Security Configuration Management in Intrusion Detection and Prevention Systems

by

Khalid Alsubhi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2016

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Intrusion Detection and/or Prevention Systems (IDPS) represent an important line of defense against a variety of attacks that can compromise the security and proper functioning of an enterprise information system. IDPSs can be network or host-based and can collaborate in order to provide better detection of malicious traffic. Although several IDPS systems have been proposed, their appropriate configuration and control for effective detection/prevention of attacks and efficient resource consumption is still far from trivial. Another concern is related to the slowing down of system performance when maximum security is applied, hence the need to trade off between security enforcement levels and the performance and usability of an enterprise information system.

In this dissertation, we present a security management framework for the configuration and control of the security enforcement mechanisms of an enterprise information system. The approach leverages the dynamic adaptation of security measures based on the assessment of system vulnerability and threat prediction, and provides several levels of attack containment. Furthermore, we study the impact of security enforcement levels on the performance and usability of an enterprise information system. In particular, we analyze the impact of an IDPS configuration on the resulting security of the network, and on the network performance. We also analyze the performance of the IDPS for different configurations and under different traffic characteristics. The analysis can then be used to predict the impact of a given security configuration on the prediction of the impact on network performance.

Acknowledgements

All praise is due to Allah who guided me through out this research and beyond.

My deepest gratitude is to my advisor, Prof. Raouf Boutaba, for his guidance, support, patience, and encouragement. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. He was always available, professional, and friendly. I have learned from him to think differently, be optimistic, and be self motivated. This work would have not been done without his advice and valuable comments.

I would like to thank Prof. Ashraf Aboulnaga, Prof. Bernard Wong, Prof. Xuemin (Sherman) Shen and Prof. Oliver Festor for serving as the committee members for my thesis. I am grateful to them for their fruitful discussions and comments.

Special acknowledgment is due to King Abdulaziz University and the Saudi Arabian Cultural Bureau in Canada for their financial support to complete my PhD degree.

My sincere gratitude goes to all my colleagues Nizar Bouabdallah, Yasir Alhazmi, Faten Zahani, and Issam Aib for their time and cooperation. Many thanks also go to the administration support staff in the School of Computer Science, especially Margaret Towell, for their help and efforts. Special thanks go to Noura Limam for her encouragement and support.

I am forever indebted to my family for for their encouragement, continuing support, prayers, and love. My father, Mr. Ateatallah Alsubhi, and my mother, Salma Alharbi, and my siblings, have been the greatest supporters in my life.

Dedication

..to my Father and Mother

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Security threats come in a variety of forms, and new attacks are crafted on a daily basis from a variety of malicious sources and for different reasons, ranging from competitor-planned intrusions to amateur explorations. Intrusion types include worms, spam, viruses, buffer-overflow, denial-of-service (DoS), rootkits, malicious logins, etc. The increasing network connectivity of today's computing environments has significantly increased the potential damage that can be caused by these intrusions. The possible economic loss from a cyber-attack is sometimes unfathomable. For instance, the five most costly viruses of all time came to USD 102 billion in losses, where the most expensive single virus was MyDoom at USD 38.7 billion [60].

Among other security enforcement tools and mechanisms, Intrusion Detection and Pre-

vention Systems (IDPS) [235] represent an important line of defense against the variety of attacks that can compromise the security and proper functioning of an enterprise information system. IDPSs can be signature-based or anomaly-based. Signature-based IDPSs, such as Snort [230] and Bro [223], are the most dominant and are based on pre-knowledge of attack signatures, which help identify malicious from benign traffic. Anomaly-based IDPSs work differently in that they learn about the normal behavior of a system and then raise alerts whenever abnormal behavior is detected. IDPSs can be network- or host-based, and can collaborate in centralized or distributed clusters in order to provide better detection of malicious traffic across a distributed networked system.

The issue of optimal IDPS configuration and provisioning has always been difficult to deal with, mainly due to the overwhelming number of parameters to tune. IDPSs are generally shipped with a number of attack detection libraries (also known as categories [230] or analyzers [223]) with a considerable set of configuration parameters. The current version of the Snort IDPS (version 2.9) [230], for example, has approximately 21,500 signature rules in fifty eight categories. Each IDPS also comes with a default configuration to use when no additional information or expertise is available. It is not trivial to determine the optimal configuration of an IDPS because it is essential to understand the quantitative relationship between the wide range of analyzers and tuning parameters. This explains the reason why current IDPSs are configured and tuned based simply on a trial-and-error approach. Although there have been recent approaches, such as in [237, 252, 273], to optimize IDPS resource consumption, we still need to deal with resource constraints and make the best use of an IDPS with available resource budgets.

The proper configuration and management of an IDPS for effective attack detection

and prevention and efficient resource consumption is very challenging. For instance the configuration of the IDPS to enforce a maximum security level can negatively impact the performance and usability of the enterprise information system being protected. A trade-off between security and performance is necessary in this case which can be achieved through proper configuration. The protection capability of an IDPS realized by deploying preventive rules can significantly improve the security of the network. However, this can have a negative impact on network performance in terms of delay as the number of preventive rules increase. In addition, the IDPS may not be able to cope with the increase in the amount of processed traffic, mainly because of limited resources in terms of CPU and memory. The goal of this thesis is to study the trade-off between security and performance. The approach envisioned to achieve this goal is to analyze the performance of a security system and configure it in such a way to reduce performance bottleneck and preserve a sufficient level of security enforcement. The key difficulty faced here is the determination of the large number and variety of configuration parameters required for the analysis.

1.2 Challenges for IDPS Configuration Management

Although many IDPS systems have been proposed in the past, their proper configuration and management for effective detection and prevention of attacks are still far from trivial. A particularly important issue is the significant slowing down of system performance when maximum security is applied; hence arises the need to tradeoff between security enforcement levels on one hand and the performance and usability of an enterprise information system on the other. In our work, we focus on designing a framework for an effective

and dynamic IDPS configuration which addresses such tradeoff. In particular, we identify several key properties required for effective and manageable IDPSs. In this section, we discuss common weaknesses of existing IDPSs and the key properties we want to consider in our framework.

1.2.1 Security Enforcement and Performance

A primary requirement for the deployment of any security technology is the ability to protect against a range of attacks. Another requirement is related to avoiding performance degradations in the network due to unnecessary security measures. Current IDPSs do not provide adequate means of achieving these two conflicting requirements. We believe it is essential to strike a balance between performance and security requirements [29]. IDPSs in a detection mode (passive mode IDS) scrutinize copies of the packets sent over the network and raise alerts whenever hostile content is found. In contrast, IDPSs in a preventive mode (inline mode IPS) have the extra capability of preventing the attacks. IDSs fulfill network performance requirements but display poor defense capabilities, as attacks succeed. On the other hand, IPSs can shield networks by rejecting packets that match any malicious pattern, but this can negatively impact network performance as traffic increases. Therefore, modeling and analyzing the performance of IDPSs can be extremely useful in gaining a deeper understanding of IDPSs' behavior and characteristics. Furthermore, quantitatively evaluating the impact of IDPS configurations on the security of the enterprise information system is highly useful for security administrators for optimizing security configurations and policies; for instance, predicting the impact of enabling some preventive rules in IDPS

configurations on the resulting security of the system and on network performance is extremely valuable.

1.2.2 Dynamic Adaptation

The goal of dynamic adaptation of a security system is to adjust security enforcement level while preserving good system performance, or vice versa. The current trend lays an emphasis on fine-tuning the IDPS configuration to suit the environment and operating conditions where the IDPS will be deployed for better performance [259]. Some IDPSs are carefully designed to be very "lightweight" or are specially configured with high-end hardware (e.g., RealSecure with AppSwitch [180]) to cope with high-speed and high-volume traffic. Most IDPSs are statically configured without considering changes in the operational conditions. When the IDPS is under an overload condition due to a surge in the traffic (e.g., under DDoS attack), the value of the IDPS will drop as its configuration is not valid anymore. This problem is most often dismissed with the argument that it is acceptable if the IDPS drops some packets when there is a traffic spike. However, the fact is that the attacker can deliberately increase the traffic to "hide" the attack. In such cases, the reason for deploying the IDPS itself is lost. A report from NIST [235] shows that most current IDPSs are not capable of keeping up with high loads, or even moderate loads in some cases. The above points suggest the need for an IDPS to react to the changes in operating conditions and adjust itself in order to achieve the maximum benefit.

1.2.3 Performance Prediction and QoS Requirements

An IDPS may be able to efficiently detect and prevent attacks which can compromise network performance, however this often comes at a significant increase in processing delays as the attack signature database grows. Also, the IDPS may be unable to cope with increases in the amount of traffic due to limited resources in terms of CPU and memory. This can lead to an increase in queue length, resulting in longer waiting times for packets, and eventually packet losses. In this case, the IDPS itself becomes the network bottleneck. As a result, operators of IDPS face significant challenges in determining how to best configure and provision their systems. To do so, they need to understand and predict the resources consumption of such systems. Analyzing the performance of the IDPS in terms of resource consumption for different configurations and under different traffic characteristics allows the prediction of the impact on network operation, and is therefore valuable for achieving the best tradeoff between security objectives and QoS requirements.

1.3 Proposed Research Problems for Dissertation

In this dissertation, we provide an overview of existing IDPS designs and analyze their strengths and common weaknesses. We identify several key desired properties of IDPSs and propose a generalized framework which incorporates these properties for effective and manageable IDPSs. We also shed light on the significance of IDPS's performance using a quantitative approach in terms of the average service time under various conditions. Particularly, the execution time of rule-checking process in IDPS is re-examined in this work.

We also analyze the performance of the IDPS in terms of resource consumption for different configurations and under different traffic characteristics, which can help in predicting the impact on network operations. We then leverage the above analysis to provide mathematical derivations for key performance metrics, namely: throughput, queuing delay, system utilization, and packet loss at the IDPS level. Finally, based on our previous findings, we investigate cloud-based provisioning of IDPS as service. In summary, we identify five open problems significant enough to form the basis of my dissertation, which include: (i) policy-based adaptation for IDPS , (ii) performance analysis in IDPS, (iii) IDPS configuration management tradeoffs, (iv) queuing analysis in IDPS, and (iiv) elasticity of cloud-based IDPSs.

The remainder of this dissertation is organized as follow. In Chapter 2, we provide an overview of intrusion detection and prevention systems, their configuration management, and existing adaptive IDPS solutions finishing this chapter with a summary of related work in IDPS area. In Chapter 3 we propose a security configuration management framework for IDPSs. In Chapter 4, we develop a new analytical model to investigate the relationship between the IDPS performance and its configuration. Chapter 5 analyzes the impact of security enforcement levels on the performance and usability of the network. Chapter 6 aims to analyze the performance of the IDPS under different traffic characteristics. In Chapter 7 we introduce Elastic Virtual IDPS (vIDPS) problem and propose an efficient solution called *Simple Lazy Facility Location (SLFL)* to solve it. Finally, we draw our conclusions and potential future work in Chapter 8.

Chapter 2

Background and Related Work

2.1 Intrusion Detection and Prevention Systems

Intrusion detection is the process of monitoring and analyzing network traffic or computer activities for signs of possible malicious incidents, such as malware (e.g., worms, spyware) or unauthorized access or misused privileges that violate computer/network security policies. The intrusion prevention process works similar to that of detection, but with the additional function of stopping threatening incidents.

An Intrusion Detection System (IDS) is a software/hardware system designed to automate the intrusion detection process. It inspects the events occurring in a computer system or network for any suspicious activities. An IDS is designed to report intrusions that violate the system security policy to a security administrator, who could quickly act to minimize the damage caused by the intrusion. An Intrusion Prevention Systems (IPS),

however, has all the capabilities of an intrusion detection system plus the additional ability to stop potential attacks. In comparison with an IDS, an IPS can respond to a detected attack and prevent it from succeeding. The prevention can be done in many ways, such as filtering the malicious traffic using a firewall or terminating the network connection or user session that is involved in the attack. IDS and IPS technologies offer many of the same capabilities, and administrators can usually disable prevention features in IPS products, causing them to function as IDSs. Accordingly, for brevity the term "Intrusion Detection and Prevention Systems" (IDPS) is used throughout the rest of this thesis to refer to both IDS and IPS technologies. Any exceptions to that are specifically noted.

