Czech Technical University in Prague

Faculty of Electrical Engineering

Doctoral Thesis

August 2016 ***Karel Bartoš***

Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Computer Science

Network Traffic Representations for Adaptive Intrusion Detection

Doctoral Thesis

Karel Bartoš

Prague, August 2016

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of study: Information Science and Computer Engineering

Supervisor: Martin Rehák, Ph.D., Ingénieur ECP
Supervisor-Specialist: Ing. Tomáš Pevný, Ph.D.

Acknowledgments

This work would not exist without the help and support of many people. First of all, my deepest gratitude goes to my supervisor and friend Martin Rehák, who gave me the opportunity to be part of his amazing dream. Ten years ago, Martin brought me to the fascinating world of network security and helped me to make my first research steps. Without his continuous guidance, support, and bright ideas, this thesis would not have been possible.

I would like to thank my colleagues and friends, namely Martin Grill, Jan Jusko, and Jan Stiborek for their friendship and support throughout the years and for making our team so great and powerful. My thanks also go to Tomáš Pevný for giving me valuable feedbacks. A very special thank you to prof. Michal Pěchouček for giving me the opportunity to participate in extraordinary projects and for leading our team into success and recognition.

I am thankful to all my colleges from Cisco and Agent Technology Center for providing me the chance to work with exceptional people. Thanks also go to Michal Sofka and Vojtěch Franc for their numerous advice about my research, to our security expert Verónica Valeros for providing me with high-quality labeled data for many of my experiments, and to Vojtěch Krmíček for helping me with the sampling implementation.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was funded by the ITC-A of the U.S Army (contracts N62558-07-C-001, W911NF-08-1-0250, W911NF-10-1-0070 and W911NF-12-1-0028); ONR Global under the Department of the Navy (contract N62909-11-1-7036); Air Force Office of Scientific Research, Air Force Material Command, USAF (grant FA8655-10-1-3016); Czech Ministry of Education (grants ME10051 and MEB111008) and the Czech Ministry of Interior (grants VG2VS/189 and VG20122014079).

Above all I extend my profound gratefulness to my family and especially to my wife Simona who has always stood by my side with her moral support and inspiration. I would not have been able to complete this thesis without her unconditional love and encouragement.

Abstract

New and unseen polymorphic malware, zero-day attacks, or other types of advanced persistent threats are usually not detected by traditional security systems. This represents a challenge to the network security industry as the amount and variability of attacks has been increasing. In this thesis, we propose three key approaches, each dealing with this challenge at different levels of abstraction.

In order to cope with an increasing volume of network traffic, we propose the adaptive sampling method based on two concepts that mitigate the negative impact of sampling on the raw input data: (i) Features used by the analytic algorithms are extracted before the sampling and attached to the surviving flows. The surviving flows thus carry the representation of the original statistical distribution in these attached features. (ii) Adaptive sampling that deliberatively skews the distribution of the surviving data to over-represent the rare flows or flows with rare feature values. This preserves the variability of the data and is critical for the analysis of malicious traffic, such as the detection of stealthy, hidden threats. Our approach has been extensively validated on standard NetFlow data, as well as on HTTP proxy logs that approximate the use-case of enriched IPFIX for the network forensics.

Next, we propose a novel representation and classification system designed to detect both known as well as previously unseen security threats. The classifiers use statistical feature representation computed from the network traffic and learn to recognize malicious behavior. The representation is designed and optimized to be invariant to the most common changes of malware behaviors. This is achieved in part by a feature histogram constructed for each group of network connections (flows) and in part by a feature self-similarity matrix computed for each group. The parameters of the representation (histogram bins) are optimized and learned based on the training samples along with the classifiers. The proposed approach was deployed on large corporate networks, where it detected 2,090 new variants of malware with 90% precision.

Finally, we propose a distributed and self-organized mechanism for the collaboration of multiple heterogeneous detection systems. The mechanism is based on a game-theoretical approach that optimizes the behavior of each detection system with respect to other systems in highly dynamic environments. The game-theoretical model specializes the detection systems on specific types of malicious behaviors to collaboratively cover a wider range of attack classes. According to our experimental evaluation on the real network traffic, the proposed mechanism shows clear improvements caused by mutual specialization of individual detection systems.

All three approaches can be combined into a unified collaborative fusion system, analyzing the input network traffic at different levels of abstraction. The benefits of such combination were demonstrated in the final experiment, where we combined the proposed adaptive sampling with a collaborative mechanism for detection systems deployed in multiple networks.

Contents

Chapter 1

Introduction

As computer networks evolved into a highly dynamic and ubiquitous infrastructure, more and more companies and organizations deploy additional security systems to maintain their environment secure. Increasingly sophisticated threats require the development of new solutions and architectures capable of addressing a comprehensive set of vulnerabilities. To address these problems, researchers and engineers have designed various security mechanisms protecting the availability, confidentiality, and integrity [141] of critical systems and networks.

Besides the increasing variability of network threats, the volume of network traffic has also been steadily increasing over the last years [5]. At the same time, the delivery of critical services from cloud data centers has increased not only the volume of traffic, but also the complexity of transactions. It has also redefined the need for network monitoring and long-term storage of network and transaction logs. The increase in volume also brings computational problems for more sophisticated detection and classification algorithms [86], as they may easily become increasingly difficult to compute on the full traffic log. Storing large amounts of traffic monitoring data also complicates network forensics and increases data retention and investigation costs.

Intrusion detection systems (IDS) have become necessary mechanisms protecting large enterprise networks. They are typically combined with other existing network security devices (e.g. firewalls, authentication servers, and anti-virus programs) to enhance the overall network security capabilities. However, each of these systems operates individually, with no interactions with other security systems. New emerging class of cyber attacks called collaborative attacks [155] launched by multiple attackers are highly effective against stand-alone detection systems, so the next-generation security devices based on cooperative and collaborative defense are required.

With the increase in the number of deployed detection systems and with the ever-growing complexity of the network environment, security administrators are becoming overwhelmed with lots of false alerts and other unnecessary or redundant information, weakening the overall efficiency of security monitoring. An intelligent mechanism is required to extract relevant information from the network traffic into high-level alerts and combine the alerts across all IDS systems.

Alert correlation and knowledge fusion have become widely used in the cooperative intrusion detection systems as techniques capable of grouping alerts from multiple sources together. While these methods are able to remove alert redundancy, taking the next step and moving towards a collaborative adaptation of individual detection systems on the current or future network threats still represents a significant challenge in the research community.

We believe that intrusion detection systems, and all network security devices in general, should operate in a collaborative and uniform architecture, fusing data into information and knowledge to provide useful and tangible situational awareness to network security engineers.

In this thesis, we propose three key approaches, each processing the network traffic at different level of abstraction, namely an adaptive sampling that reduces large amounts of low-level data while preserving most of the malicious activity in the network, an invariant representation of network traffic suitable for classifying unseen malware variants at the medium level, and a collaborative model for high-level collaboration of heterogeneous detection systems. All three approaches can be combined into a unified collaborative fusion system that transforms low-level data into high-level knowledge and are typically combined with additional techniques in order to build a full-fledged IDS system [2, 123]. From our experimental evaluation, we will show significant improvements of the proposed approaches against the state-of-the-art methods.

1.1 Research Problems

This thesis addresses the following research problems:

RP1 - How to select limited amount of network traffic for analysis with minimal impact on system efficacy?

A lot of sophisticated detection algorithms suffer from high computational requirements making them impractical or even impossible to deploy in real network environments. When the computational complexity of such methods is artificially reduced to make them applicable, the ability to detect advanced threats is typically significantly decreased. An alternative and widely-used solution is to employ random sampling, which however could be devastating for any of the subsequent postprocessing. We propose a better alternative.

RP2 - How to represent network traffic to detect new and previously unseen threats?

New and unseen polymorphic malware, zero-day attacks, or other types of advanced persistent threats are usually not detected by signature-based security devices, firewalls, or anti-viruses. This represents a challenge to the network security industry as the amount and variability of incidents has been increasing. Consequently, this complicates the design of learning-based detection systems relying on features extracted from network data. The problem is caused by different and evolving joint distribution of observation (features) and labels in the training and testing data sets.

RP3 - How to create a detection system resilient to evasion?

Security systems are prime and highly valuable targets for the attackers. Various security threats have been used to infiltrate IDS systems or decrease their performance and detection capabilities [48]. Deployment of robust representations and methods would decrease the possibility of attacker's manipulation with the system.

RP4 - How to utilize more detection systems and build a collaborative-adaptive system?

Detection of more sophisticated collaborative attacks [155] requires cooperation of various next-generation intrusion detection systems. This problem is partially related to alert correlation, which is motivated by the problem not to overwhelm security engineer with large number of (false) simple alerts, as well as to provide more comprehensive overview of the network security state. However it concentrates mainly on attacks launched by a single attacker, so designing an applicable framework of monitoring and defensive mechanisms against multiple collaborative attackers still remains a great challenge.

1.2 Key Contributions

This thesis has four main contributions:

- **Adaptive sampling method [16, 18, 19]** - We propose a new adaptive sampling method. This method is suitable for network behavior analysis techniques or network forensics thanks to the efficient sampling strategy, which selects individual flows to minimize the information redundancy of the network traffic. Furthermore, we introduce the concept of late sampling, where the algorithm first computes feature statistics and then samples the input data. The precomputed statistics are unbiased and can be used for the sampling method and the subsequent processing with minimal impact on the computational complexity of the system. Extensive experimental evaluation is provided to verify the proposed approach w.r.t. anomaly detection, information loss, and network forensics. Special emphasis is given to the analysis of real proxy logs and the impact of sampling on current network security threats. The proposed adaptive sampling addresses *RP1*.
- **Bag invariant representation for classification of malicious traffic [21, 22, 23]** - We propose a supervised approach that is able to detect previously unseen types of malware categories from a limited amount of training samples. Unlike classifying each category separately, which limits the robustness, we propose an invariant training from malware samples of multiple categories. Instead of classifying network flows individually, we propose to group flows into bags, where each bag contains flows that are related to each other. To enforce the invariant properties of the representation, we propose to use a novel approach, where the features are derived from the self-similarity of flows within a bag. These features describe the dynamics of each bag and have many invariant properties that are useful when finding new malware variants and categories. The proposed bag representation describing malware dynamics further increases the level of abstraction of the underlying network traffic in the proposed fusion model. The proposed approach addresses *RP2* and *RP3*.
- **Method for optimizing the representation automatically from the input data [23]** - To optimize the parameters of the representation, we propose a novel method that combines the process of learning the representation with the process of learning the classifier. The resulting representation ensures easier separation of malicious and legitimate communication and at the same time controls the complexity of the classifier. We evaluated the proposed representation on real network traffic of multiple companies. Unlike most of the previously published work, we performed the evaluation on highly imbalanced datasets as they appear in practice (considering the number of malicious samples), with most of the traffic being legitimate, to show the potential of the approach in practice. The proposed method contributes to address *RP2* and *RP3*.
- **Models for distributed collaboration of multiple IDS systems [14, 15, 17, 20]** - We propose two collaboration models for heterogeneous detection systems. The first model is suitable for detection systems deployed in various parts of the same network, where we propose to specialize inner models of the detectors towards a superior efficacy on a subset of attacks. Specialized detection systems produce less false alerts and their mutual collaboration leads to better efficacy results. The second model is designed for the collaboration of detection systems deployed across multiple networks, where we propose to acquire a global intelligence from all collaborating systems. We assume that the detection systems are consist

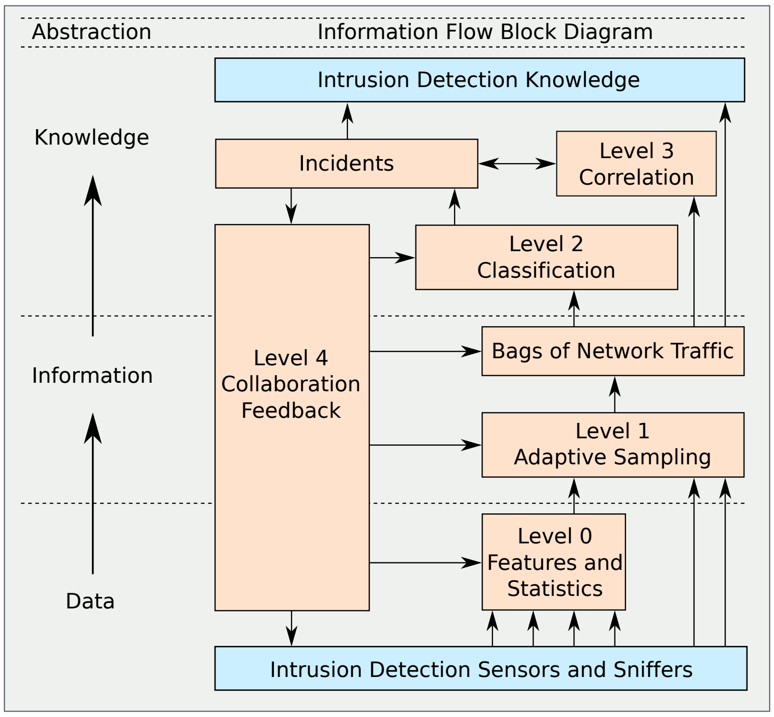

Abstraction
Information Flow Block Diagram
Intrusion Detection Knowledge
Knowledge
Incidents
Level 3 Correlation
Level 2 Classification
Level 4 Collaboration Feedback
Information
Bags of Network Traffic
Level 1 Adaptive Sampling
Level 0 Features and Statistics
Data
Intrusion Detection Sensors and Sniffers

Fig. 1.1: All individual contributions of the thesis can be combined together and cover all levels of abstraction of the proposed collaborative fusion model.

of sophisticated algorithms to be able to detect advanced threats. Such algorithms are typically computationally intensive and cannot process the whole input data in time. Therefore, we propose a collaborative-adaptive sampling component that is deployed together with each detection system. The component uses the global intelligence to significantly increase the sampling rate of malicious or suspicious data so the sophisticated detection methods are able to detect the attacks in time. This model prevents the attacks targeting multiple organizations and networks from being globally successful and reusable. The proposed collaboration models address *RP3* and *RP4*.

We propose a collaborative fusion model illustrated in Figure 1.1 that is composed of the individual contributions and covers the processing at all levels of abstraction: data, information, knowledge, and intelligence. The proposed architecture increases the level of abstraction of incoming network traffic as the data traverses through the system into a humanly understandable and actionable intelligence. With precomputed statistics at Level 0, the adaptive sampling at Level 1 reduces the amount of network traffic with significantly smaller loss of critical information, allowing more sophisticated classification algorithms at Level 2 to detect novel intrusions and threats. In the distributed or collaborative setting, Level 3 collects and correlates alerts collected from multiple systems and networks. The acquired global intelligence is valuable for a global situational security awareness. The feedback calculated at Level 4 is propagated to lower levels to dynamically adapt and reconfigure each collaborating system, which increases the overall efficacy and threat coverage.

Overall, all four contributions of the thesis enable an intelligent decomposition of IDS systems into a larger and distributed system. We propose to separate relatively fast operations (feature extraction, sampling, bag creation) from the more advanced transformations applied in later stages. Together with the proposed collaboration mechanism, this idea enables seamless dispersal of the IDS inspection on network devices and increases the overall security of the networks.

1.3 Outline of the Thesis

This thesis is structured as follows:

Chapter 2 reviews related work. Relevant research areas include extracting information and knowledge from the network traffic, with the emphasis on sampling and classification methods. Existing work on cooperation and collaboration among multiple detection systems is also included.

Chapter 3 presents the details about the proposed adaptive sampling method suitable for anomaly detectors, classifiers, or other types of postprocessing. The benefits of the proposed method are verified on various types of network data acquired from different networks.

Chapter 4 describes a novel approach for representing and classifying malicious behaviors from the network traffic. We discuss the invariant properties of the proposed representation and the corresponding application in classifying network threats as a means of extracting information and knowledge from the network data.

Chapter 5 proposes a fully-distributed model for collaboration of multiple detection systems deployed in one network. The model ensures mutual specialization of individual detectors to increase the number and the diversity of detected attacks.

Chapter 6 contains an experimental evaluation of the proposed collaborative-adaptive model designed for the collaboration of detection systems across multiple networks. The model continuously collects and correlates results provided by the detection systems into a global intelligence. The intelligence is used to automatically configure the collaborating systems, which prevents attacks from being globally successful and reusable.

Chapter 7 summarizes the contributions of this thesis and provides the list of related publications and patents.

Chapter 2
Related Work

This chapter is divided into three main sections, reflecting the decomposition of the problem into three parts according to the level of abstraction in the proposed model illustrated in Figure 1.1: sampling of network traffic, classification of network traffic, and collaboration of multiple detection systems. The first section summarizes existing sampling methods proposed to reduce large volumes of network traffic, highlighting their benefits and limitations. Next section describes recent research in the field of network traffic classification. Final section discusses three types of security architectures w.r.t. the collaboration possibilities of inner components. The last section concludes with the overview of relevant algorithms for distributed collaboration.

2.1 Sampling of Network Traffic

Various sampling methods has been proposed to tackle the problem of reducing the network traffic for further processing or analysis (such as quality of service provisioning, traffic profiling and control, fault detection, service level agreement verification (SLA), or intrusion detection [97]). Depending on their purpuse, these methods are applied on various types of network data (such as network packets, CISCO NetFlow [1], HTTP/HTTPS proxy logs, or other IPFIX format). In the following, we will summarize the existing approaches and emphasize their strengths and weaknesses.

2.1.1 Packet Sampling Methods

The most traditional and widely-used sampling is random sampling of network packets (denoted as *random packet sampling*). Each packet is sampled according to the sampling rate which is predefined in the beginning and is fixed for all incoming packets. This method has gained an increased popularity thanks to its simplicity and low computational complexity, as it is easy to implement it and deploy on routers in networks of arbitrary size. Even though random packet sampling is useful in keeping the traffic volumes under control, its simplicity negatively influences any reasoning built on the top of sampled data, such as anomaly-based network intrusion detection [66].

Several modifications and improvements of random packet sampling has been proposed to decrease the negative effect of information loss. Generally, most of these approaches adjusted the sampling method to deal with a so called *elephant and mice phenomenon*. This phenomenon says that a small percentage of network flows typically accounts for a large percentage of the total traffic. More specifically, $10 - 20\%$ flows are responsible for approx. 80% of the total packets [41]. This means that random packet sampling implicitly emphasizes a minority of large flows (with lots of packets) at the expense of the majority of smaller flows, which brings a significant bias for flow-based network traffic monitoring and analysis.

Sample and hold sampling [60] was proposed to identify elephant flows with lots of packets and then sample them with minimal packet loss. The method starts with sampling of each packet randomly. The method creates a new flow entry in the flow memory for every sampled packet belonging to a new and unseen flow. Once the flow entry is created, it is updated with every subsequent packet belonging to the flow and a corresponding counter is held in a hash table till the end of the measurement interval. It means that when a packet of a flow is sampled, all the subsequent packets of this flow will be sampled as well, which reduces the bias for longer flows with many packets at the expense of the rest of the flows.

Stratified packet sampling [41] was proposed to provide an unbiased estimation of flow sizes (in terms of number of packets and bytes). The method divides time into predetermined, non-overlapping intervals called strata (or blocks). For each block, they sample packets with the same probability (via random sampling). At the end of each block, flow statistics are estimated. Both methods [41, 60] are aimed for an accurate estimation of elephant flows, however majority of flows with fewer packets may not be selected at all, which could be devastating for finding smaller and mostly hidden connections of malicious communication. Non-linear adaptive sampling [76] was proposed to increase the number of small-sized flows in the sampled set. The first packet of each flow is always selected to the final set and then the sampling rate decreases as the number of sampled packets for a given flow increases. The sampling is suitable for packet-based passive measurement, as it significantly reduces the overall number of packets while providing an unbiased estimation of flow sizes. However, the number of flows in the sampled set remains constant.

A packet-sampling method leveraging results from anomaly detectors was proposed as progressive security-aware sampling [6]. The method relies on a collaboration of anomaly detectors located on multiple hop nodes in the network. At each node, the detectors evaluate incoming packets and include the decision into the header of each packet. The decision is then used at the next node to sample more anomalous traffic. While evaluating packets before sampling is beneficial for further analysis and monitoring, it does not allow deployment of advanced and computationally more expensive algorithms.

As more sophisticated detection algorithms have been proposed to fight against cybercrime, researchers started to compare the impact of packet and flow sampling on network traffic distributions and other features that are used in these algorithms. It has been proven [75] that second order statistics cannot be inferred when sampled with packet sampling, as opposed to flow sampling, some of these statistics can be retrieved. Moreover, packet sampling produces accurate estimates of byte and packet counts, but inaccurate estimates of flow counts [34], as volume metrics are less resilient to sampling than entropy-based summarizations [34]. The impact of packet sampling on anomaly detection metrics (volume and entropy) as well as on 3 algorithms for scan detection (TRW [85], TAPS [140], Xu [154]) has been extensively studied in [34] and [103]. Both studies, together with other published work [69, 101] came to the same conclusion that flow sampling is superior to packet sampling when it comes to anomaly detection. Another detailed study [37] confirmed the negative effect of packet sampling on the classification of network traffic.

The main reason for packet sampling being inferior is threefold: (a) packet sampling causes a fundamental bias called flow thinning (smaller flow size distribution) that causes high false positive and false negative rates [103], (b) packet sampling brings traffic feature set distortion towards either uniformity or biased towards large-sized flows, which significantly reduces the recall, and (c) trying to minimize loss of information at packet level is not efficient for flow-based approaches, as there is only small reduction of data at the flow level [69]. The impact of sampling on portscan detection [113, 112] revealed that flow sampling is not always better than packet sampling when equal amount of packets is sampled in both cases. Since there are no guaranties of how many flows will be sampled, the flow reduction effect is again questionable.

2.1.2 Flow Sampling Methods

The above mentioned results motivated the researchers to put the main focus to flow-based sampling. Smart sampling introduced by [55] is a flow-level analogy to sample-and-hold sampling. Large flows (with more bytes) are selected with higher probability then small flows. More specifically, a flow record representing a flow of x bytes is sampled with probability $p_z(x) = \min\{1, x/z\}$. Flows of size greater than manually threshold z are always selected, while smaller flows are sampled with probability proportional to their size. Smart sampling can be useful for estimating flow length distribution [56] (e.g. for network usage monitoring), but it is not suitable for general anomaly detection methods [101]. A combination of packet and flow-based sampling [156] was designed to estimate network flow characteristics (namely packet lengths and byte sizes) using expectation-maximization (EM) algorithms, however the convergence of the proposed algorithms is rather slow.

Selective sampling [9] was proposed as an flow-based alternative to non-linear adaptive [76] and smart sampling. The method is based on common observation that small flows (with small number of packets) are usually the source of malicious traffic (e.g. DDOS or worm propagation). Therefore selective sampling samples small flows with higher probability and flows that are larger in size than manually predefined threshold are sampled with probability inversely proportional to their size:

$$p(x) = \begin{cases} c & x \leq z \\ z/(n \cdot x) & x > z, \end{cases} \tag{2.1}$$

The authors of selective sampling introduced a new trend in sampling, where the methods are specifically designed for anomaly detection algorithms [8]. This approach deliberatively looses some information in order to magnify specific types of anomalies, which did not have to be visible even from the unsampled set. This way, a carefully-designed sampling can actually improve the effectiveness of some detection algorithms. Even though the selective sampling was designed only for an entropy based anomaly detector, the idea of customizing sampling methods for anomaly detection has proven strong in the following years and our work builds on the top of it.

Sampling Method	*Sampling Level*	*Volume Preservation*	*Distrib. Preservation*	*Anomaly Detection*
Random Packet Sampling	P	○	○	×
Sample and Hold Sampling [60]	P	●	×	×
Adaptive Packet Sampling [41]	P	●	×	×
Adaptive Non-Linear Sampling [76]	P	●	○	×
Progressive Sampling [7]	P	×	×	●
Random Flow Sampling	F	●	●	○
Smart Sampling [55]	F	●	×	×
Selective Sampling [9]	F	×	×	●
Proposed Adaptive Late Sampling	F	○	○	●

Table 2.1: The overview of selected sampling methods, levels (P - packet, F - flow) and their suitability for anomaly detection and for preserving traffic volumes and distributions. Legend: × - not suitable, ○ - partially suitable, ● - suitable.

The use of sampled data for more advanced analysis, such as Network Behavior Analysis (NBA) or intrusion detection [139], is problematic [102], as any sampling necessarily impacts the effectiveness of the anomaly detection and data analysis algorithms. These algorithms are based on pattern recognition and statistical traffic analysis, and the distortion of traffic features can significantly increase the error rate of these underlying methods by breaking their assumptions about traffic characteristics [37]. The loss of information introduced by sampling methods also negatively impacts any potential postmortem or forensics investigation. Moreover, the impact of sampling on new or extended data formats (e.g. HTTP proxy logs, NetFlow [1], or IPFIX protocol extended with new HTTP fields [147]) has not been studied yet.

2.2 Network Traffic Classification

Network perimeter can be secured by a large variety of network security devices and mechanisms, such as host-based or network-based Intrusion Detection Systems (IDS) [129]. We briefly review both systems, focusing our discussion on network-based IDS, which are the most relevant to the presented work.

2.2.1 Host-based IDS

Host-based IDS systems analyze malicious code and processes and system calls related to OS information. Traditional and widely-used anti-virus software or spyware scanners can be easily evaded by simple transformations of malware code. To address this weakness, methods of static analysis [108], [135] were proposed. Static analysis, relying on semantic signatures, concentrates on pure investigation of code snippets without actually executing them. These methods are more resilient to changes in malware codes, however they can be easily evaded by obfuscation techniques. Methods of dynamic analysis [107], [126], [159] were proposed to deal with the weaknesses of static analysis, focusing on obtaining reliable information on execution of malicious programs. The downside of the dynamic analysis is the necessity to run the codes in a restricted environment which may influence malware behavior or difficulty of the analysis and tracing the problem back to the exact code location. Recently, a combination of static and dynamic analysis was used to analyze malicious browser extensions [81].

2.2.2 Network-based IDS

Network-based IDS systems are typically deployed on the key points of the network infrastructure and monitor incoming and outgoing network traffic by using static signature matching [70] or dynamic anomaly detection methods [39]. Signature-based IDS systems evaluate each network connection according to the predefined malware signatures regardless of the context. They are capable of detecting well-known attacks, but with limited amount of detected novel intrusions. On the other hand, anomaly-based IDS systems are designed to detect wide range of network anomalies including yet undiscovered attacks, but at the expense of higher false alarm rates [39].

2.2.2.1 Classification of Network Traffic Services

Network-based approaches are designed to detect malicious communication by processing network packets or logs. An overview of the existing state-of-the-art approaches is shown in Table 2.2. The focus has been on the traffic classification from packet traces [27], [105], [136], [150], as this source provides detailed information about the underlying network communication. Due to the still increasing demands for larger bandwidth, analyzing individual packets is becoming intractable on high-speed network links. Moreover, some environments with highly confidential data transfers such as banks or government organizations do not allow deployment of packet inspection devices due to the legal or privacy reasons. The alternative approach is the classification based on network traffic logs, e.g. NetFlow [1], DNS records, or proxy logs. The logs are extracted at the transport layer and contain information only from packet headers.

Methods introduced in [58] and [88] apply features extracted from NetFlow data to classify network traffic into general classes, such as P2P, IMAP, FTP, POP3, DNS, IRC, etc. A comparison and evaluation of these approaches can be found in a comprehensive survey [91]. The work in [11] provides the description of traffic patterns for selected applications and proposes a classification technique to identify video, audio and file transfer traffic tunneled over HTTP.

A combination of host-based statistics with SNORT rules to detect botnets was introduced in [74]. The authors showed that it is possible to detect malicious traffic using statistical features computed from NetFlow data, which motivated further research in this field. An alternative approach for classification of botnets from NetFlow features was proposed in [29]. The authors of [120] have used normalized NetFlow features to cluster flow-based samples of network traffic into four predefined categories. As opposed to our approach, the normalization was performed to be able to compare individual features with each other. In our approach, we extended this idea and use normalization to be able to compare various malware categories. While all these approaches represent relevant state-of-the-art, network threats evolve so rapidly that these methods are becoming less effective due to the choice of features and the way they are used.

Approach	Type	Method	Features	Target class	Testing Data			Malicious	Mal:All
					Type	Year	All samples	samples	ratio
Wang [150]	U	anomaly detection	packet payload	worms, exploits	packets	2003	531,117	N/A	N/A
Kruegel [93]	U	anomaly detection	URL query parameters	web malware	proxy logs	2003	1,212,197	11	1:100k
Gu [74]	U	clustering	host statistics+SNORT	botnet	NetFlow	2007	100,000k	5,842k	1:17
Bilge [29]	S	random forest	flow size, time	botnets	NefFlow	2011	78,000,000	36	1:2.2M
Antonakakis [10]	S	multiple	NXDomains	dga malware	DNS data	2011	360,700	8008	1:45
Bailey [12]	S	hierarch. clustering	state changes	malware	executions	2007	4,591	4,591	1:1
Kapravelos [87]	S	similarity of trees	abstract syntax tree	web malware	JavaScript	2012	20,918,798	186,032	1:112
Choi [42]	S	SVM + RA*k*EL	URL lexical, host, dns	malicious flows	proxy logs	2009	72,000	32,000	1:2
Zhao [161]	S	active learning	URL lexical + host	malicious flows	proxy logs	2009	1,000,000	10,000	1:100
Huang [77]	S	SVM	URL lexical	phishing	proxy logs	2011	12,193	10,094	1:1
Ma [100]	S	multiple	URL lexical + host	malicious flows	proxy logs	2011	2,000,000	6,000	1:333
Invernizzi [79]	U	graph clustering	proxy log fields	mw downloads	proxy logs	2012	1,219	324	1:4
Soska [137]	S	random forests	content of web pages	infected websites	web pages	2014	386,018	49,347	1:8
Nelms [110]	S	heuristics	web paths	mw downloads	proxy logs	2014	N/A	150	N/A
Our approach	S	learned repr.+SVM	learned bag dynamics	malicious flows	proxy logs	2015	15,379,466	43,380	1:355

Table 2.2: Overview of the existing state-of-the-art approaches focusing on classification of malicious traffic (U = unsupervised, S = supervised). In contrast to the existing work, our approach proposes novel and optimized representation of bags, describing the dynamics of each legitimate or malicious sample. The approach is evaluated on latest real datasets with a realistic ratio of malicious and background flows (proxy log records).

2.2.2.2 Classification of Malware over HTTP(S)

One of the largest changes in the network security landscape is the fact that HTTP(S) traffic is being used not only for web browsing, but also for other types of services and applications (TOR, multimedia streaming, remote desktop) including lots of malicious attacks. According to recent analysis [79], majority of malware samples communicate via HTTP. This change has drawn more attention to classifying malware from web traffic. In [93], the authors proposed an anomaly detection system composed of several techniques to detect attacks against web servers. They divide URIs into groups, where each group contains URIs with the same resource path. URIs without a query string or with return code outside of interval [200, 300] are considered as irrelevant. The system showed the ability to detect unseen malware samples and the recall will be compared with our proposed approach in Section 4.6. In [137], the authors introduced a method for predicting compromised websites using features extracted from page content and Alexa Web Information Service.

Having sufficient amount of labeled malware samples at disposal, numerous approaches proposed supervised learning methods to achieve better efficacy. Clasifying DGA malware from DNS records based on connections to non-existent domains (NXDomains) was proposed in [10]. Even though several other data sources were used to detect malware (such as malware executions [12] or JavaScript analysis [87]), the most relevant work to our approach uses proxy logs [42], [77], [100], [161], [110].

The clustering of malware signatures was proposed in [115]. DGA-based malware was analyzed in [10], while the user agent anomalies were studied in [90]. The authors of [65] performed a detail analysis of botnet behavior including HTTP channels. This paper focuses on HTTP traffic, summarizes the previous malware analysis, and include other types of malicious behaviors.

In all these methods, proxy log features are extracted from real legitimate and malicious samples to train a data-driven classifier, which is used to find new malicious samples from the testing set. There are five core differences between these approaches and our approach: (1) we do not classify individual flows (in our case proxy log records), but sets of related flows called bags, (2) we propose a novel representation based on features describing the dynamics of each bag, (3) the features are computed from the bags and are invariant against various changes an attacker could implement to evade detection, (4) parameters of the proposed representation are learned automatically from the input data to maximize the detection performance, (5) the proposed classification system was deployed on corporate networks and evaluated on imbalanced datasets (see Table 2.2) as they appear in practice to show the expected efficacy on these networks.

2.3 Collaborative Intrusion Detection Systems

Nowadays, Intrusion Detection Systems [129] (IDS) play a crucial role in protecting computer networks against various types of attacks and other malicious behavior. As already mentioned in the previous section, every IDS can be categorized into two groups: *host-based* or *network-based*. Host-based IDSs are deployed on each host individually, where they analyze processes and system calls related to OS information. Network-based IDSs usually cover larger segment of network infrastructure (subnet) and monitor incoming and outgoing network traffic.

Moreover, IDS systems can be categorized into *anomaly-based* or *signature-based* IDS according to the detection principles the system is based on. Anomaly-based IDS is able to detect novel and zero-day attacks by employing various types of anomaly detection methods [39]. These methods usually create a model of normal (legitimate) behavior and continually recognize deviations from that behavior, which are marked as network anomalies. The downside of this approach includes mainly higher false positive rate (legitimate behavior marked as malicious). Signature-based IDS [128] performs signature matching of observed network traffic with predefined patterns of intrusive behavior. These systems are less error prone, however their database needs to be updated each time a new intrusion is observed to be able to detect it.

2.3.0.1 Taxonomy

Before we will discuss the CIDS in more detail, it should be noted that the term Collaborative IDS is sometimes used synonymously with the term Cooperative IDS. We suggest to respect the differences of these two terms as described in [36], however for the sake of simplicity, we will denote both systems as CIDS.

Cooperative Intrusion Detection System is a distributed system, where participants are able to exchange information related to the intrusion detection. Each individual entity operates on its own and provides the outcome to the rest of the entities, which may create additional benefits, but not completely new opportunities.

Collaborative Intrusion Detection System is a dynamic, distributed system where participants interact between each other to enable adaptation on their environment and their tasks, or making teams and other organizational structures. In such configurations they are able to solve problems that would not be solvable when working separately.

Event represents a symbolic description of a particular network activity (service, behavior). It may corresponds to both legitimate and malicious activity.

Alert is defined as a symbolic description of a particular network activity (service, behavior) reported by IDS, which is considered as malicious by the system.

In the following, we will describe three above-mentioned architecture designs of cooperative and collaborative intrusion detection systems (denoted as CIDS): *centralized*, *hierarchical*, and *fully distributed*. We will point out the benefits and limitations of each architecture and describe selected contributions, proposed frameworks, and techniques in more detail.

2.3.1 IDS Architectures

The first IDS prototypes were designed in the 1980s to protect security assets by using local anomaly detection and expert systems [53]. Later in the 1990s, first distributed intrusion detection systems based on the centralized approach [134, 52] were proposed. These systems present centralized view of distributed architecture by aggregating information from multiple sources, which goes hand to hand with scalability problems and a single point of failure security risks. Hierarchical and multi-layered approaches [119] introduced in late 1990s partially solved the scalability, while additional challenges dealing with reducing false alerts were discussed and analyzed. Since 2000, researchers have proposed several fully distributed IDS architectures [49, 157, 158]

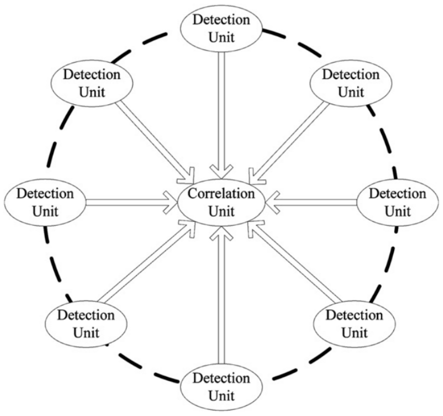


Fig. 2.1: Centralized CIDS architecture [162].

targeting scalability and a single point of failure issues. However with the strong increase of network traffic, researchers concentrated on targeting new challenges, e.g. computational and communication complexity or reducing extensive amount of (false) alerts.

Most recently, Cooperative and Collaborative Intrusion Detection Systems (CIDS) have been discussed in the context of novel and more sophisticated attack scenarios across multiple enterprise networks. Data privacy and trust between participating parties should be resolved, as well as evasion strategies against well-informed attacker. The main advantages of collaboration among multiple heterogeneous intrusion detection systems include [36]:

- **scalability and robustness** - System is able to operate efficiently on networks (or on groups of networks) regardless of its size in number of hosts as well as in number of incoming and outgoing network traffic.
- **availability** - System is operational even if some parts are disabled by the attacker (no single point of failure is present), so the system is able to compensate lack of central components.
- **teamwork** - Division and sharing of tasks can improve the overall performance and the effectiveness, resulting in coordinated decisions that can compensate potential individual shortcomings.
- **complex overview** - Having the overall network security at disposal enables the awareness of distributed cooperative/collaborative attacks.

2.3.1.1 Centralized CIDS

Initial CIDS architecture concepts were designed as centralized solutions, in which alerts locally collected on multiple detectors are sent to single central correlation unit, which performs the fusion process as illustrated in Fig. 2.1. Although this architecture is easy to deploy, it may become a single point of failure representing high security risk. When disabling the central node, the correlation process is deactivated, which could compromise the whole network security. The central correlation unit also has to deal with large amounts of alerts, so care must be taken when dealing with centralized architectures. On the other hand, the total awareness of the whole network state may produce best detection efficiency.

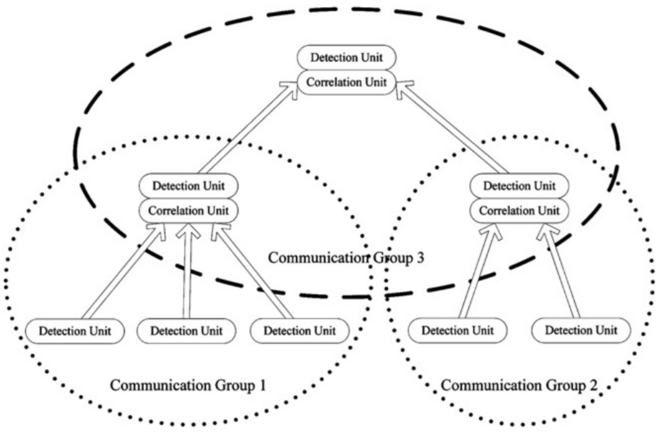


Fig. 2.2: Hierarchical CIDS architecture [162].

The NSTAT system proposed in [89] is an example of centralized CIDS. It uses client/server model consisting of several hosts as a source of multiple audit trails, and a central server machine making the final assessments about possible network security threats. The audit trails from monitored clients are collected into a single chronological data stream, which is analyzed in the central unit by using state transition analysis in real time. NSTAT focuses only on signature actions of penetration attacks, which limits the detection capabilities. Therefore the authors suggest to integrate NSTAT system with other CIDSs to increase the overall detection potential.

2.3.1.2 Hierarchical CIDS

More advanced hierarchical CIDS architecture is partitioned into individual groups according to geographical location, types of services running within the network, anticipated types of attacks etc. Each group includes a correlation node, which handles all alerts generated by all detection mechanisms within the group and passes the results to the correlation node on higher hierarchical level for further analysis. An example of this approach is illustrated in Figure 2.2. This type of architecture reduces the danger of a single point of failure, however it does not introduce fully scalable design. Furthermore, the correlation nodes process alerts within their hierarchical structure and do not posses with the whole network awareness, which can weaken the effectiveness of the correlation results. Last but not least, failure of central analysis component located at the very top of the architecture could result in the total destruction of the system.

