt a payload-based approach (cf. Section 3.3.2) to demonstrate that it is possible to effectively reveal and classify application flows by inspecting application layer protocols. Risso et al. introduced a taxonomy of payload-based classification methods [13] and argued that they are mainly based on *pattern verification*. A key challenge in encrypted traffic classification is to replace traditional *pattern verification* with more sophisticated methods based on *statistical fingerprints*, for instance, by identifying groups of bits or bytes that exhibit unique distributions. Indeed, few researchers attempted to create such *statistical fingerprints* [18, 22, 23]. Bonfiglio et al. in their inspiring work have investigated Skype traffic transported mainly by UDP [18]. Skype traffic presents a major challenge for classification, because of proprietary software and internal encryption methods (cf. Chapter 8). However, they concluded that the Skype messages can be identified by examining the initial portion of the payload—so-called Start of Message (SoM). Specifically, authors examine randomness of initial groups of bits by means of a Chi-Square test. Some blocks of bits are random, whereas some other appear to be deterministic or mixed. While their innovative approach can be successfully extended to other traffic classification problems, the method depends on the observation of specific fields in the proprietary Skype protocol. Our method applies to a general case of the standard TLS/SSL encryption protocol.

In our previous work [23] (cf. Part III), we have considered the problem of detecting encrypted TCP Skype traffic tunneled over SSL and classifying Skype service flows such as voice calls, skypeOut, video conferencing, chat, file upload and download. The initial classification phase is based on Statistical Protocol IDentification (SPID) algorithm [22] that analyzes some statistical values of the packet payload. Our experiments showed that inspecting only the first five packets containing the payload is sufficient to reveal encrypted TCP Skype flows tunneled over SSL with Precision, Recall, and F-Measure equal to 100%.

11.2 TLS/SSL Overview

Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) are cryptographic protocols that provide secure communication between two parties over the Internet [104]. They encapsulate application protocols such as HTTP or FTP.

Figure 11.1 illustrates the structure of TLS/SSL and its components:

- *Record Protocol*: compresses and encrypts upper-layer data using the security

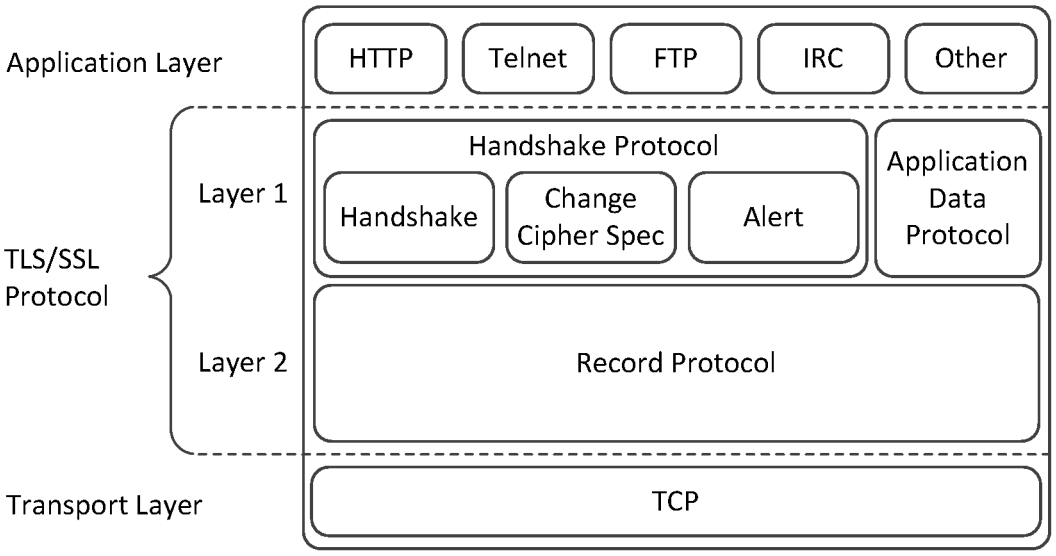

Figure 11.1: TLS/SSL protocol structure.

parameters configured by the Handshake Protocol.

- *Application Data Protocol*: provides application layer data to the Record Protocol.

- *Handshake Protocol*: negotiates parameters of a TLS/SSL session. Two communicating parties agree on the protocol version to use, they optionally authenticate each other, exchange information on the session ID, select cryptographic and compression algorithms, as well as the shared secret used to generate keys.

- *Change Cipher Spec Protocol*: signals modifications to encryption strategies. The protocol consists of a single message sent by either the client or the server to inform the other party that successive records will use the newly negotiated cryptographic algorithm and keys.

- *Alert Protocol*: reports an error condition or a change in status of the session.

Figure 11.2 presents an example message exchange between a client and a server during a TLS/SSL session with a full handshake.

The initial message exchange of `Client Hello` and `Server Hello` establishes the attributes: Protocol Version, Session ID, Cipher Suite, and Compression Method. Moreover, two random values are generated and exchanged: ClientHello.random and ServerHello.random (our Timestamp Classifier TC considers this value). The

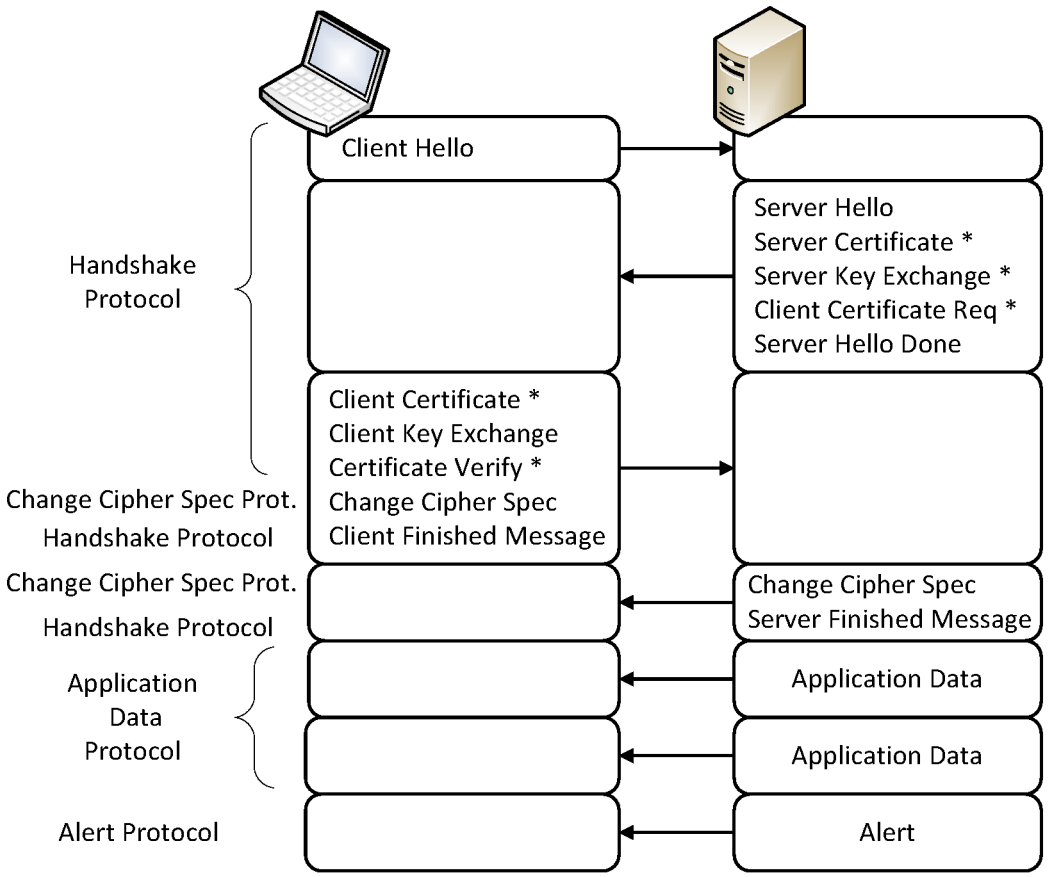

Figure 11.2: Message exchange during a TLS/SSL session with a full handshake.

key exchange uses up to four messages: server `Certificate`, `Server Key Exchange`, client `Certificate`, and `Client Key Exchange`. Then, the client sends `Change Cipher Spec` and the next `Finished` message is encrypted with the new algorithms and keys. In response, the server sends its own `Change Cipher Spec` message and the `Finished` message under the new cipher specification, which completes the TLS/SSL handshake and the two parties can exchange application layer data. The server terminates the session with an `Alert` message.

The exchange is an example—a session can be shortened by resuming previous sessions using the Session ID or it can be significantly modified depending on application requirements. Note that during the TLS/SSL handshake much information is sent as plaintext. After the handshake, only the type, the length of a record, and the TLS/SSL version are not encrypted.

Chapter 12

Classifiers

Contents

In this chapter, we use the principles of TLS/SSL protocol design to classify encrypted applications. In particular, we propose two classifiers that exploit different aspects and characteristics derived from TLS/SSL messages.

12.1 Markov Classifier

Our *Markov Classifier (MC)* takes into account message types in a TLS/SSL session observed at a client or a server: we refer to the server-side *MC* as *MCS* and to the client-side *MC* as *MCC*. Depending on the network environment, we expect slightly different characteristics for the client side, whereas the service-side model should be representative of all networks. Moreover, the separation of client- and server-side classifiers helps tackling the problem of asymmetric routing (if a network has two edge routers, routes may be asymmetric so each router can only gather information on a flow in one direction).

We use the following compact notation of messages types—the decimal protocol types and the Handshake message types present in TLS/SSL headers (cf. Figure 12.1).

To define the state space used in classification based on first-order homogeneous Markov chain, let us consider again the message exchange presented in the previous section (cf. Figure 11.2). This time, however, we will translate the client-server

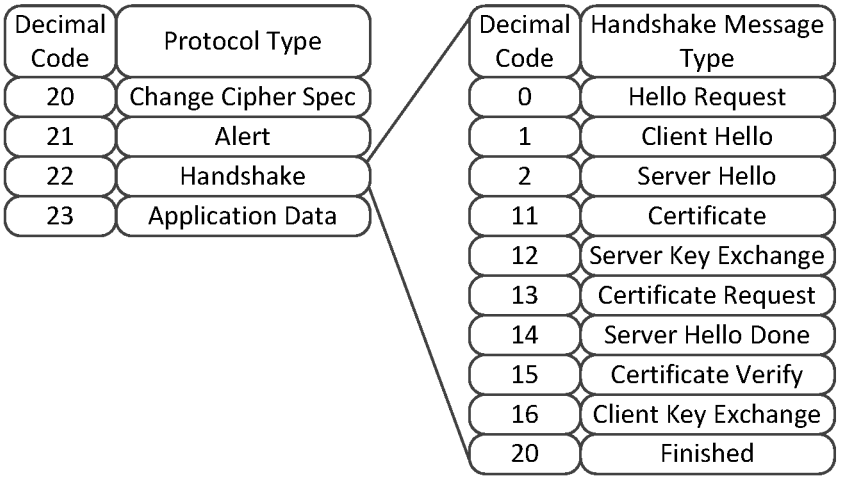

Figure 12.1: TLS/SSL protocol types and their corresponding decimal codes.

communication to corresponding decimal codes (cf. Figure 12.1) extracted from the TLS/SSL headers.