2.1.1 Types of IDPSs

Based on the data monitored, IDPSs can be host-based or network-based. A host-based IDPS (HIDPS) runs on an individual host or device in the network. It monitors inbound/outbound traffic to/from a computer as well as internal activities such as system calls and system logs. A HIDPS can only monitor an individual host and may not have a global view of the network activities. Some examples of HIDPSs are OSSEC [153] and Tripwire [170]. Network-based IDPSs (NIDPS) monitor the traffic to/from the network. A NIDPS contains sensors to sniff packets, and a data analyzer to process and correlate data. Alarms are raised whenever suspected intrusions are found. However, a NIDPS does not have knowledge about the internal activities of individual computers. Examples of NIDPSs are Snort [230] and Bro [223].

2.1.2 Detection Methodologies

IDPSs can be categorized into signature-based and anomaly-based, depending on the detection technique. Signature-based IDPSs compare data packets with the signatures or attributes of known intrusions in the database to decide whether the observed traffic is malicious or not. A signature-based IDPS is efficient in detecting known intrusions with fixed signatures. However, it is not efficient for detecting unknown intrusions or intrusions with polymorphic signatures. Pattern matching algorithms are in the heart of many IDPSs' signature-matching engines, as they determine the process of looking for specific patterns in network traffic. Pattern matching algorithms can be classified into single- and multi-pattern algorithms. In single-pattern matching algorithms, each pattern is searched in a given text individually. Knuth-Morris-Pratt [175] and Boyer-Moore [64] are some of the most widely used single pattern matching algorithms. Multi-pattern string matching algorithms search for a set of patterns in a body of text simultaneously. This is achieved by preprocessing the set of patterns and building an automaton that will be used in the matching phase to scan the text. Multi-pattern matching scales much better than algorithms that search for each pattern individually. Multi-pattern string matching algorithms include Aho-Corasick [21], Wu-Manber [292] and Commentz-Walter [88].

Many research studies have focused on accelerating the pattern-matching process for high-speed IDPSs (e.g., [188, 202]). The approaches to acceleration involve designing efficient pattern-matching algorithms [33, 217] , leveraging hardware implementation [40, 179, 193, 202], and program parallelization [222]. The interest in this issue persists in the research community of intrusion detection even until recently [146, 171, 178, 187, 270]. Many

existing solutions can accelerate the matching with flexible patterns such as regular expressions, achieving a linear time complexity even in the worst case. Given their excellent performance achieving up to multi-gigabit per second, the bottleneck due to pattern-matching seems to have been well addressed.

On the other hand, anomaly-based IDPSs observe traffic or computer activity and detect intrusions by identifying activities distinct from a user's or a system's normal behavior. Anomaly-based detection can detect unknown or new intrusions. However, it usually suffers from a high false positive rate problem. One example of anomaly-based implementations is the use of data mining algorithms to extract and identify valid, novel, and useful patterns in audit data. Several data mining algorithms have been used for anomaly-based IDPS including support vector machines [67, 169], genetic algorithms [66, 186], neural networks [211, 278], and clustering [135, 207, 212], all of which improve the detection accuracy of IDPS and assist in detecting new attacks.

2.2 IDPS Security Configuration Management

A set of parameters defines the configuration of an IDPS. The settings of these parameters at any instance of time influence the IDPS's detection capabilities as well as its performance characteristics. The issue of IDPS configuration management has always been difficult to deal with, mainly due to the overwhelming number of parameters to tune.

Signatures and Coverage: Signature-based IDPSs compare patterns against observed events to identify possible incidents. Each input (packet, event, flow, etc.) is com-

pared against thousands of signatures in a more-or-less brute-force manner. Signatures in most cases are written precisely to handle vulnerabilities, exploits, and other unknown bad sequences. However, poorly written signatures can lead to false positive detection where legitimate traffic mistakenly matches the attack signatures [91]. In an IDPS, each signature is defined by a rule that describes a known intrusion threat. A rule can be either a detective or a preventive rule. The detective rule aims to inspect a copy of a packet transmitted over the network and generate an alert when a malicious pattern exists in the packet header and/or content. Unlike the detective rule, the preventive rule is configured to be in-line so that traffic will be dropped if it carries a malicious pattern that matches the rule.

The IDPS coverage can be determined by the number of rules. The security administrator is responsible for including and excluding rules according to the specific needs of the protected network environment. For instance, Snort allows the enabling/disabling of rule libraries or individual rules through a set of configuration files. Furthermore, the security administrator can specify the mode of the rules as either detective or preventive mode.

2.2.1 Performance Analysis

The performance of IDPSs has attracted much attention in the past decade, especially with the ever-increasing volume of Internet traffic. If an IDPS is not efficient enough to keep up with the network speed, it results in dropping packets and consequently missing intrusions. Worse yet, if the system operates in a preventive mode (in-line), it will impose a direct limitation on the performance of an operational network notably by delaying the traffic.

A wide range of studies has been conducted to tackle the issues of IDPS performance, including modeling the performance, speeding up the deep packet inspection, adapting the configurations at the run-time system operations, and characterizing the operational performance with real traffic. In the following, we review some of the components that are important in characterizing and enhancing the performance of an IDPS.

2.2.2 Dynamic Adaptation and Re-configuration

Traditional intrusion detection and prevention systems rely heavily on the extensive knowledge of security experts about the networked system to be protected. However, today's networks are highly dynamic due to the frequent changes in the software/hardware deployed in the environment. Furthermore, security threats are constantly evolving and becoming more sophisticated. Current real-time IDPSs are statically configured and do not have the ability to adapt their configurations and workload at run-time, depending on the changing operational conditions [263]. Therefore, as operating conditions change due to fluctuation in the traffic profile or when under an overload attack, the original configuration becomes a non-optimal one and reduces the value of the IDPS. Performance adaptations via dynamic reconfiguration are necessary for an IDPS to counter such attacks. We argue that IDPS reconfigurability and performance adaptation must be considered not only during IDPS design and implementation, but also during IDPS operation after its deployment. Indeed, an IDPS should accomplish performance adaptation by optimizing its configuration at run-time.

2.3 Related Work

In this section, we will discuss the works from the literature that are closely related to each of the contributions of this thesis.

2.3.1 Policy-based Dynamic Adaptation

Using policies to drive security management has been extensively studied in the past [63]. However, existing work did not address the problem from the point of view of dynamic adaptation for the sake of balancing system performance and security.

The authors of [82] sought to transform an IDS system into an IPS by proposing a policy management for firewall devices integrated with intrusion prevention capabilities. They proposed an attack response matrix model which maps intrusion types to traffic enforcement actions. Their proposal is, however, only at the design level, and no concrete implementation or policy specifications have been provided. In addition, the authors do not consider trading off security enforcement levels and system performance but only how to transform an IDS into an IPS using policies.

Various proposals by Carver et al. [71–73] focused on dynamic mapping through an agent architecture (AAIRS). In this system, a host is monitored by multiple IDSs, and alarms are generated as needed. These alarms are first evaluated by the Master Analysis agent, which then notes the level of confidence of the attack and sends it to an Analysis agent, which then produces a tailored plan of response for the attack. Based on the same concern of capturing and modeling of security requirements, Albuquerque et al. pro-

posed policy-based management framework to automatically derive security configuration for enterprise networks [98]. Their proposal tackles the issue during the design stage of security configuration. However, coping with ongoing changes during operation time is not considered in this proposal.

The researchers in [128] put forward a management tool called MIRAGE to analyze, refine, and automatically deploy configuration settings related to network layers (i.e., VPN routers or firewalls). Their approach does not, however, cover security requirements' design phase. As well, updates of the components' configurations are not managed when comparing any discrepancies between the actual and the desired configurations. An interesting approach was proposed by Hassan [152] for the refinement of network security policies with regard to the low level security mechanisms. The model is policy-based in automating security mechanism implementation, and assesses the balance between the high-level security policy and the corresponding low-level security mechanism. While the approach focuses on full automation, no guidance is provided to check ambiguities and resolve any policy conflicts.

A system was proposed by Saleh et al. [234] to use a signature-based approach for attack detection, with prevention done by a packet-filtering firewall. The system utilized the Snort Network Intrusion Detection System (NIDS), which provides alerts on attack detection and forwards the alerts from NIDS to the firewall, which uses XML to form the incoming and outgoing network traffic rules. The proposed technique was suggested by the authors to be the most effective when the servers are placed in a "Demilitarized Zone". The proposed system's advantage is that it works when the attack signatures are already known, but it is unable to detect attacks whose signatures are unknown, and so the system

administrator must analyze any alerts sent by the NIDS. It is also prone to high rates of false positives and negatives, although it can be improved by modifying it to detect novel attacks [234].

A cost-sensitive model was proposed by Lee et al. [181] based on three factors: 1) operational cost, i.e., cost of processing; 2) damage cost, i.e., cost of damage when the IDS fails to work properly; and 3) response cost, , i.e., cost to apply the response to an attack. A cost-sensitive and response cost model were also proposed by Balepin et al. (2003), who provided a model to evaluate responses based on local resource dependency. Their method was to evaluate the system state in order to compute the cost of response. They found it difficult to produce a model that noted each resource's value, so resources were ranked by importance and high priority resources were allocated static costs.

In [259], Tanachaiwiwat et al. proposed a method to represent the reports of different IDSs into a matrix. These metrics give the number of triggered alarms depending on the attack, but also the number of false positives. This is followed by a risk assessment stage, where the possible cost of an attack and of the corresponding countermeasure is derived. Teo and Ahn [261] proposed a policy framework called Chameleos-x designed to enforce security policies on different kinds of equipment. Chameleos-x has three major components including a Management Console, a Translator, and Enforcement Monitors. The Management Console is used to send policies to the Enforcement Monitors that are located on each host. The Enforcement Monitors invoke the Translator to adapt the received policies into a system-specific ones which are then executed. However, their proposal lacks the support for explicit modeling of the network architecture, therefore it is difficult to understand the coverage and the applicability of the policies in different environments.

Finally, a business perspective to configuration management was provided as part of the Power prototype [74]. The latter supports the creation of policy hierarchies by means of a tool-assisted policy refinement, but the syntax used is a Prolog-like language, far from both business and administrators' views. The framework developed in the scope of the Positif project [47] provides an initial method to integrate policy specifications and anomaly detection systems (such as IDS) through policy-driven configuration of security services of networked systems within a single administrative domain. The desired behavior of the information security system (i.e., the IDS) can be implicitly achieved by means of policy specification. However, their proposal has not been implemented into a real system.

2.3.2 Performance Analysis and Optimization

The performance analysis of an IDPS involves many issues that go beyond the IDPS itself. Hardware platform, operating system, or ever the configuration setup of the IDPS are examples of such issues. However, the most important performance metric of an IDPS is to measure the systems ability to process traffic at high speed with minimum packet loss while working in real-time. Recent studies show that the IDPS rule checking process is a performance bottleneck where over 70% of the total CPU processing time of modern IDPSs is consumed by this process [69, 106, 107, 238, 287]. Accordingly, researchers have focused on devising solutions and algorithms, either software- or hardware-based, to improve the performance of this process. However, there has been little research on the issue of dynamic adaptation to balance system performance and security.

Significant efforts have been made in the past to improve the performance of the IDPS.

Schaelicke et al. [236] propose a methodology to measure the performance of rule-based IDPSs. They performed measurements for different packet payload sizes on a 100Mbit link, which was nearly saturated during the evaluation. The number of rules kept increasing until packet loss was reported, the maximum number of supported rules was then used as an indicator of performance. Furthermore, their study showed that the hardware platform, system parameters (*i.e.*, memory and CPU), and the operating system are important factors for improving the performance of IDPSs. In [107], authors present a model for monitoring the resource usage of IDPSs, and then predicting their resource consumption. Their goal is to fine tune the security level based on resource consumption. This work is clearly helpful to IDPSs operators in determining suitable security configurations and is closely related to ours. However, our investigations go a step further by analyzing the impact of IDPS configuration on average service time.