In [119], the authors introduce a general framework of hierarchical CIDS called EMERALD for large enterprise networks. The framework consists of three layers: *service layer*, *domain-wide layer*, and *enterprise-wide layer*. Service layer contains dynamically deployable *service monitors*, which perform network traffic surveillance and may interact with the environment by providing localized analysis to other monitors. These monitors are independently tunable by using several parameters (e.g. type of detection method, subscription list for sending the results etc.) and are able to provide both signature analysis and statistical profiling. The second layer called domain-wide layer operates with *domain monitors* that aggregate and correlate results received from the service monitors and report domain threats to the network administrator. Finally, enterprise-wide layer analyzes attacks across the whole network infrastructure, providing a complex view

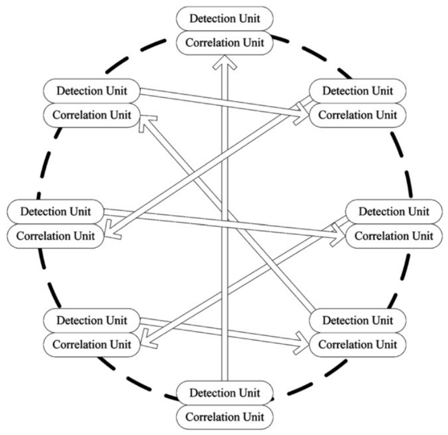


Fig. 2.3: Fully distributed CIDS architecture [162].

on the overall network state. The authors do not discuss any communication issues, however the proposed framework represents an interesting contribution to the research community, providing foundations to other CIDS architectures.

Servin and Kudenko [130] proposed hierarchical CIDS to detect DDoS attacks. The system is composed of *sensor agents* that monitor local state information and send that information in form of communication signals to *central agents* located in the upper part of the hierarchy. Each central agent processes the signal received from multiple sensor agents and learns by using reinforcement learning, where an action signal should be generated and passed through to other central agent in order to gain a positive reward. Finally the agent on the top of the hierarchy uses reinforcement learning to determine whether or not to trigger an alert to the network administrator. The goal of this learning process is that every agent would know in each state the action to execute to obtain a positive reward. The downside of this approach is that even though the architecture is hierarchical, it exhibits a single point of failure, as compromising of a single central agent on the top will incapacitate the whole alert generation procedure.

Mutual dependences among various computer networks are discussed by Li et al. [96]. The authors built hierarchical CIDS by using three types of detectors: local, regional, and global. *Local detectors* reside on end hosts and their task is to detect potential intrusions and report them to regional level, where *regional detectors* discovers regionally visible attacks by using discrete-time Hidden Markov Models, assuming mutual dependence among end hosts within the region. A region is a group of hosts sharing similar attributes, for example network proximity, local host properties (operating system types), or policy constraints. Finally, *global detectors* collect results from regions that are mutually independent, so sequential hypothesis testing technique can be applied to reveal more complex attack scenarios.

2.3.1.3 Fully Distributed CIDS

Fully distributed architecture generalizes the hierarchical concept into peer-to-peer relation among the individual parts of the system. Graphical illustration of this architecture is depicted in Fig. 2.3.

Yegneswaran [158] designed a distributed CIDS called DOMINO and demonstrated the utility of sharing information between multiple IDS systems in a cooperative infrastructure. They used information-theoretic approach, more specifically Kullback-Leibler [67] distance metric to quantify additional information gained by adding new nodes to the system. The results show that even a small number of nodes can significantly enhance the overall detection capabilities of the system.

Locasto [98] addresses the problem of data privacy in a fully distributed CIDS by using Bloom filters [31]. Bloom filters are compact (reduction in data size), resilient (no false negatives), and secure (one-way data encoding) data structure that satisfies the privacy needs for participating on collaborative defense across various enterprise networks. However some false positives may occur depending on the number of elements.

Host-based distribute CIDS proposed by Dash [49] uses *local* and *global detectors* to detect slow network intrusions. Each local detector resides on one end-host machine, uses a binary classifier to analyze both incoming and outgoing traffic and sends the output to randomly selected global detector. Global detectors draw conclusions about the received signals by using simple aggregation techniques (e.g. summing the number of positive counts) or Bayesian networks. The authors introduced interesting approach to boost the effectiveness of numerous week detectors, however these detectors are supposed to be host-based and binary. We believe such limitations may cause the deployment of the system on large enterprise networks impractical. Furthermore the authors do not discuss the communication issues and load balancing between communicating parties.

Dayong [157] proposed to employ multi-agent approach in P2P distributed CIDS. They suggest to deploy on each host two types of agents handling the detection process and communication. Another two types of mobile agents are able to travel from one host to another to collect information regarding cooperative attack scenarios.

2.3.1.4 CIDS Summary

According to the proposed approaches and techniques, it is possible to identify five key aspects of general Collaborative Intrusion Detection Framework [36]: *Communication Scheme* (type of communication used), *Organizational Structure* (topology of the whole system architecture), *Group Formation* (whether the system allows creation of smaller teams and coalitions), *Information Sharing and Interoperability* (whether the data, information, and/or knowledge is shared), and *System Security* (collaborative trust management, adversary evasion countermeasures). However, the key criterium with respect to alert fusion is the system architecture. An overview of three main types of CIDS architectures together with their major benefits and limitations is presented in Tab. 2.3.

2.3.2 Cooperative Multi-Agent Learning

In recent years, several applications of various machine learning techniques have been studied to enhance the detection performance of intrusion detection systems. However, most of the research aim has been concentrated on either adaptation methods for a single IDS [121, 125] or

Architecture	Examples	Advantages	Disadvantages
Centralized	[89]	Efficient in small environments	Single point of failure; Poor scalability
Hierarchical	[96, 119, 130]	Better scalability	Single point of failure at the top and poor correlation effectiveness at the bottom of the hierarchy
Distributed	[49, 157, 98, 158]	No single point of failure; Fully scalable	Higher communication load; Uncertain detection accuracy; Simpler existing alert correlation methods

Table 2.3: An overview of three main types of CIDS architectures with corresponding benefits and limitations.

data/information/knowledge fusion methods of multiple CIDS [116, 146, 149] to reduce false alerts and increase the overall detection awareness. Representative selection of approaches proposed in the field of alert correlation and fusion is described in the previous section. In this section, we will concentrate on the machine learning approaches (more specifically on cooperative multi-agent learning techniques), allowing the system to distributively learn and adapt, which is highly desirable in dynamic network environments.

The field of cooperative multi-agent learning has been extensively studied by the research community. The proposed approaches can be categorized by using several criteria. According to the feedback the critic/environment provides to the learner, we distinguish *unsupervised*, *supervised*, and *reward-based learning* [111]. In supervised leaning techniques, a critic provides correct solutions to learners, in contrast with unsupervised learning, where no feedback is provided at all. In reward-based learning, each learner receives some reward based on the behavior of the learner. Furthermore, several hybrid techniques have been proposed, e.g. *semi-supervised learning* methods usually take advantage of a small set of correct output data to enhance the learning process on larger set of unknown data [33].

As mentioned above, supervised learning methods provide most valuable feedback, allowing to learn from the ground truth. However, applying supervised machine learning methods in network environment is not practical (and mostly hardly achievable) due to extensive cost of obtaining labeled data. Moreover, in such a dynamic environment as computer networks, the ground truth could change rapidly making the labeled data unsatisfactory or obsolete. Despite of these difficulties, there has been few attempts to create larger sets of labeled data [64]. However most of them are from controlled or artificially created environments and were proven to be not representative [4].

Semi-supervised techniques usually require small amount of labeled data that are used to improve performance on larger set of unlabeled data. Typical example of semi-supervised learning technique is *multi-view approach*, where the data attributes are split into various attribute subsets (views). On the other side, in *consensus training*, multiple classifiers with different views are combined to form a high-level ensemble classifier.

Mao et al. [104] proposed multi-view *co-training method* [33] to leverage unlabeled data to enhance overall detection performance. They used data from two sources (network and host) and apply modified co-training algorithm to train classifiers. Data instances with high uncertain classification are labeled by an expert and returned back to the system. Chiu et al. [40] proposed

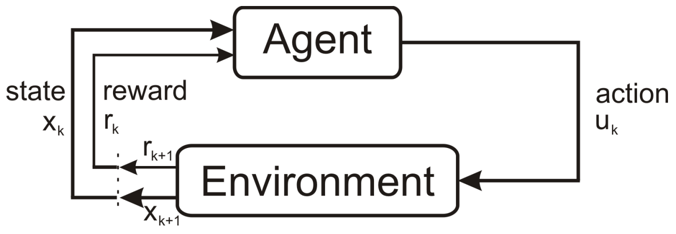


Fig. 2.4: Interaction of a single agent with the environment in reinforcement learning [144].

algorithm called *two-teachers-one-student* (a combination of multi-view and consensus training technique) to reduce the number of reported false alerts.

Event though semi-supervised techniques significantly decreased the amount of required labeled data, they still need some representative labeled samples in order to work correctly, and what is more, the performance of the algorithms strictly depends on labeled samples. However, selecting representative labeled sample set could be problematic and even small deviations may introduce considerable errors.

Collaborative learning by knowledge exchange was studied by Fisch et al. [63]. The authors proposed distributed collaboration mechanism based on sharing static patterns of novel attacks between probabilistic classifiers. Their off-line learning technique consists of two stages: *rule premises training* in an unsupervised fashion (variational Bayesian inference) and supervised *class labeling* for training of rule conclusions. This technique involves numerous interactions with security administrator, which may be impractical in real network environments.

In real-time or online network intrusion detection, collaborative intrusion detection devices do not receive any correct answers about the actions they take. However, it is possible to provide some feedback describing reaction of the unknown environment on selected action. Thus detection systems have to perform some actions first in order to learn the experience which action brings best results in which state. For this reason, we believe that applying reinforcement learning in CIDS can dramatically improve overall performance of the system.

Thus in the following, we will focus mainly on reward-based learning methods (more specifically on reinforcement learning methods) and their applications in network security domain. We will introduce current related work in IDS/CIDS learning followed by introduction to *reinforcement* and *team learning* techniques. Beside reinforcement learning, there has been proposed other types of reward-based approaches, e.g. evolutionary computation, simulated annealing, or stochastic hill climbing. Unlike reinforcement learning, these methods learn behaviors of the agents directly without any value function and we will not describe them further.

2.3.3 Cooperative Reinforcement Learning

Reinforcement learning (RL) has been widely studied as a popular and effective technique for cooperative multi-agent learning in domains, where feedback is provided after a sequence of actions performed in the environment [144], typically expressed as penalties or rewards. RL methods are based on trial and error interaction with the environment, where predefined formulas are updated to generate expected utility values. These values are used further in the exploration and exploitation process of the state space. A single-agent RL interaction is illustrated in Fig. 2.4.

Generally, the proposed approaches can be divided into *searched-based* algorithms, *model-based*, and *model-free* methods. An example of searched-based technique is evolutionary computation, where an evolutionary algorithm creates initial population of randomly generated individuals, and uses selection, breeding, and mutation to produce new individuals. Model-based methods [28] typically stochastically sample the environment to build a model and then approximate functions to generalize beyond experienced states. They require less exploration of the environment, but do not scale well to larger problems. On the other side, model-free methods [143, 151] do not require any model and approximate the optimal policy only by interactions with the environment. However, most of the proposed techniques have been derived from a model-free algorithm called *Q-learning* [151].

We believe that creating a model of network traffic environment is infeasible due to its highly dynamics, and searched-based techniques are impractical due to high number of dynamically changing states and variables. On the other hand, model-free methods provide a mechanism to enable multi-agent learning without these restrictions, but the dynamics of the system may influence their convergence properties. However such influence may be desirable in network security environments, where ensures unpredictability of the system. That is why we will focus on model-free techniques (and especially on *Q-learning* [151] algorithms) more in detail.

Various challenges have been identified, including:

- **co-learning** - In order to coordinate the overall behavior of the system, each agent should observe actions performed by the rest of the team and infer relevant knowledge into individual learning process.
- **non-stationarity** - Most single-agent RL algorithms assume stationary environments for their convergence properties. Making the environment non-stationary may violate such properties.
- **scalability** - The proposed algorithms are often designed for smaller sets of agents, which makes their deployment into real world problems impractical.
- **exploration and exploitation** - How to balance trade-off between exploration and exploitation represents traditional RL challenge. An agent that collects information from the environment should perform some exploration to avoid suboptimal solutions. On the other hand, the agent should not spent too much time in areas with low rewards.

In the following, we will present general definitions and ideas of reinforcement learning approaches.

2.3.3.1 Single Learner

In single-agent reinforcement learning, an agent or group of agents uses common resources to gain common knowledge of the environment. When more agents are involved in the learning process with a single learning knowledge, we call this approach as *team learning* [111]. The environment can be described by using *Markov decision process* (MDP), which is defined as a tuple (X, U, f, ρ), where X denotes the finite set of *environment states*, U denotes the finite set of *agent actions*, $f : X \times U \times X \to [0, 1]$ is the *state transition probability function*, and $\rho : X \times U \times X \to R$ is the *reward function*. An agent performs action $u_k \in U$ to change the state of the environment from $x_k \in X$ to $x_{k+1} \in X$ according to the state transition function $f(x_k, u_k, x_{k+1}) \in [0, 1]$ and receives corresponding reward $r_{k+1} = \rho(x_k, u_k, x_{k+1}) \in R$.

The values of both functions depend only on two consequent states, so the long-term effects of selecting an action is unknown. The behavior of each agent is driven by its policy $h : X \times U \to [0, 1]$, which defines which action should be taken in which state. The goal in each step k is to maximize the expected discounted return over the probabilistic state transitions

$$R_k = \sum_{j=0}^{\infty} E(\gamma^j r_{k+j+1}).$$

Value R_k represents total reward accumulated in the long run, which can be bounded by using the discount factor $\gamma^j \in [0, 1)$.

In other words, the goal of each agent is to maximize its overall long-term performance with only one-step performance information. To achieve this goal, we usually define the *action-value function* (Q-function) $Q^h : X \times U \to R$, which expresses the expected return of a state-action pair (x, u) given the policy h:

$$Q^h(x, u) = E\{\sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \mid x_k = x, u_k = u, h\}.$$

Furthermore, the optimal Q-function is defined as $Q^*(x, u) = max_h Q^h(x, u)$, which satisfies the Bellman optimality equation:

$$Q^*(x, u) = \sum_{x' \in X} f(x, u, x')[\rho(x, u, x') + \gamma \max_{u'} Q^*(x', u')], \quad \forall x \in X, u \in U.$$

This equation states that the optimal value of selecting action u in state x can be computed as a summation of expected immediate reward and the expected (discounted) optimal value of the next state x'. The learning goal can be achieved by computing Q^* and selecting corresponding action according to the greedy policy applied to Q^* value:

$$\overline{h}(x) = \arg\max_u Q(x, u).$$

Classic technique to deal with exploration/exploitation challenge is ϵ-greedy action selection [144], where an agent selects greedy action (producing highest known reward) with probability $1 - \epsilon$ or an explorative action with probability ϵ. Note that all actions with nonzero probability will be selected after sufficiently large amount of iterations. It has been proven that some algorithms (e.g. Q-learning) converge to the optimal value function for nonzero exploration probability. Unfortunately the convergence speed may be very slow and depends on the number of actions mostly exponentially. Furthermore, another drawback of ϵ-greedy algorithm is that it chooses equally among all actions. Therefore, various other exploration/exploitation techniques have been proposed, including *Boltzmann Exploration* [144], *Optimistic Initial Values* that benefits rarely visited areas, or other more sophisticated algorithms [145].

In Q-learning, the current estimate of Q^* is computed according the following equation:

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha_k[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1}, u') - Q_k(x_k, u_k)],$$

where $\alpha_k \in (0, 1]$ is the learning rate that specifies the size of the adjustment towards new change. Q-learning is widely used algorithm with proven convergence and has provided an inspiration for more advanced RL algorithms (e.g. $Q(\lambda)$ [114]).

Team learning is an extension of single-agent reinforcement learning, where single learner discover the optimal policy for a team of agents. Team learning can be divided into two groups: *homogeneous* and *heterogeneous learning*. The former uses a single-agent behavior for all agents, the latter may create different types of behaviors, which allows agent specialization. Main disadvantage of team learning is the explosion of the state space, which could be problematic for methods that explore the space of state utilities (e.g. reinforcement learning methods).

Collaborative intrusion detection systems based on team learning would use a single learning mechanism for all system nodes, which could be undesirable for the security reasons. However, the presented methodology can be extended to multi-agent learning, which we briefly discuss in the following section.

2.3.3.2 Multiple Learners and Concurrent Learning

Multi-agent reinforcement learning generalizes MDP to stochastic game. A *stochastic game* is a tuple $(X, U_1, \ldots, U_n, f, \rho_1, \ldots, \rho_n)$, where n is the number of agents, X is a discrete set of states in the environment, U_i represents a set of actions available to agent i, $f : X \times \mathbf{U} \times X \to [0, 1]$ is the state transition probability function ($\mathbf{U} = U_1 \times \ldots \times U_n$), and $\rho_i = X \times \mathbf{U} \times \to R$ is the reward function of agent i.

Thus the change in state as well as the reward received depends on the joint action of all agents $\mathbf{u}_k = (u_{1,k}^T \ldots, u_{n,k}^T)$. Moreover, the policies of all individual agents $h_i : X \times U_i \to [0, 1]$ form the joint policy $\mathbf{h}$. Finally the Q-function of each agent is a function of the joint action conditioned on the joint policy: $Q_i^{\mathbf{h}} : X \times \mathbf{U} \to R$.

According to the relation between the reward functions, we can define fully cooperative scenarios, where all agents have the same goal ($\rho_1 = \ldots = \rho_n$) as well as fully competitive scenarios ($\rho_1 = -\rho_2$).

Single-agent learning algorithms in stationary environments usually explore the state space and may converge to local or global optimum. The situation becomes more difficult in dynamic environments, where the environment changes through time so the agent has to shift the optimal behavior with respect to the change of the environment. Moreover in multi-agent learning, the agents can adaptively modify each others' learning environment. To model and predict the dynamics of multi-agent learning, Vidal and Durfee [148] proposed technique of using parameters such as rate of behavior change per agent or learning rate to approximate the error in the decision function of the agents during the learning process.

In a fully cooperative scenario, the reward is divided among agents w.r.t. their interactions and current state of the environment. Most approaches uses general-sum stochastic games [35, 164], where the reward is assigned unequally. In other words, increasing the reward of one agent does not imply increasing the rewards of other agents, which may result in non-cooperative behavior. Recent approaches combine techniques from reinforcement learning and game theory to enable the application in dynamic environments [44, 131].

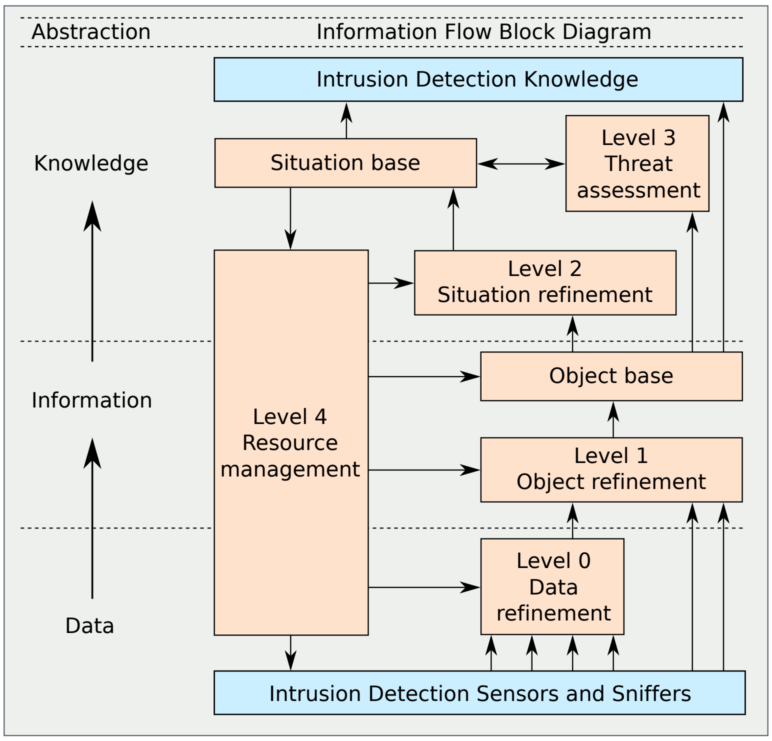

Abstraction
Information Flow Block Diagram
Intrusion Detection Knowledge
Knowledge
Situation base
Level 3 Threat assessment
Level 2 Situation refinement
Information
Object base
Level 4 Resource management
Level 1 Object refinement
Data
Level 0 Data refinement
Intrusion Detection Sensors and Sniffers

Fig. 2.5: Overview of the architecture proposed by Bass [25] for creating cyberspace situational awareness.

2.3.4 Knowledge Fusion in Network Security

Knowledge fusion has become widely used in collaborative intrusion detection systems as a technique capable of addressing novel challenges that have emerged in recent years. Bass [24, 25] proposed high-level fusion framework illustrated in Figure 2.5. This framework uses decision-support process from standard military doctrine called OODA (observe, orient, decide, act) that can be mapped into the three levels of abstraction: *data*, *information*, and *knowledge*. Data represents the observations and measurements, forming information when placed in some context. Knowledge is information explained and understood.

This framework extends the typical functionality of intrusion detection systems by introducing threat modeling phase and resource management feedback loop. The main goal of the proposed architecture is to present generated network security events at the level of human understanding of the network security threats. In the following, we will describe the purpose of each layer presented in Figure 2.5:

Level 0 The lowest layer ensures calibration and filtering of raw data from multiple sensors and alignment of measurements.

Level 1 The second layer performs data correlation and assigns weighted metrics based on relative importance to intrusion detection primitives. The output is an object, which represents an instance of the phenomenon that should be analyzed.

Level 2 In this layer, aggregation of intrusion detection primitives is performed based on their dependencies, point of origin, common protocols and targets, attack rates, and other high-level attributes. The output of this layer is the situational knowledge.

Level 3 The highest level is able to provide final assessments and implications of current situation by correlation with security policies.

Level 4 The last level serves as a fusion feedback loop. By using current situation knowledge, more thorough analysis can be made at different levels.

This alert fusion framework was further discussed by Corona et al. [43], where they used the proposed scheme to identify current research aims on information fusion according to the level of abstraction. Authors also extend the data fusion procedure with data organization and reconciliation. They claim that data should be organized by using several features describing their network context, e.g. topology, acquisition time, service type etc. Furthermore they discuss reconciliation and validation of raw data by using multiple sensors. Although the proposed suggestions and enhancements about data validation may refer to relevant security issues, they are outside of the scope of this thesis.

2.3.4.1 From Data to Information

Related work on data and object refinement includes numerous research approaches and techniques dealing with acquiring raw network data, their calibration, filtering, and representation. Numerous sampling techniques have been proposed to reduce the computational overhead of IDS systems deployed on high-speed network links. The techniques are described in Section 2.1 in more detail.

Another example of such refinement is anomaly detection assessment, where several detection methods evaluate the incoming traffic according to the degree of anomaly [13, 123]. The optimal aggregation function is selected and used to combine all individual assessments.

2.3.4.2 From Information to Knowledge

Nowadays sophisticated security threats are extremely difficult to detect by traditional network security monitoring devices. These threats may use cooperative or collaborative techniques to become stealthy for non-cooperative detection systems [155]. These attacks can be successfully detected when multiple systems operate in a cooperative mode, sharing results and gained experience. To provide most valuable and meaningful feedback, each system should be able to extract some knowledge from the observed data, which can be passed to other systems to create a complex network security awareness by using techniques from information and knowledge fusion. Generally, we distinguish between two types of information fusion techniques:

1. *classifier combination* - These techniques use various classifiers in order to take unique decisions about the patterns (single network packets or flows).
2. *alert correlation* - High-level description of the attacks generated by multiple intrusion detection systems.

Combining heterogeneous network security systems is motivated by the fact that it is extremely difficult for a single detection algorithm to detect all kinds of network threats. On the

Approach	Examples	Advantages	Disadvantages
Similarity based	[83, 146]	Easy to implement and use; Lower computational complexity; No prior knowledge needed	Unable to detect more sophisticated attacks
Attack scenario	[51, 116, 149]	Effective on predefined attacks	Unable to detect novel attacks; Attack library definition
Attack prerequisites & consequences	[45, 46, 163]	Able to detect novel attacks	Building complex attack database; High computational complexity
Filtering	[118]	Lower computational complexity; No prior knowledge needed	Poor scalability; Extensive knowledge of the environment needed

Table 2.4: An overview of existing alert fusion techniques with corresponding benefits and limitations.

other hand, different detection techniques are likely to make different types of errors. From this reason, it is reasonable to apply some techniques and methods from well-studied field of combining classifiers [92] to increase the overall detection potential but at the same time efficiently handle false positives and other classification errors. In our prior work, these techniques were applied when combining various anomaly detection methods processing the same low-level data [123]. Section 2.2 outlines the state of the art for classification methods in network security.

However, high-level information and knowledge from detection systems should be processed by alert correlation techniques. In the following, we will concentrate on the state-of-the-art in alert correlation. Alert management and correlation is concentration on finding mutual relationship between the alerts and has been studied by researches and vendors as a technique to reduce the number of false alarms generated by the network security mechanisms, to increase the level of abstraction of the alerts, and to recognize larger attack scenarios.

The proposed approaches can be categorized into four types of techniques [57, 162], which we discuss further in more detail:

- similarity between alert attributes [83, 142, 146],
- predefined attack scenarios [51, 116, 149],
- attack prerequisites and consequences [45, 46, 163],
- filtering techniques [118].

An overview of described alert fusion approaches together with their benefits and limitations is presented in Tab. 2.4.

The research in alert management focuses not only on novel correlation techniques and algorithms, but also on a formal definition of the problem. For example M2D2 data model [106] proposed by Morin et. al. formalizes an information model designed for network security information representation. The model includes four types of information (information about system characteristics, vulnerabilities, security tools, and events observed), which makes the model generalized enough to provide sufficient amount of relevant information for further correlation process of multiple information sources.

2.3.4.3 Similarity Between Alert Attributes

Each alert generated by IDS has several attributes associated with it, e.g. source and destination IP address, port numbers, protocol, timestamps. The most intuitive way how to correlate alerts is to group them through the similarity of their attributes. The main challenge is how to define the similarity metrics. The existing techniques proposed in the category are able to reduce large amounts of (false) alerts by aggregating similar alerts corresponding to the same attack or legitimate behavior.

Valdes and Skinner [146] proposed a three-layered probabilistic model that aggregates low-level events from heterogeneous sensors into security alerts on higher level of abstraction. At the first level, the mechanism aggregates the most primitive network events (e.g. TCP connections) into more advanced entities, which are further collected from various sensors by the correlation utility to perform alert correlation. The correlation procedure groups alerts into *meta alerts* (alerts of higher level of abstraction) by using feature similarity metrics. The authors also introduced an expectation of similarity E_i, a relationship between new alert and one meta alert, which expresses prior expectations that the feature X_i of alert X should match (based on predefined incident class similarity matrix) and serves as a weight factor in the similarity function between alert X and Y:

$$SIM(X,Y) = \frac{\sum_i E_i \cdot SIM(X_i, Y_i)}{\sum_i E_i}.$$

An approach proposed by Julisch [83] is based on identification of root causes that represent the reasons for which alarms occur. The alarms with similar attributes are grouped to clusters with their own root cause. This method concentrates on high-level alert analysis and tries to decrease the number of alarms by reducing the number of observed root causes.

2.3.4.4 Predefined Attack Scenarios

The approaches based on predefined attack scenarios are designed to reveal more complex attacks, usually executed in several steps. These methods are effective in detecting well-documented attacks that are included in attack scenario database by security experts, while novel attacks are usually missed.

In [51], Debar and Wespi proposed an aggregation and correlation component built on the top of an event managing product. They introduced an approach that combines similarity-based method with technique based on predefined attack scenarios by specifying four requirements for the alert correlation process: prevent *flooding* (i.e. operator overload) and *false positives*, put the information into *context* by aggregating related events and improve the *scalability* by level-of-detail/vulnerability selection management. The proposed aggregation and correlation process can be divided into three steps. The first step includes the *preprocessing of alerts* received from heterogeneous sensors into unified data model for intrusion detection alerts based on three attributes: probe or detector, source, and target. The consequent steps involve two types of alert correlation procedures: *relationship correlation* and *relationship aggregation*. The relationship correlation is based on explicit static rules and ensures identification of duplicates and consequences, while relationship aggregation groups similar alerts together by using several types of predefined situations based on the combination of three attributes: source, target, and alert class.

Perdisci et al. [116] proposed alert fusion technique, where signature-based IDS alerts are assigned to *meta-alerts* predefined by the user. These alerts generated by heterogeneous IDSs are aligned into the Intrusion Detection Message Exchange Format (IDMEF) [50], which is a standard message formate for intrusion detection reporting. The proximity to meta-alerts is defined with similarity metrics by using IP addreses, ports, etc. When an alert is not assigned to any meta-alert, it is reported to security experts for further label analysis.

Multi-dimensional alert correlation technique proposed by Zhou et al. [149] uses predefined patterns of attack types consisting of standard traffic features (IP addresses, ports, etc.). The incoming alerts are first correlated by using the predefined attack patterns, followed by filtering mechanism to remove insignificant or redundant pattern instances.

2.3.4.5 Attack Prerequisites and Consequences

Correlation methods based on attack prerequisites and consequences (sometimes named multi-stage approaches) try to reconstruct complex attack scenarios by discovering the causality relationship between the alerts. Typically, these methods use first order logic or attack modeling languages (LAMBDA [46]). The downside of this approach is that a large library of predefined state transitions is required for the algorithm to work properly, which is very expensive to create.

Cuppens [45] proposed an alert management mechanism composed of three functions: *alert management function* receives alerts generated by different IDS and stores them for further analysis in form of relational database; *alert clustering function* uses predefined expert rules (in form of predicates) to build clusters of alerts that correspond to the same occurrence of attack; *alert merging function* is responsible for merging previously-built clusters according to attack classifications, source/target, or temporal information.

Cuppens and Miege extended this mechanism by definition of additional *correlation function* [46]. The purpose of this function is to automatically generate a set of correlation rules (by using abductive reasoning), which can be applied on alerts to recognize more complex attack scenarios. The correlation process has an explicit phase, where the security administrator defines connections between events according to his knowledge, as well as implicit phase, where relations between events are determined by the statistical properties. The authors also discussed the preliminary possibilities of intention recognition (to anticipate the intruder's intentions) and consequent reaction of the network security administrator. The main disadvantage of this approach is strong influence of expert rules and knowledge on the overall effectiveness of the mechanism – these rules are mostly domain dependent and very expensive to define.

In [163], the authors present a technique for correlating alerts from signature-based NIDS by searching for specific alert relations. Several alert features were extracted (e.g. network addresses, type of service, credentials etc.) to find connections to other alerts by using requires/providers model and applying predefined inference rules. However, this technique does not perform well on multi-sensor attacks and other more sophisticated threats and is impractical due to relatively high computational requirements. Moreover, the inference process operates with detailed information provided by signature based NIDS, which could be hardly applicable in encrypted or security-sensitive environments.

2.3.4.6 Filtering Techniques

Definition of a complex library of state transitions represents very complicated and expensive task. Filtering techniques have been proposed to remove the need of creating such libraries.

Prospective alerts are prioritized according to their impact to targeted systems by using specific filtering methods.

Porras et al. [118] designed a system capable of collecting, correlating, and prioritizing alerts generated from multiple heterogeneous network security devices, e.g. firewalls, various antivirus software, or network-based and host-based IDSs. Incoming alerts are first transformed into an internal report format, then evaluated and ranked according to the *relevancy* (based on the topology of their targeted IP addresses), *priority* (degree to which an attack was successful), and the likelihood of success. Once ranked, correlation process filters alerts by using predefined security policies. Proposed approach is able to analyze heterogeneous alerts, however at a very high price – deep expert knowledge of network security perimeter as well as definition of security policies is essential for the proposed mechanism to function properly.

Chapter 3
Adaptive Sampling

In order to cope with an increasing volume of network traffic, flow sampling methods are deployed to reduce the volume of network traffic data collected and stored for monitoring, attack detection, and forensic purposes. Sampling necessarily frequently changes at least some of the statistical properties of the data and can reduce the effectiveness of subsequent analysis or processing. In this chapter, we propose two concepts that mitigate the negative impact of sampling on the data:

- Late sampling is based on a simple idea that the features used by the analytic algorithms can be extracted before the sampling and attached to the surviving flows. The surviving flows thus carry the representation of the original statistical distribution in these attached features.
- The second concept we introduce is that of adaptive sampling. Adaptive sampling deliberatively skews the distribution of the surviving data to overrepresent the rare flows or flows with rare feature values. This preserves the variability of the data and is critical for the analysis of malicious traffic, such as the detection of stealthy, hidden threats.

Our approach has been extensively validated on standard NetFlow data, as well as on HTTP proxy logs that approximate the use-case of enriched IPFIX for the network forensics.

New adaptive sampling method described in this chapter is the first contribution of the thesis. It resides at Level 1 of the proposed fusion architecture (see Figure 3.1) and reduces the incoming network traffic by putting the data into context with the precomputed statistics acquired at Level 0. The proposed approach guarantees that the number of sampled flows does not exceed the predefined limits necessary for higher layers of the collaborative fusion model and at the same time optimizes the distribution and representation of sampled flows.

3.1 Early and Late Sampling

As already discussed in the previous section, any sampling that is performed on the raw input data (e.g. network packets or flows) negatively impacts the traffic feature distributions. In the following, we will denote this traditional way of sampling as **early sampling**. An example of such negative impact in one dimension (number of flows originating from each source IP address) is illustrated in Figure 3.3. In contrast to the original distribution shown in Figure 3.2, early sampling may not only affect the shape of the distribution, but also downright eliminate most

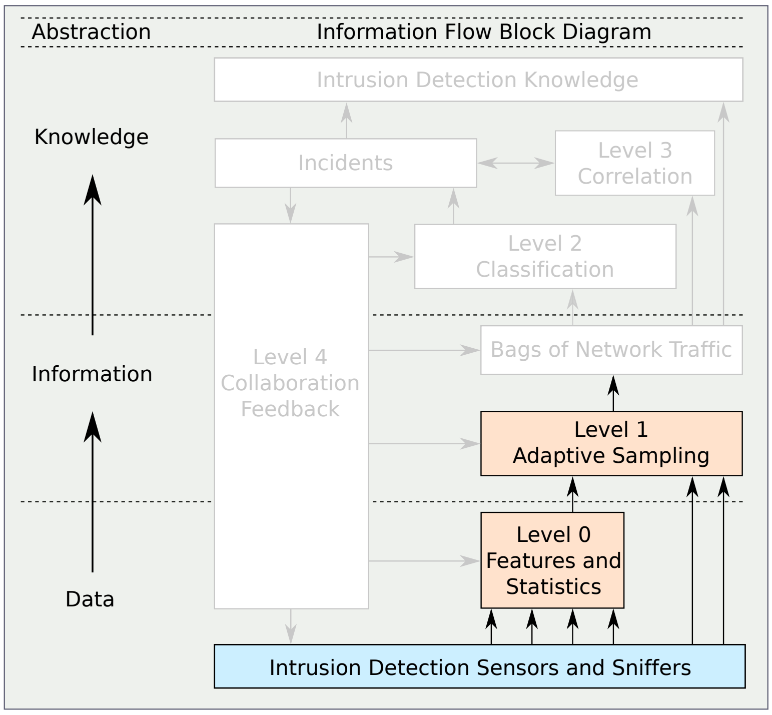

Abstraction
Information Flow Block Diagram
Intrusion Detection Knowledge
Knowledge
Incidents
Level 3 Correlation
Level 2 Classification
Level 4 Collaboration Feedback
Information
Bags of Network Traffic
Level 1 Adaptive Sampling
Level 0 Features and Statistics
Data
Intrusion Detection Sensors and Sniffers

Fig. 3.1: Adaptive sampling resides at Level 1 of the proposed architecture and reduces the incoming network traffic by putting the data into context with the precomputed statistics acquired at Level 0.

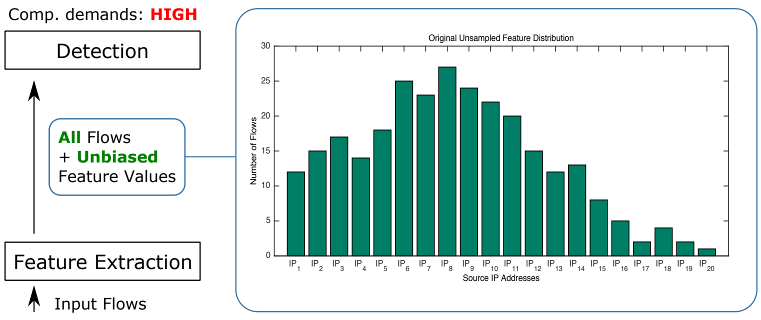


Fig. 3.2: Number of flows for 20 source IP addresses computed from the original (unsampled) data. The distribution is unbiased, however the computational demands of the detection algorithms are high.

of the rare feature values. And the flows related to the eliminated values are typically the ones that should be discovered by anomaly detection/detectors.

The impact on the specific anomaly detector would be twofold. First, most detectors use the shape of the feature distribution in order to build an internal predictive model [138]. Therefore, any significant change in the shape of the distribution directly impacts the efficacy of the anomaly detection techniques. Moreover, the sampling has disproportionally eliminated the rare or unusual feature values. These values are typically those that can be identified as anomalous and potentially malicious. Effectively, early, unbiased sampling has a filter-like effect on the

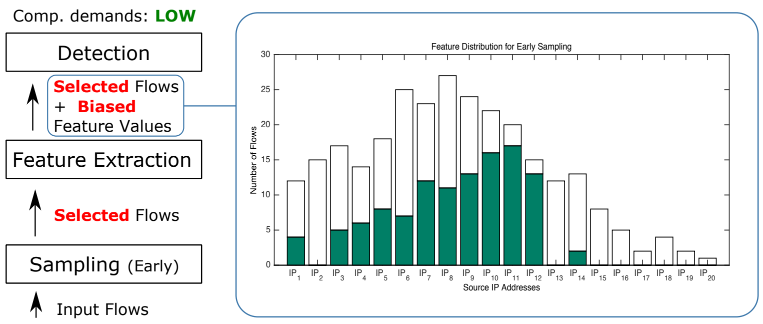


Fig. 3.3: The effect of early random sampling on feature distribution. Due to the fact that the flows are first randomly selected and the feature values are computed afterwards, early random sampling negatively shifted the feature distribution. You can see that flows originated from 8 source IP addresses were eliminated completely (pure white bars), while the rest of the values are highly imprecise.

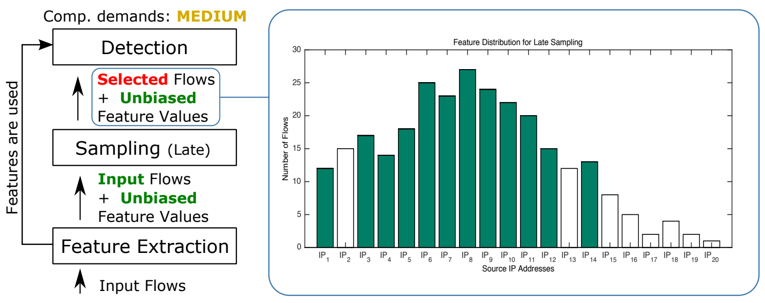


Fig. 3.4: Late random sampling preserves unbiased feature values of flows that are selected by the sampling. However, source IP addresses with small number of flows are still eliminated.

anomaly detection method behind it. As we will demonstrate in Section 3.4, early sampling considerably increases the minimal size of the detectable incidents.

To solve this problem, we need to differently restate the optimal sampling problem. We can no longer improve the sampling method in isolation, but we need to optimize the performance of the combination of the sampling and the specific anomaly detector. Working with the combination allows us to introduce two complementary techniques.