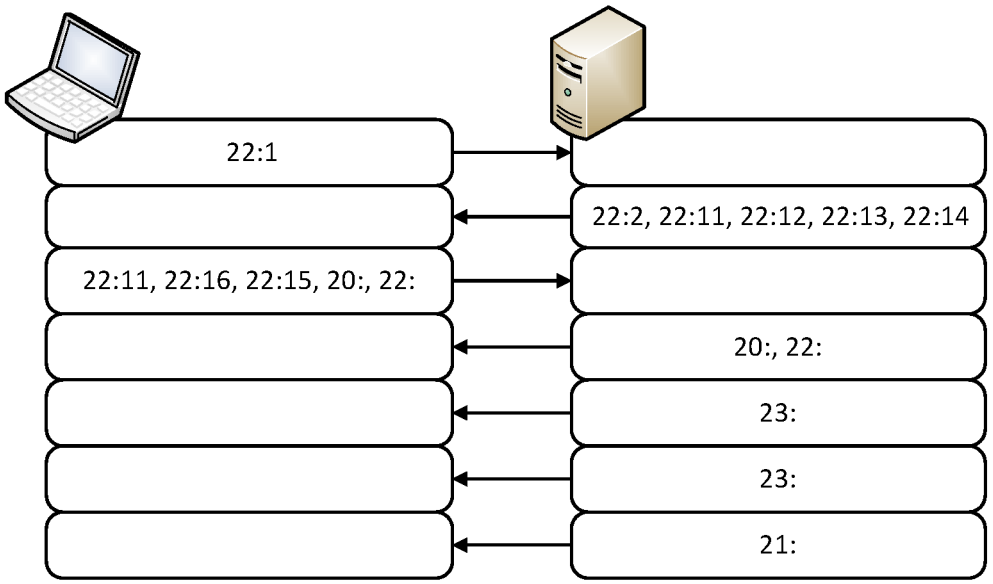

Figure 12.2: Message exchange during a TLS/SSL session with a full handshake in decimal codes.

As shown in the schematic diagram reported in Figure 12.2, we represent the `Client Hello` message as 22:1, whereas, e.g., `Server Key Exchange` message as 22:12. Moreover, all message types remain unencrypted until the end of the handshake procedure. Note that after `Change Cipher Spec` (represent as 20:), only

protocol types remain visible, because the message types are encrypted. Therefore, we cannot extract either handshake `Finished` message (represented as 22:) or underlying type of `Alert` protocol (represented as 21:). To summarize, from the diagram presented in Figure 12.2 we can distinguish one Markov chain per direction representing the underlying application. The client-side session, corresponding to MCC classification, is composed of two states, whereas the server-side session, associated with MCS classification, consist of five states.

We consider discrete-time random variable X_t for any $t = t_0, t_1, ..., t_n \in T$. It takes values $i_t \in \{1, ..., s\}$ corresponding to the observed TLS/SSL message types during a session. We assume that X_t is a first-order Markov chain [105, 29]:

$$\begin{aligned} P(X_t = i_t | X_{t-1} = i_{t-1}, X_{t-2} = i_{t-2}, \ldots, X_0 = i_0) \\ = P(X_t = i_t | X_{t-1} = i_{t-1}). \end{aligned} \tag{12.1}$$

We further assume that the Markov chain is homogeneous, i.e. a state transition from time $t-1$ to time t is time-invariant:

$$P(X_t = i_t | X_{t-1} = i_{t-1}) = P(X_t = j | X_{t-1} = i) = p_{i,j}, \tag{12.2}$$

with the transition matrix [105, 29]:

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,s} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ p_{s,1} & p_{s,2} & \cdots & p_{s,s} \end{bmatrix}, \tag{12.3}$$

where: $\sum_{j=1}^{s} p_{i,j} = 1$. We denote by:

$$Q = [q_1, q_2, \ldots, q_s], \tag{12.4}$$

the Initial Probability Distribution (IPD) where $q_i = P(X_t = i)$ at time $t = 0$.

Finally, the probability that a sequence of states $X_1, \ldots, X_T$ representing a single TLS/SSL session occurs is as follows:

$$P(\{X_1, \ldots, X_T\}) = q_i * \prod_{t=2}^{T} p_{i_{t-1}, i_t}. \tag{12.5}$$

To illustrate our approach, we present the observed transition probability matrices and the initial probability distributions of the MCS models for selected applications.

12.1.1 Example Server-side Markov Models

To illustrate our approach, we present in Figures 12.3, 12.4, 12.5, 12.6, and 12.7, the observed parameters of the server-side Markov models for chosen applications: the initial probability distribution, the transition probability matrix, and the termination probability distribution. We only provide four models out of seven applications as representative examples: PayPal, Twitter, Dropbox, and Gadu-Gadu. We also consider Skype as a special case. The model parameters come from the most representative dataset in our experiments (cf. Section 13.1.1 for the description of the `Campus2` dataset). For Skype, the measured data comes from the traffic dataset generated for Skype service flow classification [23] (cf. Section 9.2.1).

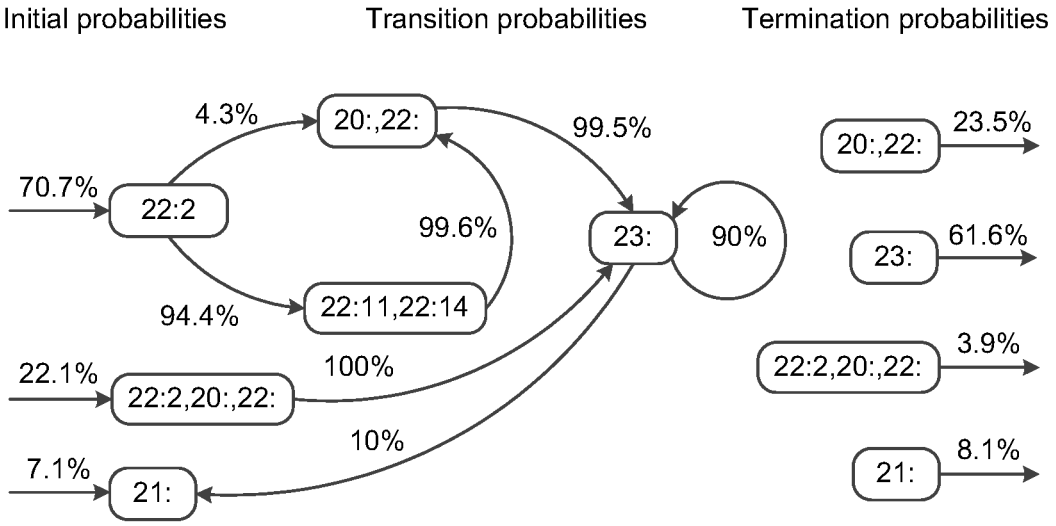

Figure 12.3: Parameters of the Markov model for PayPal.

12.1.1.1 PayPal

Figure 12.3 for PayPal shows that 92.8% of all initial sessions start with the `Server Hello` message, whereas 7.1% are `Alert` messages indicating handshake failure and closing the session even before the authentication process. Moreover, in case of successfully established sessions, the server always sends `Change Cipher Spec` indicating modifications in ciphering strategies. In 66.74% of sessions, the client authenticates the server, whereas in the remaining cases, the session is resumed using the Session ID (the handshake procedure is shortened to the `Server Hello` message). Finally, from the termination probability distribution, we can conclude that in most cases successful sessions do not end up with the `Alert` message coming from the server.

12.1.1.2 Twitter

Figure 12.4 indicates that 55% of TLS/SSL sessions are resumed from previously negotiated ones. Contrary to PayPal, almost 70% of remaining sessions do not change ciphering strategy after the server authentication procedure (no `Change Cipher Spec` message). Moreover, sessions tend to be rather short and are composed of `Application Data` message followed by `Alert` message (with the probability 58.6%) terminating the session.

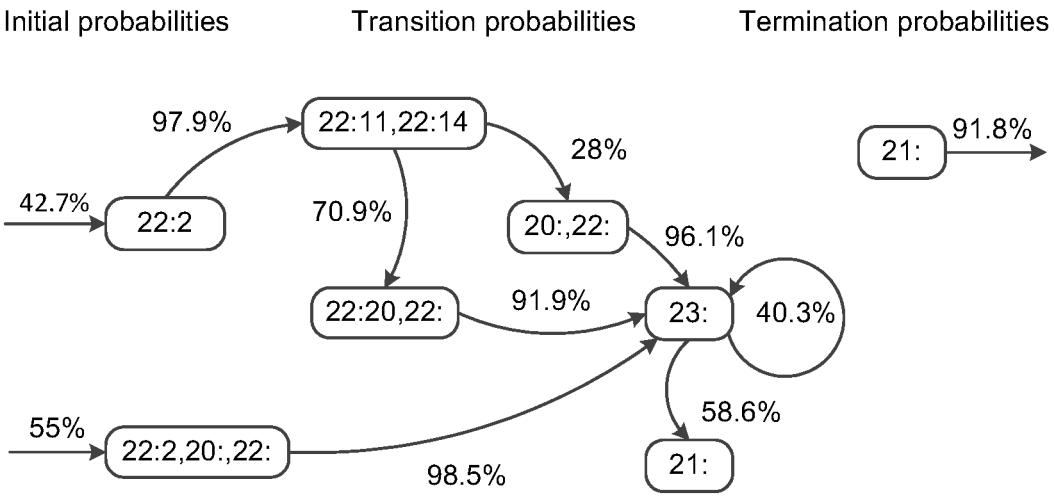

Figure 12.4: Parameters of the Markov model for Twitter.