Lee et al. [180] propose a technique to measure the performance of an IDS by quantifying the benefits and costs of detection rules. Their goal is to dynamically determine the optimal configuration for an overloaded IDS to prevent data dropping under resource constraint and to trigger adaptation to current conditions. Their work is similar to ours in that it measures the expected service time of different IDS configuration sets to determine the optimal one. However, defining the cost and benefit metrics precisely is not an easy task and varies from one environment to another. Furthermore, considering the preventive capability of an IDPS, the analysis presented by Lee et al. seems inadequate. This is due to violation of the strict QoS requirement in terms of end-to-end delay caused by the prevention process.

2.3.3 Markovian based Queuing Analysis Model

Few studies have utilized mathematical models to analyze IDPS performance in different configurations. Most of these used queuing theory to model DDoS attacks and counter these attacks by devoting resources, regardless of how effective the detection and subsequent filtering algorithms are [50, 194, 283]. Resource demands and QoS were usually only estimated for benign users in a DDoS battle in these models. Yu et. al. [296] defined a queueing model DDoS attack mitigation in cloud environments. Their focus was to allocate necessary resources to the IDPS automatically and dynamically when a cloud-hosted server comes under DDoS attack. Similarly, Chang [75] considered a simple queueing model for a SYN flooding attack, which is a very common DoS attack. The impact of DoS attacks was analyzed by Khan and Traore [167] on three parameters used for attack detection: response time, queue-growth-rate, and arrival rate. Two queueing models were proposed by Long et al. [191] for the DoS attacks in order to measure packet loss probability, packet delay, and jitter. A generalized multi-class Erlang and Engset mixed-loss model was used by Huang et al. [159] to analyze a network under DDoS attacks, and the model was later extended to analyze 3G wireless cellular networks [160]. Different from the studies found in the literature, our work uses a more general queuing model with an embedded two-dimensional Markov chain, which more accurately captures the actual dynamics of DoS attacks.

The impact of an IDPS on network performance has received far less attention. Setting configuration parameters appropriately for the IDPS to ensure detection/prevention accuracy while avoiding negative impact on the quality of service is still a challenging problem. Hess et al. [158] tried to mitigate the impact of IPS services on the end-to-end delay.

They proposed an architecture consisting of an overlay network of IPSs implemented on programmable routers. However, such an architecture cannot be easily deployed, as it requires router programmability, which is not a common feature in commercial off-the-shelf routers. Dreger et al. studied the trade-off between security level and resource consumption [106, 107], while Lee et al. [180] put forward a method to determine the performance of an IDPS by quantifying the benefits and drawbacks of each detection rule. In order to reach the best possible configuration for an overloaded IDS, they proposed a heuristic-based algorithm. However, they did not rely on any analytical model, and thus it is hard to predict in advance the resulting performance for a given configuration.

In the area of firewalls, Salah et al. [233] derived an analytical queuing model based on an embedded Markov chain in order to analyze the performance of rule-based firewalls. As well, they provided closed-form expressions for a number of significant performance metrics, such as CPU utilization, service time, and mean throughput. Similar to our work, they assumed that the arrival rate follows a Poisson distribution and the service times are independent and exponentially distributed. However, their model cannot be utilised in the IDPS context because it has different processing stages compared to firewalls.

To the best of our knowledge, none of the existing works has proposed queuing-based modeling for IDPS systems that involve both header and content analysis stages. We develop an embedded Markov chain model, making the setting of the IDPS configuration possible analytically. Accordingly, the security administrator is able to select the appropriate configuration that achieves the desired trade-off between security and network performance.

2.3.4 Elastic Cloud-based IDPS Provisioning

Many mechanisms have been proposed to support elasticity through horizontal scaling, vertical scaling, and migration techniques [127].

Horizontal Scaling: Amazon Auto Scaling Group [244] offers tenant controlled horizontal scaling based on tenant-defined thresholds. Microsoft Azure [5] adapts the number of instances based on time, history, or size of workload. Clayman et al. [86] focus on dynamic virtual nodes placement to satisfy increasing demand by installing new virtual routers; however, no mechanism to release resources is supported. Stratos [130] uses a simple packing technique to elastically scale resources.

Vertical Scaling: CloudScale [247] and PRESS [138] scale by releasing or allocating CPU resources while ignoring network resources. Vertical scaling mechanisms are limited to individual physical machines [239]. Furthermore, changing compute or memory resources on-the-fly is not supported in most cases. In addition, vertical scaling requires rebooting the system causing potential SLA violations. Therefore, Cloud Providers (CPs) do not encourage vertical scaling mechanisms.

Migration: Migration is a popular technique to achieve elastic VM placement and server and network consolidation. pMapper [274] considers VM migration cost in its greedy heuristics to solve the optimal VM placement problem. Entropy [157] models the optimal VM placement as a variant of the vector bin-packing. However, both pMapper and Entropy ignore bandwidth requirement and locality in placement decisions. Kingfisher [246] employs both vertical and horizontal elasticity mechanisms by allocating more host resources, installing and migrating VM instances. It optimizes host resources from tenants'

standpoint while also considering the overhead (in terms of delay) introduced by these mechanisms. Here again, network resources such as bandwidth are also ignored.

MCRVMP [56] addresses the static VM placement problem to satisfy time-varying traffic demands of the VMs in addition to CPU and memory requirements. TVMPP [204] strives to reduce the aggregate traffic. However, it can lead to placement solutions with congested links, since it does not consider link capacity constraints. The authors of NAVP [196] focus on consolidating as much traffic demands as possible over the same set of network links in order to reduce the total energy consumption.

Recently, Bouet et al. [62] studied the placement of virtual Deep Packet Inspection (DPI) engines to optimize both the number of installed engines and their network footprint. However, the formulated problem is static and cannot handle elastic IDPS placement.

Chapter 3

AdapSec: Adaptive Security Management Framework

3.1 Introduction

Security configuration management is the task performed by network administrators to adequately configure security tools such as IDPSs in order to ensure enterprise network security and lower the risk of security breaches. In Chapter 2, we surveyed a number of existing IDPS security configuration management techniques that aim to provide solutions that lower the cost of configuration management, enhance efficiency and improve reliability. However, most of the previous studies focused either on improving the security of the enterprise regardless of the impact of its performance and productivity or vice versa. Only few studies addressed the tradeoffs between network security and performance. In this

chapter we propose an Adaptive Security management framework (AdapSec) for intrusion detection and prevention systems. The main goal of this framework is to provide configuration tools that dynamically adjust the parameters of the IDPS according to the high level objectives specified by the security administrator. It also includes tools that continuously monitor the performance of the security system and assist the administrator in taking decision regarding the security and performance. The security administrator has always to evaluate security risks and interact with the AdapSec to provide high level objectives in terms of security and performance. He can also increase and decrease the priority of some rules according to his assessment of vulnerabilities and threats.

We also highlight the need for and present a policy-based approach for the configuration and control of the security enforcement mechanisms of an enterprise information system. The approach relies on dynamic adaptation of security measures, based on the assessment of system vulnerability and threat prediction, and provides several levels of attack containment. The use of policy-based dynamic adaptation approach in IDPSs not only effectively improve the detection accuracy of the IDPS, it also preserve a good system performance and utilization. Our proposal is practically viable and can be used to assist security analysts to operate IDPS in more dynamic and adaptive way.

In evaluating the effectiveness of our policy-based adaptation approach, we have conducted comprehensive experiments based on a real implementation. More specifically, we have implemented a dynamic policy-based adaptation mechanism between the Snort signature-based IDPS and the lightweight anomaly-based *FireCol* IDS [29, 124]. We performed our experiments using the DARPA 2000 and 1999 intrusion detection evaluation datasets to show the viability of our approach. Our experimental results demonstrate how

good attack detection is feasible with a low resource overhead when the right set of rule categories and configuration parameters are enabled. The performance gains in terms of resource utilization as well as the ability to detect security threats and respond to them at the right time provided a proof of concept at least for the *FireCol*/Snort adaptation use case.

The remainder of this chapter is organized as follows. We proceed with describing our intrusion detection and prevention system and its security control system in Section 3.2 and 3.3 respectively. The policy-based security management design is presented in Section 3.4. In section 3.5, our *FireCol* anomaly-based IDS is described. Section 3.6 presents the experiments we have conducted using dynamic adaptation policies, *FireCol*, and Snort. Section 3.7 discusses the challenges related to policy-based security adaptation. Finally section 3.8 concludes Chapter with a discussion of future work.

3.2 Intrusion Detection and Prevention System

AdapSec has two major components, namely the security control system and the intrusion detection and prevention system. As shown in Figure 3.1, the intrusion detection and prevention system includes four components:

- **Monitoring module** is responsible for gathering statistics about (1) the incoming traffic (e.g., arrival rate, packet headers), (2) the IDPS performance (e.g., packet analysis time, packet drop rate, outgoing throughput) and (3) the IDPS resource consumption (e.g., CPU and memory). The collected information are then transferred

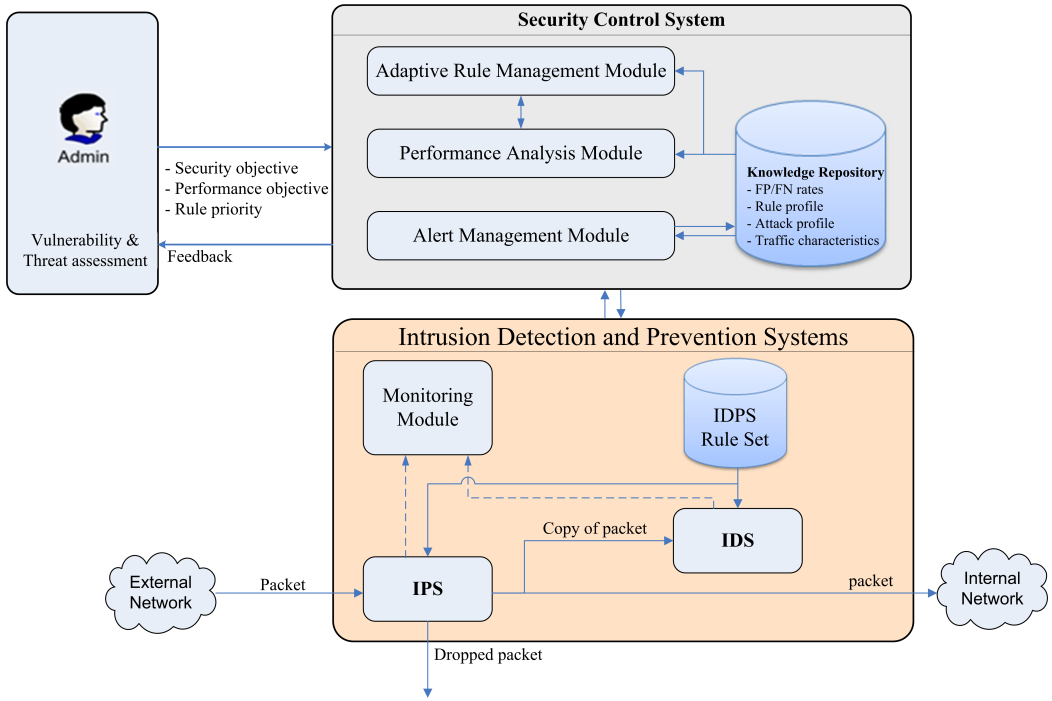


Figure 3.1: AdapSec architecture design

to the security control system for further analysis.

- **IDPS rule set** is the most critical component in IDPS which mostly define its configuration. A rule-set database contains a variety of pre-defined patterns that are matched against incoming packets for possible malicious contents. Two types of rules are considered, namely: detective and preventive rules. Detective rules are checked offline by the intrusion detection module whereas the preventive rules are checked inline by the intrusion prevention module.

- **Intrusion prevention module** is in charge of identifying and blocking anomalous traffic. The IPS operates inline where all traffic has to be queued and analyzed.

The intrusion prevention module drops traffic that is recognized as malicious by the preventive rules.