First, we propose **late sampling** that is based on a simple intuition. Statistical machine learning techniques rarely work with the raw data. Rather, they are applied in two steps. In the first step, the detector extracts features from the flows. In the second step, the features are used to build and maintain a statistical model. In late sampling, features are extracted from the full traffic **before** the sampling and *enrich* the sampled data with the features built from the full, unbiased data. This approach is based on the assumption that the computational cost related to the feature extraction and maintenance is significantly lower than the cost of the AD method itself. The difference between the traditional early and the proposed late sampling is depicted in Figures 3.3 and 3.4.

Bias introduced by the sampling is significantly reduced, which rapidly improves the detection performance of AD methods (verified experimentally in Section 3.4). The remaining bias is mainly due to the possible elimination of the specific flow record, but does not affect the values of the features associated with the flow (illustrated in Figure 3.4). Features are frequently extracted

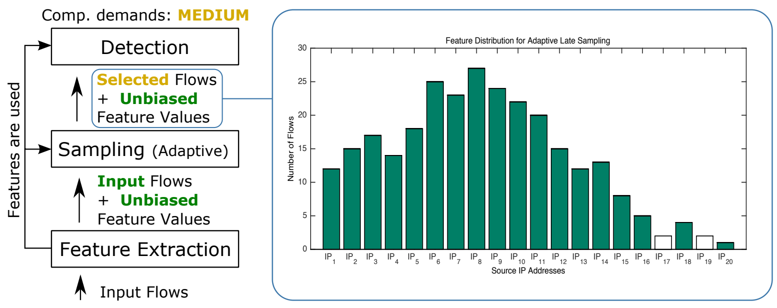


Fig. 3.5: Adaptive sampling increases the variability of flows (and the number of source IP addresses) selected to the sampled set, while late sampling preserves unbiased feature values for selected flows. Their combination (late adaptive) leads to the highest amount of preserved feature values.

from a group of flows (such as all the flows from one host/port over the selected time period), and are attached to each flow from the group. Therefore, even if we remove some flows from the group, the information associated with the remaining flows correctly reflects the properties of all flows from the group. Features extracted from the data can be used for the sampling method itself, helping it to select the most valuable and informative data.

Second, features important for the detector are known beforehand, so we can optimize for their preservation by the modification of the sampling algorithm itself. Therefore, we propose **adaptive sampling** (presented in Section 3.3) that modifies the sampling rate of flows w.r.t. their feature values to maximize the variability and minimize the redundancy. The combination of adaptive and late sampling minimizes the bias of feature distributions important for the consequent anomaly detection methods, as illustrated in Figure 3.5. Late sampling allows the adaptive sampling to emphasize the conservation of the variability in the data, as the proportions have been conserved by feature extraction.

In the next section, two measures designed to describe the statistical impact on the sampled data are proposed. These measures are used for the comparison of individual sampling methods in Section 3.4.

3.2 Sampling Measures

Comparing individual sampling approaches with each other has not been always straightforward, mainly due to the missing metrics. The these fills this gap by introducing two measures that will be used for comparing various sampling methods from the network security perspective.

Intuitively, the ideal sampling should be a process in which the number of samples and their distributions are selected in such a way that the loss of feature values is minimal. However, when dealing with the ideal sampling for network security purposes (i.e anomaly detection, network classification, or network forensics), different features and statistics impact the performance of the system differently, depending on the type of system or usage. This is because the individual methods or components of the system use only a subset of features received from the network traffic, or their feature sensitivity is variable. This fact implies that some features are more important for the given system, algorithm, or consequent analysis. The thesis starts with presenting the notation, followed by the definitions of the measures.

One flow (i.e. one network connection) is defined as a set of packets having the same source and destination IP address, source and destination port, and protocol [1]. Besides these basic features, each flow contains additional fields such as number of bytes or packets transferred, timestamp etc[1]. One flow will be denoted as φ and a set of n flows as $\Phi = \{\varphi_1, \ldots, \varphi_n\}$. Next, k-th feature of φ will be denoted as f_k and k-th feature value of φ as $\varphi^{(k)}$. For example, f_1 meaning source IP address implies that $\varphi^{(1)}$ denotes concrete source IP address of φ. Anomaly detection and classification methods compute distributions and statistics of these features for their algorithms and models. That is why it is important to preserve as many of these statistics and distributions as possible.

The distributions can be described by *feature statistics* which are computed from feature values. We distinguish between two types of feature statistics: *count features* f^c and *entropy features* f^e. These two types of statistics are used in most of the existing anomaly detection and classification methods and their preserving is important for the correct analysis of the sampled set. Count features indicate numbers of flows related to φ through the feature f_k, i.e. with feature value $\varphi^{(k)}$. For example, f_k meaning source IP address implies that $f^c_{\varphi^{(sIP)}}$ denotes the number of flows with the same source IP $\varphi^{(k)}$. Count features across more (first q) features will be denoted as $f^c_{\varphi^{(1,\ldots,q)}}$. Implicitly, we will use count features in number of flows unless told otherwise. Entropy features describe the entropy of feature f_k from flows related to φ through the feature f_l.

Furthermore, we will denote the original finite unsampled set as Φ_U and the finite sampled set as Φ_S. Thus $f^c_{\varphi^{(sIP)}}(\Phi_U)$ denotes the number of flows from the original set with exactly the same source IP address as has flow φ. And $f^{e^{(sPrt)}}_{\varphi^{(dIP)}}(\Phi_S)$ is the entropy of source ports from the sampled set, whose flows target exactly the same destination IP address as flow φ.

Definition 1: Let Φ_S be a set of flows sampled from the original set Φ_U with the sampling probability $p(x)$. Then the **reversibility degree of count feature** $f^c_{\varphi^{(k)}}$ is defined as:

$$r^c_k = \frac{1}{\mid \Phi_S \mid} \cdot \sum_{\forall \varphi \in \Phi_S} \left(\frac{f^c_{\varphi^{(k)}}(\Phi_S)}{f^c_{\varphi^{(k)}}(\Phi_U)} \right) \in [0,1]\,. \tag{3.1}$$

The reversibility degree shows how well the original (unsampled) feature values have been preserved in the sampled set Φ_S. Value $r^c_k = 1$ indicates that the feature values of feature f_k in the sampled set Φ_S were not affected by the sampling, while $r^c_k = 0$ means that all feature values in the sampled set has been completely lost.

To define the reversibility degree of entropy feature, we first describe the relative uncertainty described in [153]. The relative uncertainty of an entropy feature $f^{e^{(k)}}_{\varphi^{(l)}}$ is a normalized entropy:

$$ru^{e^{(k)}}_{\varphi^{(l)}} = \frac{f^{e^{(k)}}_{\varphi^{(l)}}}{\log f^c_{\varphi^{(l)}}} \in [0,1].$$

[1] Please, note that the specific set of features depends on the type of network data, such as NetFlow [1] or proxy logs.

Note that $\forall \varphi : 0 \leq f^{e^{(k)}}_{\varphi^{(l)}} \leq \log f^{c}_{\varphi^{(l)}}$, and $f^{e^{(k)}}_{\varphi^{(l)}} = 0$ for the case, where all values are the same (no diversity), while maximal value $f^{e^{(k)}}_{\varphi^{(l)}} = \log f^{c}_{\varphi^{(l)}}$ implies that the values occurred only once (maximal diversity).

Definition 2: Let Φ_S be a set of flows sampled from the original set Φ_U with the sampling probability $p(x)$. Then the **reversibility degree of entropy feature** $f^{e^{(k)}}_{\varphi^{(l)}}$ is defined as:

$$r^{e}_{k,l} = 1 - \frac{1}{\mid \Phi_S \mid} \cdot \sum_{\forall x \in \Phi_S} \mid ru^{e^{(k)}}_{\varphi^{(l)}}(\Phi_U) - ru^{e^{(k)}}_{\varphi^{(l)}}(\Phi_S) \mid \in [0,1]. \tag{3.2}$$

Please, note that late sampling, unlike traditional early sampling, has the reversibility degree of count and entropy features equal to 1, as all feature values are precomputed in advance from the original set Φ_U. Therefore, late sampling has a great advantage against other sampling methods. However, the reversibility degree of features is not the only measure the sampling method should optimize. For example, in the case of many missing flows with unique feature values, the sampling would be far from optimal. For this reason, we define a second measure called feature variability.

Definition 3: Let Φ_S be a set of flows sampled from the original set Φ_U with the sampling probability $p(x)$. Then the **coverage degree of feature** f_i is defined as:

$$c_i = \frac{\mathtt{d}_i(\Phi_S)}{\mathtt{d}_i(\Phi_U)} \in [0,1], \tag{3.3}$$

where $\mathtt{d}_i(\Phi_S), \mathtt{d}_i(\Phi_U)$ stands for the number of distinct values of f_i in Φ_S and Φ_U.

The coverage degree determines the ratio of how many unique values have remained in the sampled set Φ_S. Value $c_i = 1$ means that Φ_S contains all unique values of feature f_i from the original set Φ_U.

These two quality measures describe the conflicting demands in the anomaly detection and classification domain – to retain as much of the unique feature values as possible (measured with the coverage degree) with minimal loss in precision (measured with the reversibility degree). The proposed combination of late and adaptive sampling ensures maximum reversibility and better coverage degree than random sampling (see Section 3.4 for details). A high reversibility degree is desired in methods based on statistical modeling, while maximizing the coverage degree is essential for knowledge-based approaches that depend on specific values of the individuals.

3.3 Adaptive Sampling Technique

This section describes an adaptive, feature-aware sampling technique specifically designed for the purposes of anomaly detection, classification, and network forensics. The main ideas behind the proposed method are the following: (1) the incremental value of flows in a single set (defined by one or more common feature values) decreases with the growing number of similar flows already in the set, and (2) the system computes the feature statistics *before* the sampling procedure, so the statistics are computed from the original, full set of data. Before we present the algorithm itself, we first discuss these two assumptions in more detail.

The first idea is based on the assumption that if the same feature value occurred many times in the full set Φ_U, the benefit of adding this value to the sampled set Φ_S decreases in time. In other words, instead of selecting a large number n_1 of the same values, it is reasonably better to select only a satisfactory large subset of these values n_2 ($n_2 < n_1$) without compromising its great magnitude. This way, the algorithm is able to decrease the redundancy and increase the variability of feature values. The second idea of computing feature statistics before sampling (i.e. late sampling) is advantageous against the traditional early sampling as described in Section 3.1.

A majority of anomaly detection and classification methods use feature values in their algorithms. The are also important for both live and postmortem network forensics, as they help to identify the intention and the identity of the attacker. The effectiveness of a specific system usually depends on the values of features important for the system, and this set of features varies from system to system. Based on this fact, the features are divided into two categories. Features with large impact on the system or method belong to *primary features*, while the rest of the features are denoted as *secondary features*. Primary features are selected a priori based on the properties of the subsequent algorithms and methods.

Definition 4: Let $f_1, \ldots, f_k$ be primary features. We define primary probability p_p as the probability that a flow related to φ through features $f_1, \ldots, f_k$ is selected to the sampled set:

$$p_p(\varphi|f_1, \ldots, f_k) = \begin{cases} s^{(a)} & f^c_{\varphi(1,\ldots,k)} \leq t \\ s^{(a)} \cdot \frac{\log t}{\log f^c_{\varphi(1,\ldots,k)}} & f^c_{\varphi(1,\ldots,k)} > t \end{cases} \tag{3.4}$$

where $s^{(a)} \in [0,1]$ is the baseline sampling rate that can be computed according to Equation 3.7 from Theorem 1 introduced later in this Section. The threshold t defines a point in the distribution, where the sampling method starts setting the probability proportionally to the size of the feature value. The higher the feature value, the lower the sampling rate assigned. The position of the threshold t can be assigned manually, or preferably the threshold can be computed automatically by the Algorithm 2 to adapt on the incoming data. The effect of t is discussed later in this section in more detail.

This modification from random sampling slightly shifts the original probability distribution for flows with feature values above the threshold. We argue that reducing the size of attacks with higher feature values does not harm the system effectiveness, as these large attacks are mostly detectable with ease. Decrease in the sampling rate for frequent values allows the algorithm to increase the sampling rate for smaller and mostly hidden attacks without any change in the total number of sampled flows.

Definition 5: Let $f_1, \ldots, f_k$ be primary features and let f_i be a secondary feature. We define secondary probability, which is a probability that a flow related to φ through the feature f_i is selected to the sampled set, as:

$$p_s(\varphi|f_i) = \begin{cases} \max\left(ru^{e^{(i)}}_{\varphi(1,\ldots,k)}, d\right) & f^c_{\varphi(1,\ldots,k)} > t \\ 1 & \text{otherwise,} \end{cases} \tag{3.5}$$

where $d \in (0,1)$ is the lower bound, which penalizes redundant flows (i.e. flows with the same feature values).

Flows with the same feature values have $ru^{e^{(i)}}_{\varphi(1,\ldots,k)} = 0$ and $p_s(\varphi|f_i) = d \leq 1$. The secondary probability for these flows is decreased, as their information value is lower. Parameter d determines the minimal sampling rate that is applied when all values of a secondary feature f_i are the same. The value of d should respect the importance of f_i in the given system and it is the second parameter of the algorithm. Note that the decrease is applied only on flows with primary feature values above the threshold t (i.e. on flows with frequently used primary feature values). Flows with rare or unique values are not affected, as they have higher information value and decreasing the sampling rate for such flows is not desired.

Definition 6: Let $f_1, \ldots, f_k$ be primary features and $f_{k+1}, \ldots, f_n$ secondary features. Then the probability that the adaptive sampling will select flow φ is defined as follows:

$$p(\varphi) = p_p(\varphi \mid f_1, \ldots, f_k) \cdot \prod_{i=k+1}^{n} p_s(\varphi \mid f_i). \tag{3.6}$$

Network traffic is highly correlated and the features are not independent. However, it is not practical to compute a joint probability distribution over all features. The proposed adaptive sampling makes a compromise solution, taking joint distribution only from the primary features and the secondary features is considered as independent. This definition makes a reasonable trade-off between the precision of the probabilities and the computational complexity.

The proposed adaptive sampling is able to modify the sampling rate to reflect feature distributions of the network traffic. It selects flows according to the size of their feature values in order to suppress large, visible and easily detectable events, and to reveal some interesting facts from the smaller ones, while the feature distributions are slightly shifted for the benefit of the anomaly detection.

Theorem 1: Let T_{flows} be the maximal number of flows selected from $\varPhi_U$ into the sampled sets $\varPhi_S^{(1)}, \ldots, \varPhi_S^{(m)}$ (typically, it corresponds to the maximal number of flows the system is able to process) by using the adaptive sampling with the primary feature f_i and adaptive sampling rate $s^{(a)}$ computed as:

$$s^{(a)} = \frac{T_{flows}}{\sum_{f^c_{\varphi(i)} \leq t} 1 + \sum_{f^c_{\varphi(i)} > t} \frac{\log t}{\log f^c_{\varphi(i)}}}. \tag{3.7}$$

Then it holds:

$$\overline{\varPhi}_S = \lim_{m \to \infty} \left(\frac{1}{m} \cdot \sum_{i=1}^{m} |\varPhi_S^{(i)}| \right) \leq T_{flows}.$$

Note that even if the set $\varPhi_U$ is finite, it is possible to create an infinite sequence of its subsets, where the subsets do not have to be exclusively unique.

Proof: Let us assume that $T_{flows} \leq |\varPhi_U|$. Then we can express T_{flows} by using sampling rate $s^{(a)}$ and threshold t from Equation 3.7 as follows:

$$T_{flows} = s^{(a)} \cdot \left(\sum_{f^c_{\varphi(i)} \leq t} 1 + \sum_{f^c_{\varphi(i)} > t} \frac{\log t}{\log f^c_{\varphi(i)}} \right)$$

The summations are over all flows φ satisfying the threshold conditions. Now we can express $\overline{\varPhi}_S$ from Equations 3.4 and 3.6 as:

$$\overline{\varPhi}_S = s^{(a)} \cdot \left(\sum_{f^c_{\varphi(i)} \le t} 1 + \sum_{f^c_{\varphi(i)} > t} \frac{\log t}{\log f^c_{\varphi(i)}} \right) - \varepsilon_p = T_{flows} - \varepsilon_p \le T_{flows},$$

where $\varepsilon_p \geq 0$ represents decrease in number of sampled flows caused by secondary probabilities. Thus computing the parameter s according to the Equation 3.7 guarantees the theorem statement. $\square$

Theorem 1 provides for the adaptive sampling an upper bound in number of sampled flows. Knowing the computational constrains of the system, the operator can predefine the maximum number of sampled flows T_{flows} in some period of time and the algorithm will automatically adjust the sampling rate $s^{(a)}$ to maximize (but not exceed) this limit. Next, Theorem 2 describes the meaning of the threshold t and the relation between the adaptive and random sampling.

Theorem 2: Let t be the threshold of the adaptive sampling. If $t \to \infty$, then the results of the adaptive sampling converge to the results obtained by random sampling.

Proof: When $t \to \infty$, it is possible to find $T \in \mathbb{N}$ such that:

$$t \to \infty \Rightarrow (\exists T \in \mathbb{N})\ (\forall \varphi \in \varPhi_U,\ \forall f_i)\ (T \geq f^c_{\varphi(i)}).$$

Defining $t = max_{\varphi \in \varPhi_U}\{f^c_{\varphi(i)}\}$ implies that $\sum_{f^c_{\varphi(i)} > t} \frac{\log t}{\log f^c_{\varphi(i)}} = 0$. Then the adaptive sampling rate $s^{(a)}$ from Equation 3.7 can be simplified as:

$$s^{(a)} = \frac{T_{flows}}{\sum_{f^c_{\varphi(i)} \le t} 1} = \frac{T_{flows}}{\sum_{\varphi \in \varPhi_U} 1} = \frac{T_{flows}}{|\varPhi_U|} = s^{(r)}. \quad \square$$

Theorem 2 describes the meaning of the threshold t. Adaptive sampling with small t strongly emphasizes flows with rare values of primary features at the cost of the rest of the features and flows. This effect is illustrated in Figure 3.6. On the other hand, large t shifts the adaptive sampling towards a random sampling, where all samples are distributed equally regardless of the information gain (also shown in Figure 3.6).

Finally, we will discuss possible scenarios for the proposed algorithm. The sampling will be most effective on datasets with a few very large samples (feature value is highly above the threshold t) and many small samples (with feature values below t). In this scenario, the large samples would be sampled with significantly smaller probability than the small samples, which greatly increases the variability of the sampled set. In case the samples are equally distributed in all feature statistics (i.e. all features will have uniform distributions), the adaptive sampling is unable to make any preference and it would select all samples with the same probability. The proposed algorithm is designed to be easy to use and implement. The pseudo-code of the algorithm with one primary feature f_i and one secondary feature f_j is described in Algorithm 1.

As illustrated in Figure 3.6, threshold t significantly influences the properties of the adaptive sampling. Therefore, it should be carefully defined with respect to the deployed environment, detection engine or the purpose of the subsequent analysis. In some cases, it may be unclear how to define an optimal value for t. We propose to use Algorithm 2 to set the threshold t dynamically and optimally for the given data and environment. The input for the algorithm is a value v that is more intuitive and better describes the effects of the adaptive sampling. This value v is the point where the adaptive and random sampling samples with equal (or almost equal) probability, as shown in figure on the right hand side of the Algorithm 2.

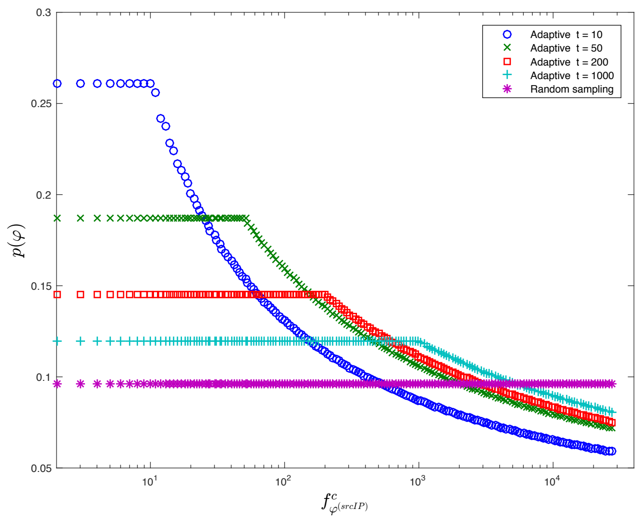


Fig. 3.6: Impact of the threshold t on the probability of being sampled $p(\varphi)$. Adaptive sampling with smaller threshold values (e.g. $t = 10$) emphasizes flows with rarely visited source IP addresses (i.e. with smaller $f^c_{\varphi(sIP)}$), as they have almost three times higher probability of being sampled when compared to the random sampling. On the other hand, highly redundant flows (i.e. flows with high value of $f^c_{\varphi(sIP)}$) are sampled with smaller probability than random sampling. However, thanks to their large number of occurrences, they will be sampled into the set $\varPhi_S$ as well.

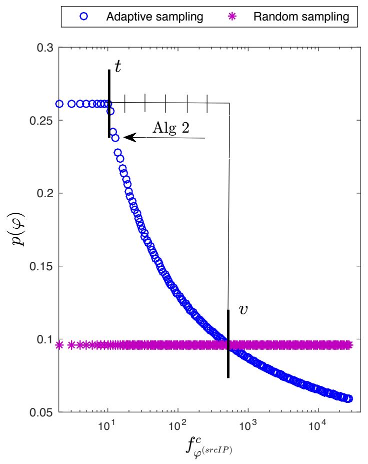
Algorithm 1 Adaptive Sampling Technique

```
$\varPhi_U$ = original (not sampled) set of flows $\varphi$
$\varPhi_S$ = sampled set
$t$ = predefined threshold (input parameter)
$T$ = maximum size for $\varPhi_S$

function SampleBatch
    $\varPhi_S$ = empty set of incidents
    $s^{(a)}$ = baselineSampling($\varPhi_U, t, T$)
    for $\varphi : \varPhi_U$ do
        $p(\varphi)$ = samplingProb($\varphi, s^{(a)}$)
        if random.nextDouble() $\leq p(\varphi)$ then
            $\varPhi_S.add(\varphi)$
        end if
    end for
    return $\varPhi_S$
end function

function BaselineSampling($\varPhi_U, t, T$)
    $T_{sum} = 0$
    for $\varphi : \varPhi_U$ do
        if $f^c_{\varphi(i)} \leq t$ then
            $T_{sum} = T_{sum} + 1$
        else $T_{sum} = T_{sum} + \frac{\log t}{\log f^c_{\varphi(i)}}$
        end if
    end for
    return $\frac{T}{T_{sum}}$
end function

function SamplingProb($\varphi, s^{(a)}$)
    $p = s^{(a)}$
    if $f^c_{\varphi(i)} > t$ then
        $p = p \cdot \frac{\log t}{\log f^c_{\varphi(i)}} \cdot p_s(\varphi|f_j)$
    end if
    return $p$
end function
```

Given value v, Algorithm 2 computes automatically the threshold t in such a way that the adaptive sampling will sample flows with feature values from interval $< 0, v)$ with greater probability than random sampling, while flows with feature values from interval $(v, +\infty)$ will be sampled with lower probability. Parameter v is typically set on the boundary of the sensitivity of the system, meaning that attacks with primary feature values higher than v can be easily detected or identified. Adaptive sampling increases the number of flows with low occurrences, improving the sensitivity of the system or the information value of the sampled data. In those

Algorithm 2 Dynamic computation of the threshold t

```
function ComputeThreshold($v, \Phi_U, T$)
    $t = v$
    $s^{(r)} = \frac{T}{|\Phi_U|}$
    $\tilde{\mathbf{f}}_i^c = \{f^c_{\varphi_1^{(i)}}, \ldots, f^c_{\varphi_k^{(i)}} \mid \forall i : f^c_{\varphi_i^{(i)}} < v\}$
    sortDownwards($\tilde{\mathbf{f}}_i^c$)
    for $u : \tilde{\mathbf{f}}_i^c$ do
        $s^{(a)}$ = baselineSampling($\Phi_U, u, T$)
        if $s^{(a)} \cdot \frac{\log u}{\log v} < s^{(r)}$ then return $t$
        end if
        $t = u$
    end for
    return $t$
end function
```



cases where computing the threshold t with Algorithm 2 cannot be realized, it is also possible to set $t = v$, which still leads to better or equal performance when compared to random sampling.

3.4 Experimental Evaluation

The goal of the sampling evaluation is to verify the benefits of late adaptive sampling technique in several aspects. The evaluation is based on the comparison of adaptive and random flow sampling techniques[2] in early and late configuration on real network traffic data. Packet sampling methods were not included due to significantly worse results at the flow level when compared to flow sampling [102].

The evaluation consists of four types of experiments with three different sets of data. The overview of the datasets used in the evaluation is provided in Table 3.1. Dataset A contains a one-day mix of a university network traffic (NetFlow format) with large horizontal and vertical scans and a small SSH brute force attack. The attacker launches a large scanning activity, which hides a more serious SSH brute force attack (of much smaller intensity) against other victim in the same network. Dataset B contains only one day of the university network traffic, which was partially labeled (approx. 45% of all flows) by security experts into various types of network behaviors, such as p2p, scanning, ssh cracking etc. Finally, the dataset C is a capture of one day of proxy logs (HTTP/HTTPS traffic) from a large company. Table 3.1 also shows the maximum number of flows in a 5-minute time interval and the requested maximum number of sampled flows. The corresponding sampling rates for random sampling are 1:6, 1:7, and 1:10 respectively (to fulfill the requested size of sampled data).

The datasets used in the evaluation include university and corporate networks. The impact of both large-scale and small targeted attacks was analyzed using two types of network data (NetFlow and proxy logs). The datasets cover wide statistical diversity of the network traffic, which allows to draw general conclusions about the performance of the proposed method.

[2] Random flow sampling was chosen for its massive deployment in practice.

Dataset	Data source	Description	Flows in 5 minutes	Sampled size	Speed
dataset A	NetFlow	University + attacks	6.2M	1.0M	10Gb
dataset B	NetFlow	University traffic	0.7M	0.1M	1Gb
dataset C	Proxy logs	Large company	2.0M	0.2M	> 10Gb

Table 3.1: Overview of the datasets used in the evaluation, together with their type, description, speed, and maximal input and predefined output/sampled size (in number of flows).

First, the performance of each sampling method was measured to show the differences in the computational complexity, followed by the evaluation of how the sampling methods influence traffic feature distributions. The impact of sampling on various existing anomaly detection methods was also evaluated. The experimental evaluation is concluded with the impact on the network forensics. The reason for using different sets of data is because of the nature of the experiments: e.g. network traffic from a 10-gigabit link is suitable for measuring the computational performance of sampling.

In the first three of our experiments, the parameters of the adaptive sampling were the following: $f^{c}_{\varphi(sIP)}$ was the primary feature, $f^{e^{(sP)}}_{\varphi(sIP)}$, $f^{e^{(dIP)}}_{\varphi(sIP)}$, and $f^{e^{(dP)}}_{\varphi(sIP)}$ were the secondary features, $d = 0.8$ and $t = 1000$. The rationale behind setting $d = 0.8$ is that $d \in (0, 1)$ should not be very close to 1 (which would minimize the effect of secondary probabilities). Setting $d = 0.8$ will ensure that flows with frequent source IP addresses and unique values of secondary features will be sampled two times more often (see Equations 3.5 and 3.6) than flows where the values of secondary features are identical, which further increases the diversity of the sampled set. The adaptive early sampling method computes only count statistics for primary feature and the rest of the statistics (entropies and other counts) is computed from the sampled set.

3.4.1 Sampling Performance

In this experiment, the performance of sampling methods was evaluated in terms of CPU time needed for sampling the input data and creating statistics for further post-processing (anomaly detection, classification, etc.). The experiment was performed with a 5-minute block of network traffic from two networks (dataset A and B) on four sampling methods: random early, adaptive early, random late, and adaptive late. In early sampling, the input flows are first sampled and then the system computes the statistics from the smaller sampled set. Both late sampling techniques compute statistics before sampling, which requires more computational time as you can see in Table 3.2. Note that the relative differences between the individual methods and sizes of the input data are more important than the absolute values, as they strictly depend on the specific computational assets used in the evaluation (4-core 3.2GHz processor).

Random early sampling method requires minimal computational time, but this sampling is devastating for any anomaly detection method as we will show further in our experimental evaluation. On the other hand, both late sampling methods (adaptive late and random late) are the most CPU time consuming, however this extra time is well invested w.r.t. the quality of results. We can see that the difference between random early and random late is in the extra time needed to compute all network statistics from the original network traffic. This time is affordable considering the fact that the input data represents 5 minutes of network traffic. Even

Input data	Random E	Adaptive E	Random L	Adaptive L
6,200,000 $\rightarrow$ 1,000,000	9531	13182	16972	17269
700,000 $\rightarrow$ 100,000	826	1419	2386	2730

Table 3.2: CPU time in ms needed to sample the input flows from datasets A and B by using four types of sampling techniques. On dataset A, random early (Random E) sampling was 1.8 times faster than adaptive late (Adaptive L).

Sampling	$c_{(sIP)}$	$c_{(dIP)}$	$c_{(sPrt)}$	$c_{(dPrt)}$	$c_{(Prot)}$	$c_{(Bytes)}$	$c_{(Pkt)}$
Random E	0.023	0.016	0.070	0.067	0.406	0.071	0.080
Random L	0.023	0.016	0.070	0.067	0.391	0.071	0.080
Adaptive E	0.060	0.013	0.174	0.162	0.469	0.143	0.151
Adaptive L	0.108	0.009	0.273	0.253	0.453	0.215	0.228

Table 3.3: Coverage degree of selected features computed from dataset A by using four types of sampling methods: random early, random late, adaptive early, and adaptive late. Higher values mean better coverage - minimal value is 0, maximal is 1.

though random late sampling computes the statistics from the original data, the method itself does not use them during the sampling process (unlike adaptive late). The statistics may be used by the anomaly detection methods or any other type of postprocessing. In contrast with adaptive late, adaptive early computes before sampling only count statistics for primary features. Thus, the difference between adaptive late and adaptive early expresses additional computational cost, when entropies are computed from the original set instead of the sampled set. Note that the majority of the detection algorithms require much more CPU time to process the corresponding amount of network traffic.

3.4.2 Preserving Feature Variability

This section provides a comparison of four evaluating sampling techniques in measures defined in Section 3.2. Since the reversibility degree (Equation 3.1 and 3.2) for late sampling approaches is 1 (maximal), we will compare the sampling methods based on the coverage degree (Equation 3.3) of the following features: source and destination IP addresses, source and destination ports, protocols, bytes, and packets. The methods were evaluated on dataset A to show the ability of preserving feature statistics on high-speed network links.

The individual values of the coverage degree is shown in Table 3.3. You can see that random early and random late sampling show almost identical behavior, which is expectable because none of them uses precomputed statistical information for the sampling itself. The late adaptive sampling outperforms the rest of the methods in almost all feature statistics, which means that it preserves more variability of the network traffic.

Overview of the impact of sampling on anomaly detection				
Network Behavior	Random E	Random L	Adaptive L	No sampling
horizontal scan	0.671	0.755	0.732	0.692
malicious	0.537	0.564	0.557	0.545
p2p	0.472	0.525	0.530	0.516
scan sql	0.846	0.858	0.869	0.845
skype supernode	0.432	0.496	0.502	0.458
ssh cracking	0.521	0.522	0.581	0.622
ssh cracking resp	0.531	0.576	0.575	0.674
vertical scan	0.700	0.793	0.799	0.780
average (stdev)	**0.571 (0.144)**	**0.636 (0.142)**	**0.643 (0.137)**	**0.641 (0.132)**

Table 3.4: Mean of AUC values across all of the anomaly detection methods for different types of malicious network behaviors and sampling methods. The results are measured on dataset B. Higher values are better. Even though it is not a backbone link, there is a measurable difference between random early sampling and the rest of the methods. The results of random late, adaptive late, and no-sampling method are comparable.

3.4.3 Impact on Anomaly Detection

In this Section, the impact of the sampling methods on the accuracy of six anomaly detection methods is analyzed. Namely, the method called *Minds* [59] models differences of the count features in time, *Xu* [153] defines static classification rules with relative uncertainty. Methods *F-D* and *F-S* are two newer methods [117] that extend the work of [94] and [95] with the prediction model of the network traffic by using PCA. Other two variants of this approach but with different features are called *Flags-F-D* and *Flags-F-S*. *Taps* [140] method is specially designed to detect scans and uses sequential hypothesis testing. These anomaly detection methods were integrated into the intrusion detection system – CAMNEP [123]. The detection methods require computation of the following statistics: distributions of flows/bytes/packets having the same sIP/dIP (or combination of the same sIP and dPrt or same dIP and sPrt), entropy of sP/dIP/dP having the same sIP, and entropy of sIP/sP/dP having the same dIP.

The impact of sampling on the above-mentioned anomaly detection methods was repeatedly evaluated on datasets A and B described in Table 3.1. We used 20 evaluations for each dataset and sampling method. On dataset B, the AUC (Area Under the ROC Curve) was calculated. Unknown (unlabeled) traffic was considered as legitimate. The evaluation results are shown in Table 3.4. Random early sampling decreases the detection capabilities by a measurable difference. On the other hand, both random late and adaptive late sampling show comparable results to the case when no sampling method is used, which means that concept of late sampling is suitable for subsequent anomaly detection or network classification.

The reason why the adaptive late sampling was only slightly better than random late sampling is because of the fact that the dataset B did not contain any larger network incidents that could be suppressed by the adaptive sampling. In the second part of this analysis, the evaluation is performed on dataset A to show the unique properties of the adaptive sampling technique on backbone links. You can see that the adaptive late was negligibly better than no-sampling method, which is thanks to emphasizing smaller attacks, so they are more visible and easily detectable. This effect will become more evident in the following analysis on dataset A.

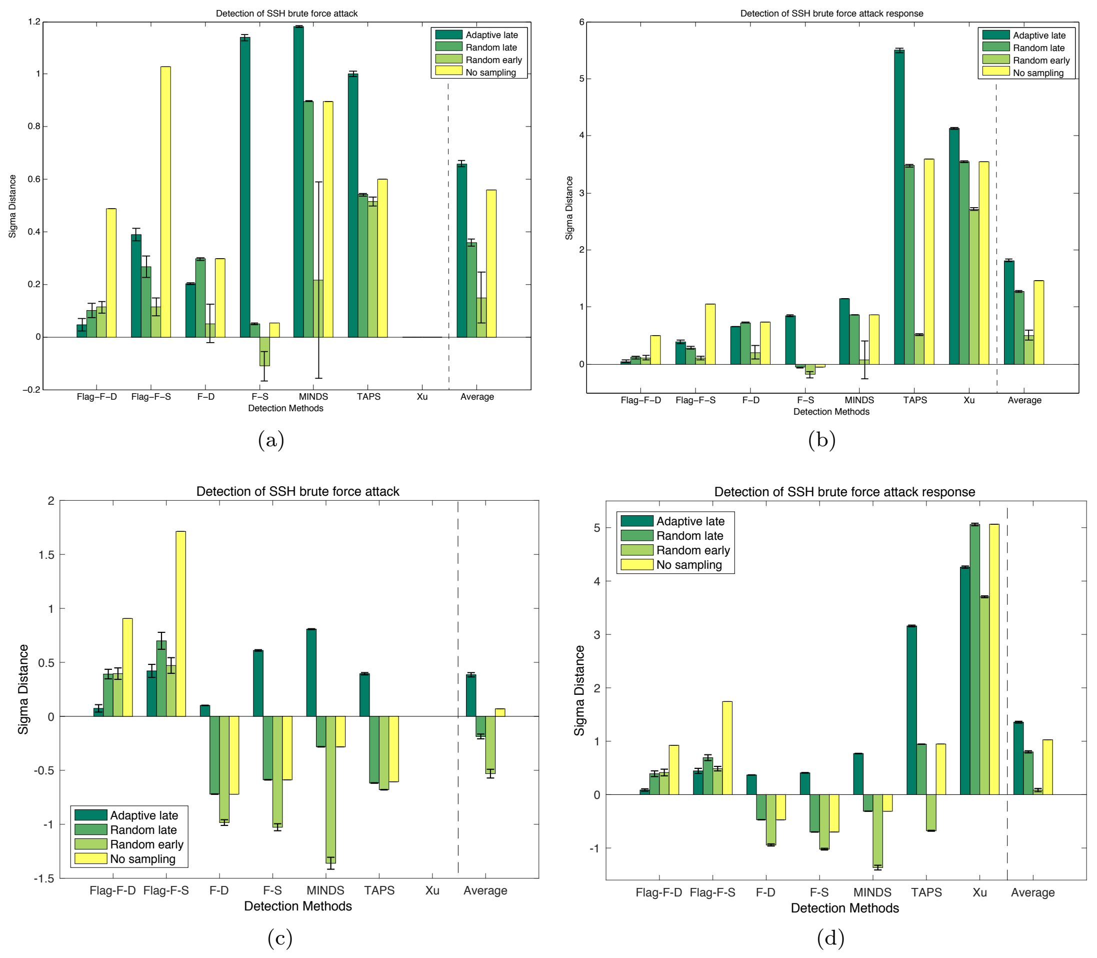


Fig. 3.7: Quality of detection of SSH brute force attack (requests and responses) by using different anomaly detection methods and sampling techniques on dataset A. First scenario with only brute force attack (a) and response (b) shows considerable detection improvements for late and adaptive sampling. Large scans included into the second scenario as a distraction make it more difficult for the methods to detect the hidden BF attack. Still, the combination of adaptive and late sampling is able to detect it – figure (c) for request and (d) for response.

The next evaluation is performed on dataset A and is divided into two scenarios: with and without large scans. The benefit of the adaptive sampling will be demonstrated on these two scenarios. Note, large scans can be easily detected, so the main focus is to analyze the detection quality of the hidden SSH brute force attack (in both scenarios). For each scenario, we have evaluated all detection methods individually and repeatedly 20 times, and the mean and standard deviation of the results is shown in Figures 3.7 and 3.8. The detection methods assign to every flow an anomaly score from interval $[0, 1]$, where 0 stands for least and 1 for most anomalous flow. The detection quality for each anomaly detection method is computed as a sigma distance from the mean as follows:

$$Q = (\mu_{attack} - \mu_{all})/\sigma, \tag{3.8}$$

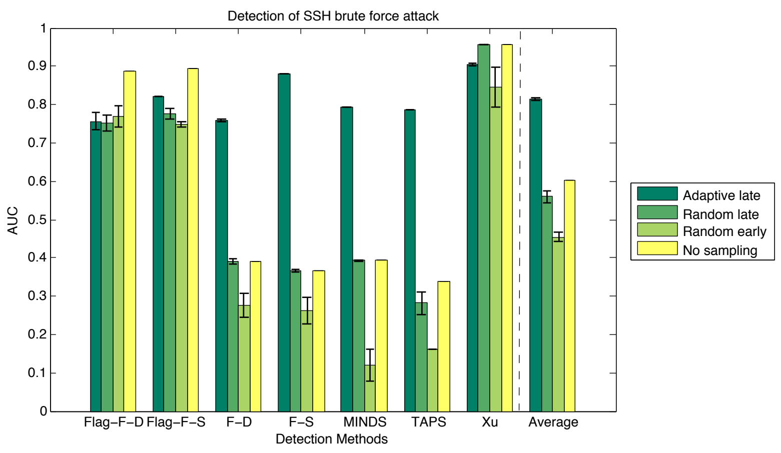


Fig. 3.8: Averge AUC values with the standard deviations of SSH brute force attack (request and response together) for various detection methods measured on dataset A. Both late sampling techniques outperformed random early and are more stable. Moreover, adaptive late sampling was able to decrease the amount of noise caused by the large scans and outperformed also no-sampling method.

where μ_{attack} is anomaly value of the attack and μ_{all} and σ is the mean and standard deviation of the anomaly values of all flows. Higher positive value means that the attack is better separated from the rest of the traffic. On the other hand, negative value means unsatisfactory detection results, as the corresponding anomaly detector did not consider the attack as anomaly.