12.1.1.3 Dropbox

The great majority of initial connections (98.3%) never resume previous TLS/SSL sessions, as shown in Figure 12.4. Furthermore, we can observe the `Server Key Exchange` message that contains some additional to `Certificate` message cryptographic information allowing the client to communicate the premaster secret. Contrary to previously presented PayPal and Twitter we do not observe at all `Change Cipher Spec` message at the initial handshake process. Sessions are composed of multiple `Application Data` messages which reflects the application character. Again, the majority of sessions (84.7%) is terminated by sending the `Alert` message. Despite the fact that sessions are highly consistent and message sequences repetitive, we can observe quite a few unusual states signaled by alert protocol.

12.1.1.4 Gadu-Gadu

As we can observe in Figure 12.6 Gadu-Gadu presents three possibilities to establish a session. The primary (64.5%) consist of typical `Server Hello` message

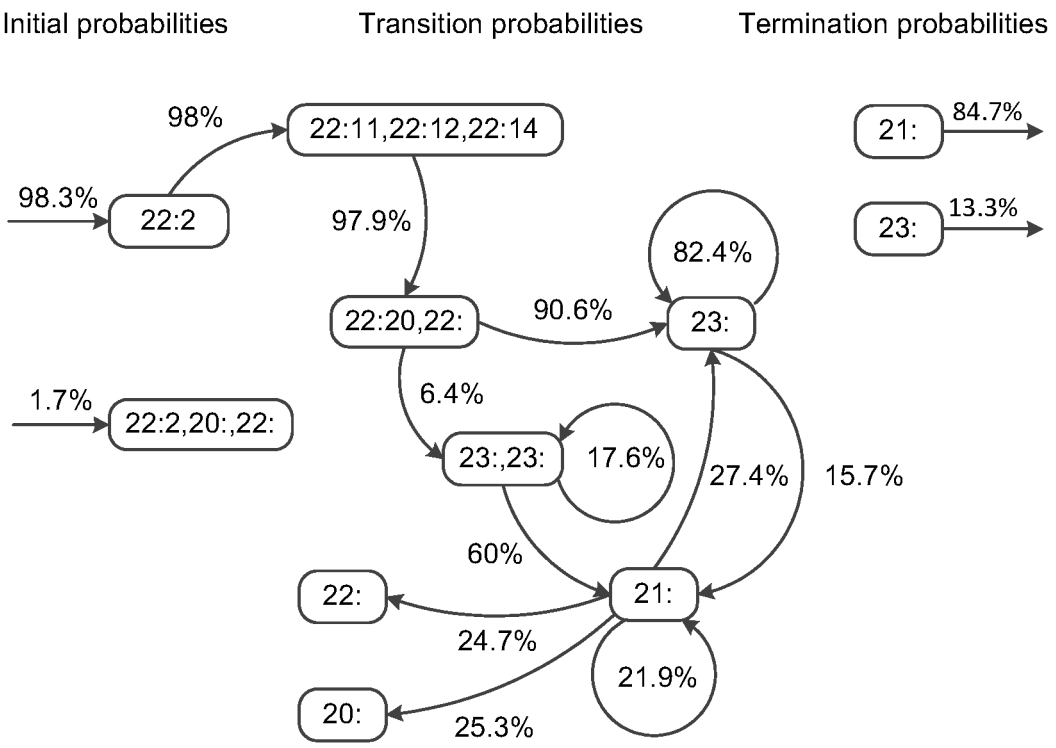

Figure 12.5: Parameters of the Markov model for Dropbox.

followed by `Certificate` and `Server Hello Done` message. In 95.7% cases `Change Cipher Spec` protocol message comes after. The second TLS/SSL handshake procedure additionally includes `Server Key Exchange` message. Finally, 15.9% sessions are being resumed. The figure suggests that on average Gadu-Gadu sessions consist of significantly larger number of messages in comparison with PayPal or Twitter. Moreover, we observe individual segments composed of multiple `Application Data` messages which presumably implies that application layer messages are relatively short—this feature stands out from previous cases.

12.1.1.5 Skype

Finally, we present the example of the Markov chain analysis of the Skype traffic tunneled through TLS/SSL. Figure 12.7 shows the Markov model parameters based on the analysis of 102 Skype flows coming from the `Skype` dataset (cf. Section 9.2.1). Skype traffic represents a special case—most of sessions (33.5%) are limited to a single `Server Hello` message. The transition matrix is built upon only 5 states and contains 24 non-zero elements. There are no messages indicating authentication process. We can also observe that the Markov chain does not reflect the phase of application data exchange.

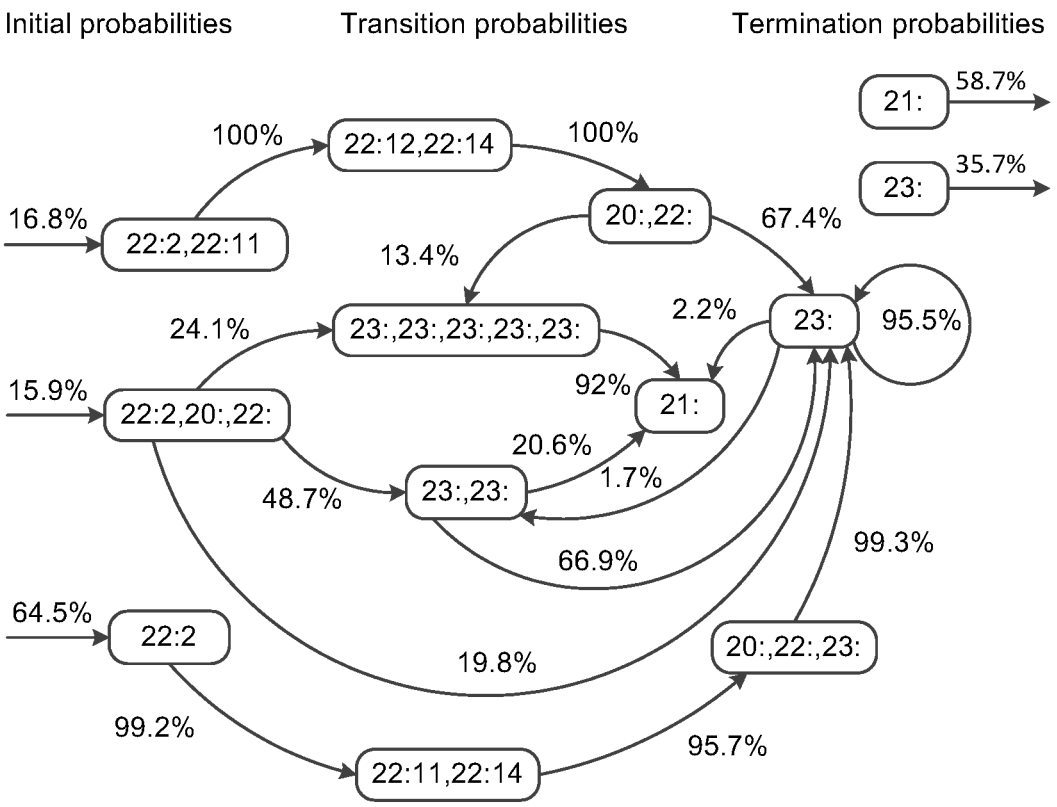

Figure 12.6: Parameters of the Markov model for Gadu-Gadu.

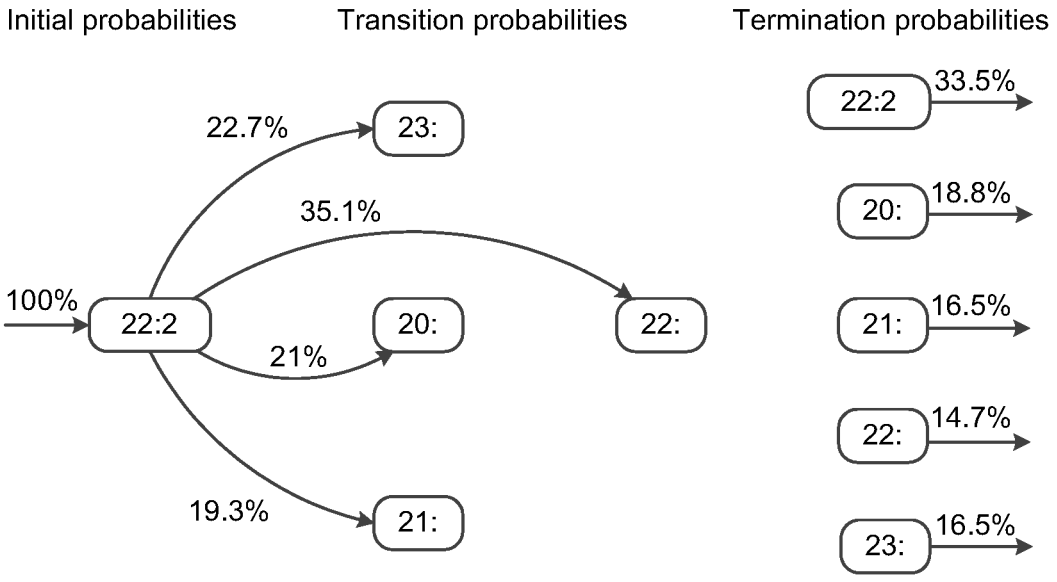

Figure 12.7: Parameters of the Markov model for Skype.

Actually, Skype is a proprietary piece of software that uses its own internal encryption mechanisms and a complex connection protocol designed for bypassing firewalls and establishing communication regardless of network policies [92, 93, 23]. Skype randomly selects ports and can switch to port 443 if it fails to establish a connection on chosen ports (cf. Chapter 8). Such technique is sufficient to bypass network-layer firewalls, however, it results in establishing a particular TLS/SSL session.

Table 12.1: Number of non-zero transition matrix elements for different applications

Application	# training flows (# servers)	# transition matrix elements
Gadu-Gadu	1196 (51)	63
MBank	2665 (3)	29
Opera	4357 (13)	26
PayPal	434 (6)	16
Mozilla	2669 (21)	38
Twitter	1530 (13)	36
Dropbox	3134 (317)	43

12.1.2 Discussion

The most important conclusion that we can draw from the examples is that the parameters of the Markov models for chosen applications differ a lot, which is the basis for accurate discrimination between applications. We have also found that the number of transition matrix elements in each application model significantly differ. Table 12.1 presents the number of non-zero elements in the transition matrix, the number of training sessions, and the number of servers that generated them, in the most representative `Campus2` dataset.