- **Intrusion detection module** receives copy of all packets in order to inspect them to detect suspicious activities that match the detection rules. It then reports detected intrusions to the security control system in order to evaluate the severity of the rule that has triggered the alert. The security control system can eventually then reclassify this rule as preventive.

3.3 Security Control System

The security control system is in charge of storing statistics and logs reported by the IDPS, analyzing them and adjusting the configuration of the IDPS according to the required objectives not only in terms of security but also in terms of performance as well. It mainly consists of four components:

- **Knowledge repository** stores the statistics collected from the monitoring module or provided by the performance analysis module.

- **Performance analysis module** provides models and tools used to analyze the performance of a particular IDPS configuration. For instance, we include in this module a new probabilistic model to evaluate the relationship between the IDPS configuration and its performance (i.e., the average service time per packet). We also develop an analytical model based on embedded Markov chains, which allows to predict the

impact of the configuration on network performance (e.g., the throughput, queuing delay, system utilization, and packet loss caused by IDPS). Such a model can help the security administrator to find the right trade-off between security enforcement levels and Quality of Service (QoS) requirements.

- **Adaptive Rule Management Module** is in charge of adapting the IDPS configuration based on runtime statistics collected by the monitoring module and analyzed by the performance analysis module. This module mainly selects which rule is preventive and which rule is detective. Rules in detective mode have no impact on the network performance (e.g., delay) as they are checked separately and offline by the intrusion detection module using a copy of the traffic. Prevention rules are checked inline by the intrusion prevention module and hence may impact the network performance, (e.g., a high processing delay can result in a high packet delay). We propose a rule mode selection optimization technique that determines an appropriate IDPS configuration set in order to maximize the security enforcement levels while minimizing network performance degradation.

- **Alert management module** receives reports from IDPS system about detected and prevented events. Security administrator then analyzes these reports for drawing a conclusion about the accuracy and performance of the IDPS.

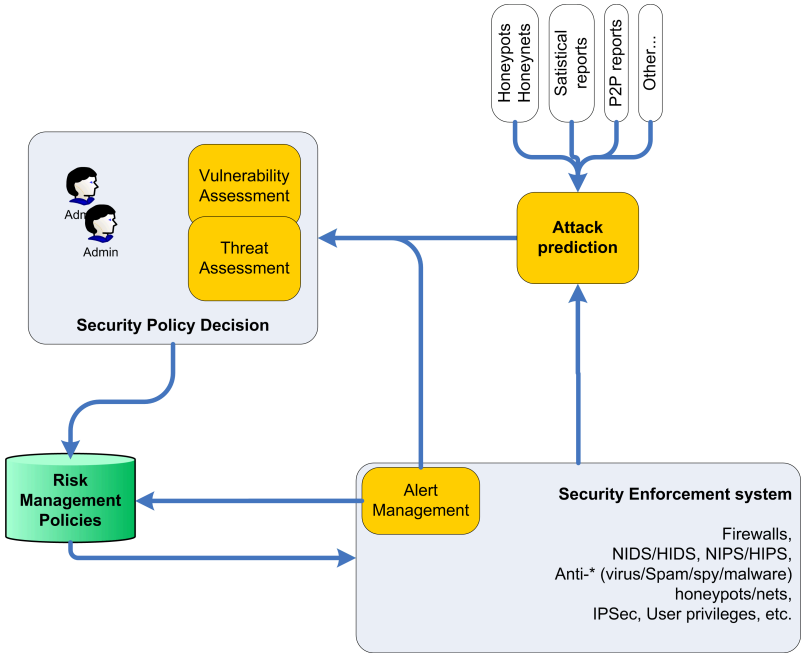


Figure 3.2: Policy-based Dynamic Adaptation Module

3.4 Policy-based Dynamic Adaptation

The policy-based security management design we propose is part of the adaptive rule management module presented in the previous section. It uses system administrator policies to balance the security measures employed by the enterprise information system. As depicted in Figure 3.2, the adaptive management of the enterprise security is provided at the low level through the repository of risk management policies. This repository is maintained by the security administrator(s) based on previous experince as well as the input from deployed threat prediction tools.

Threat prediction tools assess the degree of vulnerability of the enterprise system with

regard to a range of potential attacks. They gather information from external threat evaluation sources and generate reports on the risks that are likely to threaten the information system in the near future. Threat evaluation sources can be of different kinds, internal and external. Honeypots [225] represent an effective means to detect and learn about security threats. Statistical reports and peer security alerts from trusted parties represent another valuable source of information. For instance, in this work we use *FireCol* [125] as an alert source for Distributed Denial of Service (DDoS) attacks. The *FireCol* system forms rings of protection around a subscribed customer and lies outside the premises of the enterprise network.

The overall assessment of the system security and potential forthcoming threats is translated into risk management policies (Figure 3.2). These policies adapt the behavior of the underlying security measures accordingly. The translation is done by the threat assessment and security policy decision component. As it is not trivial to automatically assess and decide which security policy to enforce, human administrators are expected to be involved in this process.

The underlying security enforcement system comprises all those measures, tools, and devices that collectively participate in implementing the overall security policy. This includes the dynamic (re)configuration of host-based and network-based intrusion detection and prevention systems (IDPSes), firewalls, proxies, application server security components, activity reporting and event analysis components, etc.

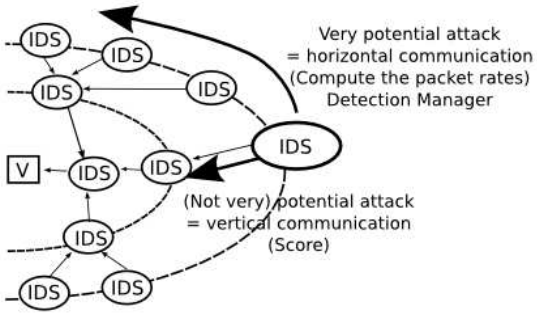


Figure 3.3: Sample Simulation Topology of *FireCol*

3.5 Threats Prediction and Assessment

3.5.1 Description

We adapted *FireCol* [125] in our security management framework to enable it to detect the majority of the attacks with low false positive detection rates. The *FireCol* system aims to detect DDoS attacks as far as possible from the victim and as close as possible to the attack source. It can be used both as an IDS or IPS. As depicted in Figure 3.3, *FireCol* instances can be distributed over ISP routers forming rings of protection around the customer.

A list of n rules $R_1, R_2, ...R_n$ describing packet patterns is input to *FireCol*. After that, simple metrics related to rule-matching are maintained. One metric is related to the frequency f_i of the matching of rules R_i (Eq. 3.1) during a predefined detection window dw. It is computed by dividing the number of packets matching this rule F_i by the total number of packets.

Another metric is the entropy (Eq. 3.2) which is an indicator of the diversity of traffic. For instance, the entropy is 1 when the distribution is uniform.

The last metric is the relative entropy metric (Eq. 3.4) which permits to compare easily if the current distribution f is close to the profile f' (higher than a certain threshold ω). In the first step of the detection mechanism, *FireCol* determines whether the traffic has changed before continuing, otherwise it indicates that there is no attack because it would have been detected before. Therefore, the detection process continues only if the relative entropy is higher than a threshold ω.

$$f_i = \frac{F_i}{\sum_{j=1}^{n} F_j} \tag{3.1}$$

$$H = -E[log f_i] = -\sum_{i=1}^{n} f(i) log_n(f_i) \text{ , (entropy)} \tag{3.2}$$

$$\psi_i = log \frac{f_i}{f'_i} \tag{3.3}$$

$$K(f, f') = \sum_{i=1}^{n} f_i \ \psi_i \text{ , (relative entropy)} \tag{3.4}$$

$$\frac{f_i(t)}{f_i(profile)} > 1 + \gamma, \ 0 \leq \gamma \leq 1, f_i(t) > \epsilon \tag{3.5}$$

$$Score_i = f_i \times b_k \text{ , (confidence score)} \tag{3.6}$$

Based on the use of the decision table 3.1, a confidence score for each pre-selected rule is derived (Eqs. 3.5, 3.6.) Basically, a high frequency means a possible DDoS attack. However, this detection approach cannot be used because the frequency of rules depend on each other. Thus *FireCol* considers the entropy for describing the randomness or diversity

Table 3.1: *FireCol* decision table

Case	Entropy	Frequency	Conclusion	Score factor
1	High	High	Potential	b_1
2	Low	High	Medium threat	b_2
3	High	Low	Potential later	b_3
4	Low	Low	No potential	$b_4 = 0$

of a selected set of features of traffic. More detail about the decision table are given in [125].

After that, the current score, the previous score affected by an ageing factor a, and the score from upstream *FireCol* instances are combined in order to obtain the effective score of a rule. If the score is higher than a certain threshold τ, a horizontal communication (Figure 3.3) between the *FireCol* instances on the ring structure is launched to compute the overall data rate and compare it with the capacity of the client. Otherwise, the decision is delegated by sending the score to the next upstream *FireCol* instance and so on [125]. Here we did not consider a ring architecture and we used *FireCol* on a single host instead.

3.5.2 Complexity

Although *FireCol* implements many stages during the detection process, the operations are simple. For example, we can compute the worst case complexity (all stages are needed) as follow. Considering that the frequency and entropy are computed only at the end of the detection window dw, *FireCol* has just to count the number of packets for each rule. Assuming p_{dw} is the number of packets during dw, there are p_{dw} operations to increment the counter. To compute the frequencies at the end, *FireCol* has to sum all counters (*i.e.*,

n operations where n is the number of rules.) Then we have the following steps:

- Compute the frequency of each rule (Eq. 3.1): n operations
- Compute the entropy (Eq. 3.2): $2n$ operations
- Compute the relative entropy (Eq. 3.4): $2n$ operations + 1 comparison
- Extract suspect n_2 rules (Eq. 3.5): 1 operation
- Examine extracted rules (Table 3.1): $n_2 + 1$
- Compute the score for the case 1,2,3 of the decision table (n_3 rules) (Eq. 3.6): n_3 operations
- Compute the confidence level: $2n_3$ operations

The main question is about the values of n_2 and n_3. We have $n \geq n_2 \geq n_3$ because few rules are kept at each stage for further analysis. It can be formalized by using an exponential distribution *i.e.*, $n_2 = n \times p(\frac{f_i(t)}{f_i(profile)} > 1 + \gamma) = e^{-\lambda(1+\gamma)}$ where γ is the parameter of the exponential law. Similarly, n_3 can be calculated. As a result, *FireCol* has $O(n)$ complexity even if all the rules are selected *i.e.*, $n = n_2 = n_3$.

3.6 Performance Evaluation

3.6.1 Experimental Settings

In order to test the validity of policy-based adaptation of security configuration while keeping the use case simple, we focus on DDoS attacks and define adaptation policies that help adapt Snort security level based on alert inputs from the *FireCol* and from Snort itself. The policies are described in Table 3.2.

Snorts rules (signatures) are organized into classes. Each class contains a number of rules that are related to a known attack type. For example, the FTP class contains all rules related to attacks on FTP servers. Snort allows the enabling/disabling of rule classes (categories) or individual rules through a set of configuration files. Snort (v 2.8.6) has a set of 58 categories.

For the sake of our experiments, we defined three levels of detection capability for Snort. The *medium* detection level contains rules that are enabled by default when deploying Snort. The total number of rules in this detection level is 8722 categorized into 32 libraries. The *minimum* detection level is defined by choosing the minimal set of rules that detect essential attack types. This level includes 11 libraries containing 4152 rules. The third level of detection is the *maximum* level, which includes all the libraries and has 9205 rules categorized into 51 libraries. As it is not trivial, section 3.7 elaborates on the issue of choosing the appropriate set of categories for each detection level.

3.6.2 Scenarios and Policies Modeling

We conduct experiments using different Snort detection capabilities to scan the DARPA 2000 LLDoS 1.0 [177] and 1999 [189] datasets, which last for three hours and one week respectively. As predicted in [28], relying only on a single instance of *FireCol* may entail many false alerts. In this experiment we use Snort as a means to confirm the validity of a *FireCol* alert. In our case, *FireCol* is deployed at a single location close to the victim.