Figures 3.7(a) - (b) illustrate the results from the first scenario without the large scans (only with the small brute force attack) of every individual detection method and a simple average of these methods. As you can see from Figure 3.7(a), the adaptive sampling separates the brute force attack from the rest of the traffic better than any other technique (even slightly better than no sampling), which is visible in the rightmost part of the figure. On the other hand, random early sampling shows very unsatisfactory results. Moreover, as opposed to early sampling, late sampling methods provide stable results, having only small standard deviation thanks to the precomputed statistics. Since the anomaly detection algorithms are deterministic, no-sampling method provided always the same results.

Figure 3.7(b) illustrates the sigma distance of the brute force attack response. Similarly as in the previous case, random early sampling is significantly worse than the rest of the methods. No-sampling is slightly better than random late. As the percentage of flows related to the attack is higher in Φ_S than in Φ_U, adaptive late was able to outperform no-sampling. Based on the analysis so far, late sampling is very important for the methods to be able to detect the attack and the adaptive late outperformed random late sampling.

The benefits of the adaptive late sampling become even more visible in the second case illustrated in Figures 3.7(c) and (d). Here, the small brute force attack is hidden behind the extensive scans. The results for the adaptive late sampling are almost identical with the previous scenario, which means that the inserted attacks had minimal negative influence. This is caused by the adaptive properties of the sampling – the amount of flows related to the large scan activity was decreased, allowing the detection methods to concentrate on the rest of the flows (including the attack) in more detail. We have also performed two-sample t-test for unequal variances to verify that the results of the adaptive sampling are better than the results obtained by other

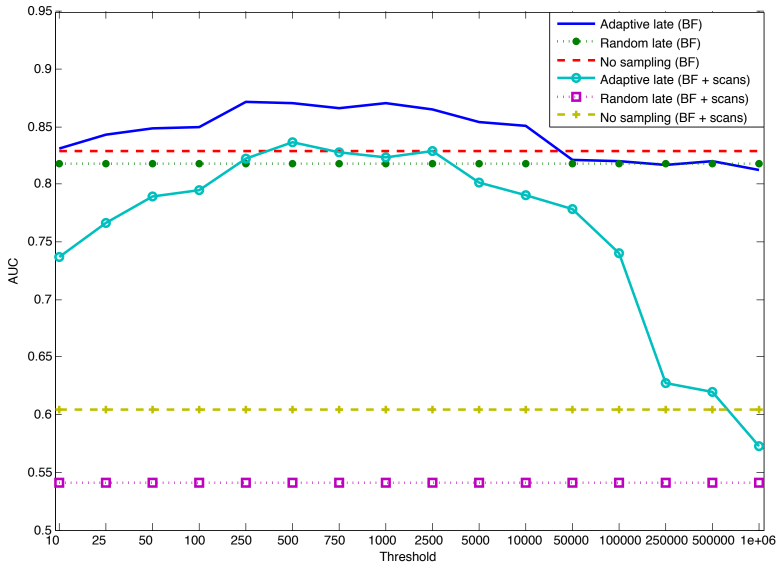


Fig. 3.9: The influence of the threshold t on average AUC values for both scenarios in dataset A – with brute force (BF), and with brute force and large scans (BF + scans). Comparison with random late and no-sampling is also included. For $t \in [25, 10000]$, the AUC value of the adaptive late is significantly better. As t goes to infinity, the results converge to the results obtained by random late method (as stated in Theorem 2).

methods. In all comparisons, the P-value was smaller than 0.001, which provides the evidence to reject the null hypothesis of equal means.

Figure 3.8 depicts the average AUC value of the attack and attack response and also confirms the results obtained so far. Finally, Figure 3.9 shows the influence of the threshold t on average AUC values for both scenarios in dataset A – with brute force (BF), and with brute force and large scans (BF + scans). Comparison with random late and no-sampling is also included. You can see that there is a large interval ($t \in [25, 10000]$), where the AUC value of the adaptive late sampling is significantly better, and as t goes to infinity, the results converge to the results obtained by random late approach (as stated in Theorem 2).

3.4.4 Impact on Network Forensics

The last part of the evaluation describes the effectiveness of the proposed adaptive late sampling for the purpose of network forensics. Retrieving true and relevant information is the key factor in network forensics. However, due to the still increasing amount of network traffic, it is not feasible to store and efficiently retrieve the full communication of the monitored network for longer time periods. To be able to use sampling in this context, the sampled data must contain as much feature variability (the ratio of retained unique values) as possible, while redundant flows can be easily discarded.

In this evaluation, flows (connections) from HTTP(S) proxy logs of a large worldwide company were analyzed for a period of one day – dataset C. Due to the large amount of network traffic (reaching 300 million flows per day only for HTTP(S)), network analysts have to decide between

All Flows				
Feature	Adaptive t=2	Adaptive t=10	Random	No-sampling
number of flows	28,595,886	28,794,294	28,796,348	281,614,703
number of users	102,689	102,501	102,264	110,898
number of domains	327,736	266,271	216,902	349,351
number of user agents	40,080	39,344	38,702	83,065
unique bytes down	371,563	360,155	337,818	892,642
unique bytes up	78,270	74,985	69,852	212,030

Table 3.5: Comparison of results obtained with the adaptive, random, and no sampling on the selected set of features – measured on all flows.

Malware Flows				
Feature	Adaptive t=2	Adaptive t=10	Random	No-sampling
number of flows	1,356	936	420	4,729
number of users	23	21	15	24
number of domains	38	28	19	53
number of user agents	20	18	14	21
unique bytes down	146	110	72	260
unique bytes up	4	3	3	4

Table 3.6: Comparison of results obtained with the adaptive, random, and no sampling on the selected set of features – measured on malware flows.

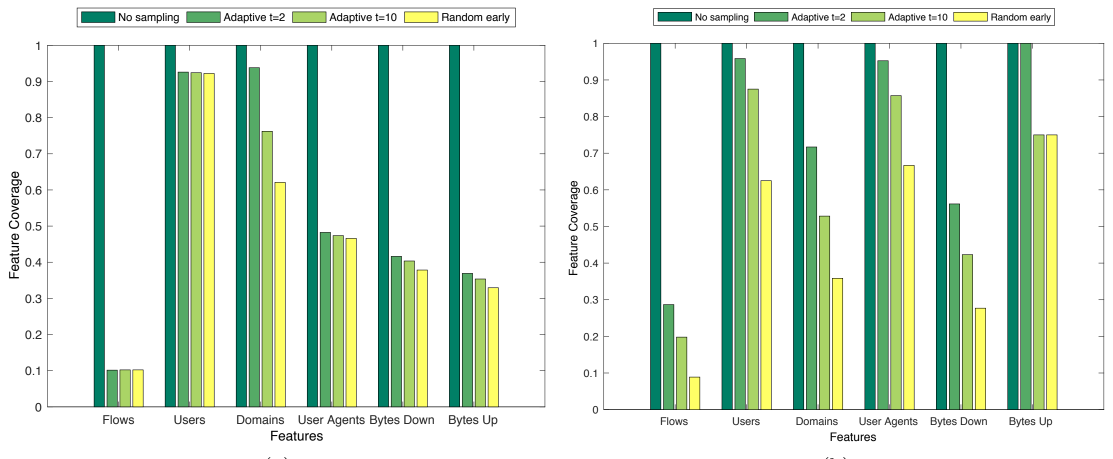
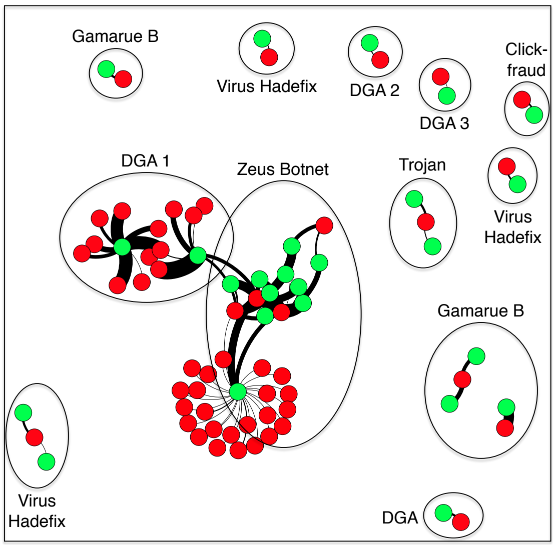
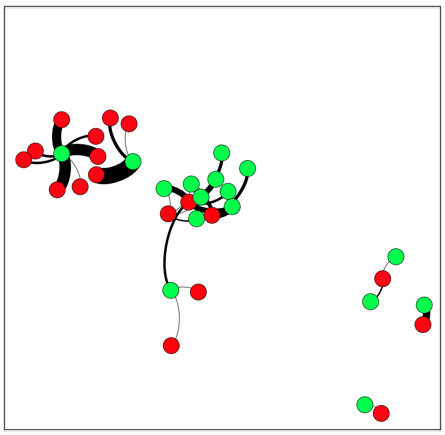


Fig. 3.10: Evaluation of four types of configurations (no sampling, adaptive t=2, adaptive t=10, and early random sampling) with the coverage measure on six different features (number of flows, users, domains, user agents, unique bytes down, and unique bytes up).

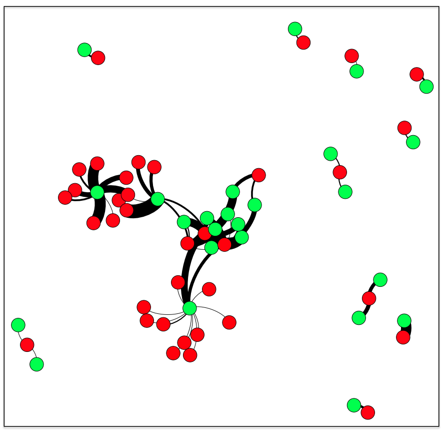
the increase of the costs for storage, or storing all the data for a limited time, or sampling the data and storing the sampled data for longer time periods. This analysis will show that for the given costs, sampling the data adaptively while extending the storage period represents a promising option. The output limit for the number of flows per 5-minute batch was 200k, which corresponds to the sampling rate approx. 1:10. Dataset C was analyzed by various security devices. The reports of the devices were verified manually and 24 confirmed infected users were found. Four types of configurations were compared: no sampling (i.e. analysis of the original



(a)



(b)



(c)

Fig. 3.11: Graph of malicious communication between the attackers (red nodes) and infected hosts (green nodes). The graph is computed from dataset C for: (a) original data without any sampling, (b) random sampling with rate 1:10, and (c) adaptive sampling with baseline rate around 1:10 and $t = 2$. Random sampling missed most of the individual attacks and large number of malicious domains, while the adaptive sampling preserved more malicious flows and most of the individual infections.

data), random sampling, and adaptive sampling in two configurations (with threshold $t = 2$ and $t = 10$). Low values of t were intentionally chosen to boost feature variability. The purpose of this evaluation is to demonstrate that the adaptive sampling is able to preserve critical information about the infected users even in significantly smaller sampled data.

Table 3.5 describes the impact of sampling on the proxy log features (feature coverage measure from Section 3.2) that are important for network forensics: number of unique flows, users, domains, user agents, and unique values of bytes up and bytes down. Each feature was preserved differently, e.g. there was only a slight loss in number of users, while the user agents or bytes were preserved with higher loss. However, adaptive sampling with ($t = 2$) achieved the best results in all features. The biggest difference is in the number of domains (as domains were set as the primary feature), where the adaptive sampling has only a minor loss, while random sampling lost 40% of the domains. The relative comparison of the methods is also depicted in Figure 3.10(a). The evaluation on the malicious flows resulted with the same conclusion, as described in Table 3.6 and illustrated in Figure 3.10(b).

Figure 3.11 shows the graph of malicious communication between the attackers (red nodes) and infected hosts (green nodes) for the original data (i.e. no sampling), and random and adaptive ($t = 2$) sampling. The thickness of the edges represents the amount of communication in number of flows. Therefore, these three graphs show the loss of information introduced by each sampling method. Random sampling (Figure 3.11(b)) missed most of the individual attacks (as they are hard to find due to their small sizes). The connection between the two large clusters was also lost. Finally, only two malicious domains were retained from the group of domains attacking a single user (located at the bottom in the center). On the other hand, the adaptive sampling with $t = 2$ (Figure 3.11(c)) performed significantly better. Most of the individual attacks together with the connection link between the two large clusters were preserved. Also more malicious domains were found from the group of domains attacking a single user. These results confirm the properties of the proposed approach. Thanks to the adaptive properties described in Section 3.3 and also thanks to the stability of the results shown in Section 3.4.3, any sampled set acquired with the adaptive late sampling will have similar composition in terms of favoring flows with unique feature values. And malicious traffic is typically connected with these unique feature values. Therefore, preserving non-redundant information is very important for any type of network forensics, where the identification of the attackers and the corresponding consequences needs to be done.

Overall, the evaluation shows that the adaptive sampling (unlike random sampling) is a valuable alternative when the data reduction needs to be applied without significant loss of information.

3.5 Summary

We presented two concepts that mitigate the negative impact of sampling on the data. Late sampling is based on a simple idea that the features used by the analytic algorithms can be extracted before sampling and attached to the surviving flows. The surviving flows thus carry the representation of the original statistical distribution in these attached features. The second concept we have introduced is adaptive sampling. Adaptive sampling deliberately skews the distribution of the surviving data to overrepresent the rare flows or flows with rare feature values.

Furthermore we defined two quality measures to evaluate the properties of the adaptive and random sampling. The quality metrics are general enough, allowing any sampling method to quantify the quality of the results from the anomaly detection standpoint.

Our approach has been extensively validated on standard NetFlow data, as well as on HTTP proxy logs that approximate the use-case of enriched IPFIX for the network forensics. Feature values remained unbiased thanks to the late sampling, while the feature variability was boosted by the adaptive sampling. The combination of the late and adaptive concepts achieved the best results in all evaluations and datasets and was able to retain almost all malware threats. The method has proven to be very promising when combined with anomaly detection or classification, and provides superior results for the network forensics. The proposed idea will be also effective on any future malware behaviors, with a significant exception: if the malware starts to mimic frequently used legitimate traffic, the feature values will become identical and the probability of malware flow selection would drop accordingly. This would also decrease the efficacy of any anomaly detection algorithm.

Chapter 4
Invariant Representation and Classification

Current network security devices classify large amounts of the malicious network traffic and report the results in many individually-identified incidents, some of which are false alerts. As we already described in Chapter 3, the proposed adaptive sampling method can be used to significantly reduce the amount of raw input data while optimizing the distribution and representation of sampled flows for methods at higher levels of the fusion model. At higher levels, a lot of malicious traffic remains undetected due to the increasing variability of malware attacks. As a result, security analysts might miss severe complex attacks because the incidents are not correctly prioritized or reported.

The network traffic can be classified at different levels of detail. Approaches based on packet inspection and signature matching [70] rely on a database of known malware samples. These techniques are able to achieve results with high precision (low number of false alerts), but their detection ability is limited only to the known samples and patterns included in the database (limited recall). Moreover, due to the continuous improvements of network bandwidth, analyzing individual packets is becoming intractable on high-speed network links. It is more efficient to classify network traffic based on *flows* representing groups of packets (e.g. NetFlow [1] or proxy logs [99]). While this approach has typically lower precision, it uses statistical modeling and behavioral analysis [39] to find new and previously unseen malicious threats (higher recall).

Statistical features calculated from flows can be used for unsupervised anomaly detection, or in supervised classification to train data-driven classifiers of known malicious traffic. While the former approach is typically used to detect new threats, it suffers from lower precision which limits its practical usefulness due to large amount of false alerts. Data-driven classifiers trained on known malicious samples achieve better efficacy results, but the results are directly dependent on the samples used in the training. Once a malware changes the behavior, the system needs to be retrained. With continuously rising number of malware variants, this becomes a major bottleneck in modern malware detection systems. Therefore, the robustness and invariance of features extracted from raw data plays the key role when classifying new malware.

The problem of changing malware behavior can be formalized by recognizing that a joint distribution of the malware samples (or features) differs for already known training (source) and yet unseen testing (target) data. This can happen as a result of target evolving after the initial classifier or detector has been trained. In supervised learning, this problem is solved by domain adaptation. Under the assumption that the source and target distributions do not change arbitrarily, the goal of the domain adaptation is to leverage the knowledge in the source domain and transfer it to the target domain. In this work, we focus on the case where the conditional distribution of the observation given labels is different, also called a conditional shift.

The domain adaptation (or knowledge transfer) can be achieved by adapting the detector using importance weighting such that training instances from the source distribution match the target distribution [132]. Another approach is to transform the training instances to the domain of the testing data or to create a new data representation with the same joint distribution of observation and labels [26]. The challenging part is to design a meaningful transformation that transfers the knowledge from the source domain and improves the robustness of the detector on the target domain.

We present a new optimized invariant representation of network traffic that enables domain adaptation under conditional shift. The representation is computed for bags of samples, each of which consists of features computed from network traffic logs. The bags are constructed for each user or device and contain all network communication with a particular hostname/domain. The representation is designed to be invariant under shifting and scaling of the feature values and under permutation and size changes of the bags. This is achieved by combining bag histograms with an invariant self similarity matrix for each bag. All parameters of the representation are learned automatically for the training data using the proposed optimization approach.

The proposed invariant representation is applied to detect malicious HTTP traffic and resides at Level 2 of the proposed architecture – see Figure 4.1. We will show that the classifier trained on malware samples from one category can successfully detect new samples from a different category. This way, the knowledge of the malware behavior is correctly transferred to the new domain. Compared to the baseline flow-based representation or widely-used security device, the proposed approach shows considerable improvements and correctly classifies new types of network threats that were not part of the training data.

4.1 Formalization of the Problem

This chapter deals with the problem of creating a robust representation of network communication that would be invariant against modifications an attacker can implement to evade the detection systems. The representation is used to classify network traffic into positive (malicious) or negative (legitimate) category. The labels for positive and negative samples are often very expensive to obtain. Moreover, sample distribution typically evolves in time, so the probability distribution of training data differs from the probability distribution of test data. This complicates the training of classifiers which assume that the distributions are the same. In the following, the problem is described in more detail.

Each sample (in our case each sampled flow φ) is represented as an n-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^n$. Samples (flows) are grouped into bags, with every bag represented as a matrix $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{n \times m}$, where m is the number of samples in the bag and n is the number of features. The bags may have different number of samples. A single category y_i can be assigned to each bag from the set $\mathcal{Y} = \{y_1, \ldots, y_N\}$. Only a few categories are included in the training set. The probability distribution on training and testing bags for category y_j will be denoted as $P^L(X|y_j)$ and $P^T(X|y_j)$, respectively. Moreover, the probability distribution of the training data differs from the probability distribution of the testing data, i.e. there is a domain adaptation

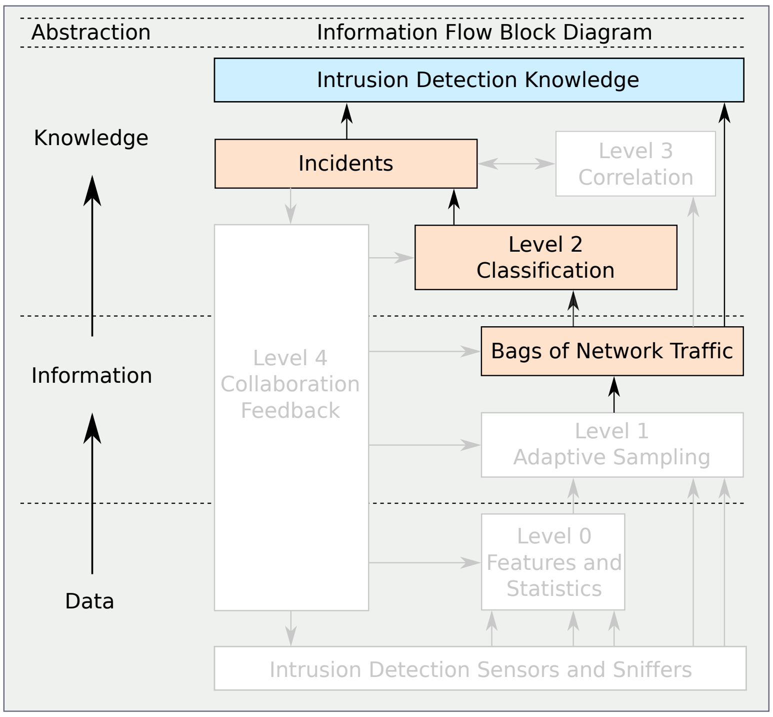

Abstraction
Information Flow Block Diagram
Intrusion Detection Knowledge
Knowledge
Incidents
Level 3 Correlation
Level 2 Classification
Information
Level 4 Collaboration Feedback
Bags of Network Traffic
Level 1 Adaptive Sampling
Data
Level 0 Features and Statistics
Intrusion Detection Sensors and Sniffers

Fig. 4.1: Classification module resides at Level 2 of the proposed architecture. The module transforms sampled network traffic into a representation of bags (groups of flows). The bages are classified into malicious incidents pointing to infected users or devices.

problem [30] (also called a conditional shift [160]):

$$P^L(X|y_j) \neq P^T(X|y_j), \;\; \forall y_j \in \mathcal{Y}. \tag{4.1}$$

The purpose of the domain adaptation is to apply knowledge acquired from the training (source) domain into test (target) domain. The relation between $P^L(X|y_i)$ and $P^T(X|y_i)$ is not arbitrary, otherwise it would not be possible to transfer any knowledge. Therefore there is a transformation τ, which transforms the feature values of the bags onto a representation, in which $P^L(\tau(X)|y_i) \approx P^T(\tau(X)|y_i)$. The goal is to find this representation, allowing to classify individual bag represented as X into categories $\mathcal{Y} = \{y_1, \ldots, y_N\}$ under the above mentioned conditional shift.

Numerous methods for transfer learning have been proposed (since the traditional machine learning methods cannot be used effectively in this case), including kernel mean matching [68], kernel learning approaches [54], maximum mean discrepancy [80], or boosting [47]. These methods try to solve a general data transfer with relaxed conditions on the similarity of the distributions during the transfer. The downside of these methods is the necessity to specify the target loss function and availability of large amount of labeled data.

This chapter proposes an effective invariant representation that solves the classification problem with a covariate shift (see Equation 4.1). Once the data are transformed, the new feature values do not rely on the original distribution and they are not influenced by the shift. The parameters of the representation are learned automatically from the data together with the classifier as a joint optimization process. The advantage of this approach is that the parameters are optimally chosen during training to achieve the best classification efficacy for the given classifier, data, and representation.

4.2 Invariant Representation

The problem of domain adaptation outlined in the previous section is addressed by the proposed representation of bags. The new representation is calculated with a transformation that consists of three steps to ensure that the new representation will be invariant under scaling and shifting of the feature values and under permutation and size changes of the bags.

4.2.1 Scale Invariance

As stated in Section 4.1, the probability distribution of bags from the training set can be different from the test set. In the first step, the representation of bags is transformed to be invariant under scaling of the feature values. The traditional representation X of a bag that consists of a set of m samples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ can be written in a form of a matrix:

$$X = \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_m \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ & & \vdots & \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{pmatrix}, \tag{4.2}$$

where x_{lk} denotes k-th feature value of l-th sample. This form of representation of samples and bags is widely used in the research community, as it is straightforward to use and easy to compute. It is a reasonable choice in many applications with a negligible shift in the source and target probability distributions. However, in the network security domain, the dynamics of the network environment causes changes in the feature values and the shift becomes more prominent. This shift can be also caused by the volumetric changes introduced by sampling. As a result, the performance of the classification algorithms using the traditional representation is decreased.

In the first step, the representation is improved by making the matrix X to be invariant under scaling of the feature values. **Scale invariance** guarantees that even if some original feature values of all samples in a bag are multiplied by a common factor, the values in the new representation remain unchanged. To guarantee the scale invariance, the matrix X is scaled locally onto the interval $[0, 1]$ as follows:

$$\tilde{X} = \begin{pmatrix} \tilde{x}_{11} & \ldots & \tilde{x}_{1n} \\ & \vdots & \\ \tilde{x}_{m1} & \ldots & \tilde{x}_{mn} \end{pmatrix} \quad \tilde{x}_{lk} = \frac{x_{lk} - \min_l(x_{lk})}{\max_l(x_{lk}) - \min_l(x_{lk})} \tag{4.3}$$

4.2.2 Shift Invariance

In the second step, the representation is transformed to be invariant against shifting. **Shift invariance** guaranties that even if some original feature values of all samples in a bag are increased/decreased by a given amount, the values in the new representation remain unchanged. Let us define a *translation invariant distance function* $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ for which the following holds: $d(u, v) = d(u + a, v + a)$.

Let x_{pk}, x_{qk} be k-th feature values of p-th and q-th sample from bag matrix X. Then the distance between these two values will be denoted as $d(x_{pk}, x_{qk}) = s^k_{pq}$. The distance $d(x_{pk}, x_{qk})$ is computed for pairs of k-th feature value for all sample pairs, ultimately forming a so called self-similarity matrix S^k. Self-similarity matrix is a symmetric positive semidefinite matrix, where rows and columns represent individual samples and (i, j)-th element corresponds to the distance between i-th and j-th sample. Self-similarity matrix has been already used thanks to its properties in several applications (e.g. in object recognition [84] or music recording [109]). However, only a single self-similarity matrix for each bag has been used in these approaches. This paper proposes to compute a set of similarity matrices, one for every feature. More specifically, a per-feature set of self-similarity matrices $\mathcal{S} = \{S^1, S^2, \ldots, S^n\}$ is computed for each bag, where

$$S^k = \begin{pmatrix} s^k_{11} & s^k_{12} & \ldots & s^k_{1m} \\ & & \vdots & \\ s^k_{m1} & s^k_{m2} & \ldots & s^k_{mm} \end{pmatrix}. \tag{4.4}$$

The element $s^k_{pq} = d(x_{pk}, x_{qk})$ is a distance between feature values x_{pk} and x_{qk} of k-th feature. This means that the bag matrix X with m samples and n features will be represented with n self-similarity matrices of size $m \times m$. The matrices are further normalized by local feature scaling described in Section 4.2.1 to produce a set of matrices $\tilde{\mathcal{S}}$. This transformation is illustrated in Figure 4.2 as step (3).

The shift invariance makes the representation robust to the changes where the feature values are modified by adding or subtracting a fixed value. For example, the length of a malicious URL would change by including an additional subdirectory in the URL path. Or, the number of transfered bytes would increase when an additional data structure is included in the communication exchange.

4.2.3 Permutation and Size Invariance

Representing bags with scaled matrices $\{\tilde{X}\}$ and sets of locally-scaled self-similarity matrices $\{\tilde{\mathcal{S}}\}$ achieves the scale and shift invariance. **Size invariance** ensures that the representation is invariant against the size of the bag. In highly dynamic environments, the samples may occur in a variable ordering. **Permutation invariance** ensures that the representation should also be invariant against any reordering of rows and columns of the matrices. The final step of the proposed transformation is the transition from the scaled matrices $\tilde{X}$, $\tilde{\mathcal{S}}$ (introduced in Sections 4.2.1 and 4.2.2 respectively) to normalized histograms. For this purpose, we define for each bag:

$$\boldsymbol{z}^X_k := \text{vector of values from } k\text{-th column of matrix } \tilde{X}$$

$$\boldsymbol{z}_k^{\mathcal{S}} := \text{column-wise representation of upper triangular matrix created from matrix } \tilde{S}^k \in \tilde{\mathcal{S}}.$$

This means that $\boldsymbol{z}_k^X \in \mathbb{R}^m$ is a vector created from values of k-th feature of $\tilde{X}$, while $\boldsymbol{z}_k^{\mathcal{S}} \in \mathbb{R}^r, r = (m-1) \cdot \frac{m}{2}$ is a vector that consists of all values of upper triangular matrix created from matrix $\tilde{S}^k$. Since $\tilde{S}^k$ is a symmetric matrix with zeros along the main diagonal, $\boldsymbol{z}_k^{\mathcal{S}}$ contains only values from upper triangular matrix of $\tilde{S}^k$.

A normalized histogram of vector $\boldsymbol{z} = (z_1, \ldots, z_d) \in \mathbb{R}^d$ is a function $\boldsymbol{\phi}\colon \mathbb{R}^d \times \mathbb{R}^{b+1} \to \mathbb{R}^b$ parametrized by edges of b bins $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_b) \in \mathbb{R}^{b+1}$ such that

$$\boldsymbol{\phi}(\boldsymbol{z}; \boldsymbol{\theta}) = (\phi(\boldsymbol{z}; \theta_0, \theta_1), \ldots, \phi(\boldsymbol{z}; \theta_{b-1}, \theta_b)),$$

where

$$\phi(\boldsymbol{z}, \theta_i, \theta_{i+1}) = \frac{1}{d} \sum_{j=1}^{d} [\![z_j \in [\theta_{i-1}, \theta_i)]\!]$$

is the value of the i-th bin corresponding to a portion of components of $\boldsymbol{z}$ falling to the interval $[\theta_{i-1}, \theta_i)$.

Each column k of matrix $\tilde{X}$ (i.e. all bag values of k-th feature) is transformed into a histogram $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ with predefined number of b bins and $\boldsymbol{\theta}_k^X$ bin edges (boundary). Such histograms created from the columns of matrix $\tilde{X}$ will be denoted as *feature values histograms*, because they carry information about the distribution of bag feature values (illustrated in Figure 4.2 as step (2)). On the other hand, histogram $\boldsymbol{\phi}(\boldsymbol{z}_k^{\mathcal{S}}, \boldsymbol{\theta}_k^S)$ created from values of self-similarity matrix $\tilde{S}^j \in \tilde{\mathcal{S}}$ will be called *feature differences histograms*, as they capture inner feature variability within bag samples. This transformation is illustrated in Figure 4.2 as step (4).

Overall, each bag is represented as a large concatenated feature map $\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})\colon \mathbb{R}^{n \times (m+r)} \to \mathbb{R}^{2 \cdot n \cdot b}$ as follows:

$$\left(\boldsymbol{\phi}(\boldsymbol{z}_1^X, \boldsymbol{\theta}_1^X), \ldots, \boldsymbol{\phi}(\boldsymbol{z}_n^X, \boldsymbol{\theta}_n^X), \boldsymbol{\phi}(\boldsymbol{z}_1^{\mathcal{S}}, \boldsymbol{\theta}_1^S), \ldots, \boldsymbol{\phi}(\boldsymbol{z}_n^{\mathcal{S}}, \boldsymbol{\theta}_n^S)\right), \tag{4.5}$$

where n is the number of the original flow-based features, m is the number of flows in the bag, and b is the number of bins. The whole transformation from input network flows to the final feature vector is depicted in Figure 4.2. As you can see, two types of invariant histograms are created from values of each flow-based feature. At the end, both histograms are concatenated into the final bag representation $\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})$.

4.3 Learning Optimal Histogram Representation

The bag representation $\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})$ proposed in Section 4.2 has the invariant properties, however it heavily depends on the number of bins b and their edges $\boldsymbol{\theta}$ defining the width of the histogram bins. These parameters that were manually predefined in Section 4.2 C influence the classification performance. Incorrectly chosen parameters b and $\boldsymbol{\theta}$ leads to suboptimal efficacy results. To define the parameters optimally, we propose a novel approach of learning these parameters automatically from the training data in such a way to maximize the classification separability between positive and negative samples.

When creating histograms in Section 4.2.3, the input instances are vectors $\boldsymbol{z}_k^X$ and $\boldsymbol{z}_k^{\mathcal{S}}$, where $k \in \{1, \ldots, n\}$. The algorithm transforms the input instances into a concatenated histogram

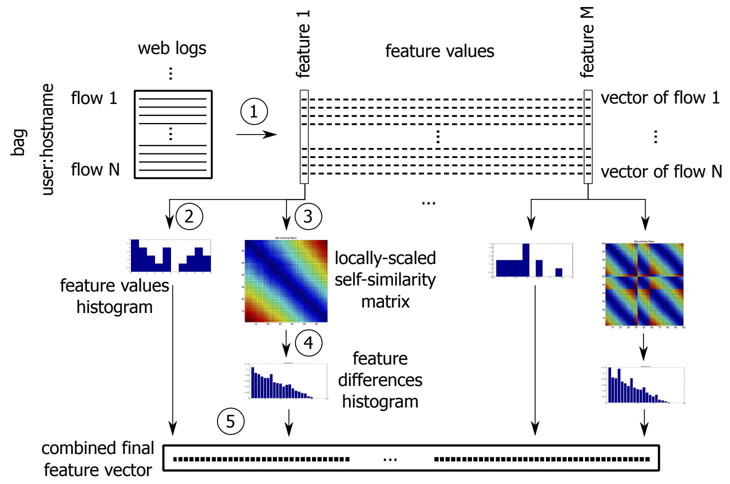


Fig. 4.2: Graphical illustration of the individual steps that are needed to transform the bag (set of flows with the same user and hostname) into the proposed invariant representation. First, the bag is represented with a standard (flow-based) feature vector (1). Then feature values histograms of locally scaled feature values are computed for each feature separately (2). Next, the locally-scaled self-similarity matrix is computed for each feature (3) to capture inner differences. This matrix is then transformed into feature differences histogram (4), which is invariant on the number or the ordering of the samples within the bag. Finally, feature values and feature differences histograms of all features are concatenated into resulting feature vector.

$\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})$. To keep the notation simple and concise, we will denote the input instances simply as $\boldsymbol{z} = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n) \in \mathbb{R}^{n \times m}$ (instead of $\boldsymbol{z} = (\boldsymbol{z}_1^X, \ldots, \boldsymbol{z}_n^X, \boldsymbol{z}_1^{\mathcal{S}}, \ldots, \boldsymbol{z}_n^{\mathcal{S}})$), which is a sequence of n vectors each of dimension m.

The input instance $\boldsymbol{z}$ is represented via a feature map $\boldsymbol{\phi}\colon \mathbb{R}^{n \times m} \to \mathbb{R}^{n \cdot b}$ defined as a concatenation of the normalized histograms of all vectors in that sequence, that is, $\boldsymbol{\phi}(\boldsymbol{z}; \boldsymbol{\theta}) = (\boldsymbol{\phi}(\boldsymbol{z}_1; \boldsymbol{\theta}_1), \ldots, \boldsymbol{\theta}(\boldsymbol{z}_n; \boldsymbol{\theta}_n))$, where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$ denotes bin edges of all normalized histograms stacked to a single vector.

We aim at designing a classifier $h\colon \mathbb{R}^{n \times m} \times \mathbb{R}^{n+1} \times \mathbb{R}^{n(b+1)} \to \{-1, +1\}$ working on top of the histogram representation, that is

$$h(\boldsymbol{z}; \boldsymbol{w}, w_0, \boldsymbol{\theta}) = \operatorname{sign}(\langle \boldsymbol{\phi}(\boldsymbol{z}, \boldsymbol{w}) \rangle + w_0) = \operatorname{sign}\left(\sum_{i=1}^{n} \sum_{j=1}^{b} \phi(\boldsymbol{z}_i, \theta_{i,j-1}, \theta_{i,j}) w_{i,j} + w_0 \right) . \tag{4.6}$$

The classifier (4.6) is linear in the parameters $(\boldsymbol{w}, w_0)$ but non-linear in $\boldsymbol{\theta}$ and $\boldsymbol{z}$. We are going to show how to learn parameters $(\boldsymbol{w}, w_0)$ and implicitly also $\boldsymbol{\theta}$ via a convex optimization.

Assume we are given a training set of examples $\{(\boldsymbol{z}^1, y^1), \ldots, (\boldsymbol{z}^m, y^m)\} \in (\mathbb{R}^{n \times m} \times \{+1, -1\})^m$. We fix the representation $\boldsymbol{\phi}$ such that the number of bins b is sufficiently large and the bin edges $\boldsymbol{\theta}$ are equally spaced. We find the weights $(\boldsymbol{w}, w_0)$ by solving

$$\min_{\boldsymbol{w} \in \mathbb{R}^{b \cdot p}, w_0 \in \mathbb{R}} \left[\gamma \sum_{i=1}^{n} \sum_{j=1}^{b-1} |w_{i,j} - w_{i,j+1}| \; + \frac{1}{m} \sum_{i=1}^{m} \max\left\{0, 1 - y^i \langle \boldsymbol{\phi}(\boldsymbol{z}^i; \boldsymbol{\theta}), \boldsymbol{w} \rangle \right\} \right] . \tag{4.7}$$

The objective is a sum of two convex terms. The second term is the standard hinge-loss surrogate of the training classification error. The first term is a regularization encouraging weights of neighboring bins to be similar. If it happens that j-th and $j+1$ bin of the i-the histogram have the same weight, $w_{i,j} = w_{i,j+1} = w$, then these bins can be effectively merged to a single bin because

$$w_{i,j}\phi(\boldsymbol{z}_i;\theta_{i,j-1},\theta_{i,j}) + w_{i,j+1}\phi(\boldsymbol{z}_i;\theta_{i,j},\theta_{i,j+1}) = 2w\phi(\boldsymbol{z}_i;\theta_{i,j-1},\theta_{i,j+1})\,. \tag{4.8}$$

The trade-off constant $\gamma > 0$ can be used to control the number of merged bins. A large value of γ will result in massive merging and consequently in a small number of resulting bins. Hence the objective of the problem (4.7) is to minimize the training error and to simultaneously control the number of resulting bins. The number of bins influences the expressive power of the classifier and thus also the generalization of the classifier. The optimal setting of λ is found by tuning its value on a validation set.

Once the problem (4.7) is solved, we use the resulting weights $\boldsymbol{w}^*$ to construct a new set of bin edges $\boldsymbol{\theta}^*$ such that we merge the original bins if the neighboring weights have the same sign (i.e. if $w^*_{i,j}w^*_{i,j+1} > 0$). This implies that the new bin edges $\boldsymbol{\theta}^*$ are a subset of the original bin edges $\boldsymbol{\theta}$, however, their number can be significantly reduced (depending on γ) and they have different widths unlike the original bins. Having the new bins defined, we learn a new set of weights by the standard SVM algorithm

$$\min_{\boldsymbol{w}\in\mathbb{R}^n, w_0\in\mathbb{R}} \left[\frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m}\max\left\{0, 1 - y^i\langle\boldsymbol{\phi}(\boldsymbol{z}^i;\boldsymbol{\theta}^*), \boldsymbol{w}\rangle\right\}\right]\,.$$

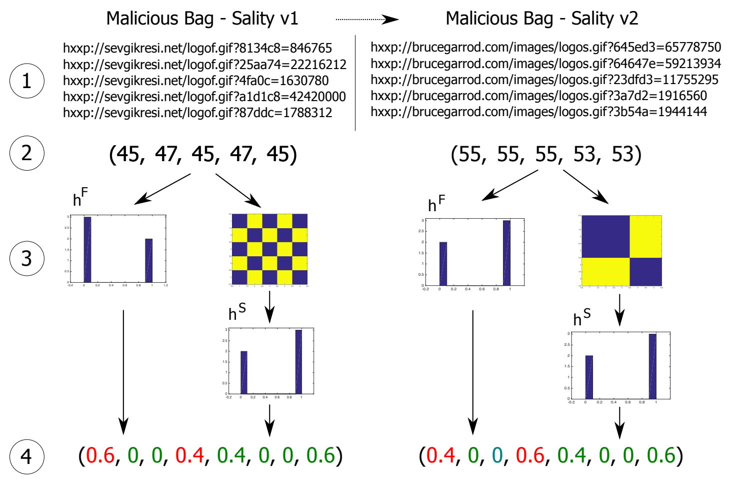
Note that we could add the quadratic regularizer $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$ to the objective of (4.7) and learn the weights and the representation in a single stage. However, this would require tuning two regularization parameters (λ and γ) simultaneously which would be order of magnitude more expensive than tuning them separately in the two stage approach.