12.1.3 Training Phase

To build the Markov models representing the applications behavior, our classifier needs a training phase during which it analyzes ground-truth data containing application flows. The classifier analyzes traces pre-processed and filtered out by tshark [106] so that only TLS/SSL encrypted packets are passed to further inspection. Then, it uses the DNSC classification to create a benchmark dataset in which application flows are identified with a high confidence level (cf. Section 13.1.1). The

classifier only considers a limited number of fields—it first extracts the IP source and destination addresses, the source and destination ports to create unique identifiers of each session. Further, it builds 14 single-directional models corresponding to 7 chosen applications based on all unencrypted TLS/SSL protocol and message types.

12.1.4 Classification Phase

The classifier first pre-process the test dataset to extract application flows and then applies a decision process based on the Maximum Likelihood criterion [107]. Classification corresponds to a multi-hypothesis decision problem. More specifically, we consider seven hypothesis $H_i, i = 1, \ldots, 7$ corresponding to each of seven applications. We apply a classical approach based on Maximum Likelihood criterion—we select the hypothesis under which the data sequence Y is most likely:

$$H = \arg\max_{H_i} \log L(\{Y_1, \ldots, Y_T\}|H_i), \tag{12.6}$$

where $L(\{Y_1, \ldots, Y_T\})$ is the likelihood of the input data sequence under each hypothesis: $L(\{Y_1, \ldots, Y_T\}) \equiv P(\{X_1, \ldots, X_T\})$ (cf. Eq. 12.5) is the probability of a message sequence computed over each of the application models.

12.2 Timestamp Classifier

The second classifier analyzes the probability distribution of the *gmt_unix_time* field in the TLS/SSL `Server Hello` message. The initial handshake `Client Hello` and `Server Hello` messages include a random structure used later in encryption composed of two fields: *gmt_unix_time* (4 bytes) and *random_bytes* (28 bytes) [104]. Depending on applications, the *gmt_unix_time* field contains different timestamps: the current time and date set by the sender clock, a random sequence of 32 bits, or a constant value, and in particular, 0.

The Timestamp Classifier extracts the *gmt_unix_time* timestamp from the `Server Hello` message and the packet reception instant from the capture file. It keeps only the first 3 most significant bytes of the value to neglect possible small time differences between the sender and the device that captures packets. 3 bytes are considered separately as integer values. More formally, let us define $X_i, Y_i \in 0, \ldots, 255$ as random variables of possible values of byte $i \in 1, 2, 3$ of the *gmt_unix_time* field and the packet reception timestamp, respectively. $\delta_i = |X_i - Y_i|$, is a deviation measure between the *gmt_unix_time* field and the corresponding packet timestamp.

Table 12.2: Content characteristics of the *gmt_unix_time* field

gmt_unix_time	δ_i
Current time	Const (= 0)
Random	Rnd
Deterministic	Const

The relationship between *gmt_unix_time* field and δ_i is summarized in Table 12.2: we expect obtained fingerprints to present random or deterministic distributions depending on the application.

At the initial training phase, the method creates statistical fingerprints of the timestamp deviation for each application traffic. During the classification phase, the method computes the deviation and compares with the traffic models by means of the Kullback-Leibler (K-L) divergence [28]:

$$D(P||Q) = K\text{-}L(P,Q) = \sum_{\delta_i \in 0,\ldots,255} P(\delta_i) log_2 \frac{P(\delta_i)}{Q(\delta_i)}. \tag{12.7}$$

The K-L divergence is a measure of the difference between two probability distributions $P(\delta_i)$ and $Q(\delta_i)$. $P(\delta_i)$ represents the distribution of the byte frequency of δ_i in an observed flow and $Q(\delta_i)$ is the distribution corresponding to one of seven application models. Classification consists of comparing $P(\delta_i)$ with all known application models and selecting the one with the smallest average divergence. When TC is a part of the classification framework explained in Section 12.3, then we consider probability distributions $Q(\delta_i)$, where δ_i refers to the analyzed session for all application models.

After the analysis of $Q(\delta_i)$ for seven chosen applications over the `Campus2` dataset and for Skype over the `Skype` dataset, we have observed four groups of applications.

The largest group represented by Gadu-Gadu, Mozilla, Twitter, and Dropbox, has the same $Q(\delta_i)$ distribution determined by the current time. Another group contains PayPal with a uniform distribution. The $Q(\delta_i)$ distributions of MBank and Opera indicate that in both cases around 80% of all sessions has the *gmt_unix_time* field determined by the clock while the remaining 20% of values are evenly distributed. Finally, the most interesting statistical fingerprint corresponds to the Skype traffic tunneled through TLS/SSL protocol: the *gmt_unix_time* field is deterministic and interpreted by network protocol analyzers as `Jan 31, 2004 18:23:18 CET`—the whole 32-byte long random structure is in fact deterministic (note that it is normally used in the encryption process). Nevertheless, it is not so important

for Skype—it uses TLS/SSL only to establish a tunnel bypassing firewalls and it encrypts its data with a proprietary protocol.

So, in general, we can only determine a class of applications by inspecting the timestamps. However, in the case of Skype, the deterministic value of *gmt_unix_time* gives us an accurate signature for identifying the Skype traffic tunneled over TLS/SSL.

12.3 Classification Framework

Our classification framework is built upon the *Naive Bayes Classifier (NBC)* [30] that combines previously described methods, i.e. two Markov Classifiers corresponding to server and client-side models and the Timestamp Classifier that considers the randomness of timestamps. The Naive Bayes Classifier has been used extensively in the domain of traffic classification [3, 18] and proved to be very effective despite its simplicity [30].

The Naive Bayes Classifier applies the Bayes theorem with a strong (naive) assumption of the independence of input *features* describing an object. Let vector $F = F_1, \ldots, F_n$ represent the set of n features used to categorize an object in one of C classes. By applying the Bayes theorem we can quantify probability $P(C|F)$ that the object represent class C using the *a-priori* probability $P(F|C)$:

$$P(C|F) = \frac{P(C \cap F)}{P(F)} = \frac{P(F|C)}{P(F)} * P(C). \tag{12.8}$$

As we assume that each feature F_i is conditionally independent of another feature F_j, where $i \neq j$ and $i, j \in \{1, \ldots, n\}$, we can write:

$$P(F|C) = \prod_i P(F_i|C). \tag{12.9}$$

In our classification framework, different classifiers play the role of features. More specifically, class C is associated with one of seven applications, whereas F_i may represent the server-side or/and the client-side message chain or/and the timestamp fingerprint of the session to analyze. The final decision discriminating between different applications is made based on the Maximum Likelihood criterion (cf. Eq. 12.6). Finally, in the presented framework, we naively assume the independence between various classifiers, e.g., between MCS applied to messages coming from the server side and MCC used to compute the probability of message sequence originated at the client side. However, as our framework does not require a

strong independence assumption of underlying classifiers, NBC results in very good accuracy shown in the next section.

Chapter 13

Method Evaluation

Contents

13.1 Experiments

In this section, we present the results of applying the proposed classification framework to trace datasets.

13.1.1 Datasets

We have gathered three datasets at two edge routers located in Poland. `Campus1` and `Campus2` datasets come from a link connecting a campus network of the AGH University of Science and Technology in Cracow to the Internet. The link has the capacity of 70 Mbits/s for incoming and 30 Mbits/s for outgoing traffic. `Campus1` dataset contains a one day long trace starting on Thursday, March 1, 2012, whereas the 24 hours long `Campus2` dataset was obtained starting on Saturday, March 26, 2012. Both datasets contain traffic generated by standard services such as web, chat, mail, VoIP, file transfer, or streaming applications. The `Campus2` dataset is the most heterogenous one with numerous applications and a large number of active online users reaching 500 people (majority of users are university students and faculty). Due to strict policies enforced by firewalls and restrictions for certain streaming and p2p applications, users commonly tunnel restricted traffic.

`Enterprise` dataset contains traces gathered during a 93 hour period starting on Sunday, July 1, 2012 on a 20 Mbit/s link connecting a small IT company to

the Internet. The traffic reflects the company profile and mainly contains office applications such as mail, VoIP, or web. There are no strict firewall rules applied at the edge router. `Enterprise` dataset contains more homogeneous traffic than the two datasets captured on the campus.

To establish the ground truth, we have developed a *Domain Name System Classifier (DNSC)* to extract and classify TLS/SSL application flows according to their domain names. More specifically, DNSC matches hostnames against our array of signature strings like for example *twitter, twttr* in case of Twitter. The method is simple and results in a very high confidence level confirmed by manual payload inspection. Nevertheless, we might not cover all instances of signatures for a particular application. Another constraint of the approach is that we cannot obtain the instances of applications if we are not able to resolve IP addresses into the corresponding domain names. To overcome these limitations, we have used in our experimental evaluation only the traffic for which the IP address resolution was possible and corresponding strings are straightforward and unambiguous.

Table 13.1: Applications, the number of application flows, the number of servers vs. number of clients in three datasets

Application	`Campus1`	`Campus2`	`Enterprise`
PayPal	546 (9 - 96)	434 (6 - 97)	172 (13 - 11)
Twitter	1411 (17 - 29)	1530 (13 - 30)	157 (11 - 6)
Dropbox	1160 (171 - 95)	3134 (317 - 133)	177 (31 - 9)
Gadu-Gadu	987 (50 - 321)	1196 (51 - 343)	30 (17 - 4)
MBank	321 (2 - 49)	2665 (3 - 51)	44 (2 - 6)
Opera	3246 (15 - 140)	4357 (13 - 132)	2034 (13 - 16)
Mozilla	2436 (20 - 271)	2669 (21 - 292)	2867 (24 - 68)

Table 13.1 shows the parameters of three datasets: the number of application flow samples and in the brackets, the number of servers versus the number of clients that use the service (for example, `Campus1` dataset contains the traces of 321 users who have connected to 50 Gadu-Gadu servers in 987 flows).

Table 13.2 presents more statistics on the relationship between datasets: the corresponding number of servers and clients common to respective datasets (e.g., there are 6 common PayPal servers in `Campus1` and `Campus2` datasets as well as 46 common clients). Their purpose is to estimate the applicability of datasets. From the joint analysis of two presented tables, in case of Gadu-Gadu for example, we can expect very good classification results if the server-side behavior is computed on the basis of the `Campus2` dataset and is applied to the `Enterprise` dataset (`Campus2`

Table 13.2: Applications and the corresponding number of servers and clients common to respective datasets

Application	Campus1 ∩ Campus2	Enterprise ∩ Campus1	Enterprise ∩ Campus2
PayPal	6 - 46	2 -	2 -
Twitter	12 - 8	9 -	9 -
Dropbox	113 - 65	19 -	21 -
Gadu-Gadu	44 - 230	16 -	16 -
MBank	1 - 12	1 -	1 -
Opera	13 - 105	12 -	11 -
Mozilla	17 - 169	6 -	6 -

contains almost all Gadu-Gadu servers accessed in `Enterprise`, 16 out of 17). By contrast, we may expect slightly worse results for Gadu-Gadu client models because of a small number of common clients (`Campus2` has 343 clients different from 4 clients in the `Enterprise` dataset).