We introduce in this study a confidence level between zero and one associated to each *FireCol* alert. The maximal score of a *FireCol* alert is the maximal frequency multiplied by the maximal factor b_1. When the detection is triggered, this maximal score is affected by the ageing factor a. So we obtain:

$$max_{score} = max_{frequency} \times b_1 \times a \tag{3.7}$$

Considering the most aggressive attack with a constant frequency of 1 during all detection windows, the maximal possible score is:

$$max_{score}(0) = b_1 \tag{3.8}$$

$$max_{score}(i) = (max_{score}(i-1) \times a) + b_1 \tag{3.9}$$

When i tends to ∞, this term tends to

$$m = \frac{b_1}{1-a} \tag{3.10}$$

Table 3.2: Policies for Dynamic Detection Capabilities

$\mathbf{p_1}$	**on** FireCol alert a **do** increase security by one level **when** a.score = medium
$\mathbf{p_2}$	**on** FireCol alert a **do** increase security by two levels **when** a.score = high
$\mathbf{p_3}$	**on** Snort alert a **do** increase security by one level **when** a.score = high
$\mathbf{p_4}$	**on** low security threat **do** switch to performance mode.
$\mathbf{p_5}$	**on** DDoS attack **do** enforce DDoS mitigation, notify remote mirror servers, increase backup-policy level.

Hence, the confidence level of the score s is:

$$confidence = \frac{s - \tau}{m - \tau} \tag{3.11}$$

The dynamic security-level adaptation policies are listed in Table 3.2. For the purpose of this evaluation, the inputs to them are alerts from either the *FireCol* or Snort. Whenever the *FireCol* generates an alert with high confidence ($> 60\%$), Snort detection is upgraded by two levels (policy p_2). If the *FireCol* generates an alert with a medium confidence ($> 40\%$), the detection level of Snort is increased by one level (policy p_1). Low confidence *FireCol* alerts are ignored ($\leq 40\%$). For Snort, we only consider high-priority alerts by augmenting the detection level by one (policy p_3). Furthermore, we define a *caution window* cw for the medium and maximum detection levels. This window represents the period where Snort should stay in a particular detection level before it switches down to a lower

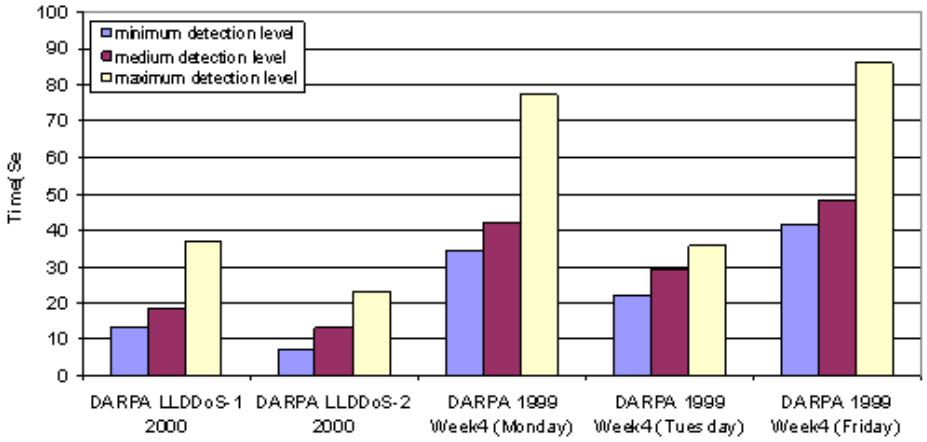


Figure 3.4: Snort performance over multiple datasets using different Detection Levels

level in case no abnormal behavior has been observed (policy p_4).

3.6.3 Detection levels and Cost

Figure 3.4 represents the running time of Snort with three different detection levels (minimum, medium, and maximum) over a number of datasets. Each group of bars corresponds to a particular dataset and each bar represents the detection level used in Snort to scan the dataset. Although, the medium and maximum detection levels of Snort include nearly the same number of rules (the medium level has 95% of all rules), Snort performance was different for these levels. This is because the default-disabled rule sets are computationally expensive and time consuming. Since these results are conducted by scanning the tcpdump file of the datasets, Snort did not drop any packet due to unrestricted time limit. However, this may not be the case in a real time environment with high-rate links (i.e. gigabit)

Table 3.3: *FireCol* alerts for the DARPA 2000 dataset ($dw = 60s$)

Time	Destination	Confidence
09:21:36	172.16.112.50	0.04
10:04:36	172.16.112.50	0.14
10:08:36	172.16.112.50	0.24
10:31:36	172.16.113.204	0.29
10:57:36	172.16.112.50	0.15
10:58:36	172.16.112.50	0.30
11:04:36	172.16.113.105	0.59
11:28:36	131.84.1.31	0.67
11:33:36	172.16.112.50	0.05

where some packets may be dropped to keep up with the rate. Dropping packets has the undesired outcome of the possibility of missing some attacks.

In the first experiment, we use the DARPA 2000 LLDOS 1.0 dataset [177] which contains traffic collected from two sensors installed in the DMZ and the Inside parts of the evaluation network. We focus our analysis on the inside sensor. The dataset features a series of attacks carried out over multiple sessions. These sessions start with a scanning the network in order to launch a DDOS attack against an off-site server. The sessions are grouped into five phases. First, the attacker starts by scan of the network (IPsweep) looking for any live IP addresses. Then, it looks for the sadmind daemon of live IP addresses. The attacker exploits the sadmind vulnerability in order to install an mstream Trojan, which is required for launching the DDoS attack. Finally, the DDoS attack is launched against the off-site server if successful.

Using *FireCol*, and as shown in Table 3.3, the DDoS victim is identified to be 131.84.1.31.

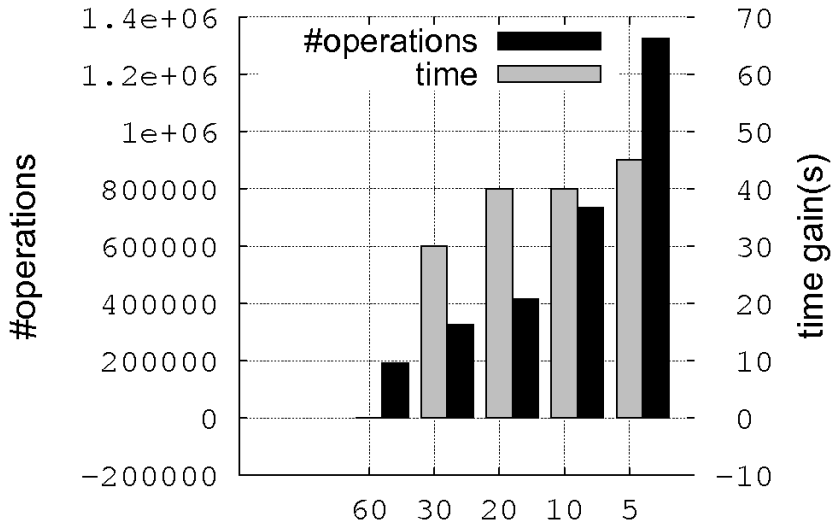


Figure 3.5: Impact of Detection window size on the *FireCol* overhead. Time reduction in seconds comparing with a 60 seconds detection window and number of mathematical operations

In this experiment the *FireCol* was run with a detection window of one minute. The *FireCol* manages to detect the attack with a relatively high confidence level (67%). There are some other false positive alerts. However, most of them got filtered out using the 40-60% confidence thresholds (policies p_1 and p_2) and only one false alert for host 172.16.113.105 remained. This alert may be due to a traffic burst to which the *FireCol* is sensitive.

The detection window size is an important parameter of *FireCol*. Large values imply that some DDoS attacks will be detected too late, while small values result in increased false positives due to traffic profile fluctuation. Figure 3.5 shows the time and overhead in terms of number of operations induced by several detection window sizes for the same dataset. In all cases however, the overhead induced by *FireCol* is much less than the one

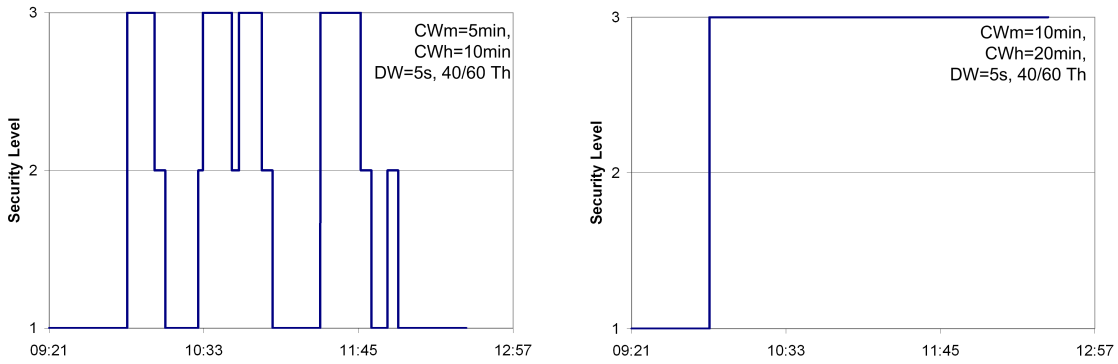
due to the execution of Snort because of the difference in detection mechanism. The delay and the number of operations are greatly reduced because there are more detection time slots. However, when the detection window size decreases, the traffic is more variable.

3.6.4 Detection Levels and Accuracy

This experiment shows how different risk management policies can significantly affect the detection capabilities of Snort. It also shows the effectiveness of detecting some attacks based on the prediction mechanism or on the assistance from other security components. For example, if the security administrator is expecting the network to be under a DoS attack, then Snort needs to be dynamically tuned for maximum DoS detection and so on. The actual difficulties ahead are related to how to effectively predict the security risk and be able to enforce the appropriate risk management policy into the relevant components of the security enforcement components.

Figures 3.6, 3.7 and 3.8 show the impact of dynamic security level adaptation policies on the behavior of Snort when scanning the DARPA dataset with different caution and detection window sizes.

Figure 3.6(a) shows the result for a caution window of $cw_m = 5$ and $cw_x = 10$ minutes for the medium and maximum security levels respectively. The first alert that changed the security level of Snort comes from *FireCol* at 09:57 with a confidence of 0.8 (policy p_2). It is worth noting that the first phase of the attacks (IPsweep) did not change Snort detection level because the severity of all the alerts generated by Snort for this phase was medium. However, for all the other attack phases, except the fourth phase which needs a



(a) CW=5/10mn

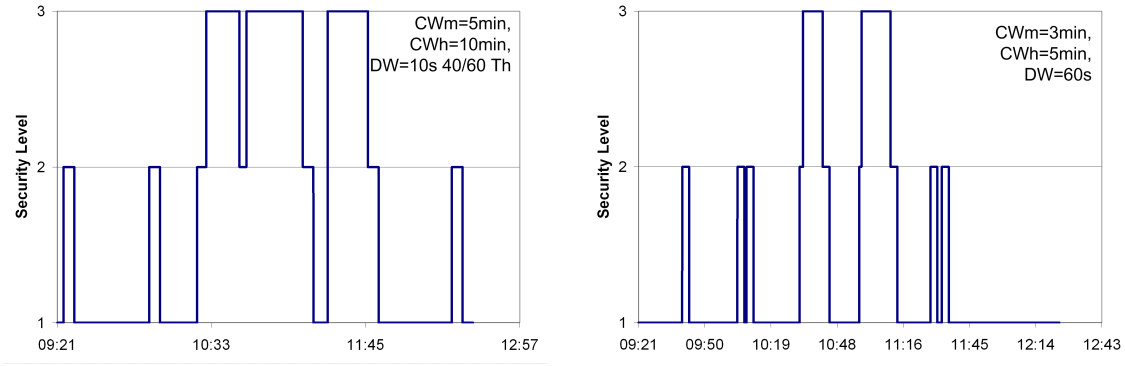
(b) CW=10/20mn

Figure 3.6: Snort Security Level Adaptation when DW=5s and Th=40-60% (DARPA 2000 dataset)

host-based IDS to be detected, Snort was in its maximum detection level. Overall, Snort was running at maximum detection 28.66% of the time, at medium detection 13.16%. This implies that it was running at minimum detection 58.17% of the time while it still managed to detect the attack phases. In terms of the time necessary to analyze the dataset, Snort took 28% less time than it did if ran at maximum detection. This is good if we know that the DARPA 2000 dataset is only three hours long and contains a five phases attack.