The proposed method is based on similar principals as we introduced in Chapter 3 describing the adaptive sampling. In this case we do not remove redundant flows but merge bins with similar information value, which results in simpler representation, smoother decision boundary, and more robust classifier.

4.4 Malware Representation Example

This section illustrates how the proposed representation (nonoptimized version) is calculated for two real-world examples of malicious behavior. Namely, two versions of a polymorphic malware Sality are compared. Sality [61] is a malware family that has become a dynamic and complex form of malicious infection. It utilizes polymorphic techniques to infect files of Widows machines. Signature-based systems or classifiers trained on a specific malware type often struggles with detecting new variants of this kind of malware. Naturally, most of the conclusions to the discussion that follows can be drawn for many other malware threats.

Figure 4.3 shows how the two Sality samples are represented with the proposed approach. First, the input flows are grouped into two bags (one bag for each Sality sample), because all flows of each bag have the same user and the same hostname (1). For the sake of simplicity, only URLs of the corresponding flows are displayed. Next, 115 flow-based feature vectors are



Malicious Bag - Sality v1
Malicious Bag - Sality v2
hxxp://sevgikresi.net/logof.gif?8134c8=846765
hxxp://sevgikresi.net/logof.gif?25aa74=22216212
hxxp://sevgikresi.net/logof.gif?4fa0c=1630780
hxxp://sevgikresi.net/logof.gif?a1d1c8=42420000
hxxp://sevgikresi.net/logof.gif?87ddc=1788312
hxxp://brucegarrod.com/images/logos.gif?645ed3=65778750
hxxp://brucegarrod.com/images/logos.gif?64647e=59213934
hxxp://brucegarrod.com/images/logos.gif?23dfd3=11755295
hxxp://brucegarrod.com/images/logos.gif?3a7d2=1916560
hxxp://brucegarrod.com/images/logos.gif?3b54a=1944144
1
2
3
4
(45, 47, 45, 47, 45)
(55, 55, 55, 53, 53)
h^F
h^S
(0.6, 0, 0, 0.4, 0.4, 0, 0, 0.6)
(0.4, 0, 0, 0.6, 0.4, 0, 0, 0.6)

Fig. 4.3: Illustration of the proposed representation applied on two versions of malware Sality. First, two bags of flows are created (1), one bag for each Sality sample. Next, flow-based feature vectors are created for each bag (2). For illustrative purposes, only a single feature is used - URL length. In the third step, histograms of feature values $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ and feature differences $\boldsymbol{\phi}(\boldsymbol{z}_k^S, \boldsymbol{\theta}_k^S)$ are created (3) as described in Section 4.2.3. Only four bins for each histogram were used. Finally, all histograms are concatenated into the final feature vector (4). Even though the malware samples are from two different versions, they have the same histogram of feature differences $\boldsymbol{\phi}(\boldsymbol{z}_k^S, \boldsymbol{\theta}_k^S)$. Since $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ is not invariant against shift, you can see that half of the values of $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ are different. Still, $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ values may play an important role when separating malware samples from other legitimate traffic.

computed for each bag (2). To simplify illustration, we show only a single feature – URL length. After this step, each Sality sample is represented with one feature vector of flow-based values. Existing approaches use these vectors as the input for the subsequent detection methods. As we will show in Section 4.5, these feature values are highly variable for malware categories. Classification models trained with such feature values loose generalization capability.

To enhance the robustness of the flow-based features, the proposed approach computes histograms of feature values $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ and feature differences $\boldsymbol{\phi}(\boldsymbol{z}_k^S, \boldsymbol{\theta}_k^S)$ (3) as described in Section 4.2.3. To make the illustration simple, only four bins for each histogram were used. Finally, all histograms are concatenated into the final feature vector (4). It can be seen that even though the malware samples are from two different versions, they have the same histogram of feature differences $\boldsymbol{\phi}(\boldsymbol{z}_k^S, \boldsymbol{\theta}_k^S)$. Since the histogram of feature values $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ is not invariant against shift, half of the values of $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ are different.

The number of histogram bins and their sizes are then learned from the data by the proposed algorithm (see Section 4.3). The proposed representation describes inner dynamics of flows from each bag, which is a robust indicator of malware samples, as we will show in the analysis of various malware families in Section 4.6. In contrast to the existing methods that use flow-based features or general statistics such as mean or standard deviation, the proposed representation reflects properties that are much more difficult for an attacker to evade detection.

4.5 Evasion Possibilities

This section discusses evasion options for an attacker when trying to evade a learning-based classification system. According to the recent work [127], the essential components for an evasion are: (1) the set of features used by the classifier, (2) the training dataset used for training, (3) the classification algorithm with its parameters. Without the knowledge of the features, the attacker is faced with major challenges and there is not any known technique for addressing them [127].

Acquire knowledge of classification algorithm with its parameters or the training data is hard if not impossible. Therefore, in the following analysis, we assume that only the features are known to the attacker. When classifying HTTP traffic from proxy logs, it is actually not difficult to create a set of common features widely used in practice. These features are the baseline flow-based features, such as those described in Table 4.1. When the attacker performs a mimicry attack, selected features of malicious flows are modified to mimic legitimate traffic (or flows marked as benign by the classifier).

In the following, we will analyze the case when the attacker performs a mimicry attack to evade detection by modifying flow attributes, such as URLs, bytes, and inter-arrival times. Other flow attributes can be altered in a similar way with analogical results. All modifications are divided into two groups, depending on whether the proposed representation is invariant against them.

The proposed representation is invariant to the following changes.

- **Malicious code, payload, or obfuscation** – The advantage of all network-based security approaches is that they extract features from headers of network communication rather than from the content. As a result, any changes to the payload including the usage of pluggable transports designed to bypass Deep Packet Inspection (DPI) devices will have no effect on the features. Some pluggable transports (e.g. ScrambleSuit) are able to change its network fingerprint (packet length distribution, number of bytes, inter-arrival times, etc.). Since the proposed representation mainly relies on the dynamics of URLs of flows in the bag, such changes will not negatively impact the efficacy, which is a great advantage against DPI devices.
- **Server or hostname** – The representation operates at the level of bags, where each bag is a set of flows with the same user and hostname/domain. If an attacker changes an IP address or a hostname of the remote server (because the current one has been blacklisted), the representation will create a new bag with similar feature values as in the previous bag with the original IP address or hostname, which is a great advantage against feeds and blacklists that need to be updated daily and are always behind.
- **URL path or filename** – Straightforward and easy way of evading existing classifiers using flow-based features or URL patterns is the change in path or filename from sample to sample. Since the variability of these features remains constant within each bag, these changes will also have no effect on the proposed representation.
- **Number of URL parameters, their names or values** – This is an alternative to URL path changes.
- **Encoded URL content** – Hiding information in the URL string represents another way to exfiltrate sensitive data. When the URL is encrypted and encoded (e.g. with base64), it changes the URL length and may globally influence other features as well. As the proposed

representation is invariant against shifting, changing the URL length will not change the histograms of feature differences.

- **Number of flows** – Another option for an attacker to hide in the background traffic is increasing or reducing the number of flows related to the attack. Such modification of the attack does not affect the representation, as long as there are enough flows to create the feature vectors.
- **Time intervals between flows** – This feature has been used in many previous approaches for its descriptive properties. It is an alternative way to the proposed representation how to model a relationship between individual flows. Our analysis revealed that current malware samples frequently modify the inter-arrival time to remain hidden in the background traffic – see Figure 4.4 for details. Therefore, we do not rely on this unstable feature that can be also influenced by network delays or failures.
- **Ordering of flows** – An attacker can easily change the ordering of flows to evade detection based on patterns or predefined sequences of flows. For the proposed representation the ordering of flows does not matter.

The proposed representation is not invariant to the following changes.

- **Static behavior** – The representation is not effective on malware behaviors, where all flows associated with a malware are identical. Such behavior has no dynamics and can be classified with flow-based approaches with comparable results. In our dataset, only 10% of flows were removed because of this constrain.
- **Multiple behaviors in a bag** – In case more behaviors are associated with a bag, such as when a target hostname is compromised and communicates with a user with legitimate and malicious flows at once, the representation does not guarantee the invariance against the attacker's changes. Such bags contain a mixture of legitimate and malicious flows and their combination could lead to a different representation. Note that there wasn't any malware sample in our data that would satisfy this condition, since the legitimate traffic has to be authentic (not artificially injected) to confuse the representation.
- **Encrypted HTTPS traffic** – Most features presented in this paper are computed from URLs or other flow fields, that are not available in encrypted HTTPS traffic. In this case, only a limited set of flow-based features can be used, which reduces the discriminative properties of the representation. However, majority of malware communication is still over HTTP protocol, because switching to HTTPS would harm the cyber-criminals' revenues due to problems with signed certificates [79].
- **Real-time changes and evolution** – In case a malware sample for a given user and hostname would start changing its behavior dynamically and frequently, the bag representation will vary in time. Such inconsistency would decrease the efficacy results and enlarge the time to detect. However, creating such highly dynamic malware behavior requires a considerable effort, therefore we do not see such samples very often in the real network traffic.

We conclude our analysis with the observation, that attackers change flow features very frequently (see Figure 4.4). An example of such change is illustrated in Figure 4.3, where the next version of Sality malware changed second-level domain, URL path, filename, and URL prameters. The goal of the proposed representation is to be invariant against most of the changes to successfully detect new, previously unseen malware variants.

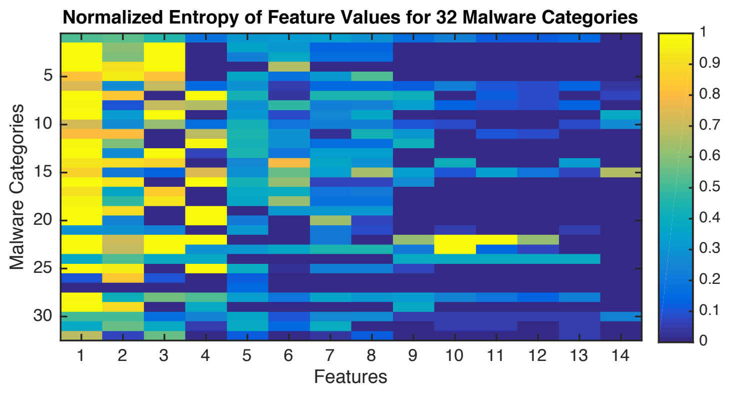


Fig. 4.4: Flow-based features (columns) are changing for most of the malware categories (rows). The figure uses normalized entropy to show the variability of each feature within each malware category. Yellow color denotes that the feature value is changed very often, while blue color means that the feature has the same values for all samples of the given category. **Features:** 1-URL, 2-interarrival time, 3-URL query values, 4-URL path, 5-number of flows, 6-number of downloaded bytes, 7-server IP address, 8-hostname, 9-URL path length, 10-URL query names, 11-filename, 12-filename length, 13-number of URL query parameters, 14-number of uploaded bytes. **Malware categories:** 1-Click-fraud (amz), 2-Asterope family 1, 3-Asterope family 2, 4-Beden, 5-Click-fraud, 6-DGA, 7-Dridex, 8-Exfiltration, 9-InstallCore, 10-Mudrop Trojan Dropper, 11-Monetization, 12-Zeus, 13-Mudrop, 14-MultiPlug, 15-mixture of unknown malware, 16-Click-fraud (tracking), 17-Poweliks family 1, 18-Poweliks family 2, 19-Qakbot Trojan, 20-Rerdom Trojan, 21-Ramnit worm, 22-RVX, 23-Sality, 24-Threats related to a traffic direction system (TDS) 1, 25-TDS 2, 26-TDS 3, 27-Tinba Trojan, 28-C&C tunneling, 29-Upatre, 30-Vawtrak, 31-Vittalia, 32-Zbot. Details about the malware categories are given in Section 4.6.

4.6 Experimental Evaluation

The proposed representation was applied to classify unseen malware bags. Next section provides the specification of datasets and malware categories, followed by the results from the experimental evaluation. We will show that an SVM classifier achieves significantly better efficacy results using the proposed representation when compared to the baseline flow-based representation that is used in previously published work.

4.6.1 Specification of the Datasets

This section provides a detailed description of the datasets, labeled malware samples, and features used in the experimental evaluation. Malware samples were obtained from several months (January - June 2015) of real network traffic of 80 international companies in form of proxy logs [99]. The logs contain HTTP and HTTPS flows, where one flow is one connection defined as a group of packets from a single host and source port with a single server IP address, port, and protocol. As flows from the proxy logs are bidirectional, both directions of a communication are included in each flow. The malware samples were obtained from findings of several network security devices based on signatures (Cisco Cloud Web Security), blacklists (Shadowserver[1]),

[1] www.shadowserver.org

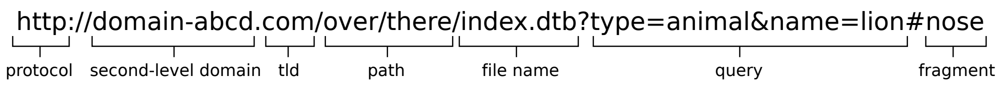

http://domain-abcd.com/over/there/index.dtb?type=animal&name=lion#nose
protocol
second-level domain
tld
path
file name
query
fragment

Fig. 4.5: URL decomposition into seven parts.

feeds (Collective Intelligence Framework CIF [62]), and static and behavioral analysis (Cisco Cloud Web Security). In many cases, human experts were involved to label novel previously undetected threats. Findings from VirusTotal[2] server were also used.

In the following, malware samples will be referred as **positive bags**, where one positive bag is a set of flows (connections) from proxy records with the same source host towards the same destination hostname. In other words, each bag contains the whole client-hostname communication for a given period of time (e.g. 1 hour).The bags that are not labeled as malicious are considered as legitimate/negative. We assume that the number of unknown malicious bags labeled as negatives is negligible when compared to the number of correctly labeled negative bags. Each bag should contain at least 5 flows to be able to compute a meaningful histogram representation from the input flows.

Each flow consists of the following fields: user name, source IP address, destination IP address, source port, destination port, protocol, number of bytes transferred from client to server and from server to client, flow duration, timestamp, user agent, URL, referer, MIME-Type, and HTTP status. The most informative field is the URL, which can be decomposed further into 7 parts as illustrated in Figure 4.5. From the flow fields mentioned above, we extracted 88 flow-based features listed in Table 4.1. Features from the right column are applied on all URL parts, including the URL itself and a referer.

4.6.2 Malicious Samples

More then 32 different malicious categories were found in the evaluation datasets. Note than most of the novel threats (typically not detected by most of existing devices) were found and confirmed manually and are placed into one malicious category called *other categories*. To illustrate the complexity of the classification problem caused by the large amount of malware families and their variability, a brief description of some categories is provided together with proxy log examples (Table B.1 in Appendix B):

- **Asterope** – Threat related to Asterope click-fraud botnet. Asterope is a Trojan bot malware which performs click-fraud by imitating the action of a user clicking on an advertisement. The bot communicates with the command-and-control server using HTTP from which it receives the next website to visit.
- **Click-fraud, malvertising-related botnet** – The main distribution channel for this threat is fraudulent software such as anti-virus, browser plugins, and software updates. The infection typically appears as a browser plugin that hijacks web browsers. It may then establish a command-and-control channel, track user activity, have rootkit capability, and perform click-

[2] www.virustotal.com

Features applied on URL, path, query, filename
length
digit ratio
lower case ratio
upper case ratio
ratio of digits
vowel changes ratio
ratio of a character with max occurrence
has a special character
max length of consonant stream
max length of vowel stream
max length of digit stream
number of non-base64 characters
has repetition of parameters
starts with number

Other Features
number of bytes from client to server
number of bytes from server to client
length of referer
length of file extension
number of parameters in query
number of '/' in path
number of '/' in query
number of '/' in referer
is encrypted

Table 4.1: List of selected flow-based features extracted from proxy logs. We consider these features as baseline (as some features were used in previously published work), and compare the baseline with the proposed representation. More detailed description of features is provided in Appendix A.

fraud through the automatic loading and clicking of unsolicited advertisements. The attacker may obtain information about the infected device and attempt to further exploit the device with additional threats.

- **DGA** – Threat that uses domain generation algorithms (DGA) and Fast-Flux to establish its command-and-control communication. Fast-Flux is a DNS technique used by botnets to hide malicious devices behind a command-and-control infrastructure of compromised hosts. These hosts act as proxies that register and de-register their IP addresses. By using a short Time To Live (TTL) value, the hostname to IP address mapping for devices in the requested domain name space will change rapidly. This results in a constantly changing list of destination IP addresses for a single DNS name and allows the attacker to distribute information about the malicious devices.
- **Dridex** – Threat related to Dridex banking trojan. Dridex is typically spread through spam campaigns and its main goal is to obtain confidential information from the user about his or her online banking and other payment systems. The trojan communicates with the command-and-control server using HTTP, P2P, or I2P protocols.
- **Monetization** – Malware monetization activity involving fake blog sites that serve as front ends for click-fraud.

Category	Samples		Signature-based device	
	Flows	Bags	TPs	Recall
Training Positives	132,756	5,011	736	0.15
Click-fraud malvertising	12,091	819	238	0.29
DGA malware	8,629	397	231	0.58
Dridex	8,402	264	31	0.12
IntallCore	17,317	1,332	0	0.00
Monetization	3,107	135	0	0.00
Mudrop	37,142	701	0	0.00
Poweliks	11,648	132	0	0.00
Zeus	34,420	1,275	236	0.19
Testing Positives	43,380	2,090	48	0.02
Training Negatives	862,478	26,825		
Testing Negatives	15,379,466	240,549		

Table 4.2: Number of flows and bags of malware categories and legitimate background traffic used for training and testing the proposed representation and classifier. Right-most column shows the amount of bags that were found and blocked by an existing signature-based device. Majority of the malicious bags from the test were missed, as the device, relying on a static database of signatures, was not able to catch evolving versions and new types of the malicious behaviors.

- **Poweliks** – Threat related to the Poweliks Trojan which downloads other malware to the infected device and can steal information. The threat is persistent and uses several mechanisms to hide itself.
- **Zeus** – Threat related to the Zeus Trojan horse malware family which is persistent, may have rootkit capability to hide its presence, and employs various command-and-control mechanisms. Zeus malware is often used to track user activity and steal information by man-in-the-browser keystroke logging and form grabbing. Zeus malware can also be used to install CryptoLocker ransomware to steal user data and hold data hostage.
- **Others categories** – Other categories include Bedep, InstallCore, Mudrop, MultiPlug, Ramdo, Rerdom, Sality, Vawtrack, Vittalia, etc.

Table B.1 from Appendix B describes an important fact about the distributions of feature values computed from individual malicious bags. As you can see, URLs within each malicious bag are similar to each other (as opposed to most of legitimate bags). This small non-zero variability of flow-based feature values is captured by the proposed representation using both types of histograms. The variability is very general but also descriptive feature, which increases the robustness of the representation to further malware changes and variants.

The summary of the training and testing set is shown in Table 4.2. Positive bags from 8 categories with the highest number of bags were added to the training set, while the rest of the malware samples from the other categories (including novel threats) were included in the testing set. This means that training and testing data are composed of completely different malware bags from different malware families. This makes the classification problem much harder, as the classifier is trained on 8 malware categories and then evaluated on malicious traffic of different categories and behaviors unseen in the training set. This scenario simulates the fact that new types of threats are created to evade detection. The benchmarking signature-based network security device (widely used in many companies) was able to detect only 2% of the malicious bags from the testing set. Anomaly detection system introduced in [93] automatically removes 38% of all malicious flows from the testing set, as they have empty URL query string. Training a classifier for each category separately is an easier task, however such classifiers are typically over-fitted to a single category and cannot detect further variations without retraining. On the other hand, training a classifier from multiple categories is more suitable for dynamic or polymorphic changes of new malware.

Negative bags were acquired from 10 hours of http network traffic of two companies. Negative bags from the first company were used for training, while bags from the second company were used for testing. Note that malicious bags found in the traffic were removed from the set of negative bags.

4.6.3 Evaluation on Real Network Traffic

This section shows the benefits of the proposed approach of learning the invariant representation for two-class classification problem in network security. Feature vectors described in Section 4.6.1 correspond to input feature vectors $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ defined in Section 4.1. These vectors are transformed into the proposed representation of histograms $\boldsymbol{\phi}(\tilde{X}; \tilde{\mathcal{S}}; \boldsymbol{\theta})$, as described in Section 4.2. We have evaluated two types of invariant representations. One with predefined number of equidistant bins (e.g. 16, 32, etc.) computed as described in Section 4.2, and one when the representation is learned together with the classifier to maximize the separability between malicious and legitimate traffic (combination of Section 4.2 and 4.3). For the representation learning, we used 256 bins as initial (and most detailed) partitioning of the histograms. During the learning phase, the bins were merged together, creating 12.7 bins per histogram on average.

Both approaches are compared with the baseline flow-based representation used in previously published work, where each sample corresponds to a feature vector computed from one flow. Results of a widely used signature-based security device are also provided (see Table 4.2) to demonstrate that the positive samples included in the evaluation pose a real security risk, as majority of them was not detected. Maximum number of flows for each bag was 100, which ensures that the computational cost is controlled and does not exceed predefined limits.

Ten flow-based feature vectors of two legitimate and five malicious bags are displayed in Figure 4.6. Each row represents one flow-based feature vector. You can see that legitimate bags have higher diversity of flow-based feature values, which is a result of higher diversity of flows within a bag (illustrated in Table B.1 in Appendix B). This diversity (also shown in Figure 4.4) makes it difficult for a flow-based classifier to learn more complex malicious behaviors, as they are not well separated from the legitimate traffic. On the other hand, feature values within malicious bags are more consistent, resulting in more bars with uniform color. This key property, which is shared across malware categories, is not visible from the flow-based features point of view. It is visible only at the level of bags.

Figure 4.7 shows how the same positive and negative samples look in the proposed representation. Zero values are depicted with dark blue color, while maximum values (equal to 1) are depicted with yellow bars. Instead of ten flow-based feature vectors, each bag is represented with a single vector describing the inner dynamics of flow-based feature values within each bag. Malicious bags have a lot of values equal to zero as opposed to legitimate bags, which increases

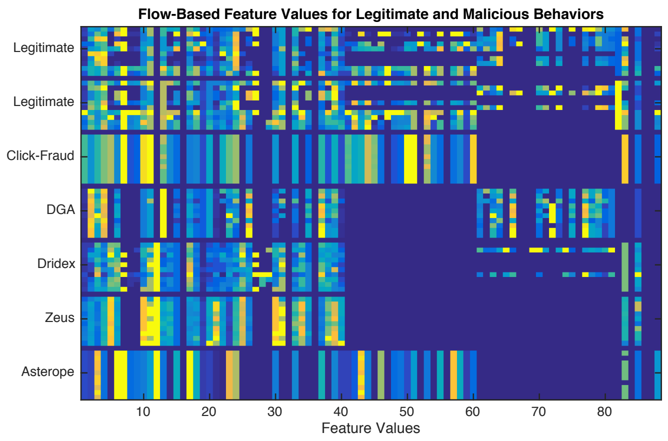


Fig. 4.6: Graphical illustration of two legitimate and five malicious bags in the baseline (flow-based) representation. Each bag is represented with 10 flow-based feature vectors (rows). You can see that feature values of legitimate vectors have high range of values, which complicates the training of classifiers. They have also significantly higher variability of feature values than malicious bags, which cannot be utilized from flow-based representation.

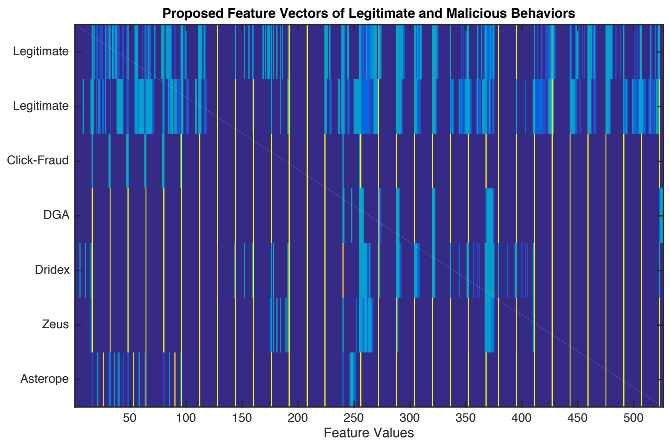


Fig. 4.7: Illustration of two legitimate and five malicious bags in the proposed invariant representation. First two legitimate bags show high variability of feature values and can be easily separated from malicious bags. On the other hand, feature values of malicious bags describe the inner similarity of flows or URLs within each bag as illustrated in Table B.1 in Appendix B. In contrast to legitimate bags, malicious bags have a lot of common feature values equal to 0 and 1, which improves their separability.

the separability of the two classes. Moreover, feature values equal to one are common for most of malicious bags across categories, which increases the descriptive value and robustness of the proposed representation. Such representation is more suitable for classification of frequently-changing malicious behaviors, as will be demonstrated further in this section.

Two-dimensional projection of the feature vectors for the flow-based and the proposed representation is illustrated in Figures 4.8 and 4.9 respectively. Bags from 32 malicious categories are displayed with red circles, while the legitimate bags are denoted with green circles. The projections show that the flow-based representation is suitable for training classifiers specialized on a single malware category. In case of the proposed representation, malicious bags from var-

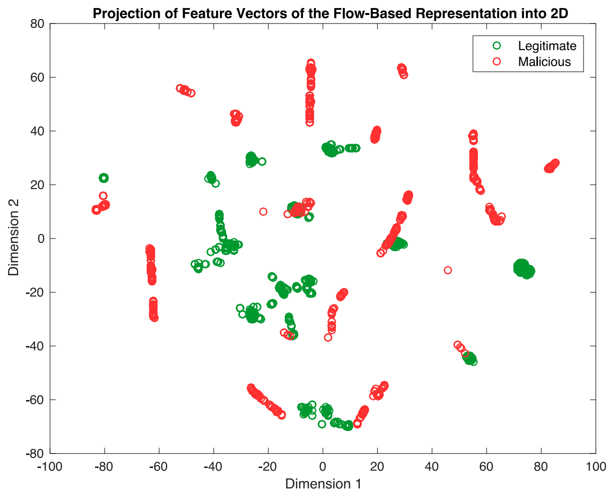


Fig. 4.8: Graphical projection of feature vectors of the baseline flow-based representation (some vectors are in Figure 4.6) into two dimensions using t-SNE transformation. Feature vectors from 32 different malware categories are displayed. Due to high variability of flow-based feature values, legitimate and malicious samples are scattered without any clear separation. The results show that the flow-based representation is suitable for training classifiers specialized on a single malware category, which often leads to classifiers with high precision and low recall.

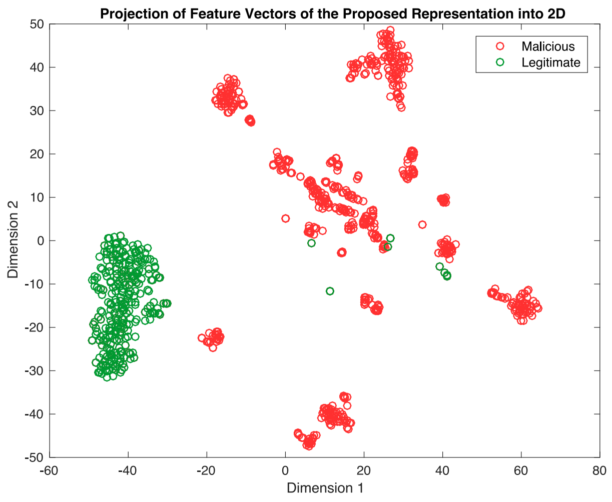


Fig. 4.9: Graphical projection of feature vectors of the proposed representation (some vectors are in Figure 4.7) into two dimensions using t-SNE transformation. Thanks to the invariant properties, malicious bags from various categories are grouped together, as they have similar dynamics modeled by the representation. Most of the legitimate bags are concentrated on the left-hand side, far from the malicious bags. This shows that training a classifier with the proposed representation will achieve higher recall with comparable precision.

ious categories are grouped together and far from the legitimate traffic, which means that the classifiers will have higher recall and comparable precision with the flow-based classifiers.

Next, we will show the properties of the proposed method of learning the representation to maximize the separation between positive and negative samples (see Section 4.3 for details). Figure 4.10 visualizes the proposed method on synthetic 2-dimensional input data. The input

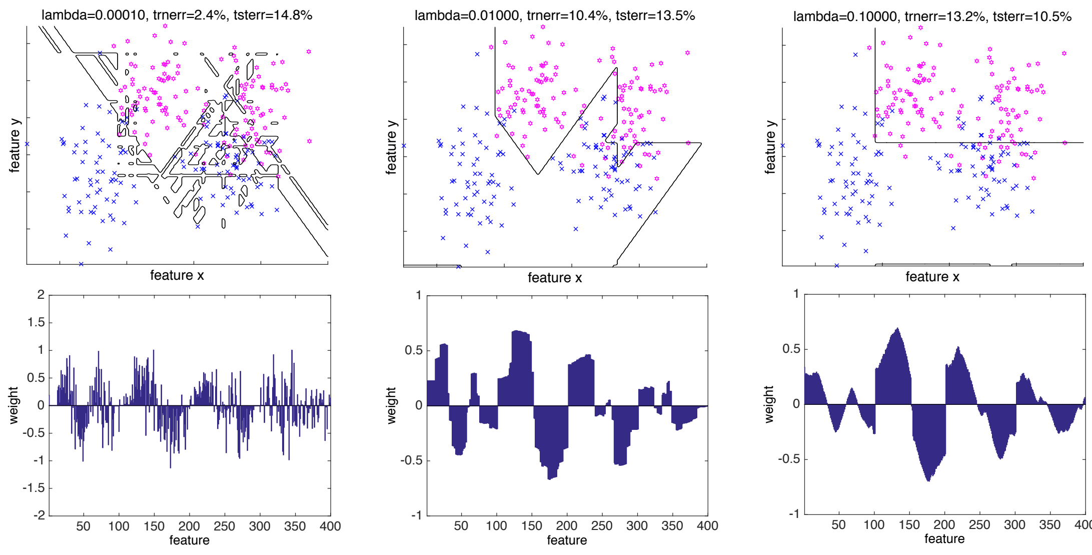


Fig. 4.10: Visualization of the proposed method of learning the invariant representation on 2-dimensional synthetic data. Figures in the upper row show the decision boundaries of two class classifier learned from the bins for three different values of parameter λ (0.0001, 0.01, 0.1) which controls the number of emerging bins (the corresponding weights are shown in the bottom row). With increasing λ the data are represented with less bins and the boundary becomes smoother and less over-fitted to the training data.

2D point $(x, y) \in \mathbb{R}^2$ is represented by 4-dimensional feature vector $(x^2, y^2, x+y, x-y)$. Each of the 4 features is then represented by a histogram with 100 bins (i.e. each feature is represented by 100 dimensional binary vector will all zeros but a single one corresponding to the active bin). Figures in the top row show the decision boundaries of two-class classifiers learned from data. The bottom row shows the weights of the linear classifier corresponding to the bins (in total 400 weights resulting from 100 bins for each out of 4 features). The columns correspond to the results obtained for different setting of the parameter λ which controls the number of emerging bins and thus also the complexity of the decision boundary. With increasing λ the data are represented with less bins and the boundary becomes smoother. Figure 4.10 shows the principle of the proposed optimization process. The bins of the representation are learned in such a way that it is much easier for the classifier to separate negative and positive samples and at the same time control the complexity of the classifier.

Figures 4.11 and 4.12 show the bins and weights learned from the training set of real network traffic. The blue vertical lines represent learned weights associated with 256 bins of a histogram computed on a single input feature. The red lines show new bins derived from the weights by merging those neighboring bins which have the weights with the same sign. Figure 4.11 shows the weights and the derived bins for a standard SVM which has no incentive to have similar weights. The histogram derived from the SVM weights reduces the number of bins from 256 to 130. Figure 4.12 shows the results for the proposed method which enforces the similar weights for neighboring bins. In this case, the weights exhibit a clear structure and the derived histogram has only 18 bins. The decision boundary is in this case smoother and the classifier trained from this representation will be more robust.

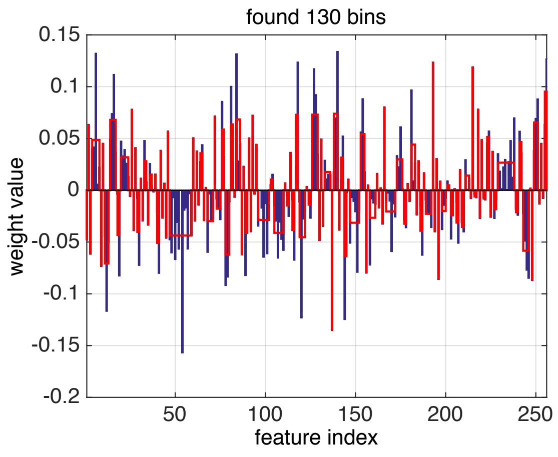


Fig. 4.11: Weights (blue bars) and derived bins of a histogram (red line) for a standard SVM and one of the invariant features. Since the bins are equidistant and predefined at the beginning, the resulting histogram (defined by the red line) has complicated structure, leading most probably to complex boundary and over-fitted results (as shown in Figure 4.10 on the left hand side).

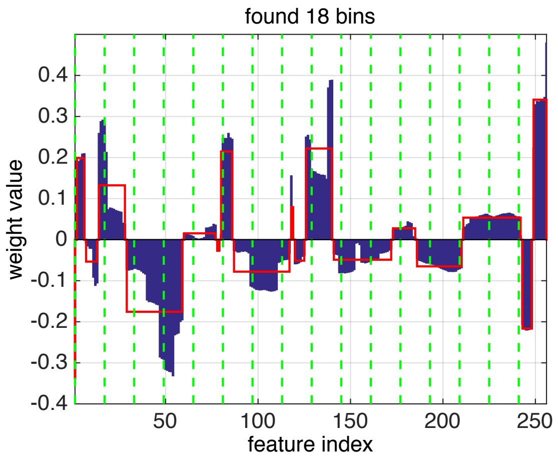


Fig. 4.12: Weights (blue bars) and derived bins of a histogram (red line) for the proposed bin optimization. In this case, the weights show a clear structure and the derived histogram has only 18 bins. The decision boundary is in this case smoother and the classifier trained from this representation will be more robust. Green dashed lines also show how the histogram bins would look like if they are positioned equidistantly (16 bins).

Next, a two-class SVM classifier was evaluated on five representations: baseline flow-based, per-feature histograms of values $\boldsymbol{\phi}(\boldsymbol{z}_k^X, \boldsymbol{\theta}_k^X)$ (bag mean), per-feature histograms of feature differences $\boldsymbol{\phi}(\boldsymbol{z}_k^S, \boldsymbol{\theta}_k^S)$ (bag variance), the combination of both (bag combined), and the combination of both with bin optimization (optimized bag combined). The training and testing datasets were composed of bags described in Table 4.2. ROC curves on training data are depicted in Figure 4.13, where five types of representations are compared. In contrast with the baseline approach, the classification model trained from the combined bag representation is accurate even for higher true positive rate values. The figure shows that flow-based classifier achieved high-precision results with only very limited recall (approx. 7%). This means that the classifier was specialized on these flows at the expense of the rest of the malicious samples. It also means

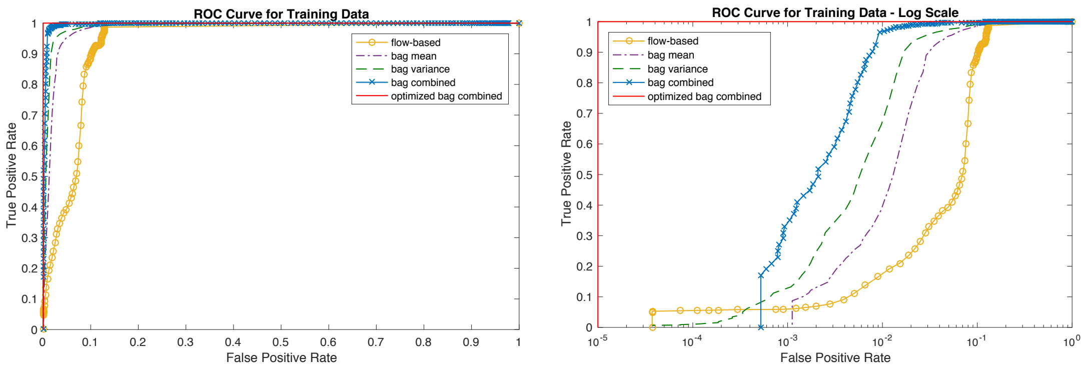


Fig. 4.13: ROC curves in linear (left) and logarithmic (right) scale of SVM classifier on training data for five types of representations: baseline flow-based, per-feature histograms of values $\mathbf{h}^F$ (bag mean), per-feature histograms of feature differences $\mathbf{h}^S$ (bag variance), the proposed combination of both (bag combined), and bag combined with optimization of parameters (optimized bag combined). Flow-based classifier detected small amount of samples (approx. 7%) with very low false positive rate. This means that the classifier was specialized on these flows at the expense of the rest of the malicious samples. On the other hand, the proposed combination of both bag approaches resulted in high recall with acceptable low false positive rate.

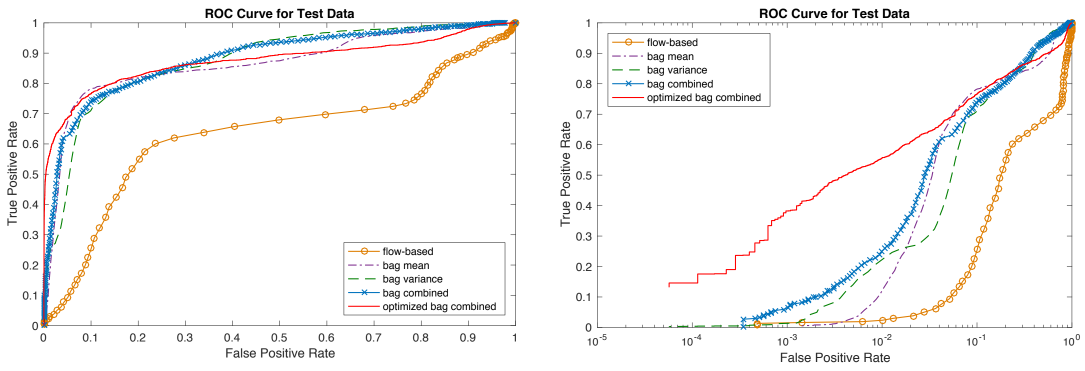


Fig. 4.14: ROC curves (linear and logarithmic) of SVM classifier on test data for five types of representations (logarithmic scale on the right). Flow-based representation shows very unsatisfactory results showing that flow-based approach cannot be applied in practice to detect unseen malware variants. The combination of feature values with feature differences histogram (bag combined) led to significantly better efficacy results. These results were further exceeded when the parameters of the invariant representation were learned automatically from the training data (optimized bag combined).

that flow-based features are good for training specialized classifiers designed for one specific behavior. On the contrary, the proposed combination of both bag representations resulted in significantly higher recall with low false positive rate.