Finally, we will often refer to Skype as an example of traffic tunneled through TLS/SSL. The evaluation runs on a set of packet traces referred to as `Skype` dataset generated in the experiments of classifying Skype service flows (cf. Section 9.2.1).

13.1.2 Criteria for Classification Performance

We assume that the classification based on the DNSC reference classifier provides the ground truth and we evaluate the proposed classifiers with respect to its classification decisions. We consider two meaningful metrics to assess the performance of a classification method: the True Positive Rate and False Positive Rate, denoted as TPR and FPR, respectively (cf. Eq. 3.1, 3.2). TPR is known as *sensitivity*, and 1− FPR is commonly referred to as *specificity*. True Positive occurs when the classification result is consistent with the classification decision taken by DNSC and the application session is correctly classified as a given application, e.g., a PayPal session is accurately recognized as PayPal. Conversely, False Positive occurs when the classification result is inconsistent with the decision taken by the reference classifier and a session is incorrectly classified, e.g., a Twitter session is falsely recognized as PayPal.

13.1.3 Classification Results

In this section, we report on the classification results of the proposed framework: we first test MCS+MCC+TC and MCS+TC on `Campus1` and `Enterprise` datasets,

respectively when `Campus2` dataset served for training.

Table 13.3: Results for MCS+MCC+TC on Campus1 dataset: applications, total number flows, number of not classified flows, absolute TP and FP as well as their rates. Training set: Campus2

Application	# flows	# missed	# TP	TPR	#FP	FPR
PayPal	546	55	437	0.89	54	0.006
Twitter	1411	30	1138	0.824	174	0.021
Dropbox	1160	1	1102	0.951	32	0.004
Gadu-Gadu	987	18	939	0.969	24	0.003
MBank	321	20	277	0.92	73	0.008
Opera	3246	224	2832	0.937	217	0.032
Mozilla	2436	5	2375	0.977	80	0.011

Table 13.3 shows the results for MCS+MCC+TC classifiers. Let us take the example of Mozilla for which we observe that the TP rate is very large (97.7%) with relatively small rate of FP (1.1%). The good results come from the fact that for Mozilla, `Campus1` and `Campus2` share common servers and clients (`Campus2` covers 17 out of 20 servers and 169 out of 271 clients that also occurred in the analyzed `Campus1` dataset, cf. Tables 13.1 and 13.2).

In the case of Twitter, we can observe less accurate results (TPR of 82.4%, FPR of 2.1%), because the overlap of clients and servers in the two datasets is not so significant. By manual inspection, we have observed that the degradation in the TP rate for Twitter is due to some similarities of its MC models with Opera and MBank, which also results in a slightly higher rate of FP for Opera and MBank, because some Twitter sessions are falsely classified as either Opera or MBank.

Table 13.4 presents the results for the MCS+TC classifiers on the `Enterprise` dataset with the `Campus2` dataset used for training. We can see that for Gadu-Gadu, the classifiers have correctly recognized all application instances with only two sessions incorrectly classified as Gadu-Gadu. A similar reasoning applies to the `Enterprise` dataset—the training `Campus2` dataset covers 16 out of 17 servers (cf. Tables 13.1 and 13.2). However, notice that 17 Gadu-Gadu sessions (that correspond to 56% of all application instances) were not classified. By manually inspecting the flows, we have observed slightly different TLS/SSL message sequences compared to those in the training phase.

All MBank sessions were correctly classified, but this time, there were no unrecognized sessions (marked in Table 13.4 as *missed*). We can explain less accurate results with TPR equal only to 64.3% for PayPal by a small number of session

Table 13.4: Results for MCS+TC on Enterprise dataset: applications, total number flows, number of not classified flows, absolute TP and FP as well as their rates. Training set: Campus2

Application	# flows	# missed	# TP	TPR	#FP	FPR
PayPal	172	29	92	0.643	212	0.04
Gadu-Gadu	30	17	13	1	2	0.000
Twitter	157	0	125	0.796	154	0.029
Dropbox	177	1	168	0.955	16	0.003
MBank	44	0	44	1	52	0.010
Opera	2034	4	1723	0.848	459	0.135
Mozilla	2867	2	2319	0.809	49	0.019

instances in the training phase. All manually inspected PayPal flows either slightly differ from the pre-computed model or they are classified as other applications for which the models were constructed using a richer set of session instances and have a complex structure allowing for diverse messages sequences.

13.1.4 Classifier Selection

In this part, we want to evaluate the impact of the proposed classifiers on the final classification decision by the framework. For all experiments reported in this section, we analyze up to five transitions of both server and client-side Markov chains (we explain the limit of five transitions in the discussion of parameter calibration in Section 13.1.5). We consider the analysis of `Campus1` and `Enterprise` datasets under the training phase on `Campus2` as well as the analysis of `Campus1` and `Campus2` datasets under the training phase on `Enterprise`. We present the results by plotting TPR on y-axes and FPR on x-axes (cf. Figure 13.1)—the values correspond to the average TPR and FPR obtained for each of seven applications, i.e. Gadu-Gadu, MBank, Opera, PayPal, Mozilla, Twitter, and Dropbox. We test our Bayes framework composed of: the single MCS, MCC, or TC classifier, and the joint MCS+TC, MCC+TC, MCS+MCC+TC classifiers.

Let us first focus on the most heterogeneous `Campus2` dataset used for training. The left-hand side of Figure 13.1 presents the results of the analysis on `Campus1` and `Enterprise` datasets. We can observe that Markov models computed for the server side give significantly better results than the models constructed at the client side regardless of the dataset used in the classification process. Server-side models are much easier to build and are more universal across different networks, while client-side models are more network-specific. Recall that clients on different networks

may usually access a large number of common servers (e.g., the Gadu-Gadu servers accessed from the campus or from the enterprise) that provide a service regardless of the client and network location, whereas there is a small number of common clients, if any (cf. Tables 13.1 and 13.2). When the MCS and MCC are combined, we obtain

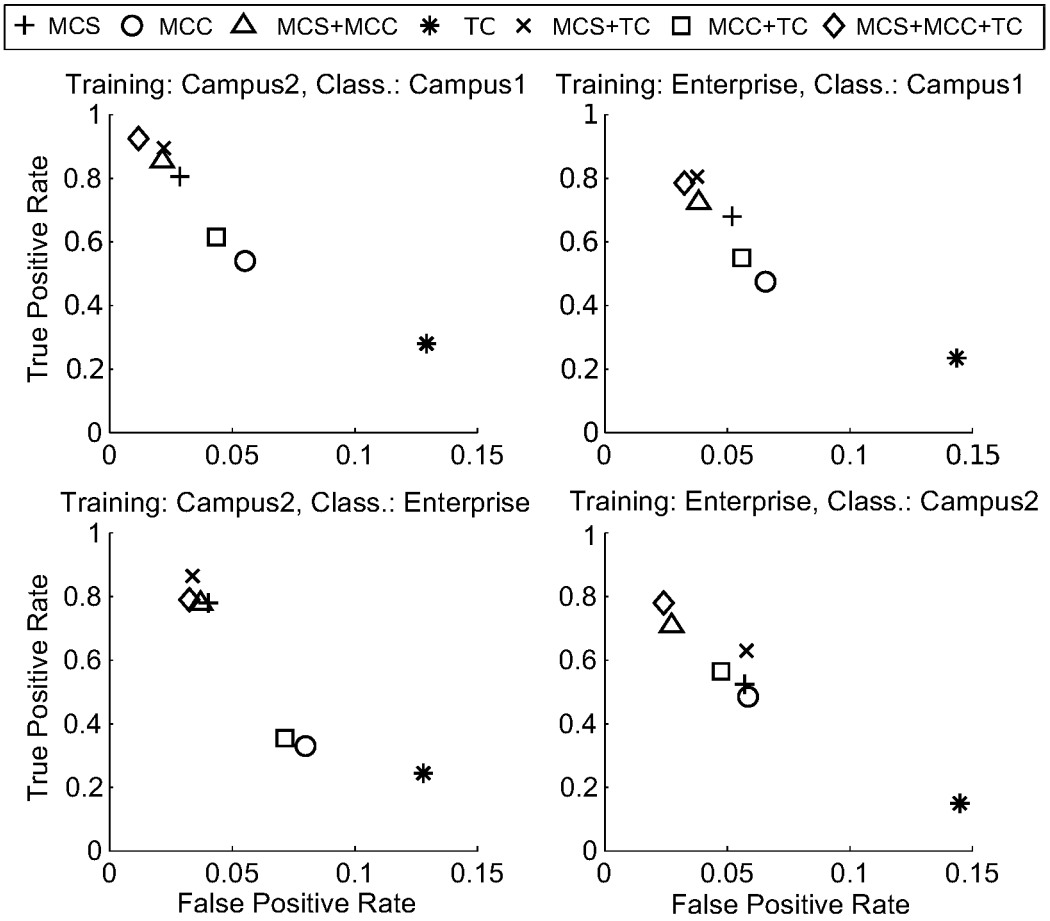

Figure 13.1: TPR vs. FPR. Comparison of different sets of classifiers for various analyzed and training datasets.

better results for the `Campus1` dataset due to similar users profile, whereas the rates of TP and FP for the `Enterprise` dataset remain almost unchanged. Single TC classifier results in very low TPR and relatively high FPR. However, when jointly used with other classifiers, it considerably improves the classification performance.

We can conclude from the experiments that the joint usage of all three classifiers gives the best results of TPR (92.4%) and FPR (less that 1.2%) if the training and testing datasets come from the same subnetworks. Conversely, combining only MCS with TC allows to achieve TPR equal to 86.5% with FPR less than 3.4%, if the model is computed over the `Campus2` dataset and analysis performed over the

`Enterprise` dataset.