Figure 3.6(b) represents the detection behavior of Snort with a large caution window of 10 minutes for the medium and 20 minutes for the maximum security levels. This results in Snort running in maximum detection level since the occurrence of the first high-level alert. This is because the high-level alerts coming either from Snort or *FireCol* make it so that the cw_x caution window never reaches an end.

In Figure 3.7(a), we use the same caution window size as in 3.6(a) but with a larger window detection size. This results in having Snort run in maximum detection level in

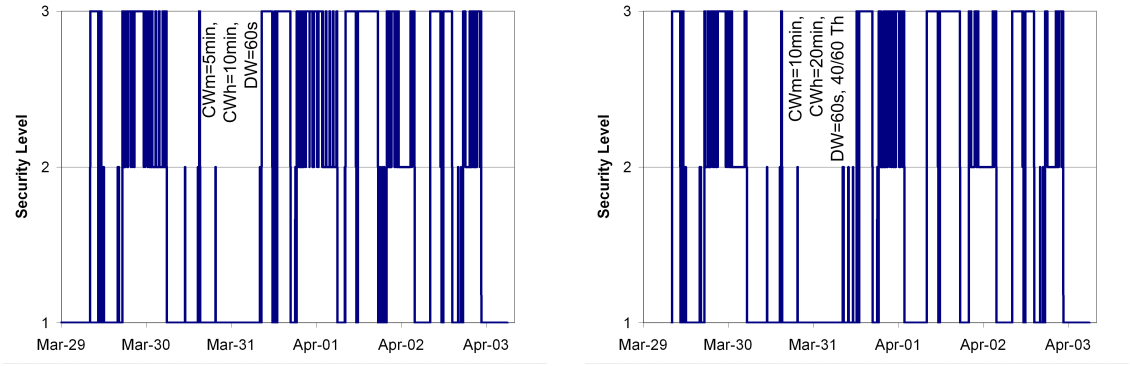


(a) (DW,CW,Th)=(10s,5/10mn,40-60%) (b) (DW,CW,Th)=(60s,3/5mn,0%)

Figure 3.7: Snort Security Level Adaptation when varying DW and Th (DARPA 2000 dataset)

only three of the four detectable attack phases. In Figure 3.7(b), we use smaller caution windows and a large window detection size of one minute, which results in reducing the overall time for the maximum and the medium detection levels and having Snort run at medium detection and not maximum detection during the DDoS attack.

Finally, Figures 3.8(a) and 3.8(b) show the dynamic security levels when using the DARPA 1999 dataset. The results are quite interesting as in the many DDoS attacks occuring in this long dataset (one week), all those which last for more than around five seconds manage to get captured by *FireCol* and hence reported to Snort which adapts its security level accordingly.



(a) (DW,CW,Th)=(60s,5/10mn,40-60%)

(b) (DW,CW,Th)=(60s,10/20mn,40-60%)

Figure 3.8: Dynamic Snort Detection Level Adaptation for DARPA 1999 Dataset

3.7 Discussion

The promises of policy-based dynamic adaptation of a security system to reduce security enforcement overhead while preserving good system performance are faced with the problem of defining the appropriate configuration policies in order to avoid an excessive exposure to real threats. In our experiments, configuration parameters, such as the caution windows of adaptation policies, the window detection size of the *FireCol*, and score thresholds are not trivial to set. For the *FireCol*, reducing the detection window size helps in the early detection of DDoS attacks but has the disadvantage of increased overhead and number of false alerts. However, in all cases, the overhead due to *FireCol* is always much lower than that entailed by Snort. In addition, depending on resource availability and attack prediction, the confidence thresholds may be lowered or increased by the system administrator. Using techniques such as neural networks or fuzzy logic, can be helpful in solving this configuration problem.

An additional concern which affects the accuracy of the detection is the selection of categories (set of rules) for each detection level as well as the number/type of security levels to define. The selection of categories may vary form an environment to another. For instance, the maximum detection level for protecting the web server of a company should include not only all the rules which detect web server specific attacks, but also those related to potential preliminary steps of these attacks, such as scanning. A possible solution for choosing the categories for each detection level can be based on common attack graphs [214] where the early steps of the attacks are included in the minimum detection level. However, the attacker may learn about the use of the caution window-based behavior and artificially introduce delay between attack steps. This can be countered by using a dynamic caution window size.

Another important point is related to the inherent support of dynamic adaptation by existing security mechanisms and tools. In this work, Snort was a hurdle in the sense that it does not yet support dynamic adaptation. We resorted to the use of virtual machines or double Snorting. However, these techniques have the main disadvantage of loosing detection state. The ideal solution would be to improve Snort source code to support the dynamic loading and unloading of rules/categories without the loss of detection state.

3.8 Conclusion

In this chapter we presented a policy-based design for adaptive risk management of an enterprise information system. Our proposal is intended to provide policy-based dynamic adaptation and reconfiguration mechanisms of the deployed levels of security measures.

The policies allow to define and dynamically maintain the right balance between effective security on one hand and system usability and performance on the other.

As a proof of concept, we carried out using dynamic adaptation experiments between the light-weight anomaly-based *FireCol* IDS and the more advanced signature-based Snort IDPS over known datasets. The experiments showed how good attack detection can still be feasible along with a low resource overhead when the right set of rule categories and configuration parameters are enabled. The performance gains in terms of resource utilization as well as the ability to detect security threats and respond to them at the right time validate our approach at least for the *FireCol*/Snort adaptation use case. Future work will investigate of attack graphs, attack statistical relationships, and learning mechanisms to define appropriate adaptation policies and security levels, and conduct experiments on other IDPS systems such as Bro.

Chapter 4

Configuration and Performance Analysis

4.1 Introduction

One of the major requirements for deploying any security technology is the ability to defend against attacks. Another requirement is to avoid network performance degradation when maximum security is not necessary. This results in a trade-off between security enforcement levels on one side and the performance and usability of an enterprise information system on the other. Existing IDPSs do not take into account these two conflicting goals. Network-based Intrusion Detection Systems (NIDSs) inspect copies of the packets that are transmitted over the network and generate alerts whenever malicious content is found. In contrast, Network-based Intrusion Prevention Systems (NIPSs) have the extra

ability to prevent the attacks from being successful. IDSs do not interfere with the network performance requirement (in terms of delay) but exhibit poor protection, as the attacks have already succeeded. On the other hand, IPSs can protect the network by dropping the malicious packets that match any attack pattern; however, this can have a negative impact on the network performance when the considered set of attack patterns increase.

Although many IDPS systems have been proposed, their appropriate configuration and control for effective attacks detection/prevention and efficient resources consumption has always been challenging [52, 99]. The evaluation of the IDPS performance for any given security configuration is a crucial step for improving their realtime capability [236]. Another concern is related to the impact of security enforcement levels on the performance and usability of an enterprise information system.

In this chapter, we analyze the impact of configuring an IDPS rule-checking process along with its consequent action (i.e. alert or drop) on the resulting security of the network, and on the average service time per event. We develop a new analytical model to investigate the relationship between the IDPS performance and its configuration. Our results show that applying different sets of rules categories and configuration parameters impacts average service time and affects system security.

The chapter proceeds with a background of the rule-checking process in Section 4.2. In Section 4.3, we present an analytical model to investigate the relationship between the IDPS performance and the rules mode selection. Section 4.4 presents the performance analysis study of the impact of IDPS configuration on average service time. It also describes the relationship between the system security level and different configuration parameters.

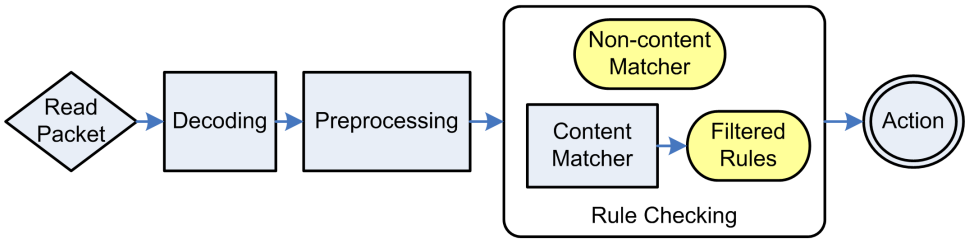


Figure 4.1: Analysis Tasks for Intrusion Detection and Prevention Systems

Finally, Section 4.5 concludes the chapter.

4.2 Rule-checking Operations

In this section, we describe the operation of existing intrusion detection and prevention systems and some of the weaknesses inherent in them. Generally, IDPSs perform a number of analysis tasks to identify malicious traffic. `SNORT`, for example, carries out the following tasks (Figure 4.1):

- Data decoding: decodes the header information of the packet and translates specific protocol elements into a data structure, for the use of the following tasks.
- Preprocessing: examines the packet for malicious activity that can not be captured by signature matching or performs a number of preliminary steps in the packet, *i.e.*, normalization, fragmentation reassembly, stream reconstruction, etc.
- Rule checking: examines the packet to determine if it is associated with an intrusion. There are two types of rules an IDPS can handle: content-based and non-content-based. The former is divided into three main sections: 1) action to be taken, 2)

header specifying protocol, IP addresses, and ports information, and 3) an option stating which parts of the packet should be inspected for determining the presence of a particular pattern, or a collection of patterns. The non-content-based rule is similar to the content-based one except that there is no pattern to look for.

- Action execution: the action describes what response an IDPS can perform when a packet matches a specified rule. The main actions include (but are not limited to): logging a packet (log), generating an alert (alert), dropping a packet (drop), terminating a connection (reject), and ignoring a packet (pass).

Our analysis will be limited to the rule-checking process along with the action associated with each rule. Once rules are selected and initialized, they are grouped by protocol type (*i.e.*, tcp, udp, icmp, ect.), and then by ports, then by those with content and those without. For each content-based group, a multi-pattern matcher is constructed for all rules by choosing a single pattern from all patterns in each rule option (*e.g.*, `SNORT` uses longest pattern). Clearly, there is no pattern matcher for non-content-based rules. When a packet arrives at the rule-checking engine, the corresponding multi-pattern matcher will be called on to filter out (for further evaluation) the rules whose single pattern are matched. The filtered rules can be large depending on the chosen patterns for the multi-pattern matcher and on the number of rules within a group(*i.e.*, http). The evaluation of these filtered rules and the non-content-based rules are applied sequentially. Once a rule matches a packet the corresponding action will be taken.

A rule can be either a detective or a preventive rule. A detective rule's action is `alert` and a preventive rule's action is `drop`. The detective rule aims to inspect a copy of a

packet transmitted over the network and generate an alert when a malicious pattern exists in the packet content. Clearly, this passive inspection mode has no impact on the network performance in terms of delay, as it checks only a copy of traffic for malicious activity, while the actual traffic is delivered successfully. However, this inspection mode exhibits a poor protection as it does not prevent an attack from succeeding. Unlike the detective rule, the preventive rule is configured to be in-line mode so that traffic will be dropped if it carries a malicious pattern that matches the rule. This preventive mode can meet the security requirement but it can have a negative impact on the network performance, especially when the attacking patterns increase.

4.3 Performance Evaluation Model

In this section, we develop an analytical model to study the impact of the vector $\mathcal{G}$ on the resulting security of an enterprise information system and on the average service time to inspect an event. We assume that the IDPS processes one event at a time. Once an event arrives, it goes through a sequence of detection and/or prevention rules according to the current configuration of the IDPS represented by vector $\mathcal{G}$. The process terminates if the event is dropped by a preventive rule or reported by a detective rule as a malicious event. In case an event is normal, the process ends when all rules are checked.