The results on testing data are depicted in Figure 4.14. Note that positive bags in the testing set are from different malware categories than bags from the training set, which makes the classification problem much harder. The purpose of this evaluation is to compare flow-based representation, which is used in most of previously published work, with the proposed invariant representation. Flow-based representation shows very unsatisfactory results, mainly due to the fact that the classifier was based only on the values of flow-based features that are not robust

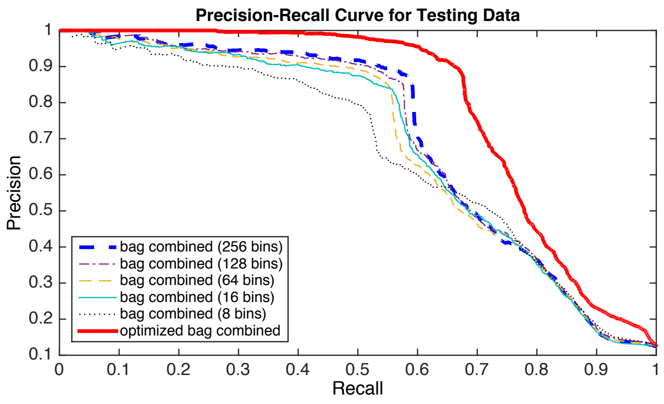


Fig. 4.15: Precision-recall curve of SVM classifier trained on the proposed representation with different number of histogram bins for each feature. All classifiers are outperformed by the classifier, where the parameters of the invariant representation are learned automatically from the data (optimized bag combined). The classifier achieved 90% precision (9 of 10 alerts were malicious) and 67% recall on previously unseen malware families.

across different malware categories (as shown in Section 4.5). The classifier based on combined bag representation performed significantly better. These results were further exceeded when the parameters of the invariant representation were learned automatically from the training data (optimized bag combined), which is shown in Figure 4.14 with logarithmic scale.

Precision-recall curve is depicted in Figure 4.15 to compare the efficacy results of classifiers based on the proposed representation with predefined number of bins per feature (8, 16, 64, 128, and 256 bins) with the same representation, but when the parameters are learned from the training data (using bin optimization from Section 4.3).

Overall, the results show the importance of combining both types of histograms introduced in Section 4.2 together, allowing the representation to be more descriptive and precise without sacrificing recall. But most importantly, when the parameters of the representation are trained to maximize the separability between malicious and legitimate samples, the resulting classifier performs in order of a magnitude better than a classifier with manually predefined parameters.

4.7 Summary

This chapter proposes a robust representation suitable for classifying evolving malware behaviors. It represents sets of network flows as bags based on the combination of invariant histograms of feature values and feature differences. The representation is designed to be invariant under shifting and scaling of the feature values and under permutation and size changes of the bags. Formally, the representation solves the domain adaptation problem in supervised learning setting, where the training and testing datasets have different probability distributions. For the network security domain, it means that the proposed approach allows the classifiers to create robust models of malicious behaviors capable of detecting previously unseen malware variants and changes.

The proposed representation was evaluated on real HTTP network traffic with more than 43k malicious samples and more than 15M samples overall. The comparison with a baseline flow-based approach and a widely-used signature-based web security device showed several key advantages of the proposed representation. First, the invariant properties of the representation result in the detection of new types of malware at a low false positive rate. Second, multiple malware behaviors can be represented in the same feature space while flow-based features necessitate training a separate detector for each malware family. Third, the combination of precise signature-based devices with data-driven classifiers trained with the proposed representation significantly increases the network threat coverage. The new detector improves the incident reporting and prioritization by including threats that are new but perhaps not yet confirmed.

The proposed representation further increases the abstraction of network traffic. At this level, the results can be reported to security administrators or can be used to build a collaborative mechanism that will be described in the next two chapters.

Chapter 5
Collaborative Adaptation Model

Security administrators deploy multiple detectors (i.e. systems detecting the presence of a threat or infection) on various locations of the network infrastructure in order to provide the full monitoring coverage, because large variety of attacks are launched from various locations inside and outside the network. The detectors are typically deployed in the network hierarchically to analyze the input data at various levels of detail. For example, one detector is deployed on the network perimeter and monitors the network traffic between LAN and Internet, while other detectors can be placed inside LAN, for example near high-value systems or users.

For this reason, different detectors frequently analyze the same input data multiple times, but their detection methods or the data granularity may by different. When such overlapping data is analyzed by the detectors running in isolation, the network administrator is overwhelmed with lots of duplicate alerts, while other intrusions might be completely missed. Existing alert correlation and fusion methods are designed to remove the duplicates, but do not address the root cause of the problem: the fact that multiple detectors produce redundant alerts. More often than not, the detectors are based on different techniques and alert correlation and merging is not a trivial issue.

We propose to tackle this problem with the collaborative specialization of the detectors, which reduces the alert redundancy and increases the overall sensitivity of a group of homogeneous or heterogeneous detectors. The proposed model is based on the idea of mutual specialization, where each detector specializes its behavior and creates unique alerts. Specialized detectors are designed to be:

- effective – their models are focused only on a subset of attacks therefore generating less false alerts.
- essential – they detect unique attacks that haven't been detected by other detectors.

The specialization is possible only for the detectors that are dynamically reconfigurable, which means they have a predefined set of parameters (e.g. thresholds, rule priorities, or other parameters modifying the inner detection models) that can be adjusted automatically as a part of the local self-adaptation process. The values of such parameters define individual configurations of the detectors. The range of the parameter values is typically predefined, which means that the set of all possible configurations of each detector is also known in advance. The configuration influences the results of a detector, which means that two configurations may lead to different (but individually still acceptable) results. Static detectors with only one configuration can still contribute to the collaboration by providing their results to reconfigurable detectors, forcing them to specialize (i.e. change their configurations) on the types of attacks that are not detected by the static detectors.

The collaboration model assumes that the detectors iteratively share their results with each other. We propose to define a similarity function for each pair of heterogeneous detectors. The function compares the results of the detector with the results obtained from the rest of the collaborating detectors to provide feedback describing how unique results were obtained by the latest configuration of the detector. Due to security and integration reasons, the detectors should not share any details about their internal models or configurations. Such information can be misused by attackers to bypass the detectors with customized attacks and penetrate inside the network, or compromise the detectors and decrease their detection capabilities.

The specialization of the detectors is directed by the proposed game-theoretical solution concept, where multiple detectors seek to optimize a global collective objective though local decision making of each detector. The decision making is realized iteratively at the level of configurations of the inner model of the detector based on the feedbacks computed by the feedback function. The goal is to select such configuration that leads to unique detections, which ensures the specialization. Such local optimization process, distributively controlled from the global perspective, reduces manual setup of the collaborating system, making the deployment of the whole system more cost-effective and straightforward. The proposed collaboration model also makes any penetration or manipulation with the system much more difficult thanks to its inherent unpredictability and robustness, which is ensured by the solution concept of correlated equilibria, where each detector's payoff is influenced by the combination of attackers' strategies and the strategies played by the other detectors.

The problem of heterogeneous collaboration in the network security domain is hard due to the incompatibility between heterogeneous detectors and due to the highly dynamic network environment. We propose to approximate this hard problem with an easier problem of finding functions that are used in the game-theoretical model suitable for such dynamic environments. The key assumption is to keep the collaborative model method-independent, allowing the collaboration of heterogeneous detectors with various detection principles and algorithms (such as combining statistical detectors with pattern-matching IDS). A general scheme of the model between two detectors is illustrated in Figure 5.1. In this model, each detector iteratively processes batches of input data and the results are distributed to other collaborating detectors through the network. Next, each detector computes the payoff describing the uniqueness of the results created with the current configuration and dynamically selects a new configuration according to the proposed game-theoretical solution concept. The new configuration is then used in the next iteration of the detector operation, and the process re-starts. This way, each detector can adequately react to the current state of the network by changing its configuration parameters. The model represents last stage of the proposed general fusion architecture, as depicted in Figure 5.2.

The proposed model has the following assumptions:

- **All-to-All Communication** - We assume all-to-all communication among the detectors to be able to distributively propagate the results to other detectors. Possible communication overhead can be reduced by grouping the results from a single detector into one message that can be periodically sent to other detectors.
- **Reconfiguration Capability** - At least some of the detection systems should be reconfigurable so they can locally adapt their internal detection models to the current state of the

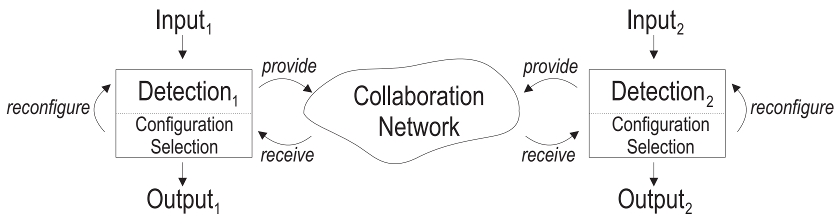


Fig. 5.1: Scheme of the first iteration of the proposed collaboration of multiple heterogeneous detectors.

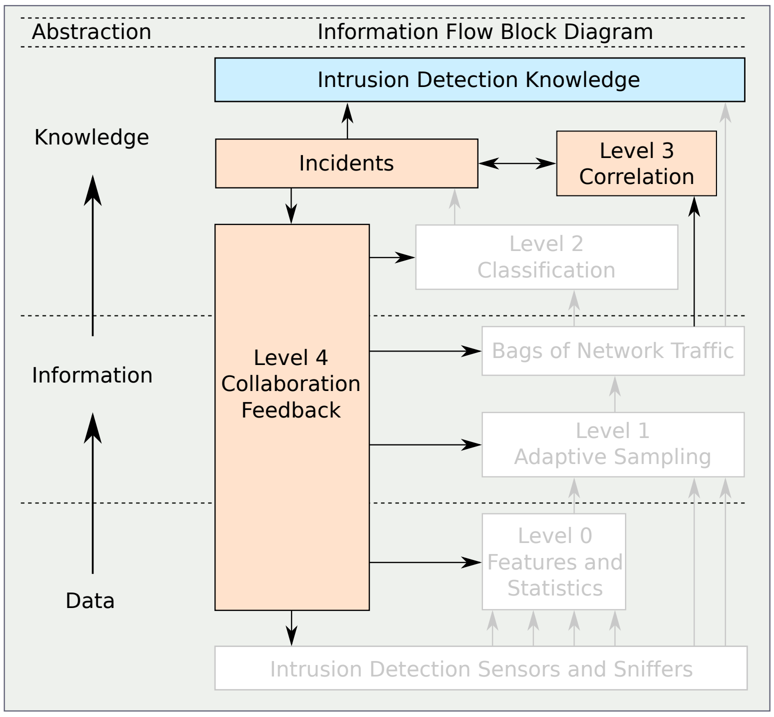

Abstraction
Information Flow Block Diagram
Intrusion Detection Knowledge
Knowledge
Incidents
Level 3 Correlation
Level 2 Classification
Information
Level 4 Collaboration Feedback
Bags of Network Traffic
Level 1 Adaptive Sampling
Level 0 Features and Statistics
Data
Intrusion Detection Sensors and Sniffers

Fig. 5.2: The collaboration components reside at Level 3 of the proposed architecture. First, the incidents from all collaborating systems are collected by the correlation component, followed by the computation of the feedback, which is then propagated to the lower levels of the system.

network traffic according to the proposed collaborative model. An example of such reconfiguration is the change in priorities, aggregation functions, thresholds, sampling rates, or other parameters influencing the system. The set of all possible configurations of each detector is predetermined in advance.

- **No Common Ontology Required** - Results of the detectors can be in various formats (e.g. Intrusion Detection Message Exchange Format [50] or STIX[1]), still the detectors are able to interact with each other. The interoperability is ensured by the functions described in Section 5.3. These functions allow the detectors to interact with each other even if their detection mechanisms are different.
- **Communication Security** - For security reasons, detectors do not provide information about their internal state or inner model, which is their most valuable asset. If this model

[1] https://stixproject.github.io/

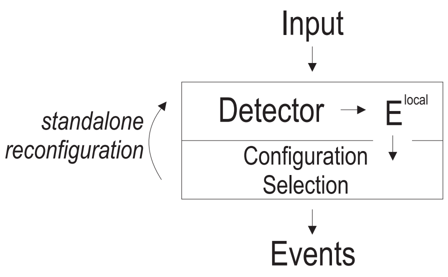


Fig. 5.3: Scheme of a standalone reconfiguration. In each iteration, the detector reuses the results E^{local} for its local (standalone) reconfiguration.

ever gets compromised, the attacker can modify the attack according to the model weaknesses and his actions would remain hidden in the background traffic. For this reason, the only information the detectors provide is the output (e.g. list of detected events) from a single period of time. Authenticated communication channel with confidentiality and integrity guarantees should be used for all communication between the detectors to reduce the possibility of attacker's manipulation.

- **Strategic Deployment -** The performance of the collaboration depends on the position of the detectors within the network infrastructure, or the size of the collaboration. Therefore, strategic deployment of the detectors is important to provide a relevant and useful information to the model. We assume that the position of the detectors in the network is predefined and cannot be dynamically changed, as manipulations with the detectors would cost additional resources and might be undesirable (e.g. against the policy).

The above-mentioned assumptions define initial conditions of the distributed collaboration controlled by the game-theoretical model explained further in Section 5.3.

5.1 Standalone Model

We define an *IDS detector* P_{D_i} as a 3-tuple:

$$P_{D_i} = (DP_i, X_i, r_i), \tag{5.1}$$

where DP_i denotes the detection profile of the detector i: it represents the detection algorithm, type of its input data, and its scope and deployment position in the network. The parameters of the detection profile are outside of control, i.e. predetermined. $X_i = \{x_i^{(1)}, \dots, x_i^{(n)}\}$ denotes a set of all possible configurations of the detection algorithm, and r_i stands for a reconfiguration function that changes the configuration of the detector. In order to keep the notation light, we will normally omit detection profile from the definition of IDS detector and denote i-th IDS detector as $P_{D_i} = (X_i, r_i)$.

The output of i-th detector in a particular time period (e.g. in every 5 minutes) is called *events*:

$$E_i = \{e_i^{(1)}, \dots, e_i^{(n)}\}. \tag{5.2}$$

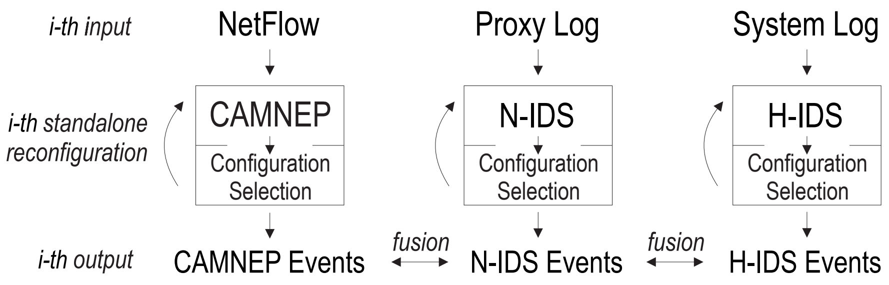


Fig. 5.4: One (i-th) iteration of the standalone reconfiguration of three heterogeneous standalone IDS detectors, where each detector analyzes its input data in isolation.

Events are called *local* if they are created by the detector itself. The events provided to the detector by a different detector are called *remote*. Local and remote events produced by and visible to the detector P_{D_i} will be denoted as $E_i^L \in \mathcal{E}_i$ and $E_i^R \in \mathcal{E}_i$ respectively, where $\mathcal{E}_i$ is a space of all possible sets of events that can be created by P_{D_i}.

Reconfiguration function that depends only on local events is called a *standalone reconfiguration function*:

$$r_i^S : X_i \times \mathcal{E} \rightarrow X_i, \tag{5.3}$$

with the corresponding *standalone IDS detector* $P_{D_i}^S = (X_i, r_i^S)$. In each iteration, the standalone reconfiguration function switches the actual configuration $x_i^{(k)} \in X_i$ of the detector $P_{D_i}^S$ into another configuration $x_i^{(l)} \in X_i$ based only on local events E_i^L. This is also illustrated in Figure 5.3. Note that explicit indication of time or iteration was omitted in our notation to keep it as light and concise as possible. A standalone IDS detector operates in isolation without any collaboration with other detectors. Graphical illustration of three standalone detectors is depicted in Figure 5.4. For example, a host-based IDS (H-IDS) reconfigures the priorities of pattern matching rules for analyzing system logs by using only a standalone reconfiguration function.

The process of selecting a suitable configuration strictly depends on the detection algorithm and the corresponding set of configurations. We define this procedure as a general function r_i^S so it can be used for any IDS detector and its specification depends on the given detector. An example of such reconfiguration is a change in parameters, thresholds, or rule priorities, modification of inner models, adding new patterns or signatures etc. With the standalone reconfiguration function, each detector is able to reconfigure its parameters to locally optimize its performance.

5.2 Collaborative Model

The proposed collaborative model requires at least two detectors P_{D_i} and P_{D_j}, both of them are able to exchange the events and at least one is reconfigurable. A *collaborative IDS detector* $P_{D_i}^C = (X_i, r_i^C)$ uses a *collaborative reconfiguration function*:

$$r_i^C : X_i \times \mathcal{E} \times \mathcal{E} \rightarrow X_i \tag{5.4}$$

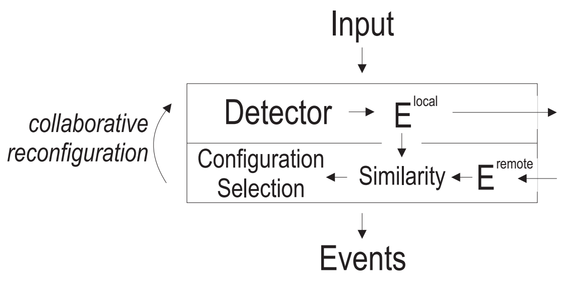


Fig. 5.5: Scheme of a collaborative reconfiguration of one detector. At each time step, the detector creates local events that are further sent to the other detectors within the collaborating group. At the same time, the detector receives remote events and computes a value describing the similarity of local events. Finally, the similarity value is used for selecting the optimal configuration that leads to specialization.

to transition between the individual configuration states. An important part of the reconfiguration is to collect all available remote events, compare them with the local events, and compute a feedback describing the uniqueness of events generated with the last configuration.

The proposed collaborative model within a single network assumes various types of detectors deployed at multiple locations of the network. The collaboration goal is to optimize the performance of each individual detector to increase the efficacy of the overall system. The model aims to provide better efficacy when compared to the case of the same detectors running in isolation.

The principle idea of the collaboration is to optimize the detectors through the specialization on some particular type of intrusions. This should be made possible by the heterogeneity of the background traffic and detector placement. More specialized detectors typically produce significantly lower number of false positives which increases the precision of the detector. The possible decrease in recall of the individual detectors is compensated by the collaboration, where the final recall is calculated from the findings of all detectors together. Since the detectors analyze overlapping network traffic, an attack that is missed by one detector will be more probably detected by a different detector when specialized on different types of attacks than the first one.

To enforce the specialization of each detector, the corresponding reconfiguration function should be able to compare local and remote events. For such comparison, we propose to use a *similarity function*:

$$s_{ij} : E_i \times E_j \to [0,1] \tag{5.5}$$

that computes the degree of similarity of local events with respect to the remote events created by different detectors.

The collaborative reconfiguration function selects the next configuration by using local events, remote events, and mapping the remote events to local events through the prism of the similarity functions. The mapping allows the r_i^C to assess which of its detected events are unique (and therefore have a high value for the global system) and which are duplicates of the events detected by one or more other detectors. The whole collaborative reconfiguration process is illustrated in Figure 5.5, while an illustration of three heterogeneous collaborative IDS detectors is depicted in Figure 5.6.

In each iteration (e.g. every 5 minutes), the collaborative reconfiguration function compares local and remote events created in the same iteration by using predefined similarity functions

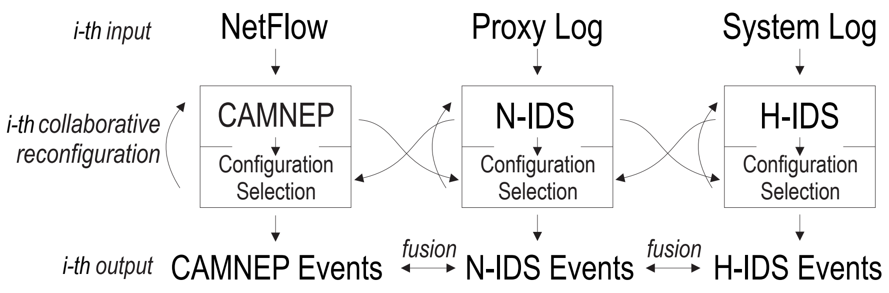


Fig. 5.6: An example of three heterogeneous collaborative IDS detectors. In each iteration, the results of each detector influence the reconfiguration of other detectors, which allows them to react on the current state of the network. Optionally, event fusion can be performed as a final step of the process.

and determines the uniqueness of local events. We propose a mechanism, where the uniqueness enters to the game model to determine a new configuration with the highest specialization.

One of the key problems is the lack of coordination between the detectors that adapt concurrently and mutually influence each other. We argue that the explicit coordination between diverse IDS types would complicate the problem and turn it into a very hard system integration problem, as all the systems would have to be able to share and understand data about the effects of each other's configuration changes. However, it has been shown that a certain class of game-theoretical techniques can help us to reach a robust solutions without explicit coordination ([122]). These will be presented in Section 5.3.

5.3 Game-Theoretical Collaboration Model

In this section we will connect the proposed collaborative model with generally defined game-theoretical problem. We formalize the distributed collaboration as a game between the attackers and a set of defenders represented by IDS detectors. Each player performs certain actions to achieve its predefined goal. An example of an attacker's goal is to exploit secret data from private network. On the other hand, defenders' goal is typically to prevent the attackers from achieving their goals, or secure specific infrastructure or type of service within the network.

More formally, the *defender* system consists of n defenders $P_{D_1}, \ldots, P_{D_n}$ represented by IDS detectors. Each defender is one player of the game with its own set of strategies. Strategies determine the behavior of each player and lead to the predefined goals. That means the strategies correspond to individual configurations X_i of the detector P_{D_i}. We will denote the set of all strategies for i-th defender P_{D_i} as $X_{D_i} = \{x_{D_i}^{(1)}, x_{D_i}^{(2)}, \ldots, x_{D_i}^{(k)}\}$, where $x_{D_i}^{(k)}$ is k-th strategy of player P_{D_i}. Similarly, the set of m *attackers* $P_{A_1}, \ldots, P_{A_m}$ uses strategies $X_{A_1}, \ldots, X_{A_m}$. By using this notation, we define the normal form of a single iteration of a dynamic sequential game with n defenders and m attackers as follows:

$$
\begin{aligned}
G &= (P, X, U, \mathcal{E}_1, \ldots, \mathcal{E}_n) && (5.6)\\
P &= \{P_{D_1}, \ldots, P_{D_n}, P_{A_1}, \ldots, P_{A_m}\} && (5.7)\\
X &= \{X_{D_1}, \ldots, X_{D_n}, X_{A_1}, \ldots, X_{A_m}\} && (5.8)\\
U &= \{u_{D_1}, \ldots, u_{D_n}, u_{A_1}, \ldots, u_{A_m}\} && (5.9)\\
u_{D_i} &: X_{D_i} \times X_{A_1} \times \ldots \times X_{A_m} \times \mathcal{E}_1 \times \ldots \times \mathcal{E}_n \to \mathbb{R} && (5.10)\\
u_{A_j} &: X_{A_j} \times \mathcal{E}_1 \times \ldots \times \mathcal{E}_n \to \mathbb{R}, && (5.11)
\end{aligned}
$$

where U denotes a predefined set of utility functions of the game. The utility function $u_{D_i} \in U$ of i-th defender calculated at the end of each iteration is a function of the strategy that was used in the iteration, the strategies of the attackers (if they are available), and the set of local and remote events detected during the iteration cycle. The utility function u_{A_j} of j-th attacker depends on his strategy and the set of all detected events. The utility function of the defender gives higher payoff to strategies that result in uniquely detected malicious events, which forces the detector to specialize and create unique detections, while the utility function of the attacker penalizes strategies that were successfully detected, motivating the attacker to perform attacks that are hard to detect.

This way, we connected a general specification of multi-player dynamic game with the proposed collaborative architecture, where the defenders (IDS detectors) select a suitable strategy (configuration) and receive the payoff (feedback), which is used for choosing a strategy for the next iteration (with the reconfiguration function). Formulating the collaboration problem with a game-theoretical formalism allows us to: 1) model the contradictory goals of the attackers and the defenders, 2) model and enforce the specialization of the defenders, and 3) propose a solution concept suitable for solving the game for the dynamic environment, leading to high utility values in long-term, which in practice means high overall efficacy results of the detectors.

The defenders should ideally optimize a shared utility function. In practice, though, the utility functions (on the defender's side) differ as the internal state of each player is inaccessible for integration and security reasons. The dynamism of the environment also causes that utility functions of the defenders can rarely be identical.

In the following, we will assume a three-player game between one attacker and two defenders. Note that this game can be easily extended to different scenarios with more players on both sides. Generally, the payoff matrix for the first defender that describes the payoff distribution according to strategies selected by the attacker (columns) and the first defender (rows), can be written as follows:

$$
\mathbb{U}_{D_1} = \begin{pmatrix} u_{D_1}(x^1_{D_1}, x^1_{A_1}, E^{(1)}_{D_1}, E_{D_2}) & \ldots & u_{D_1}(x^1_{D_1}, x^k_{A_1}, E^{(1)}_{D_1}, E_{D_2}) \\ \vdots & \vdots & \vdots \\ u_{D_1}(x^l_{D_1}, x^1_{A_1}, E^{(l)}_{D_1}, E_{D_2}) & \ldots & u_{D_1}(x^l_{D_1}, x^k_{A_1}, E^{(l)}_{D_1}, E_{D_2}) \end{pmatrix}, \qquad (5.12)
$$

where k, l denotes number of all strategies for P_{A_1}, P_{D_1} respectively. I-th column of matrix $\mathbb{U}_{D_1}$ describes how does the payoff change depending on the defender's strategies (in rows) when the attacker performed i-th strategy. The payoff matrix for the attacker $\mathbb{U}_{A_1}$ and the second defender $\mathbb{U}_{D_2}$ can be expressed similarly.

The proposed game is a sequential game of unpredictable length, where i-th iteration cycle of the game consists of the following steps:

1. **Applying optimal strategy** - At the beginning of each iteration cycle, each defender uses the strategy x that would have been optimal in the last iteration (e.g. the strategy that leads to the highest payoff in long-term) for detecting malicious activity from the network traffic. In our embodiment, this corresponds to applying the optimal configuration of the detector that leads to most unique and valuable detections. In the first iteration cycle of the game, where all strategies have equal payoff, the algorithm will choose one randomly.
2. **Collecting available information** - Next, each defender collects all available information necessary for calculating the payoff of the strategy that is being used in this iteration. This includes collecting local events generated by the detector and the current strategy, and all remote events provided by other collaborating detectors.
3. **Computing payoff values** - Once the local and all remote events are collected, the defender will evaluate the current strategy by computing the payoff value. Since we want to specialize the individual defenders, the utility function should return higher values for strategies resulting in high amount of unique local detections (events), and lower values for strategies with redundant or small amount of local detections (events). The uniqueness $\kappa_{D_i}^{(k)}$ of k-th local event $e_{D_i}^{(k)}$ is calculated through the predefined similarity function as follows:

$$\kappa_{D_i}^{(k)} = \min_{\forall j, j \neq i} \min_{\forall e_l \in E_j} (1 - s_{ij}(e_k, e_l)), \tag{5.13}$$

where $s_{ij}(e_k, e_l) \in [0,1]$ is a similarity of two events. Equation 5.13 defines the uniqueness $\kappa_{D_i}^{(k)}$ of an event e_k as the minimal value of $1 - s_{ij}(e_k, e_l)$ calculated from all remote events e_l from all collaborating detectors. In case of two defenders, the uniqueness of k-th event $e_{D_1}^{(k)}$ created by P_{D_1} is calculated as:

$$\kappa_{D_1}^{(k)} = \min_{\forall e_l \in E_2} (1 - s_{12}(e_k, e_l)). \tag{5.14}$$

Based on the uniqueness of the local events $E_{D_i}^{(p)}$ created by the defender P_{D_i} with p-th strategy $x_{D_i}^{(p)}$ to counter strategies $x_{A_1}, \ldots, x_{A_m}$ of the attackers, the payoff for $x_{D_i}^{(p)}$ is defined as:

$$u_{D_i}^{(p)} = u_{D_i}(x_{D_i}^{p}, x_{A_1}, \ldots, x_{A_m}, E_{D_1}, \ldots, E_{D_n}) = \sum_{e_k \in E_{D_i}^{(p)}} \kappa_{D_i}^{(k)} \cdot \upsilon_{D_i}(e_k, x_{A_1}, \ldots, x_{A_m}), \tag{5.15}$$

where $\upsilon_{D_i}(e_k, x_{A_1}, \ldots, x_{A_m}) : E \times X_{A_1} \times \ldots \times X_{A_m} \to [0,1]$ is a utility function describing the value of the particular event e_k (e.g. derived from the severity of the detected behavior or from the confidence of the detection method) and its relation with strategies $x_{A_1}, \ldots, x_{A_m}$. The relations can be approximated by injecting simulated attacks into the real network traffic and mapping them onto detected events. The events that correspond to known malicious behaviors will have higher utility values. If such relation is not available, the payoff does not depend on the strategies of the attackers and the utility function for the game of two defenders can be simplified as follows:

$$u_{D_1}^{(p)} = u_{D_1}(x_{D_1}^{p}, E_{D_1}, E_{D_2}) = \sum_{e_k \in E_{D_1}^{(p)}} \kappa_{D_1}^{(k)} \cdot \upsilon_{D_1}(e_k). \tag{5.16}$$

The utility function υ from Equation 5.15 or Equation 5.16 is not necessarily required. More specifically, in those cases when the detectors cannot determine the utility of generated events (e.g. event severity), we can omit the utility function in Equation 5.16 and define the feedback function only with similarity functions. This would imply that all events have the same value.

4. **Updating long-term payoff of the strategies** - As the proposed game has unpredictable number of iterations, the model maintains a long-term payoff matrix for each strategy, describing the value of the strategy from the long-term perspective. Once the payoff is computed for the current strategy $x_{D_i}^{(p)}$, the solution concept that controls the game updates its corresponding long-term payoff $\overline{u}(x_{D_i}^{(p)})$. In our case, these values describe an expected value for each configuration in terms of unique and valuable events.
5. **Selecting the optimal strategy for the next iteration** - Finally, the solution concept will select the optimal strategy for the next iteration. If the concept explores the space greedily, it will always select the strategy that leads to the maximum payoff:

$$x_{D_i}^{*} = \arg \max_{x_{D_i} \in X_{D_i}} \overline{u}(x_{D_i}). \tag{5.17}$$

   Note that the configurations of detectors receive low feedback value if they produce the same events in the same time window. This however does not influence long-term behavior of the model in this highly dynamic environment. Greedy solution concepts do not have to be the optimal choice for dynamic games, as they often converge to local optima. In the next section, we propose a new solution concept that is more robust against the changes of the game.

These five steps represent actions of the game in one iteration, which corresponds to one reconfiguration cycle of the detector. The reconfiguration cycle is performed repeatedly every 5 minutes and in each cycle, new and previously unseen network traffic is analyzed and evaluated. In practice, the proposed model requires the definition of the similarity functions s_{ij} for each couple of detector types and optionally the utility function υ_{D_i} from Equation 5.15. The rest of the reconfiguration process is defined by the game.

The notion of optimization, where players are not explicitly informed about actions played by other players, makes the game consistent with the formalization based on extensive-form games introduced in [165]. We are facing a dynamic optimization problem, where the environmental conditions imposed by the external environment can change rapidly, making the game similar to a sequence of static games with unpredictable length.

5.3.1 Summary of the Reconfiguration Process

In this section, we will briefly summarize the whole idea of the proposed collaborative reconfiguration process. Once IDS detector P_{D_i} completes the detection process, it sends the local events E_i to other detectors and at the same time collects remote events $E_1, \dots, E_{i-1}, E_{i+1}, \dots, E_n$. Once the remote events are collected, the detector P_{D_i} determines the similarity of local to remote events and computes a payoff value (Equation 5.15). This value represents how unique and valuable these local events are in the collaborative context. Finally, the payoff is passed to the reconfiguration function controlled by the solution concept (Equation 5.17), which selects new configuration (strategy) of the detector that is applied in the next iteration cycle.

5.4 Game Solution Concept

In this section, we will introduce a simple, yet effective algorithm that can be used as a solution concept for the game defined in Section 5.3. As mentioned above, algorithms in dynamic network environment should rapidly converge into equilibrium. The equilibrium can be easily changed with the change of the environment. Since the payoff received at the end of each iteration of the game can be considered as a reinforcement received from the environment, we propose to adopt a sequential reinforcement learning model. More specifically, we propose a new algorithm ε-FIRE, which is a combination of FIRE model [78, 133] with ε-greedy [144] method to guarantee that the algorithm will always find new equilibria.

The ε-greedy algorithm is an effective means of balancing exploitation and exploration in multi-agent reinforcement learning domain. This algorithm behaves greedily most of the time, which means that it selects the actions with the highest estimated reward (or rating). However every once in a while (with probability ε), it selects the action randomly from all possible actions:

1. Algorithm starts with a set of strategies X to select from. Each strategy is associated with expected utility $u(x)$.
2. The algorithm draws a random number r from the $[0, 1]$ interval. It compares r with the ε parameter (hence ε-greedy).
3. If $r < \varepsilon$, then the algorithm randomly selects one strategy from the strategy set. Otherwise, it selects the strategy with the highest expected payoff.

The convergence of the ε-greedy algorithm is guaranteed for static environments [144], because as the number of game stages increases, all actions will be selected an infinite number of times.

The results regarding the behavior of the ε-greedy algorithm in stochastic games are encouraging. In [152], the authors show that the application of ε-greedy can achieve better results than standard Q-learning approaches in a series of games. Interestingly, the authors argue that its use can achieve an average payoff higher than Nash equilibria value in some games and show it for Prisoner's dilemma in particular. However, compared to regret-minimization algorithms [32], the algorithm does not use (and it also does not need to maintain) the information about the expected payoff for each action. However, the convergence results only hold for a sequence of static games.

FIRE model [78] was originally designed for trust evaluation in open multi-agent systems. We have adopted its interaction part into our algorithm to increase convergence speed. Greedy strategy with the highest payoff is computed according to the following equation:

$$u(x) = \max_j \sum_i w_i \cdot u_i^j,$$

where w_i represents i-th weight coefficient and u_i^j represents i-th last observed payoff of j-th strategy. The weights must hold conditions:

$$W = \sum_{i=1}^{N} w_i = 1 \quad \wedge \quad w_{i-1} \leq w_i,$$

which means it gives more weight to more recent payoff. According to [78] we can use the following function to calculate the weights:

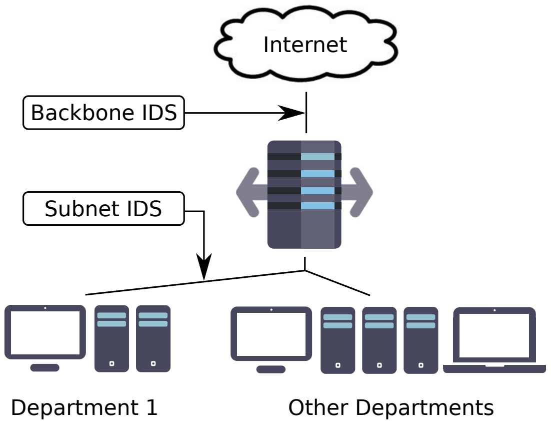


Fig. 5.7: Overview of the collaboration scenario between two IDS detectors deployed on different parts of the network. The first IDS was placed topologically in front of the second IDS.

$$w_i = \frac{1}{W} \cdot e^{\frac{i-1}{\lambda}}, \tag{5.18}$$

where $i-1$ is a time difference describing how far from the past the corresponding weight is and $\lambda = -\frac{log(C)}{N-1}$ with C as user parameter defining value of the last weight before normalization. With this weight function we can create a set of weights in exponentially descending order starting from 1 down to C.

Our ε-FIRE algorithm selects strategy x according to the following principle:

$$x = \begin{cases} \arg\max_j \sum_i w_i \cdot u_i^j & r \geq \varepsilon \\ \\ \text{random}(X) & r < \varepsilon \end{cases}$$

The ε-FIRE algorithm is suitable for highly dynamic environments, where the reward variance is larger and rewards are noisier. In such environments, ε-greedy method should perform better than any simple greedy algorithm [144]. We also argue that in highly dynamic environments, the ε-FIRE algorithm can outperform the regret minimization as the regret values based on long-term past experience may be misleading. In the proposed solution concept, these values are discounted by the use of the FIRE model.

5.5 Experimental Evaluation

This section describes the details about the distributed collaboration of two NetFlow IDS detectors deployed on different parts of the network (as shown in Fig. 5.7). The first detector (denoted as *backbone*) was placed on the backbone link and analyzes external communication from all departments, while the second detector (denoted as *subnet*) was deployed inside the network and processes the network traffic only from Department 1. It means that the communication between Department 1 and Internet is visible to both detectors and the proposed collaboration method can be used to specialize each detector.

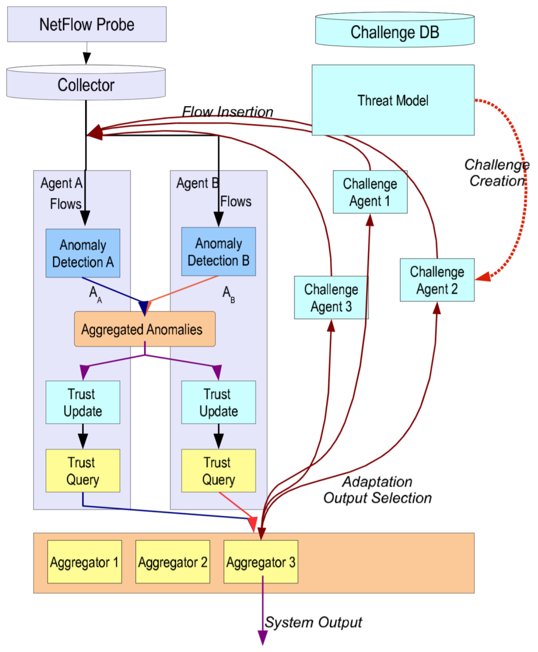


Fig. 5.8: Overview of the local self-adaptation process in CAMNEP detector.

Network traffic used in the evaluation consists of 500 minutes of real university network traffic in NetFlow format. The traffic also contains a persistent malware in various stages including the initial infection (typically from a USB drive or an email attachment), (optional) library download through the HTTP connection and principally the Command and Control traffic implemented as a periodic polling of a specific website with HTTP GET requests modeled after actual malware behavior. It may also establish and maintain the connection and download a small file from the Command and Control server.

5.5.1 IDS Nodes Description

For the evaluation, we used two CAMNEP [123] IDS detectors. CAMNEP detector analyzes NetFlow data in batches (usually 5 minutes of network traffic) with six different anomaly detection methods. Every 5 minutes, the methods assign to each flow an anomaly score. A suitable aggregation function is chosen for combining all anomaly scores of a flow into a *combined anomaly score*. The combined anomaly score is computed for all flows from the batch, creating a distribution of combined scores. Based on a self-adaptive algorithm proposed in our previous work [124], the system selects an aggregation function and determines the *threshold* separating anomalous and normal parts of the distribution. In the final stage of the processing, events are created from the anomalous part of the distribution by grouping anomalous flows with similar attributes. The local self-adaptation cycle is illustrated in Figure 5.8.

CAMNEP IDS detector has the following properties:

- **online detection** - CAMNEP detector processes batches with 5 minutes of network traffic (because of the statistical nature of the anomaly detection engine).
- **local reconfiguration** - CAMNEP detector is able to reconfigure and change its internal state by selecting an optimal aggregation function (which combines anomaly values of the individual detection methods). The aggregation function is the subject for optimization.
- **local self-adaptation** - The system optimizes its performance according to the predefined network services and attacks that are inserted from the database into the real network traffic. If an aggregation function performs well on this labeled data, it will be a suitable candidate for combining anomaly values of the real network traffic.
- **threshold computing** - Based on the local self-adaptation, the system determines the threshold dividing legitimate and malicious parts of the network traffic.

Detailed description of CAMNEP system can be found in [123].