`Enterprise` as a training dataset leads to slightly worse results than in the previous case, because the MC models do not cover enough TLS/SSL session instances to be effective compared to the associated `Campus1` and `Campus2` datasets (cf. Tables 13.1 and 13.2). The results reported in the right top graph of Figure 13.1 indicate that the set of MCS+TC performs slightly better (TPR of 80.1%, FPR of 3.8%) than the jointly used MCC+MCS+TC classification (TPR of 78.1%n FPR of 3.3%). Finally, the results obtained for the classification of the most heterogeneous `Campus2` dataset are much worse for the MCS classifier, which leads to the selection of MCS+MCC+RC for the classification framework (cf. the right bottom graph of Figure 13.1).

From the above experimental results, we can conclude that

- MCS is the essential part of the classification framework. If we build the application models on a heterogenous training dataset that covers a wide range of session instances, they can apply across different subnetworks;
- MCC can be very effective when the framework analyzes the datasets collected on the same subnetwork used for collecting training datasets;
- the joint usage of TC with Markov classifiers gives considerably better classification results compared to the performance only based on MCS and MCC.

13.1.5 Parameter Calibration

In this section, we investigate the sensitivity of a separately used server-side or client-side Markov classifier on the number of considered transitions in a given Markov chain. We perform the sensitivity analysis on the most heterogenous `Campus2` dataset pre-classified by applying DNSC classification described in Section 13.1.1. TP and FP rates represent the average values obtained for each of seven considered applications. Figure 13.2 presents the impact of the number of state transitions used in both training and testing phase on TP and FP rates. We can notice that for both TPR and FPR, classification based on only 5 transitions is roughly as accurate as classification based on 100 transitions. These results highlight the scalability of the proposed framework—it requires considering only several first TLS/SSL messages to obtain very good classification results.

13.2 Conclusion

In this part, we have defined a framework based on two complementary methods for classifying applications. The first one uses a stochastic model representing

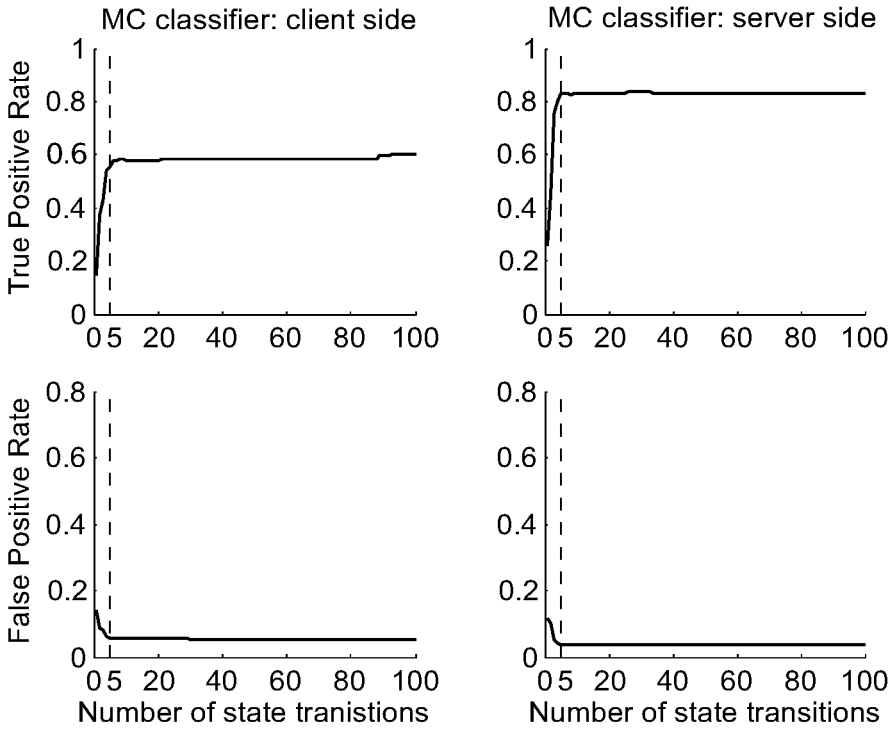

Figure 13.2: Calibration process of MC: TPR and FPR as a function of the number of state transitions for the client-side (left) and for the server-side (right) on the `Campus2` dataset.

TLS/SSL session states as a first-order homogeneous Markov chain. The session states represent the TLS/SSL protocol and message types in single-directional traffic flows (client to server or server to client). We have observed that the parameters of the Markov models for chosen applications differ a lot, which is the basis for accurate discrimination between applications. The second classifier analyzes the deviation between the *gmt_unix_time* field in the TLS/SSL `Server Hello` message and the packet arrival time. We combine the methods using a *Naive Bayes Classifier (NBC)*.

We validated the framework with experiments on three recent datasets gathered on two edge routers. We apply the proposed framework to classification of seven popular applications that use TLS/SSL for security. In conditions favorable for the classification framework, the methods achieve very good accuracy with more than 92% of the True Positive Rate and less than 1.5% of the False Positive Rate. Under adverse conditions for the methods, they obtain slightly less accurate results with the True Positive Rate ranging from 77.7% to 80.1% and the False Positive Rate between 2.4% to 3.8%.

Chapter 14

Conclusions

Even though the domain of traffic classification is relatively well explored, our primary goal is to enrich existing research efforts by our own contributions. The issues considered in this thesis were inspired by common problems existing in real - operational networks. Thus, we have tried to bridge the gap between academia and professional practice. In this chapter, we summarize the thesis claims and highlight the future directions of this research.

Efficiency and scalability

In Part II, we have proposed a detection scheme for high-volume SYN flooding attacks and low-volume portscan activity. We have demonstrated that our method achieves a high attack detection rate (True Positive Rate). Moreover, in comparison with existing methods, we have reduced the False Positive Rate, i.e., when legitimate packets are classified as malicious ones. Finally, by using sampling methods, we have significantly reduced the influence of packet sampling on the performance of the detection scheme. However, as far as scalability is concerned, we believe that the future practical implementation based on Snort [61] or Bro [108] might be even more convincing than the evaluation process based on the proof-of-the-concept algorithm presented in this thesis.

Challenges ahead

Among various challenges in the domain of traffic analysis, classifying encrypted flows seems to be the most urgent one because, an increased number of applications make use of encryption, e.g., Tor [100], I2P [101], BitTorrent [102], IMule [103], Skype [93]. In Part III, we have proposed a classification method for recognizing Skype encrypted traffic tunneled over SSL and identifying its service flows. Then, in Part IV, we have defined a more generic framework based on two complementary methods for classifying applications encrypted with TLS/SSL protocol. Our results shed a new light on the potential of approaches based on application-layer protocol analysis for encrypted and tunneled traffic.

Refute the myths

In the research community, there is a number of common beliefs that should be verified such as the opinion that port-based classification approaches are inadequate any longer in traffic classification, or that DPI methods do not scale to hight bandwidth rates. In our work, we argue the belief that payload-based methods always fail in case of encrypted traffic. In Part IV, we have investigated in-depth TLS/SSL header structure and we have proposed a framework for encrypted traffic classification. We believe that there are still many "myths" in the domain of traffic classification that should be investigated in detail and perhaps revised.

Formalization of the domain

A comparison (if possible) between different methodologies is an important part of any evaluation process. However, it is a difficult task, not only due to the lack of a shared testing platform or easily available packet traces, but basically because of the lack of a common understanding of concepts such as the definition of the classification classes. In this thesis (cf. Section 3.2), we have addressed this particular problem by proposing three classification goals, i.e., we propose to classify traffic according to its *category*, *application-level protocol*, or *application* that generates traffic. Moreover, in Section 3.3 we have presented an extended taxonomy for approaches in traffic classification based on the research presented in this thesis. For completeness, another attempt aiming at formalizing the domain based on ontology paradigms has been proposed recently [109].

Practical deployment

Many of research methodologies, especially based on statistical methods, have never been evaluated in the real- -operational networks (with some exceptions, for example, TCP STatistic and Analysis Tool (Tstat) [110] or Hybrid Traffic Identification (HTI) [24]). Thus, as mentioned before, one of interesting research directions would be practical implementation and deployment of methods presented in this thesis in a campus network.

Inter-domain portability

Although, we have presented intrusion detection as a sub-domain of traffic classification problem, it is often considered in the literature as a separate research subject. Thus, we believe that applying some of existing classification methods in detection of intrusive activity would be an interesting research subject. Even if not presented in this thesis, we have successfully applied

the method proposed in Part III to the detection of malicious traffic—the propagation of Worm.Win32.Skipi.b that spreads over the Skype messenger [111]. In the future work, we would like to explore how the proposed methods can be extended to other classification problems.

Appropriate feature selection

The problem of feature selection and parameter calibration has been well studied in the domain of traffic classification including the presented thesis. However, we believe that some standard recommendations should be introduced to separate training traces from datasets used in parameter tuning and in the evaluation process. Indeed, feature selection and parameter calibration methods tend to optimize performance results for particular datasets. Moreover, in the future work, we may consider automatic feature selection and calibration process as a part of practical implementations.

Hybrid approaches

As it was mentioned earlier, in recent years, we have observed that application developers tend to evade traffic classification by encryption and other obfuscation methods. Even some governments are interested in an anonymous p2p technology, for example, Tor project aims at protecting users' privacy [100]. As a result, more complexed, hybrid methods combining different approaches should be applied in the future such as the one presented in Part III of the thesis, which puts together traffic flow features and complex DPI elements to identify Skype service flows.

Ground truth

The last but not least issue revised in the thesis conclusions is related to pre-labeled datasets, namely to the ground-truth information, crucial for every evaluation process. In Part IV, we have developed a simple method called DNSC to extract encrypted application flows according to their domain names. Although the method can classify even encrypted traffic with high confidence level, it is characterized by a limited classification scope. As a result, we believe that a common testbed based on multiple reliable, but not necessarily scalable or light weight algorithms is required for cross-checking and generating a valid ground truth.