Table 4.1: SUMMARY OF NOTATIONS

Symbol	Meaning
$\mathcal{R}$	Set of Detection and Prevention rules in IDPS.
$\mathcal{N}$	Number of rules contained by IDPS.
E	An arriving event.
$\mathcal{G}$	Binary vector indicating whether a rule is a detective or preventive rule
$\mathcal{A}$	Set of attacks covered by IDPS
P_M	Prior probability of attack occurrence
FP	False positive probability for the detection and prevention of IDPS
FN	False negative probability for the detection and prevention of IDPS
$T(r_i)$	Processing time for rule r_i
H(k)	Vector indicating the proportion of malicious event of type i.
$B(i)$	The blocking probability of a preventing rule r_i

4.3.1 Definitions and Preliminaries

IDPS rules (or signatures) are classified into libraries. Each library contains a number of rules that are related to a known attack type. For example, the FTP library contains all rules related to attacks on FTP servers (i.e., `SNORT` (v 2.8.6) has a set of 58 libraries). Let $\mathcal{L} = \{l_1, l_2, \ldots, l_M\}$ be the set of all available libraries. We let $\mathcal{R}(l_i) = \{r_1, r_2, \ldots, r_{N_i}\}$ denote the set of a finite number of rules included in a library l_i. The total number of rules included in an IDPS is $N = |\mathcal{R}| = \sum N_i$, where $i = 1, 2, \ldots, M$.

We consider an IDPS with $\mathcal{R}$ rules or signatures. We let $\mathcal{R} = \{1, 2, \ldots, N\}$ denote a finite set of rules. We assume that the security administrator is responsible for including and excluding rules according to the specific needs of the protected network environment. For instance, Snort allows the enabling/disabling of rule libraries or individual rules through a set of configuration files. Furthermore, the security administrator can specify the mode of the rules as being either in a detective or a preventive mode. To classify a rule as to

which group it belongs to, we define a binary vector $\mathcal{G} = \{g_1, g_2, \ldots, g_N\}$ that indicates whether a rule is a detective or preventive rule (i.e., detection mode if $G(k) = 0$, prevention mode if $G(k) = 1$, where $k = 1, 2, \ldots, N$). This binary vector is defined as corresponding to rules vector $\mathcal{R}$ with $\mathcal{N}$ rules.

Each rule r_k has a processing time T_k. We consider only the time that it takes a rule to process an actual packet. Clearly, a detective rule that simply examines a copy of traffic is assumed to require no processing time on the actual traffic. The processing time t_k will be considered only if the rule r_k is in a preventive mode ($G(k) = 1$).

Each rule $r_k \in \mathcal{R}$ is responsible for only one type of malicious event. We let $\mathcal{A} = \{a_1, a_2, \ldots a_N\}$ be the set of different attacks covered by the IDPS, assuming that the occurrence of each attack is independent of the others.

We denote E as being an arriving event or flow. A malicious event E part of an attack of type k where $k \in \{0, 1, \ldots, N\}$ is denoted as E $\leftarrow a_k$. Note that an event contains at most only one type of maliciousness. We denote by E $\leftarrow a_0$ a benign event which does not contain any malicious content with regards to the different rules' restrictions R_i ($i = 1, 2, \ldots, N$).

A rule r_i announces event E as malicious with regard to attack type a_i is defined as E $\overset{r_i}{\leftarrow} a_i$. Similarly, we define E $\overset{r_i}{\leftarrow} a_0$ to indicate that the event E is announced as normal when no rule r_i reports the presence of attack a_i in it for all $i = 1, 2, \ldots, N$. The probability that rule r_k triggers an arriving event E as malicious, given that it is malicious with regards to attack type a_k is defined by: $\mathbb{P}\text{rob}\{\text{E} \overset{r_i}{\leftarrow} a_k \mid \text{E} \leftarrow a_k\}$ which is equal to the true positive probability TP_k =1-FN_k. FN_k represents the false negative rate of

rule r_k when mis-announcing a malicious event that contains an attack of type a_k. We let FP_k=$\mathbb{P}\mathrm{rob}\{\mathrm{E} \overset{r_i}{\leftarrow} a_k \mid \mathrm{E} \leftarrow a_0\}$ be the false positive rate of rule r_k, that is, the probability that rule r_k triggers an arriving event E as malicious, given that it is not malicious with regard to rule r_k.

4.3.2 Characterization of Traffic

A site-specific risk analysis provides information about the malicious activities that were encountered in the past. We believe that the risk analysis process is an important step to quantitatively measure the network security. However, our focus is not on developing a risk analysis model rather we are trying to benefit from information gathered by security administrators during the site-specific risk analysis process which includes the proportion of malicious events among all detected events, prior probability of maliciousness, false positive rate, and false negative rate. We mentioned the risk analysis model here for the sake of showing the feasibility of obtaining such parameters.

We denote P_M as the probability of maliciousness that categorizes an arriving event E to be malicious. This prior probability can be used to estimate future attacks. We denote by H(k) the vector indicating the proportion of malicious events of type i among all the malicious events for all i=1,...,N. Clearly, the sum of this vector is equal 1 ($\sum$H(i)=1, $i = 1, \ldots, N$).

4.3.3 Average Processing Time

In this section, we evaluate the average service time of an IDPS. It is the time required by the IDPS with a rule configuration $\mathcal{G}$ to successfully determine whether an arriving event is accepted as a normal event or reported/rejected with the presence of an attack in it. In the rule analysis process, preventive rules have a great impact on the service time of an IDPS. For example, a significant improvement in processing time can be achieved if a frequently triggered preventive rule is checked as early as possible because unnecessary analysis is avoided. We define $B(i)$ as the blocking probability of rule r_i. It is the probability of announcing an event as malicious by a preventing rule r_i, $\forall\ i = 1,\ldots,N$. The blocking probability of rule r_i is defined by:

$$B(i) = \mathbb{P}\text{rob}\{\text{E} \overset{r_i}{\leftarrow} a_i, \mathcal{G}(i)\} \tag{4.1}$$

where an event E is announced as malicious with an attack of type a_i by a rule r_i and the rule is a preventive rule, $\mathcal{G}(i)$=1.

In order for rule r_i to announce an event as malicious, all previous rules have to announce it as safe. In other words, an event should arrive at rule r_i before any decision is taken on it. This can be expressed as follows:

$$B(i) = \mathbb{P}\text{rob}\{\text{E} \overset{r_j}{\leftarrow} a_0(\forall j < i), \text{E} \overset{r_i}{\leftarrow} a_i\}\mathcal{G}(i) \tag{4.2}$$

Given that the event space consists of a malicious event of attack type k ($\text{E} \leftarrow a_k$) and

benign event $\mathrm{E} \leftarrow a_0$, we rewrite $B(i)$ as follow:

$$\begin{aligned} B(i) = \mathbb{P}\mathrm{rob}\{\mathrm{E} \leftarrow a_k, \mathrm{E} \overset{r_j}{\leftarrow} a_0(\forall j < i), \mathrm{E} \overset{r_i}{\leftarrow} a_i\}\mathcal{G}(i) \\ + \mathbb{P}\mathrm{rob}\{\mathrm{E} \leftarrow a_0, \mathrm{E} \overset{r_j}{\leftarrow} a_0(\forall j < i), \mathrm{E} \overset{r_i}{\leftarrow} a_i\}\mathcal{G}(i) \end{aligned} \tag{4.3}$$

Let us consider the situation when the event is malicious. Clearly, the probability of announcing an event as malicious by rule r_i depends on the probability that the event is malicious and on the probability of accepting the event as normal by all the rules previously checked. We let the first term of Equation 4.3 be $B_{mal}(i)$ and using the theorem of total and conditional probability, $B_{\mathrm{mal}}(i)$ can be written as:

$$\begin{aligned} B_{\mathrm{mal}}(i) = \sum_{k=1}^{N} \mathbb{P}\mathrm{rob}\{\mathrm{E} \overset{r_i}{\leftarrow} a_i \mid \mathrm{E} \overset{r_j}{\leftarrow} a_0(\forall j < i), \mathrm{E} \leftarrow a_k\} \\ \times \mathbb{P}\mathrm{rob}\{\mathrm{E} \overset{r_j}{\leftarrow} a_0(\forall j < i), \mathrm{E} \leftarrow a_k\} \end{aligned} \tag{4.4}$$

The first term in Equation 4.4 represents the case when the IDPS announces the event as malicious by rule r_k given that the event arrives to rule r_i and it is malicious. In this case, the probability that the IDPS correctly announces the event as malicious or mistakenly classifies it as malicious is defined as $\mathrm{PB}_{\mathrm{mal}}$. This can be calculated as follows:

$$\mathrm{PB}_{\mathrm{mal}}(k, i) = \begin{cases} 1 - \mathrm{FN}_i & \text{if } k = i \\ \mathrm{FP}_i & \text{if } k \neq i \end{cases} \tag{4.5}$$

We let $\mathrm{PE}_{\mathrm{mal}}$ stand for the second term of Equation 4.4, which represents the probability that the IDPS accepts the event as normal by all rules r_j, j=1,. . . ,i-1, earlier than the

current evaluated rule r_i where the event E is malicious. We have two cases in this situation. In the first case, the current evaluated rule r_i is the first one (i=1), where no rule has been checked so far. $\mathrm{PE}_{\mathrm{mal}}$ can be calculated as:

$$\mathrm{PE}_{\mathrm{mal}}(k,i) = H(k)P_M \quad \text{where } i = 1. \tag{4.6}$$

The second case of $\mathrm{PE}_{\mathrm{mal}}$ may be encountered when there is at least one rule r_j that has been checked before rule r_i; that is, r_i is not the first rule to be evaluated (*i.e.*, $i > 1$). Accordingly, $\mathrm{PE}_{\mathrm{mal}}$ can be calculated by:

$$\mathrm{PE}_{\mathrm{mal}}(k,i) = \begin{cases} \prod_{j=1}^{i-1} \Big(1 - (1 - \mathrm{FN}_j)\mathcal{G}(j)\Big)\mathrm{H}(k)P_M & \text{if } k \neq j \\ \prod_{j=1}^{i-1} \Big(1 - \mathrm{FP}_j\mathcal{G}(j)\Big)\mathrm{H}(k)P_M & \text{if } k = j \end{cases} \tag{4.7}$$

Now let us consider the situation when the event is normal. We are interested in the probability of announcing an event as malicious by rule r_i given that the event is safe and all previously evaluated rules r_j (*i.e.*, $j < i$) mark the event as safe. This can be written as:

$$\begin{aligned} B_{\mathrm{safe}}(i) = \mathbb{P}\mathrm{rob}\{\mathrm{E} \overset{r_i}{\leftarrow} a_i \mid \mathrm{E} \overset{r_j}{\leftarrow} a_0(\forall j < i), \mathrm{E} \leftarrow a_0\} \\ \times \mathbb{P}\mathrm{rob}\{\mathrm{E} \overset{r_j}{\leftarrow} a_0(\forall j < i), \mathrm{E} \leftarrow a_0\} \end{aligned} \tag{4.8}$$

Applying the same steps used for the malicious case yields the following equation in a

safe case:

$$B_{\text{safe}}(i) = \text{FP}_i \times \left\{ \begin{array}{ll} 1 - P_M & \text{if } i = 1 \\ \prod_{j=1}^{i-1} \Big(1 - \text{FP}_j \mathcal{G}(j)\Big)(1 - P_M) & \text{if } i > 1 \end{array} \right\} \tag{4.9}$$

Given the Equations 4.3, 4.4, and 4.9, we can calculate the blocking probability $B(i)$ of rule i as follows:

$$B(i) = \text{B}_{\text{mal}}(i) \times \mathcal{G}(i) + B_{\text{safe}}(i) \times \mathcal{G}(i) \tag{4.10}$$

where

$$\text{B}_{\text{mal}}(i) = \sum_{k=1}^{N} \text{PB}_{\text{mal}}(k, i) \times \text{PE}_{\text{mal}}(i) \tag{4.11}$$

Finally, we measure the average service time of an IDPS as follows:

$$\begin{aligned} Avg = & \left[\sum_{i=1}^{N} B(i) \sum_{k=1}^{N} T(k) G(k) \right] + \Big(1 - \sum_{i=1}^{N} B(i)\Big) \\ & \times \sum_{i=1}^{N} T(i) G(i) \end{aligned} \tag{4.12}$$