The output of each CAMNEP IDS detector is a set of events characterizing anomalous processes and services in the network. Different aggregation function typically results in different set of events. In the experimental evaluation, we manually predefined 30 different aggregation functions.

5.5.2 Experimental Scenarios

The backbone and subnet detectors were evaluated in four different scenarios to demonstrate the benefits of the proposed collaboration model. In each scenario, the detectors used a different reconfiguration function that controls the process of choosing the optimal aggregation function for the next iteration cycle (next 5-minutes of the network traffic).

In *stand-alone* scenario, both IDS detectors operate in isolation and their reconfiguration functions are based solely on the local self-adaptation process. This process choses an aggregation function that separates best labeled attacks and legitimate services inserted to the real network traffic. This scenario simulates a standard deployment of a single detector without any further interactions with the environment.

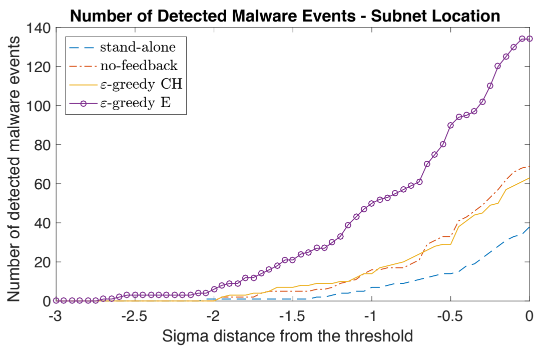
Second scenario called *no-feedback* extends the stand-alone model with an event fusion performed at the end of each reconfiguration. The fusion algorithm combines the results from both detectors into a single coherent output, showing the benefits of having more detectors at different locations.

Third scenario called *ε-greedy CH* performs with small probability ε an exploratory move and selects randomly an aggregation function from the space of all aggregation functions, while otherwise (with probability $1 - \varepsilon$) it behaves similarly to *no-feedback* reconfiguration. In our experiments, we set $\varepsilon = 0.2$ allowing the strategy to perform enough exploratory moves.

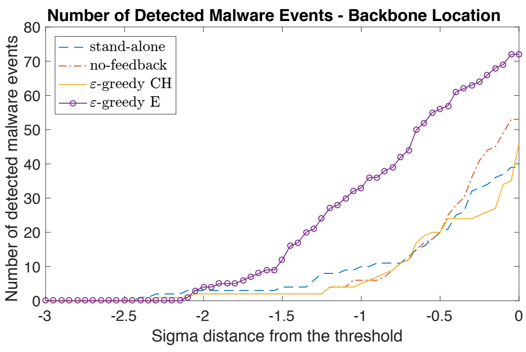
Finally, the scenario called *ε-greedy E* is a combination of local self-adaptation and the proposed ε-FIRE solution concept described in Section 5.4 with an exponential vector of weights $w = \{0.046, 0.082, 0.147, 0.261, 0.464\}$ (according to Equation 5.18). In each iteration cycle, the aggregation functions are evaluated twice: by the local self-adaptation and w.r.t the uniqueness of generated events. A simple average of both evaluations determines, which aggregation function will be chosen for the next iteration cycle. Based on the uniqueness of generated events, the reconfiguration function updates the long-term payoff of the configuration (as described in

Algorithm 3 Computing uniqueness of two sets of events

```
function COMPUTEUNIQUENESS(localEvents, remoteEvents)
    $unique = 0$
    for $localEvent : localEvents$ do
        for $remoteEvent : remoteEvents$ do
            $unique = unique + 1 - eventSimilarity(localEvent, remoteEvent)$
        end for
    end for
    return $unique$
end function

function EVENTSIMILARITY(lE, rE)
    return $2 * intersection(lE.flows, rE.flows)/(lE.flows + rE.flows)$
end function
```



(a)



(b)

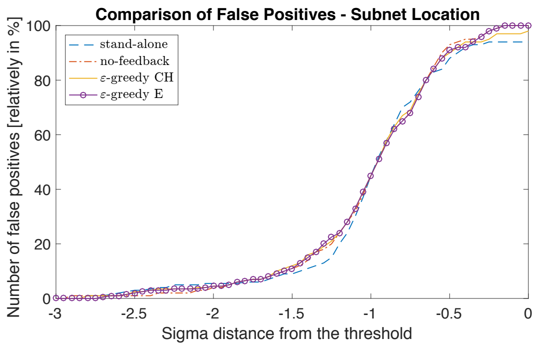
Fig. 5.9: Number of successfully detected malware events depending on sigma distance from the threshold position - subnet(a) and backbone (b) location. Higher values are better

step (4) of the game model). The uniqueness of events is calculated from the similarity of the underlying flows according to Algorithm 3. The uniqueness is computed from the ratio of similar flows in both events. Two flows are considered as similar if they have the same source and destination IP addresses, source and destination ports, and protocol.

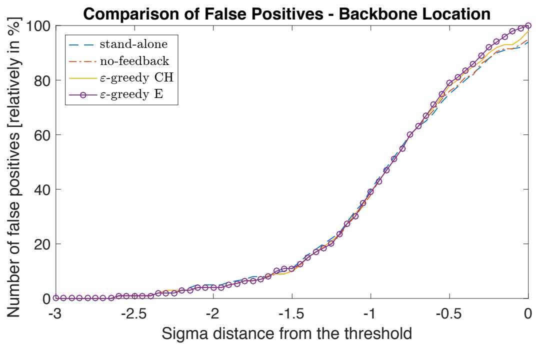
The ε-*greedy E* reconfiguration allows each detector to optimize not only w.r.t. the local self-adaptation, but also w.r.t. *uniqueness* of created events. Thus each CAMNEP detector is encouraged to specialize on unique network behaviors and significantly increases its potential of finding novel threats, while keeping false positives down by the local self-adaptation.

5.5.3 Evaluation

In our evaluation, we will show how the proposed distributed collaboration model for one network improves efficacy of the overall detection system. First, we analyzed the amount of successfully detected malware events for each of the four scenarios described in Section 5.5.2. The results are shown in Figure 5.9. Malware event is considered to be detected by CAMNEP detector if the



(a)



(b)

Fig. 5.10: Relative false positive rate of the system depending on sigma distance from the threshold - subnet (a) and backbone (b) location. Lower values are better.

selected aggregation function evaluated the corresponding flows as anomalous by putting them below the threshold defined in Section 5.5.1.

The position of the threshold separating legitimate and malicious traffic is marked with the value ”0” at the right-hand side of the figure. Sigma distance (see also Equation 3.8) of an event is defined as a difference between the threshold position and an average degree of anomaly of all flows in the event (computed by the individual CAMNEP node) divided by the standard deviation computed from anomaly values of all flows. Note that higher negative values of sigma distance from the threshold (i.e. from the threshold to the left) means that the malware event is better separated from the rest of the traffic.

Analysis from subnet detector is depicted in Figure 5.9 (a). The worst efficacy results in number of detected malicious events were obtained in the *stand-alone* scenario. The other two scenarios, namely *no-feedback* and *ε-greedy CH* performed significantly better, showing the benefits of having two detectors on different locations with the overlapping network traffic. This is where the event fusion algorithms are being applied, representing the current state-of-the-art. Finally, all three scenarios were outperformed by the proposed *ε-greedy E*, when both CAMNEP detectors interact and reconfigure according to the proposed model.

Analogical evaluation of the backbone detector is illustrated in Figure 5.9 (b). The results are similar as for the subnet detector, however the differences among non-cooperative scenarios are not so significant. *No-feedback* scenario as well as *ε-greedy CH* shows comparable results with the *stand-alone* scenario in number of detected malware. This suggests that the event fusion technique may not be sufficient. Therefore, the system requires the proposed collaborative approach (*ε-greedy E*), which ensures the diversity and specialization of the individual detectors as shown in Figure 5.9. This is especially important given the fact that the self-monitoring mechanism on both nodes was identical, and that the increased efficacy is purely a result of co-specialization.

Furthermore, all scenarios show comparable false positive rates (Figure 5.10). This observation confirms our assumption that specialized detectors might have individually lower number of true positives, but they also have lower number of false positives. The combination of positives from both specialized detectors resulted in significantly more malware detections with the number of false positives comparable to individual detectors running in isolation.

5.6 Summary

In this section, we have compared the performance of stand-alone detectors running in isolation with three other strategies, where the detectors interact with each other. We have shown that event fusion has a positive impact on the overall detection efficacy. Specifically, it increases recall while false positive rate remains effectively unchanged. Moreover, we have shown that the proposed collaborative approach between the nodes significantly boosts the overall efficacy, allowing each system to specialize on the particular network activities, while not reducing the overall precision. By deploying the collaborative approaches to more complex network infrastructures, the IDS nodes will dynamically react to the changes in the network. They are able to act as a coherent entity capable of maximizing the global network security awareness, while relying only on minimal mutual collaboration and implicit synchronization.

We placed our collaborative mechanism on the top of the proposed fusion architecture for two reasons: 1) Collaborating with high-level objects (events) significantly reduces computational complexity of the collaborative model, especially the costs necessary for synchronization and sharing the events with other systems. The overall complexity of the collaboration mainly depends on the complexity of the predefined similarity metrics. 2) Collaborating with high-level objects (events) that contain as much additional information and knowledge as possible increases the effectiveness and allows making high-level security assessments for the whole network infrastructure.

Chapter 6
Experiments with a Collaborative Model for Multiple Networks

The benefits of the collaboration shown in Chapter 5 do not have to be limited to a single network. On the contrary, a collaboration model for detectors deployed in multiple networks can significantly reduce the time necessary to detect global network threats, systematically targeting companies all over the world. A mechanism that would instantaneously propagate the knowledge about novel attacks to other collaborating detectors and networks and use this knowledge to configure these systems automatically would significantly decrease global re-usability of the novel attacks for targeting other networks.

In this chapter, we evaluate a conceptual model of such mechanism. Our collaborative model for multiple networks benefits from the recent increase of interest of cloud security monitoring and services. It creates a global framework that can access the network traffic from multiple networks at the same time without insurmountable integration, security, or bandwidth issues. Privacy issues can be solved by complete anonymization and obfuscation of all sensitive information before sending the information to other detectors. In our model, we propose to share only a limited (but still very valuable) amount of metadata, such as the list of detected malicious Internet domains, server IP addresses, autonomous systems, or feature vectors (numbers) describing the detected behaviors. This data does not contain any host/user-specific information. This list is a reduced version of the security events from the algorithm in Chapter 5, stripped of the potentially internal or private information.

If the number of networks becomes large enough, the collaboration builds a global threat intelligence model. While these models would rarely be complete, collecting such global intelligence is a significant advantage for the detectors, as global statistics and trends can be easily computed and used to strengthen the detection algorithm. The downside of this approach is the necessity to process much more traffic than many of the advanced detection or classification techniques (such as the one introduced in Chapter 4) can handle. And direct inspection in a cloud device is out of question. However, our collaboration model is designed to deal with this problem.

The proposed model is based on a different idea than the model introduced in Chapter 5, as specializing the detectors for each network separately would be undesirable due to the missing overlap of the network traffic. Global specialization would also be inappropriate due to the obvious misalignment between the utility functions of individual network owners [1]. Instead, we propose to build and share a global intelligence represented by the reduction of events to

[1] The fact that the malware was discovered somewhere else would be of little consolation to any specialized victim.

the groups of malicious behaviors and sources (such as known malicious second-level domains, server IP addresses, or autonomous systems) found in various networks. As most of the global threats are frequently targeting multiple networks, fast detection on at least one network and propagating the results to the rest of the networks in a timely fashion prevents the attack from being globally successful and makes the attack infrastructure harder to re-use.

The model has two main components: sampling and correlation component. The sampling component ensures that the amount of data that is about to be processed by the detection system does not exceed the predefined limit and at the same time, the data that is selected for the processing contains as much relevant information as possible. On a single detection system, this can be achieved by the adaptive sampling described in Chapter 3. However, we go one step further and propose a collaborative-adaptive sampling, where the parameters of the sampling are adjusted based of the feedback received from other detection systems provided by the correlation component. This way, the information about newly discovered attacks can simultaneously spread across all the detection systems, as illustrated in Figure 6.1. Depending on the security requirements and environment restrictions, the model can be deployed in a centralized or fully-distributed architecture, and as a combination of IDS on premise and cloud-based collaborative mechanism, or as a system fully-integrated in the cloud.

Sampling component can be also easily integrated with almost any kind of detector, without the modification or even a knowledge of the detector. This makes sampling an ideal component of the black-box integration framework.

6.1 Correlation Component

The purpose of the correlation component (denoted as (C) in Figure 6.1) is to collect results from all detectors and compute a global intelligence from the detected attacks. The proposed correlation mechanism is specific to this application domain and consists of two stages of clustering based on the two feature sets: a) features describing malware behavior (e.g. volume, persistence, frequency, similarity between malware samples) are used for *behavioral clustering*, and b) features describing identity of the targeted servers (hostname, server ip address, or autonomous system) are used for *identity clustering*.

Behavioral clustering groups similar types of malicious behaviors. Each cluster may consist of attacks with servers hosted in various parts of the world. Note, that as opposed to probabilistic threat propagation [38], these servers do not have to be observably interconnected for the attacks to fall into a cluster. Identity clustering complements the behavioral clustering and groups different attacks launched from or associated with the same server. Finally, the identity features are used to build the global reputation describing the amount of malicious activity associated with each domain, server IP address, or autonomous system.

Graphical illustration of the proposed behavioral and identity clustering is shown in Figure 6.2. (1) Initially, Attacker Node 1 performed two attacks of the same type (Type 1) against Company A and a different attack (Type 2) against Company B. Only one of these attacks was detected and reported as a malicious incident (red node), while the rest of the attacks were reported as borderline incidents (yellow nodes) with low severity, waiting for further confirmation. Moreover, Attacker Node 2 performed another attack (Type 2) against Company C, which

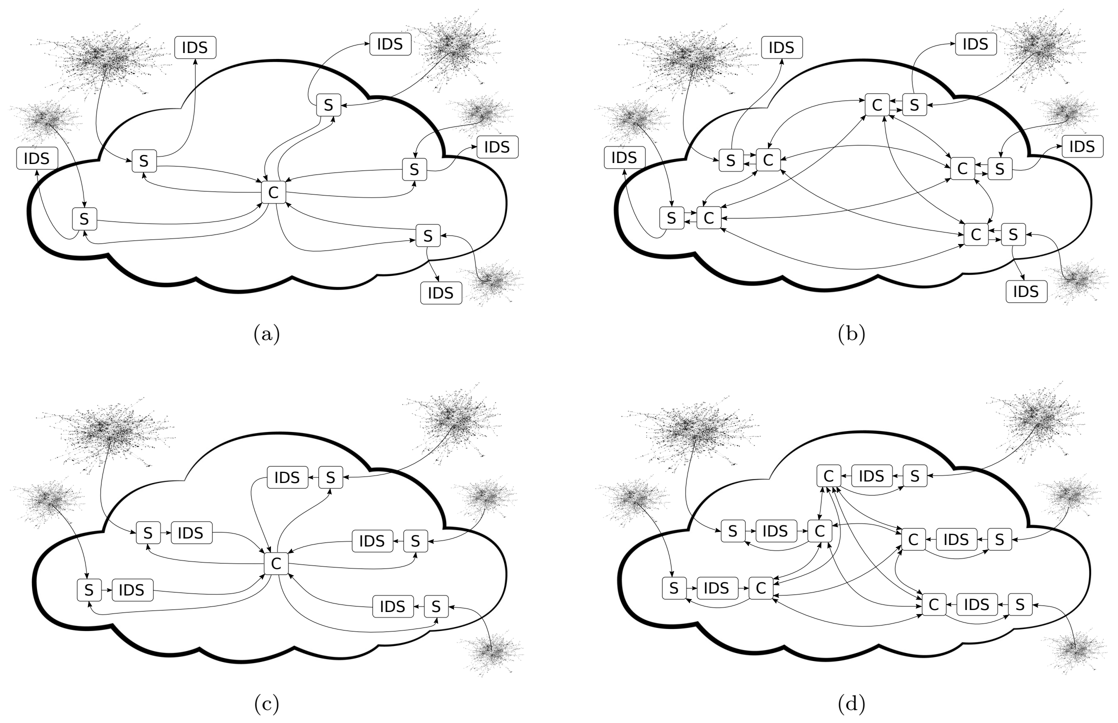


Fig. 6.1: Graphical illustration of the proposed collaborative model across multiple networks - a combination of IDS on premise with collaborative mechanism in the cloud ((a) centralized and (b) fully distributed architecture) and the solution fully integrated in the cloud ((c) centralized and (d) fully distributed architecture). The amount of data sent to the cloud from multiple networks is reduced by the sampling components (S), so the consequent IDS detector (IDS) is able to process it. The correlation component (C) collects available results from all detectors and make global knowledge, which is then pushed back to the sampling components for further reconfigurations.

was also detected as a borderline incident. Note that the attacker nodes are not observably interconnected with each other. The proposed clustering is able to automatically confirm the maliciousness of the borderline incidents by: (2) putting the incident from Company B into already detected malicious cluster Type 1 as it exhibits behavior similar to MW Type 1, and (3) inferring the maliciousness of Attacker Node 2 due to the newly discovered incident at Company B, which makes the other borderline incident launched from the same attacker node but detected in Company C even more suspicious.

This way, the knowledge gained at one network can be extended and applied at other networks to discover the structure of the malicious infrastructure behind the global threats. In the following, we will discuss both clustering approaches in more detail. We emphasize that the purpose of this section is not to propose a new clustering algorithm, but rather to apply an existing algorithm as a necessary part of our collaborative model. More detailed clustering algorithms can be found in [82].

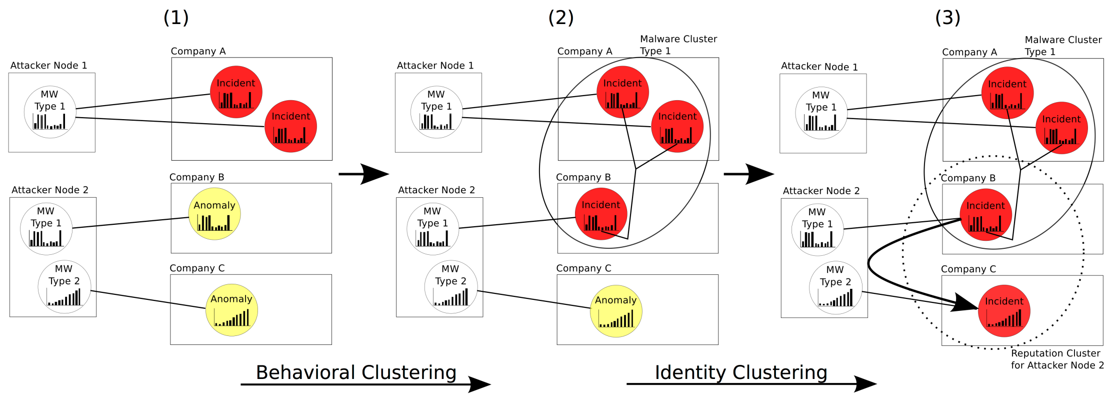


Fig. 6.2: Graphical illustration of two types of clustering approaches used by the correlation component for extracting a global security knowledge.

6.1.1 Behavioral Clustering

Behavioral clustering groups attacks with similar behavior into one cluster and separates them from the rest of the traffic. Since malware is highly evolving, the number of diverse attack techniques grows with time. This means that the full set of malicious categories is not known in advance. An unsupervised (or semi-supervised) clustering algorithm can be used to divide the detected events into clusters. The design of a specific behavioral clustering algorithm is being studied in [82] and is not part of this thesis.

6.1.2 Identity Clustering

We propose to use an identity clustering as online and interactive alternative to commonly used blacklists or malicious feeds or other static sources of information about locations typically hosting malware. It serves as a complement to behavioral clustering. Attackers typically reuse their acquired or compromised sites for multiple purposes. Once this site or server IP address is identified (e.g by behavioral clustering), it's reputation score maintained for every observable location is decreased. Later, an unknown attack launched from bad-reputation location is automatically considered as untrustful and can be processed separately with higher importance.

6.2 Collaborative-Adaptive Sampling Component

The global threat intelligence created by the correlation component is transfered back to the sampling module. The proposed adaptive sampling introduced in Chapter 3 uses precomputed statistics and other features to adaptively modify the sampling rate of the incoming network connections to emphasize smaller or unique artifacts frequently related to malicious activity.

We propose to extend the sampling algorithm (denoted as (S) in Figure 6.1) to incorporate the global threat intelligence provided by the correlation component. A naive solution is to create a database of known malicious sites (e.g. hostnames or second-level domains) and check the presence of these sites in the original unsampled data. If such domain is found, the traffic can be automatically included to the sampled data. However, the naive approach has two main disadvantages: 1) it does not control the amount of traffic going to the detectors and once an attacker launches a massive attack against the detector from bad-reputation sites, it could completely saturate the detector, and 2) to bypass the existing blacklists and feeds, attackers typically change hostnames, domains, and server IPs quite frequently, thus reducing the effectiveness of the naive approach.

Instead, we propose to maintain a global reputation of each domain, server IP address, and autonomous system and include the reputation values back to the adaptive sampling algorithm. Connections originating from bad-reputation servers (or autonomous systems) will be sampled with higher probability than connections from widely used services. More specifically, the correlation component provides through reputation clustering a reputation value $\tau^{(sld)}$ for every second-level domain, $\tau^{(sIP)}$ for every server IP address, or $\tau^{(AS)}$ for every autonomous system. The value $\tau^{(sIP)} \in [0,1]$, where $\tau^{(sIP)} = 0$ means that the corresponding server IP address is frequently associated with malware (so the server IP has very low reputation), while $\tau^{(sIP)} = 1$ denotes a server IP with the highest possible reputation.

To make the adaptive sampling collaborative, the reputation is incorporated as a part of the primary probability (see Equation 3.4) as follows:

$$p_p(\varphi|f_1,\ldots,f_k) = \begin{cases} s^{(a)} \cdot \frac{1}{\tau(\varphi)} & f^c_{\varphi(1,\ldots,k)} \leq t, \quad \forall\varphi : s(\varphi) \leq \tau(\varphi) \\ s^{(a)} \cdot \frac{\log t}{\log f^c_{\varphi(1,\ldots,k)}} & f^c_{\varphi(1,\ldots,k)} > t \end{cases} \tag{6.1}$$

where $\tau(\varphi)$ represents the final reputation of flow φ computed as a simple average of marginal values $\tau^{(sld)}$, $\tau^{(sIP)}$, $\tau^{(AS)}$ etc. This way, the collaborative-adaptive sampling can significantly boost the sampling rate for flows with bad reputation. Note that the sampling boost is provided only to the flows with feature values below the threshold, as emphasizing redundant flows would be counterproductive.

Condition $s(\varphi) \leq \tau(\varphi)$ needs to be satisfied for all flows, otherwise the primary probability would exceed the interval $[0,1]$. If a reputation value of a flow exceeds the sampling rate $s^{(a)}$, all reputation values will be scaled accordingly. The computation formula for the sampling rate $s^{(a)}$ from Equation 3.7 has to be modified as follows:

$$s^{(a)} = \frac{T_{flows}}{\sum_{f^c_{\varphi(i)} \leq t} \frac{1}{\tau(\varphi)} + \sum_{f^c_{\varphi(i)} > t} \frac{\log t}{\log f^c_{\varphi(i)}}}. \tag{6.2}$$

The algorithm starts by setting all reputation values equal to 1. Based on each update provided by the correlation component, the reputation is either decreased, or increased, or remains constant. The level of decrease depends on the number of networks reporting the location as malicious - the more networks, the more serious and widespread the infection is and the larger decrease in reputation is performed. On the other hand, the reputation can be increased when no more infections are reported for some larger time periods.

Dataset	All Flows		Malicious Samples				Infected
	Total	Max in 1 min	Flows	SLD	sIP	AS	Users
Small-1	25,971,094	45,906	13,923	57	72	20	41
Small-2	24,747,491	42,549	234	25	26	10	32
Medium-1	75,648,425	126,752	549	34	56	16	18
Medium-2	66,523,237	100,490	307	34	44	20	43
Medium-3 (L)	52,938,375	69,801	2,948	107	139	27	123
Large-1 (XL)	273,687,428	333,851	12,058	235	319	60	375

Table 6.1: Overview of the datasets used in the evaluation. Each dataset is described by the total number of flows per day and maximum per 1 minute. Details about malicious samples are also shown, included number of flows, second-level domains (SLD), server IP addresses (sIP), autonomous systems (AS), and infected users.

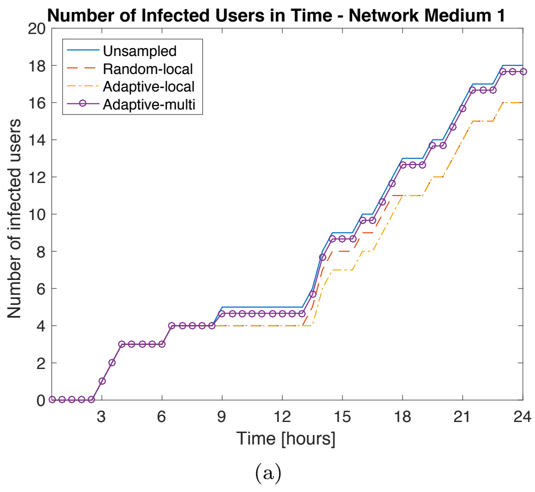
6.3 Experimental Evaluation

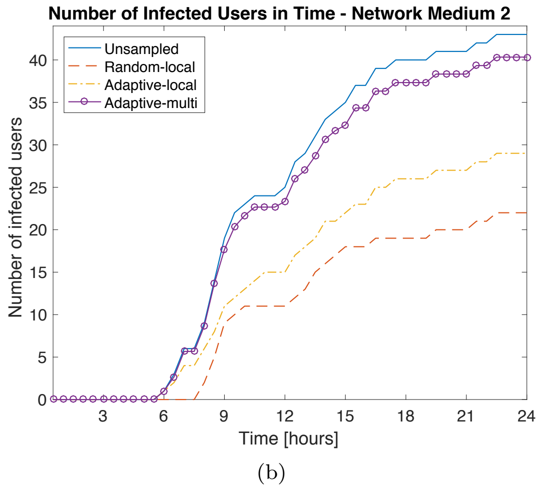
The evaluating collaboration system includes 6 detection systems analyzing HTTP(S) proxy logs from 6 different networks. Each detection system consists of several classification methods (such as the method proposed in Chapter 4). The classification methods require the extraction of advanced and computationally-intensive features to be able to detect more sophisticated attacks, which means that they are able to analyze only limited number of connections.

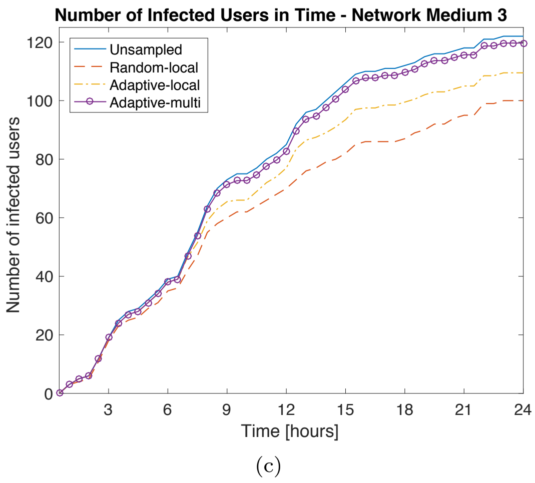
To extend the applicability of the experimental results, we will not implement the reconfiguration into the detection system (as we did in Chapter 5). Instead, we will consider each detection system as a black box with a single predefined parameter – maximum number of flows that the system is able to process in the given time interval. Depending on the computational requirements, the parameter can be predefined for every detection system individually. The goal of the collaboration is to provide global intelligence to the sampling algorithm to increase the sensitivity of sampling on probably malicious traffic. Thus, our evaluating criterion will be the amount of malicious traffic provided to the detection systems.

The details about the datasets are shown in Table 6.1. Each dataset contains 1 day of HTTP(S) network traffic of one network that was retrospectively analyzed and partially labeled by domain experts. The processing of one day of data at one network is divided into iteration cycles. In each iteration, sampled data provided by the adaptive sampling from one minute of network traffic is fed to the detection engine. The results are distributed across all six detection systems and collected in the correlation component. The correlation component maintains reputation values for all autonomous systems, server IP addresses, and second-level domains, which are updated with the newly-collected results. The updated values are provided back to the sampling algorithm to adapt the sampling rate for the next iteration. The sampling component optimizes the distribution of sampled flows, not their number which has to be under the predefined limit. In our experiments, the limit for the number of sampled flows was set to 20,000 per minute.

The proposed collaborative-adaptive sampling is compared with a stand-alone version of the adaptive and random sampling to demonstrate the improvements brought by sharing the global intelligence. The average results from 5 runs are shown in Figures 6.3 (a) - (d). Networks Small-1 and Small-2 are not displayed, because the sampling limit was too high and all the methods







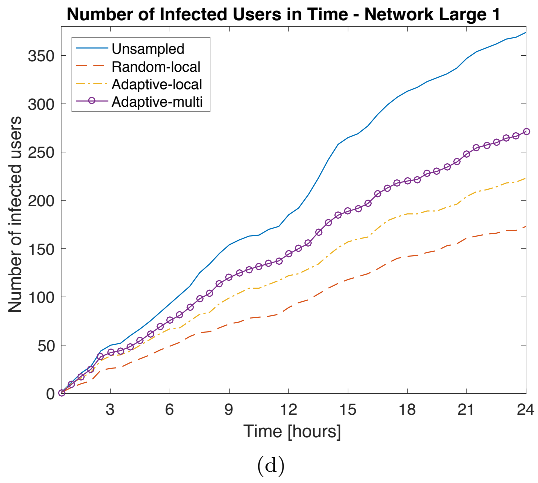


Fig. 6.3: Number of infected users from four different networks: (a) Medium 1, (b) Medium 2, (c) Medium 3, and (d) Large 1.

exhibit negligible differences. However, both networks were contributing to the collaboration by extending the global intelligence with new malicious findings.

On the rest of the networks, the differences between the methods increases with the growing size of the network. First, on network Medium-1 (Figure 6.3 (a)), the differences are still negligible. As the number of infected users increases (Figures 6.3 (b) - (d)), the adaptive sampling proposed in Chapter 3 exceeded random sapling running on each network separately, which confirms the results obtained in Chapter 3. Both stand-alone methods were outperformed by the proposed collaborative-adaptive sampling approach. These results confirms the expectations that global threats can be efficiently eliminated when the global intelligence is pushed back to the individual systems.

The results of the evaluation confirmed the importance of global intelligence for individual detectors and also confirmed a common claim that cloud detection is better. Even though the proposed collaborative model was influencing the systems only at the sampling level, it significantly improved the performance of each system on medium and large networks by extracting more infected users from the raw input data. This shows that network threats do target multiple networks and that a collective defense mechanism is necessary for preventing such threats from being globally successful. We hope that our results will motivate other researchers to continue their work towards collaborative security systems, for example by extending our model with more levels of collaboration.

Chapter 7
Conclusions

The goal of this thesis was to contribute to a set of methods for selecting and classifying malicious network behaviors at various levels of abstraction and to evaluate the proposed methods on real network traffic to demonstrate their usefulness in practice.

We have proposed a new collaborative fusion model structured into three main layers, where each layer increases the level of abstraction of the analyzing objects: from data to information, from information to knowledge, and from knowledge to global intelligence. The main contribution of this thesis consists of the individual components of the proposed model, namely the adaptive sampling that reduces large amounts of low-level data while preserving most of the malicious activity in the network, robust representation of network traffic suitable for classifying unseen malware variants, and two collaborative models for detection systems deployed in one and multiple networks.

According to the fusion model, the proposed methods and components can be combined together into a single detection system that transforms low-level data into high-level intelligence. The proposed methods have been experimentally verified on various types of real network data and most of them are being used in practice as a part of two intrusion detection systems: CAMNEP (Cooperative Adaptive Mechanism for Network Protection [123]) and CTA (Cognitive Threat Analytics [2]). The CTA is an on-line malware detection security-as-a-service product delivered by Cisco Systems. Besides the methods proposed in this thesis, CTA also contains an anomaly detection engine [71, 72, 73] and a correlation component [82]. At the time of writing this thesis, CTA daily analyzed more than 10 billion web requests generated by millions of users from large enterprise networks. The system finds daily tens of thousands network threats that evaded previously installed security measures positioning the system as the last line of the network defense. Through these systems, the methods protect millions of users all over the world from current and future sophisticated threats.

7.1 Thesis Achievements

This section summarizes the contribution of the thesis to the state-of-the-art of network security. The achievements are the following:

1. Design and implementation of the adaptive flow sampling method. The sampling process is divided into two phases and the sampling rate is computed individually for each network flow.

First, the predefined set of features or other statistical information important for the system is calculated from the original (unsampled) input data. Second, flows with rare or unique values of such features are sampled with higher probability than flows with frequently-used values. This way, the algorithm emphasizes new artifacts and unique behaviors that are usually tightly related with malicious activity. An extensive evaluation on various data sources from various networks confirms the benefits of the proposed method in terms of the efficacy of anomaly detectors or classifiers, as well as in terms of postmortem network forensics. The method enables the deployment of sophisticated detection/classification algorithms on large networks without sacrificing their detection power and thus successfully addresses research problem *RP1* of this thesis.

2. Design and implementation of a novel representation of network traffic suitable for detecting new and previously unseen malware variants. Instead of classifying flows individually, the proposed representation groups flows into bags and describes their internal structure, which makes the representation invariant against most of the changes the attackers typically employ to avoid detection. A novel method that combines the process of learning the representation with the process of learning the classifier is proposed to optimize the parameters of the representation automatically from the data without any manually-predefined parameters. The resulting representation ensures easier separation of malicious and legitimate communication and at the same time controls the complexity of the classifier built on the top of it. The classifier trained on the proposed representation achieved 90% precision (9 of 10 alerts were malicious) and detected 67% of previously unseen malware families. This achievement successfully addresses research problems *RP2* and partly *RP3*.
3. Model and solution concept for the distributed collaboration among heterogeneous detection systems deployed in various parts of the network. The model is based on the idea of local specialization, where the detection systems reconfigure their internal states to detect unique intrusions. The reconfiguration process is controlled by the proposed ε-FIRE algorithm and uses feedback describing the uniqueness of the current results w.r.t. the results provided by the rest of the detection systems in the given time interval. The proposed model of specialized detectors provides better efficacy results and enables to establish a collaboration even if the detection systems are heterogeneous, which successfully addresses research problem *RP4* when realized within the same network. Moreover, the proposed collaborative specialization also reduces the evasion possibilities for the attackers and implicitly contributes to address research problem *RP3*. Additional experiments of the collaborative model for multiple networks addresses *RP4* when realized across various networks.

Overall, all four problems addressed in the thesis enable an intelligent decomposition of monolithic IDS systems into larger and distributed system by separating of relatively fast operations (feature extraction, sampling, bag creation) from the more advanced transformation applied in later stages. This idea, together with a robust and fully-automated approach to distributed collaboration, enables seamless dispersal of the IDS inspection on network devices and increases the overall security of the networks designed for the future.