Bibliography

[1] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic Classification Using Clustering Algorithms. In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data (MineNet'06)*, pages 281–286. ACM, 2006. 5, 14, 26

[2] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'05)*, pages 229–240. ACM, 2005. 5, 14, 19, 20, 23, 77, 83, 87

[3] Andrew W. Moore and Denis Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 50–60. ACM, 2005. 5, 14, 20, 21, 23, 77, 83, 87, 101

[4] Min Zhang, Wolfgang John, KC Claffy, and Nevil Brownlee. State of the Art in Traffic Classification: A Research Review. *10th International Conference on Passive and Active Network Measurement (PAM'09) Student Workshop*, pages 1–2, April 2009. 14, 15, 17, 18

[5] Andrew W. Moore and Konstantina Papagiannaki. Toward the Accurate Identification of Network Applications. *Proceedings of the Passive and Active Measurement Workshop*, pages 41–54, 2005. 14, 18, 23, 77, 83, 87

[6] Anthony Mcgregor, Mark Hall, Perry Lorier, and James Brunskill. Flow Clustering Using Machine Learning Techniques. In *Proceedings of the 5th International Conference on Passive and Active Network Measurement (PAM'04)*, pages 205–214, 2004. 14

[7] 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)*, pages 512 – 521. ACM, 2004. 14, 18, 19, 21, 83, 87

[8] The Cooperative Association for Internet Data Analysis (CAIDA). Internet Traffic Classification. http://www.caida.org. 14

[9] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic Classification on the Fly. *ACM SIGCOMM Computer Communication Review*, 36(2):23–26, 2006. 14

[10] 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)*, pages 313–326. ACM, 2006. 14

[11] 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)*, pages 197–202. ACM, 2005. 14

[12] Christopher Kohnen, Christian Uberall, Florian Adamsky, Veselin Rakocevic, Muttukrishnan Rajarajan, and Rudolf Jager. Enhancements to Statistical Protocol IDentification (SPID) for Self-Organised QoS in LANs. In *Proceedings of the 19th IEEE International Conference on Computer Communications and Networks (ICCCN'10)*, pages 1–6. IEEE, 2010. 14, 21

[13] Fulvio Risso, Mario Baldi, Olivier Morandi, Andrea Baldini, and Pere Monclus. Lightweight, Payload-Based Traffic Classification: An Experimental Evaluation. In *IEEE International Conference on Communications, ICC'08*, pages 5869 – 5875, May 2008. 14, 18, 21, 84, 87, 88

[14] Internet Assigned Numbers Authority (IANA). http://www.iana.org/assignments/port-numbers. 17, 83

[15] Holger Dreger, Anja Feldmann, Michael Mai, Vern Paxson, and Robin Sommer. Dynamic Application-layer Protocol Analysis for Network Intrusion Detection. In *Proceedings of the 15th Conference on USENIX Security Symposium*. USENIX Association, 2006. 18, 87

[16] 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 International Conference on Emerging Networking Experiments and Technologies (CoNEXT'08)*, pages 1–12. ACM, 2008. 18, 21, 26, 83, 87

[17] Gregor Maier, Anja Feldmann, Vern Paxson, and Mark Allman. On Dominant Characteristics of Residential Broadband Internet Traffic. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference (IMC'09)*, pages 90–102. ACM, 2009. 18, 83

[18] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli. Revealing Skype Traffic: When Randomness Plays with You. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'07)*, pages 37–48. ACM, 2007. 18, 19, 20, 21, 23, 57, 60, 77, 78, 84, 87, 88, 101

[19] 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)*, pages 315 – 320. ACM, 2007. 19, 20, 83, 87

[20] Marios Iliofotou, Hyun-chul Kim, Michalis Faloutsos, Michael Mitzenmacher, Prashanth Pappu, and George Varghese. Graption: A Graph-based P2P Traffic Classification Framework for the Internet Backbone. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 55(8):1909–1920, June 2011. 20

[21] Thuy T. T. Nguyen and Grenville J. Armitage. A Survey of Techniques for Internet Traffic Classification Using Machine Learning. *IEEE Communications Surveys and Tutorials*, 10(1-4):56–76, 2008. 20

[22] Erik Hjelmvik and Wolfgang John. Statistical Protocol Identification with SPID: Preliminary Results. *Sixth Swedish National Computer Networking Workshop (SNCNW'09)*, May 2009. 21, 23, 57, 62, 88

[23] Maciej Korczyński and Andrzej Duda. Classifying Service Flows in the Encrypted Skype Traffic. *2012 IEEE International Conference on Communications (ICC'12)*, pages 1–5, June 2012. 21, 23, 58, 84, 87, 88, 94, 98

[24] Marcin Pietrzyk, Taoufik En-Najjary, Guillaume Urvoy-Keller, and Jean-Laurent Costeux. Hybrid Traffic Identification. Technical Report EURECOM 3075, Eurecom, 2010. 21, 112

[25] Guang-Lu Sun, Yibo Xue, Yingfei Dong, Dongsheng Wang, and Chenglong Li. An Novel Hybrid Method for Effectively Classifying Encrypted Traffic. In *IEEE Global Telecommunications Conference (GLOBECOM'10)*, pages 1–5, December 2010. 21, 77, 87

[26] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. Early Application Identification. In *Proceedings of the 2006 ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT'06)*, pages 1–12. ACM, July 2006. 21, 23, 77

[27] Sebastian Zander, Thuy Nguyen, and Grenville Armitage. Self-Learning IP Traffic Classification Based on Statistical Flow Characteristics. In *Proceedings of the 6th International Conference on Passive and Active Network Measurement (PAM'05)*, pages 325–328, Berlin, Heidelberg, 2005. Springer-Verlag. 21, 23

[28] Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951. 22, 62, 100

[29] Iain L. MacDonald and Walter Zucchini. *Hidden Markov and Other Models for Discrete-Valued Time Series*. Chapman & Hall, 1997. 22, 93

[30] Devinderjit Sivia and John Skilling. *Data Analysis: A Bayesian Tutorial*. Oxford University Press, July 2006. 22, 101

[31] Dongyan Zhang, Chao Zheng, Hongli Zhang, and Hongliang Yu. Identification and Analysis of Skype Peer-to-Peer Traffic. In *Proceedings of the Fifth International Conference on Internet and Web Applications and Services (ICIW'10)*, pages 200–206. IEEE Computer Society, 2010. 23, 57, 59, 77

[32] Ignasi Paredes-Oliva, Ismael Castell-Uroz, Pere Barlet-Ros, Xenofontas A. Dimitropoulos, and Josep Sole-Pareta. Practical Anomaly Detection Based on Classifying Frequent Traffic Patterns. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'12) Workshop*, pages 49–54. IEEE Press, 2012. 23

[33] Riyad Alshammari and A. Nur Zincir-Heywood. Unveiling Skype Encrypted Tunnels Using GP. *2010 IEEE Congress on Evolutionary Computation (CEC'10)*, pages 1–8, July 2010. 23, 57, 77

[34] Philip A. Branch, Amiel Heyde, and Grenville J. Armitage. Rapid Identification of Skype Traffic Flows. In *Proceedings of the 18th International Workshop on Networking and Operating Systems Support for Digital Audio and Video*, pages 91–96. ACM, 2009. 23, 57

[35] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. Quantifying Skype User Satisfaction. *ACM SIGCOMM Computer Communication Review*, 36(4), October 2006. 23, 57, 77, 78

[36] Christian Dewes, Arne Wichmann, and Anja Feldmann. An Analysis of Internet Chat Systems. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement (IMC'03)*, pages 51–64. ACM, 2003. 23

[37] Gerhard Munz, Hui Dai, Lothar Braun, and Georg Carle. TCP Traffic Classification Using Markov Models. In *Proceedings of the Second International Conference on Traffic Monitoring and Analysis (TMA'10)*, pages 127–140, Berlin, Heidelberg, 2010. 23

[38] Valentin Carela-Espanol, Pere Barlet-Ros, Marc Sole-Simo, Alberto Dainotti, Walter de Donato, and Antonio Pescape. K-Dimensional Trees for Continuous Traffic Classification. In *Proceedings of the Second International Conference on Traffic Monitoring and Analysis (TMA'10)*, pages 141–154, Berlin, Heidelberg, 2010. 23

[39] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An Analysis of Internet Content Delivery Systems. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, pages 315–327. ACM, 2002. 23

[40] Subhabrata Sen and Jia Wang. Analyzing Peer-to-Peer Traffic Across Large Networks. *IEEE/ACM Transactions on Networking*, 12(2):219–232, April 2004. 23

[41] R. Rangadurai Karthick, Vipul P. Hattiwale, and Balaraman Ravindran. Adaptive Network Intrusion Detection System Using a Hybrid Approach. In *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS'12)*, pages 1–7, January 2012. 23

[42] Haining Wang, Danlu Zhang, and Kang G. Shin. Detecting SYN Flooding Attacks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'02)*, pages 1530–1539. IEEE Press, June 2002. 23, 33, 35, 53

[43] Haining Wang, Danlu Zhang, and Kang G. Shin. Change-Point Monitoring for the Detection of DoS Attacks. *IEEE Transactions on Dependable and Secure Computing*, 4(1):193–208, October 2004. 23, 33, 35, 53

[44] Wei Chen and Dit-Yan Yeung. Defending Against TCP SYN Flooding Attacks Under Different Types of IP Spoofing. *Fifth International Conference on Networking (ICN'06)*, April 2006. 23, 33, 35, 53

[45] Maciej Korczyński, Lucjan Janowski, and Andrzej Duda. An Accurate Sampling Scheme for Detecting SYN Flooding Attacks and Portscans. *2011 IEEE International Conference on Communications (ICC'11)*, pages 1–5, June 2011. 23, 34

[46] Yipeng Wang, Zhibin Zhang, and Li Guo. Traffic Classification Beyond Application Level: Identifying Content Types From Network Traces. In *Proceedings of the 2011 ACM Symposium on Applied Computing (SAC'11)*, pages 540–541. ACM, 2011. 23

[47] Yulios Zavala, Jeferson Wilian de Godoy Stenico, and Lee Luan Ling. Internet Traffic Classification Using Multifractal Analysis Approach. In *Proceedings of the 15th Communications and Networking Simulation Symposium (CNS'12)*, pages 1–5. Society for Computer Simulation International, 2012. 23

[48] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *Proceedings of the 10th Conference on USENIX Security Symposium*, pages 25–25. USENIX Association, 2001. 23

[49] 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*, 36(5):5–16, October 2006. 23

[50] Guowu Xie, Marios Iliofotou, Ram Keralapura, Michalis Faloutsos, and Antonio Nucci. SubFlow: Towards Practical Flow-level Traffic Classification. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'12)*, pages 2541–2545. IEEE Press, 2012. 23

[51] Dario Bonfiglio, Marco Mellia, Michela Meo, and Dario Rossi. Detailed Analysis of Skype Traffic. *IEEE Transactions on Multimedia*, 11(1):117–127, January 2009. 23, 57, 77, 78

[52] Laurent Bernaille and Renata Teixeira. Early Recognition of Encrypted Applications. *Proceedings of the 8th International Conference on Passive and Active Network Measurement (PAM'07)*, 4427:165–175, 2007. 23, 77, 83, 87

[53] KC Claffy. *Internet Traffic Characterization*. PhD thesis, University of California at San Diego, 1994. 23