7.2 List of Publications and Patents

Journal Articles with Impact Factor (2)

1. **Karel Bartoš**, Martin Rehák. IFS: Intelligent flow sampling for network security: An adaptive approach. In *International Journal of Network Management (IJNM)*, volume 25(5), pages 263–282, 2015. **(80%)**
2. Martin Rehák, Michal Pěchouček, Martin Grill, Jan Stiborek, **Karel Bartoš**, Pavel Čeleda. Adaptive Multi-Agent System for Network Traffic Monitoring. In *IEEE Intelligent Systems*, pages 16–25, 2009. **(16%)**

Peer-Reviewed Journal Articles (3)

1. Jan Stiborek, Martin Grill, Martin Rehák, **Karel Bartoš**, Jan Jusko. Game Theoretical Model for Adaptive Intrusion Detection System. In *Transactions on Computational Collective Intelligence*, volume 15, pages 133–163, 2014. **(10%)**
2. Martin Rehák, Michal Pěchouček, Martin Grill, **Karel Bartoš**, Vojtěch Krmíček, Pavel Čeleda. Collaborative Approach to Network Behavior Analysis Based on Hardware-Accelerated FlowMon Probes. In *International Journal of Electronic Security and Digital Forensics*, pages 35–48, Inderscience Enterprises, 2009. **(12%)**
3. Martin Rehák, Michal Pěchouček, **Karel Bartoš**, Martin Grill, Pavel Čeleda, Vojtěch Krmíček. CAMNEP: An intrusion detection system for high speed networks. In *Progress in Informatics*, volume 4, pages 65–74, 2008. **(10%)**

Patents (3)

1. Gustav Šourek, **Karel Bartoš**, Filip Železný, Tomáš Pevný, Petr Somol. Events from network flows. US20160112442. (*US - Issued.*)
2. **Karel Bartoš**, Martin Rehák, Michal Sofka. Global clustering of incidents based on malware similarity and online trustfulness. US20160226904. (*International - Allowed by the Patent Office.*)
3. **Karel Bartoš**, Michal Sofka. Identifying threats based on hierarchical classification. US2015 0334125. (*Allowed by the US Patent Office.*)

In ISI Proceedings (15)

1. **Karel Bartoš**, Michal Sofka, Vojtěch Franc. Learning Invariant Representation for Malicious Network Traffic Detection. In *Proceedings of the 22th European Conference on Artificial Intelligence (ECAI)*, IOS Press, 2016. **(60%)**

2. **Karel Bartoš**, Michal Sofka. Robust representation for domain adaptation in network security. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 116–132, Springer International Publishing, 2015. **(70%)**
3. Vojtěch Franc, Michal Sofka, **Karel Bartoš**. Learning detector of malicious network traffic from weak labels. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 85–99, Springer International Publishing, 2015. **(25%)**
4. **Karel Bartoš**, Martin Rehák. Self-organized mechanism for distributed setup of multiple heterogeneous intrusion detection systems. In *Proceedings of the IEEE Self-Adaptive and Self-Organizing Systems - (AHANS SASO)*, pages 31–38, 2012. **(50%)**
5. **Karel Bartoš**, Martin Rehák. Trust-based solution for robust self-configuration of distributed intrusion detection systems. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, pages 121–126, IOS Press, 2012. **(50%)**
6. **Karel Bartoš**, Martin Rehák. Distributed self-organized collaboration of autonomous IDS sensors. In *Dependable Networks and Services - Proceedings of the 6th International Conference on Autonomous Infrastructure, Management, and Security (AIMS)*, pages 113–117, Springer Berlin Heidelberg, 2012. **(50%)**
7. Jan Stiborek, Martin Grill, Martin Rehák, **Karel Bartoš**, Jan Jusko. Game Theoretical Adaptation Model for Intrusion Detection System. In *Proceedings of the 10th Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, pages 201–210, Springer Berlin, 2012. **(20%)**
8. **Karel Bartoš**, Martin Rehák, Vojtěch Krmíček. Optimizing flow sampling for network anomaly detection. In *Proceedings of the 7th International Conference on Wireless Communications and Mobile Computing (IWCMC)*, pages 1304–1309, 2011. **(40%)**
9. Martin Rehák, Eugen Staab, Volker Fusenig, Michal Pěchouček, Martin Grill, Jan Stiborek, **Karel Bartoš**, Thomas Engel. Runtime monitoring and dynamic reconfiguration for intrusion detection systems. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 61–80, Springer Heidelberg, 2009. **(14%)**
10. Martin Rehák, Eugen Staab, Volker Fusenig, Jan Stiborek, Martin Grill, **Karel Bartoš**, Michal Pěchouček, Thomas Engel. Threat-model-driven runtime adaptation and evaluation of intrusion detection system. In *Proceedings of the 6th International Conference on Autonomic Computing and Communications*, pages 65–66, ACM Press, 2009. **(14%)**
11. **Karel Bartoš**, Martin Grill, Vojtěch Krmíček, Martin Rehák, Pavel Čeleda. Flow based network intrusion detection system using hardware-accelerated netflow probes. In *Proceedings of CESNET Conference - Security, middleware, and virtualization – glue of future networks*, pages 49–56, CESNET, 2008. **(20%)**
12. Martin Rehák, Michal Pěchouček, Martin Grill, **Karel Bartoš**. Trust based classifier combination for network anomaly detection. In *Proceedings of the 12th International Conference on Cooperative Information Agents (CIA)*, pages 116–130, Springer Heidelberg, 2008. **(15%)**
13. Martin Rehák, Michal Pěchouček, Martin Grill, **Karel Bartoš**, Pavel Čeleda, Vojtěch Krmíček. Collaborative approach to network behavior analysis. In *Proceedings of the 4th International Conference on Global E-Security*, pages 153–160, Springer Heidelberg, 2008. **(12%)**
14. Martin Rehák, Michal Pěchouček, **Karel Bartoš**, Martin Grill, Pavel Čeleda, Vojtěch Krmíček. Improving anomaly detection error rate by collective trust modeling. In *Proceed-*

ings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID), pages 398–399, Springer Heidelberg, 2008. **(10%)**

15. Martin Rehák, Michal Pěchouček, **Karel Bartoš**, Martin Grill, Pavel Čeleda. Network intrusion detection by means of community of trusting agents. In *Proceedings of International Conference on Intelligent Agent Technology (IAT)*, pages 498–504, IEEE Computer Society, 2007. **(10%)**

In Other Proceedings (9)

1. Veronica Valeros, **Karel Bartoš**, Lukáš Machlica. 50 thousand needles in 5 million haystacks: Understanding old malware tricks to find new malware families. To appear at *Black Hat EU*, 2016. **(33%)**
2. **Karel Bartoš**, Michal Sofka, Vojtěch Franc. Optimized invariant representation of network traffic for detecting unseen malware variants. In *Proceedings of the 25th USENIX Security Symposium*, pages 807–822 , USENIX Association, 2016. **(60%)**
3. **Karel Bartoš**, Martin Rehák, Michal Svoboda. Self-organized collaboration of distributed IDS sensors. In *Proceedings of the 9th Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, pages 214–231, Springer Berlin Heidelberg, 2013. **(50%)**
4. **Karel Bartoš**, Martin Rehák. Towards efficient flow sampling technique for anomaly detection. In *Traffic Monitoring and Analysis - Lecture Notes in Computer Science*, pages 93–106, volume 7189, Springer Berlin Heidelberg, 2012. **(50%)**
5. Martin Rehák, Michal Pěchouček, Martin Grill, Jan Stiborek, **Karel Bartoš**. Game theoretical adaptation model for intrusion detection system. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1123–1124, 2011. **(20%)**
6. Pavel Čeleda, Martin Rehák, Vojtěch Krmíček, **Karel Bartoš**. Flow based security awareness framework for high-speed networks. In *Proceedings of the Conference on Security and Protection of Information (SPI)*, pages 3–13, Univerzita Obrany, 2009. **(25%)**
7. Martin Rehák, Eugen Staab, Michal Pěchouček, Jan Stiborek, Martin Grill, **Karel Bartoš**. Dynamic information source selection for intrusion detection systems. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1009–1016, ACM Press, 2009. **(15%)**
8. Martin Rehák, **Karel Bartoš**, Martin Grill, Jan Stiborek, Michal Svoboda. Monitoring sítí pomocí NetFlow dat - od paketů ke strategiím. In *Proceedings of Česká společnost uživatelů otevřených systémů EurOpen.cz*, pages 75–81, 2009 **(20%)**
9. Martin Rehák, Michal Pěchouček, Pavel Čeleda, Vojtěch Krmíček, Martin Grill, **Karel Bartoš**. Multi agent approach to network intrusion detection (demo paper). In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1695–1696, ACM Press, 2008. **(16%)**

References

1. Cisco netflow. http://www.cisco.com/warp/public/732/tech/netflow.
2. Cisco Systems, Inc. – Cognitive Threat Analytics (CTA). http://cognitive.cisco.com/.
3. List of 1 million top web sites. http://www.alexa.com.
4. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, 3:262–294, November 2000.
5. Cisco visual networking index: Forecast and methodology, 2013 -– 2018. In *Cisco Visual Networking Index (VNI) White Papers*, http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/white-paper-listing.html, 2014.
6. S. Ali, I. U. Haq, S. Rizvi, N. Rasheed, U. Sarfraz, S. A. Khayam, and F. Mirza. On mitigating sampling-induced accuracy loss in traffic anomaly detection systems. *SIGCOMM Comput. Commun. Rev.*, 40:4–16, June 2010.
7. S. Ali, I. U. Haq, S. Rizvi, N. Rasheed, U. Sarfraz, S. A. Khayam, and F. Mirza. On mitigating sampling-induced accuracy loss in traffic anomaly detection systems. *SIGCOMM Comput. Commun. Rev.*, 40(3):4–16, June 2010.
8. G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou. Network anomaly detection and classification via opportunistic sampling. *Netwrk. Mag. of Global Internetwkg.*, 23:6–12, January 2009.
9. G. Androulidakis and S. Papavassiliou. Improving network anomaly detection via selective flow-based sampling. *Communications, IET*, 2(3):399 –409, march 2008.
10. M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 24–24, Berkeley, CA, USA, 2012. USENIX Association.
11. B. Augustin and A. Mellouk. On traffic patterns of http applications. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6, Dec 2011.
12. M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In C. Kruegel, R. Lippmann, and A. Clark, editors, *Recent Advances in Intrusion Detection*, volume 4637 of *Lecture Notes in Computer Science*, pages 178–197. Springer Berlin Heidelberg, 2007.
13. K. Bartos, M. Grill, V. Krmicek, M. Rehak, and P. Celeda. Flow based network intrusion detection system using hardware-accelerated netflow probes. In *In CESNET Conference 2008 : security, middleware, and virtualization – glue of future networks*, pages 49–56, 2008.
14. K. Bartos and M. Rehak. *Distributed Self-organized Collaboration of Autonomous IDS Sensors*, pages 113–117. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
15. K. Bartos and M. Rehak. Self-organized mechanism for distributed setup of multiple heterogeneous intrusion detection systems. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2012 IEEE Sixth International Conference on*, pages 31–38, Sept 2012.
16. K. Bartos and M. Rehak. Towards efficient flow sampling technique for anomaly detection. In A. Pescapè, L. Salgarelli, and X. Dimitropoulos, editors, *Traffic Monitoring and Analysis*, volume 7189 of *Lecture Notes in Computer Science*, pages 93–106. Springer Berlin Heidelberg, 2012.

17. K. Bartos and M. Rehak. Trust-based solution for robust self-configuration of distributed intrusion detection systems. In *20th European Conference on Artificial Intelligence (ECAI)*, pages 121–126. IOS Press, 2012.
18. K. Bartos and M. Rehak. Ifs: Intelligent flow sampling for network security–an adaptive approach. *International Journal of Network Management*, 25(5):263–282, 2015.
19. K. Bartos, M. Rehak, and V. Krmicek. Optimizing flow sampling for network anomaly detection. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 1304 –1309, july 2011.
20. K. Bartos, M. Rehak, and M. Svoboda. *Self-organized Collaboration of Distributed IDS Sensors*, pages 214–231. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
21. K. Bartos and M. Sofka. *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III*, chapter Robust Representation for Domain Adaptation in Network Security, pages 116–132. Springer International Publishing, Cham, 2015.
22. K. Bartos, M. Sofka, and V. Franc. Learning invariant representation for malicious network traffic detection. In *22th European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2016.
23. K. Bartos, M. Sofka, and V. Franc. Optimized invariant representation of network traffic for detecting unseen malware variants. In *25th USENIX Security Symposium (USENIX Security 16)*, Austin, TX, Aug. 2016. USENIX Association.
24. T. Bass. Multisensor data fusion for next generation distributed intrusion detection systems. In *In Proceedings of the IRIS National Symposium on Sensor and Data Fusion*, pages 24–27, 1999.
25. T. Bass. Intrusion detection systems and multisensor data fusion. *Commun. ACM*, 43:99–105, April 2000.
26. S. Ben-David, J. Blitzer, K. Crammer, F. Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.
27. L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian. Traffic classification on the fly. *ACM SIGCOMM '06*, 36(2):23–26, Apr. 2006.
28. D. Bertsekas. *Dynamic programming and optimal control*, volume 1, 3rd ed. Belmont, MA: Athena Scientific, 2005.
29. L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel. Disclosure: Detecting botnet command and control servers through large-scale netflow analysis. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 129–138, New York, NY, USA, 2012. ACM.
30. J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.
31. B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13:422–426, July 1970.
32. A. Blum and Y. Mansour. Learning, regret minimization and equilibria. In *Algorithmic Game Theory*, chapter 4, pages 79–101. Cambridge University Press, 2007.
33. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM.
34. D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina. Impact of packet sampling on anomaly detection metrics. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 159–164, New York, NY, USA, 2006. ACM.
35. L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156 –172, march 2008.
36. R. Bye, S. A. Camtepe, and S. Albayrak. Collaborative intrusion detection framework: characteristics, adversarial opportunities and countermeasures. In *Proceedings of the 2010 international conference on Collaborative methods for security and privacy*, CollSec'10, pages 1–1, Berkeley, CA, USA, 2010. USENIX Association.
37. V. Carela-Español, P. Barlet-Ros, A. Cabellos-Aparicio, and J. Solé-Pareta. Analysis of the impact of sampling on netflow traffic classification. *Computer Networks*, 55(5):1083 – 1099, 2011.

38. K. M. Carter, N. Idika, and W. W. Streilein. Probabilistic threat propagation for malicious activity detection. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2940–2944, May 2013.
39. V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41:15:1–15:58, July 2009.
40. C.-Y. Chiu, Y.-J. Lee, C.-C. Chang, W.-Y. Luo, and H.-C. Huang. Semi-supervised learning for false alarm reduction. In P. Perner, editor, *Advances in Data Mining. Applications and Theoretical Aspects*, volume 6171 of *Lecture Notes in Computer Science*, pages 595–605. Springer Berlin / Heidelberg, 2010.
41. B.-Y. Choi and Z.-L. Zhang. Adaptive random sampling for traffic volume measurement. *Telecommunication Systems*, 34(1-2):71–80, 2007.
42. H. Choi, B. B. Zhu, and H. Lee. Detecting malicious web links and identifying their attack types. In *Proceedings of the 2Nd USENIX Conference on Web Application Development*, WebApps'11, pages 11–11, Berkeley, CA, USA, 2011. USENIX Association.
43. I. Corona, G. Giacinto, C. Mazzariello, F. Roli, and C. Sansone. Information fusion for computer security: State of the art and open issues. *Information Fusion*, 10(4):274 – 284, 2009. Special Issue on Information Fusion in Computer Security.
44. J. W. Crandall and M. A. Goodrich. Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. *Mach. Learn.*, 82:281–314, March 2011.
45. F. Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th Annual Computer Security Applications Conference*, pages 22–, Washington, DC, USA, 2001. IEEE Computer Society.
46. F. Cuppens and A. Miège. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 202–, Washington, DC, USA, 2002. IEEE Computer Society.
47. W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.
48. N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 99–108, New York, NY, USA, 2004. ACM.
49. D. Dash, B. Kveton, J. M. Agosta, E. Schooler, J. Chandrashekar, A. Bachrach, and A. Newman. When gossip is good: distributed probabilistic inference for detection of slow network intrusions. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, pages 1115–1122. AAAI Press, 2006.
50. H. Debar, D. Curry, and B. Feinstein. The intrusion detection message exchange format (idmef). *rfc 4765 March*, (4765), 2007.
51. H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, RAID '00, pages 85–103, London, UK, 2001. Springer-Verlag.
52. A. V. Debra Anderson, Thane Frivold. Next-generation intrusion detection expert system (nides) - a summary. Technical Report SRI-CSL-95-07, SRI International, Menlo Park, CA 94025-3493, May 1995. This report was prepared for the Department of the Navy, Space and Naval Warfare Systems Command, under Contract N00039-92-C-0015.
53. D. E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13:222–232, February 1987.
54. L. Duan, I. W. Tsang, and D. Xu. Domain transfer multiple kernel learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):465–479, 2012.
55. N. Duffield and C. Lund. Predicting resource usage and estimation accuracy in an ip flow measurement collection infrastructure. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, pages 179–191, New York, NY, USA, 2003. ACM.
56. N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. *IEEE/ACM Transactions on Networking*, 13:933–946, October 2005.
57. H. T. Elshoush and I. M. Osman. Alert correlation in collaborative intelligent intrusion detection systems–a survey. *Applied Soft Computing*, In Press,, 2011.

58. J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, MineNet '06, pages 281–286, New York, NY, USA, 2006. ACM.
59. L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas. Minds - minnesota intrusion detection system. In *Next Generation Data Mining*. MIT Press, 2004.
60. C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '02, pages 323–336, New York, NY, USA, 2002. ACM.
61. N. Falliere. Sality: Story of a peer-to-peer viral network. *Rapport technique, Symantec Corporation*, 2011.
62. G. Farnham and K. Leune. Tools and standards for cyber threat intelligence projects. Technical report, SANS Institute InfoSec Reading Room, 10 2013.
63. D. Fisch, E. Kalkowski, and B. Sick. Collaborative learning by knowledge exchange. In C. Müller-Schloer, H. Schmeck, and T. Ungerer, editors, *Organic Computing — A Paradigm Shift for Complex Systems*, volume 1 of *Autonomic Systems*, pages 267–280. Springer Basel, 2011.
64. D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. Mcclung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *in Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*, 2000.
65. S. Garcia, V. Uhlir, and M. Rehak. Identifying and modeling botnet c&c behaviors. In *International Workshop on Agents and CyberSecurity ACySe 2014, in Autonomous Agents and Multiagent Systems*, 2014.
66. P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1):18–28, 2009.
67. R. Gray. *Entropy and Information Theory*. Springer-Verlag, 1990.
68. A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.
69. L. A. Grieco and C. Barakat. An analysis of packet sampling in the frequency domain. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 170–176, New York, NY, USA, 2009. ACM.
70. K. Griffin, S. Schneider, X. Hu, and T.-C. Chiueh. Automatic generation of string signatures for malware detection. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 101–120, Berlin, Heidelberg, 2009. Springer-Verlag.
71. M. Grill, I. Nikolaev, V. Valeros, and M. Rehák. Detecting dga malware using netflow. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1304–1309, May 2015.
72. M. Grill and T. Pevný. Learning combination of anomaly detectors for security domain. *Computer Networks*, pages –, 2016.
73. M. Grill, T. Pevný, and M. Rehák. Reducing false positives of network anomaly detection by local adaptive multivariate smoothing. *Journal of Computer and System Sciences*, pages –, 2016.
74. G. Gu, R. Perdisci, J. Zhang, W. Lee, et al. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium*, volume 5, pages 139–154, 2008.
75. N. Hohn and D. Veitch. Inverting sampled traffic. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, pages 222–233, New York, NY, USA, 2003. ACM.
76. C. Hu, S. Wang, J. Tian, B. Liu, Y. Cheng, and Y. Chen. Accurate and efficient traffic monitoring using adaptive non-linear sampling method. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.
77. H. Huang, L. Qian, and Y. Wang. A svm-based technique to detect phishing urls. *Information Technology Journal*, 11(7):921–925, 2012.
78. T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *ECAI*, pages 18–22, 2004.
79. L. Invernizzi, S. Miskovic, R. Torres, S. Saha, S. Lee, M. Mellia, C. Kruegel, and G. Vigna. Nazca: Detecting malware distribution in large-scale networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

80. A. Iyer, S. Nath, and S. Sarawagi. Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 530–538, 2014.
81. N. Jagpal, E. Dingle, J.-P. Gravel, P. Mavrommatis, N. Provos, M. A. Rajab, and K. Thomas. Trends and lessons from three years fighting malicious extensions. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 579–593, Washington, D.C., Aug. 2015. USENIX Association.
82. J. S. J. K. Jan Jusko, Martin Rehak and T. Pevny. Using behavioral similarity for botnet command-and-control discovery. *To appear at Intelligent Systems, IEEE.*
83. K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Trans. Inf. Syst. Secur.*, 6:443–471, November 2003.
84. I. N. Junejo, E. Dexter, I. Laptev, and P. Perez. View-independent action recognition from temporal self-similarities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):172–185, 2011.
85. J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 211–225, May 2004.
86. J. Jusko and M. Rehak. Identifying peer-to-peer communities in the network by connection graph analysis. *International Journal of Network Management*, 24(4):235–252, 2014.
87. A. Kapravelos, Y. Shoshitaishvili, M. Cova, C. Kruegel, and G. Vigna. Revolver: An automated approach to the detection of evasive web-based malware. In *USENIX Security*, pages 637–652. Citeseer, 2013.
88. T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: Multilevel traffic classification in the dark. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 229–240, New York, NY, USA, 2005. ACM.
89. R. A. Kemmerer. Nstat: A model-based real-time network intrusion detection system. Technical report, Santa Barbara, CA, USA, 1998.
90. N. Kheir. Behavioral classification and detection of malware through {HTTP} user agent anomalies. *Journal of Information Security and Applications*, 18(1):2 – 13, 2013. SETOP'2012 and FPS'2012 Special Issue.
91. H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee. Internet traffic classification demystified: Myths, caveats, and the best practices. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 11:1–11:12, New York, NY, USA, 2008. ACM.
92. J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, Mar 1998.
93. C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 251–261, New York, NY, USA, 2003. ACM.
94. A. Lakhina, M. Crovella, and C. Diot. Diagnosis Network-Wide Traffic Anomalies. In *ACM SIGCOMM '04*, pages 219–230, New York, NY, USA, 2004. ACM Press.
95. A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies using Traffic Feature Distributions. In *ACM SIGCOMM, Philadelphia, PA, August 2005*, pages 217–228, New York, NY, USA, 2005. ACM Press.
96. J. Li, D.-Y. Lim, and K. Sollins. Dependency-based distributed intrusion detection. In *Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, pages 8–8, Berkeley, CA, USA, 2007. USENIX Association.
97. H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16–24, 2013.
98. M. Locasto, J. J. Parekh, A. D. Keromytis, and S. J. Stolfo. Towards collaborative security and p2p intrusion detection. In *In Proceedings of the IEEE Information Assurance Workshop (IAW*, pages 333–339, 2005.
99. W. Lou, G. Liu, H. Lu, and Q. Yang. Cut-and-pick transactions for proxy log mining. In C. Jensen, S. Šaltenis, K. Jeffery, J. Pokorny, E. Bertino, K. Böhn, and M. Jarke, editors, *Advances in Database Technology — EDBT 2002*, volume 2287 of *Lecture Notes in Computer Science*, pages 88–105. Springer Berlin Heidelberg, 2002.
100. J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Learning to detect malicious urls. *ACM Trans. Intell. Syst. Technol.*, 2(3):30:1–30:24, May 2011.

101. J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang. Is sampled data sufficient for anomaly detection? In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, pages 165–176, New York, NY, USA, 2006. ACM.

102. J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang. Is sampled data sufficient for anomaly detection? In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 165–176, New York, NY, USA, 2006. ACM.

103. J. Mai, A. Sridharan, C.-N. Chuah, H. Zang, and T. Ye. Impact of packet sampling on portscan detection. *Selected Areas in Communications, IEEE Journal on*, 24(12):2285–2298, Dec 2006.

104. C.-H. Mao, H.-M. Lee, D. Parikh, T. Chen, and S.-Y. Huang. Semi-supervised co-training and active learning based approach for multi-view intrusion detection. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 2042–2048, New York, NY, USA, 2009. ACM.

105. D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. Inferring internet denial-of-service activity. *ACM Trans. Comput. Syst.*, 24(2):115–139, May 2006.

106. B. Morin, L. Mé, H. Debar, and M. Ducassé. M2d2: a formal data model for ids alert correlation. In *Proceedings of the 5th international conference on Recent advances in intrusion detection*, RAID'02, pages 115–137, Berlin, Heidelberg, 2002. Springer-Verlag.

107. A. Moser, C. Kruegel, and E. Kirda. Exploring multiple execution paths for malware analysis. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 231–245, May 2007.

108. A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 421–430, Dec 2007.

109. M. Müller and M. Clausen. Transposition-invariant self-similarity matrices. In *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 47–50, 2007.

110. T. Nelms, R. Perdisci, M. Antonakakis, and M. Ahamad. Webwitness: Investigating, categorizing, and mitigating malware download paths. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 1025–1040, Washington, D.C., Aug. 2015. USENIX Association.

111. L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:2005, 2005.

112. I. Paredes-Oliva, P. Barlet-Ros, and J. Sole-Pareta. Scan detection under sampling: a new perspective. *Computer*, 46(4):38–44, April 2013.

113. I. Paredes-Oliva, P. Barlet-Ros, and J. Solé-Pareta. Portscan detection with sampled netflow. In M. Papadopouli, P. Owezarski, and A. Pras, editors, *Traffic Monitoring and Analysis*, volume 5537 of *Lecture Notes in Computer Science*, pages 26–33. Springer Berlin Heidelberg, 2009.

114. J. Peng and R. J. Williams. Incremental multi-step q-learning. *Machine Learning*, 22:283–290, 1996.

115. R. Perdisci, D. Ariu, and G. Giacinto. Scalable fine-grained behavioral clustering of http-based malware. *Computer Networks*, 57(2):487 – 500, 2013. Botnet Activity: Analysis, Detection and Shutdown.

116. R. Perdisci, G. Giacinto, and F. Roli. Alarm clustering for intrusion detection systems in computer networks. *Eng. Appl. Artif. Intell.*, 19:429–438, June 2006.

117. T. Pevny, M. Rehak, and M. Grill. Detecting anomalous network hosts by means of pca. In *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, pages 103–108, Dec 2012.

118. P. A. Porras, M. W. Fong, and A. Valdes. A mission-impact-based approach to infosec alarm correlation. In *Proceedings of the 5th international conference on Recent advances in intrusion detection*, RAID'02, pages 95–114, Berlin, Heidelberg, 2002. Springer-Verlag.

119. P. A. Porras and P. G. Neumann. Emerald: Event monitoring enabling responses to anomalous live disturbances. *Proc 20th NISTNCSC National Information Systems Security Conference*, pages 353–365, 1997.

120. L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*, pages 5–8, 2001.

121. M. Rehák, M. Grill, and J. Stiborek. On the Value of Coordination in Distributed Self-Adaptation of Intrusion Detection System. In *Proceedings Web Intelligence and Intelligent Agent Technology WI-IAT11*, pages 196–203, Los Alamitos, CA, 2011. IEEE Computer Soc.

122. M. Rehak, M. Grill, and J. Stiborek. On the value of coordination in distributed self-adaptation of intrusion detection system. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 02*, WI-IAT '11, pages 196–203, Washington, DC, USA, 2011. IEEE Computer Society.

123. M. Rehak, M. Pechoucek, M. Grill, J. Stiborek, K. Bartos, and P. Celeda. Adaptive multiagent system for network traffic monitoring. *Intelligent Systems, IEEE*, 24(3):16 –25, may-june 2009.

124. M. Rehák, E. Staab, V. Fusenig, M. Pěchouček, M. Grill, J. Stiborek, K. Bartoš, and T. Engel. *Runtime Monitoring and Dynamic Reconfiguration for Intrusion Detection Systems*, pages 61–80. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

125. M. Rehak, E. Staab, V. Fusenig, J. Stiborek, M. Grill, K. Bartos, M. Pechoucek, and T. Engel. Threat-model-driven runtime adaptation and evaluation of intrusion detection system. In *Proceedings of the 6th international conference on Autonomic computing*, ICAC '09, pages 65–66, New York, NY, USA, 2009. ACM.

126. K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov. Learning and classification of malware behavior. In D. Zamboni, editor, *Detection of Intrusions and Malware, and Vulnerability Assessment*, volume 5137 of *Lecture Notes in Computer Science*, pages 108–125. Springer Berlin Heidelberg, 2008.

127. N. Rndic and P. Laskov. Practical evasion of a learning-based classifier: A case study. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 197–211, May 2014.

128. M. Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association.

129. K. Scarfone and P. Mell. Guide to intrusion detection and prevention systems (idps) recommendations of the national institute of standards and technology. *Nist Special Publication*, 800(94), 2007.

130. A. Servin and D. Kudenko. Multi-agent reinforcement learning for intrusion detection. In *Proceedings of the 5th , 6th and 7th European conference on Adaptive and learning agents and multi-agent systems: adaptation and multi-agent learning*, ALAMAS'05/ALAMAS'06/ALAMAS'07, pages 211–223, Berlin, Heidelberg, 2008. Springer-Verlag.

131. R. Sharma and M. Gopal. Synergizing reinforcement learning and game theory–a new direction for control. *Applied Soft Computing*, 10(3):675–688, 2010.

132. H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

133. C. Sierra and J. Debenham. An information-based model for trust. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, AAMAS '05, pages 497–504, New York, NY, USA, 2005. ACM.

134. S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur. Internet besieged. chapter DIDS (distributed intrusion detection system) — motivation, architecture, and an early prototype, pages 211–227. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1998.

135. D. Song, D. Brumley, H. Yin, J. Caballero, I. Jager, M. Kang, Z. Liang, J. Newsome, P. Poosankam, and P. Saxena. Bitblaze: A new approach to computer security via binary analysis. In R. Sekar and A. Pujari, editors, *Information Systems Security*, volume 5352 of *Lecture Notes in Computer Science*, pages 1–25. Springer Berlin Heidelberg, 2008.

136. H. Song and J. Turner. Toward advocacy-free evaluation of packet classification algorithms. *Computers, IEEE Transactions on*, 60(5):723–733, May 2011.

137. K. Soska and N. Christin. Automatically detecting vulnerable websites before they turn malicious. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 625–640, San Diego, CA, Aug. 2014. USENIX Association.

138. A. Sperotto, M. Mandjes, R. Sadre, P.-T. de Boer, and A. Pras. Autonomic parameter tuning of anomaly-based idss: an ssh case study. *Network and Service Management, IEEE Transactions on*, 9(2):128–141, June 2012.

139. A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An overview of ip flow-based intrusion detection. *Communications Surveys Tutorials, IEEE*, 12(3):343–356, Third 2010.

140. A. Sridharan, T. Ye, and S. Bhattacharyya. Connectionless port scan detection on the backbone. *In Malware workshop, held in conjunction with IPCCC.*, 2006.

141. P. Standards. Standards for security categorization of federal information and information systems. *Security Management*, (February), 2004.

142. S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical automated detection of stealthy portscans. *J. Comput. Secur.*, 10:105–136, July 2002.

143. R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

144. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, Mar. 1998.

145. I. Szita and A. Lőrincz. The many faces of optimism: a unifying approach. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1048–1055, New York, NY, USA, 2008. ACM.

146. A. Valdes and K. Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, RAID '00, pages 54–68, London, UK, 2001. Springer-Verlag.

147. P. Velan, T. Jirsik, and P. Celeda. Design and evaluation of http protocol parsers for ipfix measurement. In T. Bauschert, editor, *Advances in Communication Networking*, volume 8115 of *Lecture Notes in Computer Science*, pages 136–147. Springer Berlin Heidelberg, 2013.

148. J. M. Vidal and E. H. Durfee. Predicting the expected behavior of agents that learn about agents: The clri framework. *Autonomous Agents and Multi-Agent Systems*, 6:77–107, January 2003.

149. C. Vincent Zhou, C. Leckie, and S. Karunasekera. Decentralized multi-dimensional alert correlation for collaborative intrusion detection. *J. Netw. Comput. Appl.*, 32:1106–1123, September 2009.

150. K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection. In E. Jonsson, A. Valdes, and M. Almgren, editors, *Recent Advances in Intrusion Detection*, volume 3224 of *Lecture Notes in Computer Science*, pages 203–222. Springer Berlin Heidelberg, 2004.

151. C. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8(3), 1992.

152. M. Wunder, M. L. Littman, and M. Babes. Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In *ICML'10*, pages 1167–1174, 2010.

153. K. Xu, Z.-L. Zhang, and S. Bhattacharrya. Reducing Unwanted Traffic in a Backbone Network. In *USENIX Workshop on Steps to Reduce Unwanted Traffic in the Internet (SRUTI)*, Boston, MA, July 2005.

154. K. Xu, Z.-L. Zhang, and S. Bhattacharyya. Profiling internet backbone traffic: behavior models and applications. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 169–180. ACM, 2005.

155. S. Xu. Collaborative attack vs. collaborative defense. In *Collaborative Computing: Networking, Applications and Worksharing*, volume 10 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 217–228. Springer Berlin Heidelberg, 2009.

156. L. Yang and G. Michailidis. Sampled based estimation of network traffic flow characteristics. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1775–1783, May 2007.

157. D. Ye, Q. Bai, M. Zhang, and Z. Ye. P2p distributed intrusion detections by using mobile agents. In *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, pages 259 –265, may 2008.

158. V. Yegneswaran, P. Barford, and S. Jha. Global intrusion detection in the domino overlay system. In *In Proceedings of Network and Distributed System Security Symposium (NDSS*, 2004.

159. H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda. Panorama: Capturing system-wide information flow for malware detection and analysis. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 116–127, New York, NY, USA, 2007. ACM.

160. K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 819–827. JMLR Workshop and Conference Proceedings, 2013.

161. P. Zhao and S. C. Hoi. Cost-sensitive online active learning with application to malicious url detection. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 919–927, New York, NY, USA, 2013. ACM.

162. C. V. Zhou, C. Leckie, and S. Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. *Computers Security*, 29(1):124–140, 2010.

163. J. Zhou, M. Heckman, B. Reynolds, A. Carlson, and M. Bishop. Modeling network intrusion detection alerts for correlation. *ACM Trans. Inf. Syst. Secur.*, 10, February 2007.

164. Q. Zhu, H. Tembine, and T. Basar. Network security configurations: A nonzero-sum stochastic game approach. In *American Control Conference (ACC), 2010*, pages 1059 –1064, 30 2010-july 2 2010.

165. M. Zinkevich, M. Johanson, M. H. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *NIPS*. MIT Press, 2007.

Appendix A
Baseline flow-level features

The classifiers use various features depending on the classification level. At the flow level, the system decomposes each URL into seven components (protocol, second-level domain, top-level domain, path, file name, query, and fragment), where i-th component is denoted as $\mathbf{c}_i$ (see Figure 4.5). On these components, the following functions are applied.

- **URL component length** $l(\mathbf{c}_i)$ - number of characters of the i-th component.
- **Ratio of consonant to vowel changes** $r_v(\mathbf{c}_i)$ - describes the frequency of changes between consonants and vowels (this ratio is specifically suitable for recognizing generated domains):

$$r_v(\mathbf{c}_i) = \frac{number\ of\ changes\ from\ consonant\ to\ vowel}{l(\mathbf{c}_i)}.$$

- **Maximum occurrence ratio of URL characters** - according to the observations, characters of some malicious URLs are distributed randomly. The ratio useful to identify this property is defined as a maximum number of occurrences of any character divided by the total number of characters in the URL component.
- **Maximum occurrence ratio of character type** - random distribution can be identified not only for the individual characters, but also for types of the characters (e.g. letters, numbers, special characters, etc.).
- **Repetitive changes of special characters** - a lot of URL addressable forms and views are long and complicated, but they typically have repetitive changes of two characters used to fill and separate individual fields of the corresponding form: '=' and '&'. This fact is beneficial for separating these type of URLs from the rest.

Specifically for the second-level domains, the system extracts the following features:

- **Probability of component trigrams** $P_t(\mathbf{d})$ - trigrams serve as a reliable indicator whether the domain is generated or not. Note that the probability distribution of trigrams was created from the domains listed in Alexa 1 million most popular domains [3]. These domains are more suitable for creating such distribution because of better robustness when dealing with various languages (as opposed to dictionary words). The probability of a trigram $\mathbf{t}(\mathbf{A})$ being part of a domain A listed in Alexa 1M most popular sites is defined as:

$$p_t(\mathbf{t}(\mathbf{A})) = \frac{number\ of\ \mathbf{t}(\mathbf{A})\ occurrences\ in\ Alexa}{number\ of\ all\ trigrams\ in\ Alexa}.$$

Then, a rank (or upper index) is assigned to each trigram describing the trigram frequency:

$$\forall i,j \leq |\mathbf{A}|: \; p_t(\mathbf{t}^{(i)}(\mathbf{A})) \geq p_t(\mathbf{t}^{(j)}(\mathbf{A})) \;\Leftrightarrow\; i \leq j,$$

where $p_t(\mathbf{t}^{(i)}(\mathbf{A}))$ denotes i-th most-frequent trigram from the Alexa list and $i \in \{1, \ldots, 1000\}$. Finally, we define ranking probability of a trigram $\mathbf{t}(\mathbf{d})$ being part of a legitimate domain $\mathbf{d}$ as follows:

$$\hat{p}_t(\mathbf{t}(\mathbf{d})) = \begin{cases} 1-(i-1)\cdot 10^{-4} & \exists i \leq 1000 : \mathbf{t}(\mathbf{d}) = \mathbf{t}^{(i)}(\mathbf{A}) \\ 0 & \textit{otherwise} \end{cases}.$$

Then the probability of domain trigrams $P_t(\mathbf{d})$ is defined as the average of ranking probabilities $\hat{p}_t(\mathbf{t}^{(j)}(\mathbf{d}))$.

- **Maximal probability of two adjacent trigrams** $m(\mathbf{d})$ - most longer domains contain meaningful words composed of at least two frequently used adjacent trigrams. This fact makes this feature very promising for removing false positives for generated domains. Intuitively, we define probability of two adjacent trigrams:

$$p_t(\mathbf{t}_j(\mathbf{d}), \mathbf{t}_{j+1}(\mathbf{d})) = \frac{\hat{p}_t(\mathbf{t}_j(\mathbf{d})) + \hat{p}_t(\mathbf{t}_{j+1}(\mathbf{d}))}{2}. \tag{A.1}$$

Then we take maximum of all values:

$$m(\mathbf{d}) = \max_j (p_t(\mathbf{t}_j(\mathbf{d}), \mathbf{t}_{j+1}(\mathbf{d}))). \tag{A.2}$$

- **Number of suspicious trigrams** $n(\mathbf{d})$ - majority of generated domains are hard for humans to pronounce. Beside of consonant to vowel changes, this fact also increases the frequency of non-typical and rarely used trigrams captured in number of suspicious trigrams:

$$n(\mathbf{d}) = \textit{number of trigrams with } \hat{p}_t(\mathbf{t}(\mathbf{d})) = 0. \tag{A.3}$$

Besides the above mention URL features, the system also extracts the following flow-based features: flow duration, number of bytes transferred from client to server and vice versa, user agent, referrer, MIME-type and HTTP status.

Appendix B
Examples of Bags

Asterope	
hxxp://194.165.16.146:8080/pgt/?ver=1.3.3398&id=126&r=12739868&os=6.1—2—8.0.7601.18571&res=4—1921—466&f=1	
hxxp://194.165.16.146:8080/pgt/?ver=1.3.3398&id=126&r=15425581&os=6.1—2—8.0.7601.18571&res=4—1921—516&f=1	
hxxp://194.165.16.146:8080/pgt/?ver=1.3.3398&id=126&r=27423103&os=6.1—2—8.0.7601.18571&res=4—1921—342&f=1	
Click-fraud, malvertising-related botnet	
hxxp://directcashfunds.com/opntrk.php?tkey=024f9730e23f8553c3e5342568a70300&Email=name.surname@company.com	
hxxp://directcashfunds.com/opntrk.php?tkey=c1b6e3d50632d4f5c0ae13a52d3c4d8d&Email=name.surname@company.com	
hxxp://directcashfunds.com/opntrk.php?tkey=7c9a843ce18126900c46dbe4be3b6425&Email=name.surname@company.com	
Dridex	
hxxp://27.54.174.181/8qV578&$o@HU6Q6S/gz$J0l=iTTH 28%2CM/we20%3D	
hxxp://27.54.174.181/C4GyRx%7E@RY6x /M&N=sq/bW_ra4OTJ	
hxxp://27.54.174.181/gPvh+=GO/9RPPfk0%2CzXOYU%20/Vq8Ww/+a_m%7Ez	
hxxp://27.54.174.181/qE0my4KIz48Cf3H8wG%7Evpz=iJ%26fqMl%24m/46JoELp=GJww%3D%26Ib+Ar.y3 iu%2D1E/sso	
InstallCore	**Monetization**
hxxp://rp.any-file-opener.org/?pcrc=1559319553&v=2.0	hxxp://utouring.net/search/q/conducing
hxxp://rp.any-file-opener.org/?pcrc=1132521307&v=2.0	hxxp://utouring.net/go/u/1/r/1647
hxxp://rp.any-file-opener.org/?pcrc=1123945956&v=2.0	hxxp://utouring.net/go/u/0/r/2675
hxxp://rp.any-file-opener.org/?pcrc=1075608192&v=2.0	hxxp://utouring.net/search/f/1/q/refiles
Poweliks	
hxxp://31.184.194.39/query?version=1.7&sid=793&builddate=114&q=nitric+oxide+side+effects&ua=Mozilla ... &lr=7&ls=0	
hxxp://31.184.194.39/query?version=1.7&sid=793&builddate=114&q=weight+loss+success+stories&ua=Mozilla ... &lr=0&ls=0	
hxxp://31.184.194.39/query?version=1.7&sid=793&builddate=114&q=shoulder+pain&ua=Mozilla%2F5 ... &lr=7&ls=2	
hxxp://31.184.194.39/query?version=1.7&sid=793&builddate=114&q=cheap+car+insurance&ua=Mozilla%2F5 ... &lr=7&ls=2	
Zeus	
hxxp://130.185.106.28/m/IbQFdXVjiriLva4KHeNpWCmThrJBn3f34HNwsLVVsUmLXtsumSSPe/zzXtIu9SzwjI ... 3RqvGzKN5	
hxxp://130.185.106.28/m/IbQJFUVjgZn4vx4KHeNpWCmThrJBn3f34HNwsLVVsUmLfkoPaSS+S+zzXtIu9SzwjI ... 3vKwmk0oUi	
hxxp://130.185.106.28/m/IbQJFUVjiJwJBX4KHeNpWCmThrJBn3f34HNwsLVVsUmKH7ue2STvSkzzXtIu9SzwjI ... 3vKwmk0oUi	
hxxp://130.185.106.28/m/IbQNtVVji5/7Yp4KHeNpWCmThrJBn3f34HNwsLVVsUmLz4sO6YRvOjzzXtIu9SzwjI ... 3zB9057quqv	
Legitimate traffic	
hxxp://www.cnn.com/.a/1.73.0/js/vendor/usabilla.min.js	
hxxp://www.cnn.com/.element/ssi/auto/4.0/sect/MAIN/markets_wsod_expansion.html	
hxxp://www.cnn.com/.a/1.73.0/assets/sprite-s1dced3ff2b.png	
hxxp://www.cnn.com/.element/widget/video/videoapi/api/latest/js/CNNVideoBootstrapper.js	
Legitimate traffic	
hxxp://ads.adaptv.advertising.com/a/h/7g_doK40WLPMYHbkD9G2u7HSXjqz ... &context=fullUrl%3Dpandora.com	
hxxp://ads.adaptv.advertising.com/crossdomain.xml	
hxxp://ads.advertising.com/411f1e96-3bde-4d85-b17e-63749e5f0695.js	
hxxp://ads.advertising.com/ids/411f1e96-3bde-4d85-b17e-63749e5f0695	

Table B.1: URLs of flows from selected malicious categories together with two examples of legitimate URLs (at the bottom). You can see that URLs within each malicious bag are similar to each other (as opposed to most of legitimate bags). Small non-zero variability of flow-based feature values is projected into the proposed representation with both types of histograms, which makes the representation more robust to further malware changes and variants.