[54] Amir Globerson and Naftali Tishby. Sufficient Dimensionality Reduction. *The Journal of Machine Learning Research*, 3:1307–1331, March 2003. 24

[55] Isabelle Guyon and Andre Elisseeff. An introduction to Variable and Feature Selection. *The Journal of Machine Learning Research*, 3:1157–1182, March 2003. 24, 58

[56] Ha-Nam Nguyen and Syng-Yup Ohn. DRFE: Dynamic Recursive Feature Elimination for Gene Identification Based on Random Forest. In *Proceedings of the 13th International Conference on Neural Information Processing (ICONIP'06)*, pages 1–10, Berlin, Heidelberg, 2006. Springer-Verlag. 24

[57] Hua-Liang Wei and Stephen A. Billings. Feature Subset Selection and Ranking for Data Dimensionality Reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):162–166, January 2007. 24

[58] Marius Kloft, Ulf Brefeld, Patrick Duessel, Christian Gehl, and Pavel Laskov. Automatic Feature Selection for Anomaly Detection. In *Proceedings of the First ACM Workshop on AISec'08*, pages 71–76. ACM, 2008. 24

[59] G. E. P. Box. Non-Normality and Tests on Variances. *Biometrika*, 40:318–335, 1953. 24, 58, 68, 69

[60] Pedro Casas, Johan Mazel, and Philippe Owezarski. Knowledge-independent Traffic Monitoring: Unsupervised Detection of Network Attacks. *IEEE Network*, pages 13–21, 2012. 24, 25

[61] Snort: Network Intrusion Prevention and Detection System (IDS/IPS. http://www.snort.org/. 24, 111

[62] Dorothy E. Denning. An Intrusion Detection Model. In *Proceedings of the Seventh IEEE Symposium on Security and Privacy*, pages 119–131, May 1986. 25

[63] Augustin Soule, Kave Salamatian, and Nina Taft. Combining Filtering and Statistical Methods for Anomaly Detection. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC'05)*, pages 31–31. USENIX Association, 2005. 25

[64] Daniela Brauckhoff and Kave Salamatian. Applying PCA for Traffic Anomaly Detection: Problems and Solutions. *2009 IEEE International Conference on Computer Communications (INFOCOM'09)*, pages 2866–2870, April 2009. 25

[65] Gautam Thatte, Urbashi Mitra, and John Heidemann. Parametric Methods for Anomaly Detection in Aggregate Traffic. *IEEE/ACM Transactions on Networking*, 19(2):512–525, April 2011. 25

[66] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Computer Communication Review*, 35(4):217–228, August 2005. 25

[67] Kingsly Leung and Christopher Leckie. Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science (ACSC'05)*, pages 333–342. Australian Computer Society, Inc., 2005. 25

[68] Jiong Zhang. Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection. In *2006 IEEE International Conference on Communications (ICC'06)*, pages 2388–2393, June 2006. 25

[69] Francesco Gringoli, Luca Salgarelli, Maurizio Dusi, Niccolo Cascarano, Fulvio Risso, and KC Claffy. GT: Picking Up the Truth From the Ground for Internet Traffic. *ACM SIGCOMM Computer Communication Review*, 39(5):12–18, October 2009. 26

[70] Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. Quantifying the Accuracy of the Ground Truth Associated with Internet Traffic Traces. *Computer Networking*, 55(5):1158–1167, April 2011. 26

[71] BackTrack Linux - Penetration Testing Distribution. http://www.backtrack-linux.org. 26

[72] Craig Labovitz, Danny McPherson, Mike Hollyman, and Scott Iekel-Johnson. Internet Traffic Trends - A View from 67 ISPs. *North American Network Operators Meeting*, June 2008. 33

[73] Ramana Kompella, Sumeet Singh, and George Varghese. On Scalable Attack Detection in the Network. *IEEE/ACM Transactions on Networking*, 15(1):14–25, February 2007. 33, 35, 53

[74] Changhua Sun, Chengchen Hu, Yachao Zhou, Xin Xiao, and Bin Liu. A More Accurate Scheme to Detect SYN Flood Attacks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'09) Workshop*, pages 1–2. IEEE Press, April 2009. 33, 35, 53

[75] Packet Sampling (PSAMP) IETF Working Group Charter. http://www.ietf.org/html.charters/psamp-charter.html. 34, 36

[76] NetFlow Services Solutions Guide. http://www.cisco.com/. 34, 36

[77] Maciej Korczyński and Lucjan Janowski. Implementation of The Algorithm to Detect and Prevent Network Attacks Based on Rate Limiting Method. *Conference on Next Generation Services and Networks - the Technical Aspects, Application and Market*, November 2010. 34

[78] Gilles Berger-Sabbatel, Maciej Korczyński, and Andrzej Duda. Architecture of a Platform for Malware Analysis and Confinement. *3rd INDECT/IEEE International Conference on Multimedia Communications*, pages 1–7, May 2010. 34

[79] Karol Adamski, Maciej Korczyński, and Lucjan Janowski. Trace2Flow. *3rd NMRG Workshop on Netflow/IPFIX Usage in Network Management*, July 2010. 34

[80] Georgios Androulidakis, Vasilis Chatzigiannakis, Symeon Papavassiliou, Mary Grammatikou, and Vasilis Maglaris. Understanding and Evaluating the Impact of Sampling on Anomaly Detection Techniques. In *Proceedings of the IEEE Conference on Military Communications (MILCOM'06)*, pages 349–355. IEEE Press, October 2006. 36, 43, 53

[81] Jianning Mai, Ashwin Sridharan, Chen-Nee Chuah, Hui Zang, and Tao Ye. Impact of Packet Sampling on Portscan Detection. *IEEE Journal on Selected Areas in Communications*, 24(12):2285–2298, December 2006. 36

[82] Jianning Mai, Chen-Nee Chuah, Ashwin Sridharan, Tao Ye, and Hui Zang. Is Sampled Data Sufficient for Anomaly Detection? In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC'06)*. ACM, October 2006. 36

[83] TracesPlay. http://tracesplay.sourceforge.net/. 41

[84] Jieren Cheng, Jianping Yin, Yun Liu, Zhiping Cai, and Min Li. DDoS Attack Detection Algorithm Using IP Address Features. In *Frontiers in Algorithmics*, volume 5598/2009, pages 207–215. Springer Berlin / Heidelberg, June 2009. 42

[85] Shan-Qing Guo and Zhong-Hua Zhao. An Anomaly Intrusion Detection Model Based on Limited Labeled Instances. In *2008 International Symposium on Electronic Commerce and Security*, pages 283–287, August 2008. 42

[86] Bradley Efron and Rob Tibshirani. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. *Statistical Science*, 1(1):54–75, February 1986. 43

[87] Abraham Yaar, Adrian Perrig, and Dawn Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 130–143, May 2004. 53

[88] Martine Bellaiche and Jean-Charles Gregoire. SYN Flooding Attack Detection Based on Entropy Computing. In *IEEE Global Telecommunications Conference (GLOBECOM'09)*, pages 1–6. IEEE Press, November 2009. 53

[89] Kensuke Fukuda and Romain Fontugne. Estimating Speed of Scanning Activities with a Hough Transform. In *2010 IEEE International Conference on Communications (ICC'10)*, pages 1–5, Cape Town, May 2010. 53

[90] Ping Du and Akihiro Nakao. DDoS Defense Deployment with Network Egress and Ingress Filtering. In *2010 IEEE International Conference on Communications (ICC'10)*, pages 1–6, Cape Town, May 2010. 53

[91] Wesley M. Eddy. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987, August 2007. 53

[92] Salman A. Baset and Henning Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'06)*, pages 1–11. IEEE Press, April 2006. 57, 98

[93] Sean O'Neil. Skype's Biggest Secret Revealed. *2010 27th Chaos Communication Congress*, December 2010. 60, 83, 98, 111

[94] Wireshark Web Site. http://www.wireshark.org. 70

[95] SPID Web Site. http://sourceforge.net/projects/spid. 71

[96] Marcin Pietrzyk. *Methods and Algorithms for Network Traffic Classification.* PhD thesis, Telecom Paris Tech, 2011. 77, 87

[97] Charles V. Wright, Lucas Ballard, Fabian Monrose, and Gerald M. Masson. Language Identification of Encrypted VoIP Traffic: Alejandra y Roberto or Alice and Bob. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 43–54. USENIX Association, 2007. 77, 78

[98] Kenjiro Cho, Kensuke Fukuda, Hiroshi Esaki, and Akira Kato. Observing Slow Crustal Movement in Residential User Traffic. In *Proceedings of the 2008 ACM International Conference on Emerging Networking Experiments and Technologies(CoNEXT'08)*, pages 1–12. ACM, 2008. 83

[99] Haixin Duan and Xing Li. Identification of P2P Traffic Based on the Content Redistribution Characteristic. *International Symposium on Communications and Information Technologies*, pages 596 – 601, October 2007. 83, 87

[100] Tor Project: Anonymity Online. https://www.torproject.org/. 83, 111, 113

[101] I2P Anonymous Network. http://www.i2p2.de/. 83, 111

[102] Encrypting Bittorrent To Take Out Traffic Shapers. http://torrentfreak.com/, February 2006. 83, 111

[103] IMule - The Anonymous EMule. http://www.imule.i2p.tin0.de/. 83, 111

[104] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol, Version 1.2. "RFC 5246", August 2008. 83, 88, 99

[105] Nong Ye, Yebin Zhang, and C. M. Borror. Robustness of the Markov-Chain Model for Cyber-Attack Detection. *IEEE Transactions on Reliability*, pages 116–123, 2004. 93

[106] Wireshark Web Site. http://www.wireshark.org. 98

[107] John Aldrich. R.A. Fisher and the Making of Maximum Likelihood 1912-1922. *Statistical Science*, 12(3):162–176, August 1997. 99

[108] The Bro Network Security Monitor. http://www.bro-ids.org/. 111

[109] Marcin Pietrzyk, Lucjan Janowski, and Guillaume Urvoy-Keller. Toward Systematic Methods Comparison in Traffic Classification. In *2011 7th International Wireless Communications and Mobile Computing Conference (IWCMC'11)*, pages 1022–1027, 2011. 112

[110] Alessandro Finamore and Marco Mellia and Michela Meo and Maurizio M. Munafo and P. D. Torino and Dario Rossi. Experiences of Internet Traffic Monitoring with Tstat. *IEEE Network*, pages 8–14, 2011. 112

[111] Skype-New Target of the Worm Spreading via IM. http://blog.bkav.com/, May 2010. 113