4.3.4 Level of Security

The main objective of deploying any security tool is to protect the network from any malicious activities. Measuring the impact of security configurations can help security administrators in making optimal decisions about how to strengthen network security. In IDPSs, rules in preventive mode have the capability of blocking attacks once they have been matched. However, this induces a negative impact on network performance (*i.e.*, E2E delay, throughput, service usability, jitter, etc.) especially when the number of preventive

rules increases. Therefore, the main concern is to find the appropriate balance between security enforcement levels and the performance and usability of the network. Here, we evaluate the impact of a chosen IDPS configuration on the resulting security of the system. In particular, we are interested in measuring the probability of blocking an event given that it is malicious.

$$\begin{aligned} S &= \mathbb{P}\text{rob}\{\text{E} \overset{r_j}{\leftarrow} a_i \mid \text{E} \leftarrow a_i\} \\ &= \frac{\mathbb{P}\text{rob}\{\text{E} \overset{r_i}{\leftarrow} a_i, \text{E} \leftarrow a_i\}}{\mathbb{P}\text{rob}\{\text{E} \leftarrow a_i\}} \\ &= \frac{\sum_{i=1}^{N} \mathbb{P}\text{rob}\{\text{E} \overset{r_i}{\leftarrow} a_i, \text{E} \overset{r_i}{\leftarrow} a_i\} \times \mathcal{G}(i)}{\mathbb{P}\text{rob}\{\text{E} \leftarrow a_i\}} \end{aligned} \tag{4.13}$$

Using Equation 4.3 in Equation 4.13 yields:

$$S = \frac{\sum_{i=1}^{N} \sum_{k=1}^{N} \text{PB}_{\text{mal}}(i) \times \text{PE}_{\text{mal}}(i) \times \mathcal{G}(i)}{P_M} \tag{4.14}$$

4.3.5 Accuracy of Action

The capability of an IDPS to apply different rule modes (*i.e.*, alert or block) motivated the need for measuring action accuracy. Therefore, we study the action that is taken by the IDPS against an arriving event. The action of the IDPS could be either accepting or blocking an event. The accuracy of action is defined as taking the right action with regards to an arriving event. That is, the action of the IDPS is accurate if it accepts an event that

is normal and/or blocks a malicious event. We define the accuracy of action A_{acc} as the probability of either accepting a benign event or blocking a malicious one. A_{acc} can be written as follows:

$$A_{acc} = \mathbb{P}\text{rob}\{\text{E}\leftarrow a_i, \text{E} \overset{r_i}{\leftarrow} a_i\} + \mathbb{P}\text{rob}\{\text{E}\leftarrow a_0, \text{E} \overset{r_i}{\leftarrow} a_0\} \tag{4.15}$$

Using the conditional probability theorem yields:

$$\begin{aligned} A_{acc} = {} & \mathbb{P}\text{rob}\{\text{E} \overset{r_i}{\leftarrow} a_i \mid \text{E}\leftarrow a_i\}\mathbb{P}\text{rob}\{\text{E}\leftarrow a_i\} \\ & + \mathbb{P}\text{rob}\{\text{E} \overset{r_i}{\leftarrow} a_0 \mid \text{E}\leftarrow a_0\}\mathbb{P}\text{rob}\{\text{E}\leftarrow a_0\} \end{aligned} \tag{4.16}$$

By substituting 4.14 in 4.16, the action accuracy taken by the IDPS for an arriving event is as follows:

$$A_{acc} = S \times P_M + \prod_{i=1}^{N} \Big(1 - FP_i\mathcal{G}(i)\Big) \times (1 - P_M) \tag{4.17}$$

The evaluation of the IDPS action accuracy is essential not only in characterizing its capability of correctly detecting/preventing attacks but also it help security analysts in analyzing its impact on the network performance. Basically, the action taken by the IDPS when attack is announced has some influences on the usability of the protected system. In other words, blocking legitimate traffic is always undesirable and happened by the action of the IDPS rules. Therefore, IDPS with a high action accuracy is preferred and recommended

4.3.6 Accuracy of Decision

In this section, we analyze the decision accuracy made by the IDPS. This refers to the decision to announce an arriving event as malicious or not, regardless of the action taken as a result of the announcement. The decision is accurate when announcing an arriving malicious event as malicious while not doing so with the benign one. Therefore, the accuracy of the decision is defined as the probability of either triggering an event as malicious while it is malicious or not triggering the event when it is normal. This is equivalent to the complement of making a wrong decision with regards to an arriving event. That is, the decision accuracy of the IDPS is the complement of announcing an event as malicious where it is not malicious or announcing a benign event as malicious. The inaccuracy of the decision can be written as follows:

$$\overline{D}_{acc} = \mathbb{P}\text{rob}\{\text{E}\leftarrow a_i, \text{E} \overset{r_i}{\leftarrow} a_0\} + \mathbb{P}\text{rob}\{\text{E}\leftarrow a_0, \text{E} \overset{r_i}{\leftarrow} a_i\} \tag{4.18}$$

Solving this equation results in:

$$\begin{aligned} \overline{D}_{acc} = &\Big(1 - \prod_{i=1}^{N} \big(1 - FP_i\big)\Big) \times (1 - P_M) \\ &+ \sum_{k=1}^{N} FN_k \prod_{\substack{i=1 \\ i\neq k}}^{N} \big(1 - FP_i\big) \times H(k) P_M. \end{aligned} \tag{4.19}$$

The difference between the action and decision accuracy is that the former concerns the response taken by a triggered rule to either block an arriving event or to accept it.

The latter concerns the decision of announcing an arriving event as malicious or benign. Clearly, the rule's mode of an IDPS has no impact on the decision accuracy but it affects the action accuracy. For instance, the action of an IDPS is considered to be inaccurate if a malicious event matches a specified rule which is in a detective mode (alert). This is because the malicious traffic has not been blocked by the detective rule. The action accuracy can be identical to the decision accuracy when the IDPS is configured to operate entirely in IDS mode (i.e., the action of all the rules is alert) or entirely in IPS mode (see Table 4.2 when IPS=100%).

4.4 Performance Evaluation and Results

In this section, we study the impact of the IDPS configuration on the average service time. We also measure the security level of the system when choosing different configuration parameters. The results are derived using both analytical and simulation approaches. The simulations are performed using a new discrete-event simulation tool developed under Matlab [293]. In order to test the validity of our work while keeping the case simple, we assign equal processing time for all the rules (one unit of time), we let the proportion of maliciousness be equally distributed $H(i)/\mathcal{N}$, $i = 1, \ldots, N$, and we set the probability of maliciousness to be P_M=0.5. For simplicity, we assume that the false detection rates, FP and FN, are measured for the entire rule-checking engine of an IDPS.

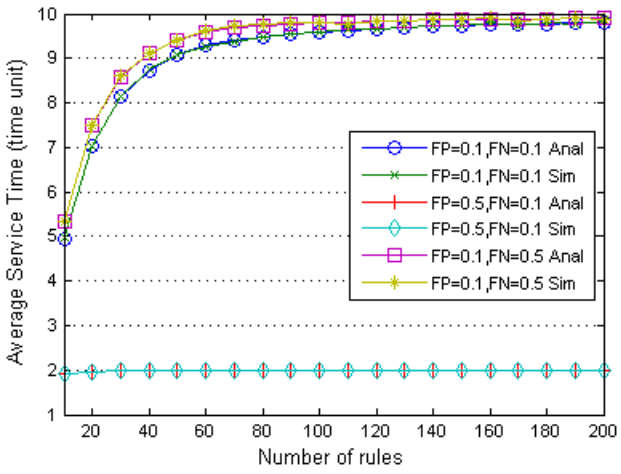


Figure 4.2: Average Service Time for Number of Rules when (P_M,IPS)=(0.5,75%)

4.4.1 Parameters estimation

The challenge involved in performance analysis of a security system so as to reduce resource utilization while preserving a good level of security enforcement is the need to obtain estimates for the various parameters used in the analysis. For instance, the false positive (FP) and false negative (FN) rates can be accomplished by either using proper training data sets or by analyzing the past behavior of the system [143]. The rule processing time T can be measured experimentally. `SNORT`, for instance, provides statistics on rule performance through a simple configuration option (*i.e.*, `profile rules`). For each rule, `SNORT` provides an estimate of how much it takes to process a packet. The prior probability of attack occurrence (P_M) and the proportion of attacks (H) can be initially estimated using a site-specific risk analysis approach and updated with new attacks accordingly.

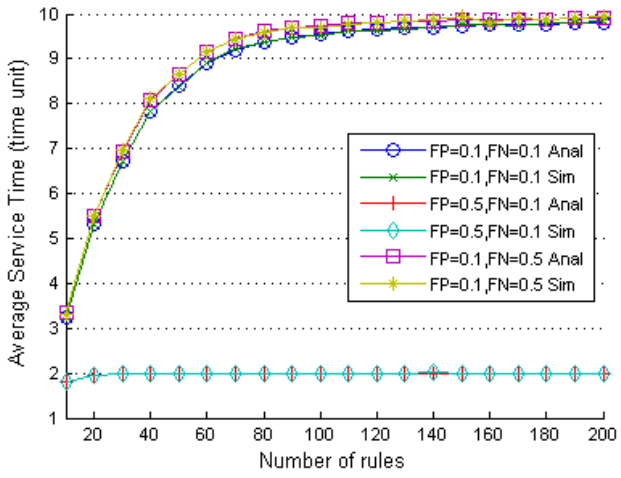


Figure 4.3: Average Service Time for Number of Rules when (P_M,IPS)=(0.5,50%)

4.4.2 Average Service Time

The average service time is calculated as the time required for an event to be completely served. An event is served once a detective/preventive rule finds a match and triggers its action or once the event is identified as normal. Our goal in this experiment is to show the impact of the increasing number of rules $\mathcal{N}$ on the average service time when processing an event using different preventive rule percentages. For each preventive percentage, we plot the average service time using different detection rates. Figures 4.2, 4.3 and 4.4 show the results when selecting the percentage of preventive rules to be 75%, 50%, and 10% respectively.

Figure 4.2 illustrates the impact of increasing the number of rules on the average service time when 75% of the rules are in preventive mode. Figure 4.2 shows that the larger $\mathcal{N}$ is, the longer service time becomes. When we reach approximately 100 rules, the system

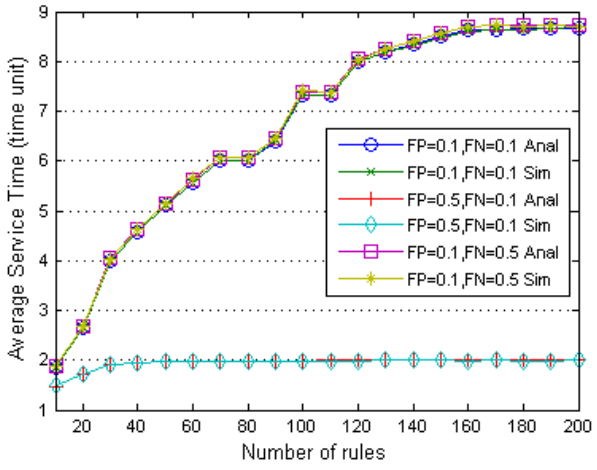


Figure 4.4: Average Service Time for Number of Rules when (P_M,IPS)=(0.5,10%)

reaches the saturation state and additional rules result in very little further impact. Prior to the saturation region, a reduction in the number of rules results in an improvement in average service time. In this situation, a malicious event is likely to be missed by the rule accountable for it and accordingly is examined by other rules that are not responsible for it. Thus, a reduction in the false negative rate yields no appreciable improvement in average service time. The false negative rate has little impact on the average service time. However, this is not the case when the value of FP increases. Rather, there is a notable reduction in average service time when the rate of false positives increases. This occurs because of an early decision made as a result of a wrong diagnosis. In this case, an increase in the number of rules has only a slight impact on average service time.

When the percentage of preventive rules is 50%, Figure 4.3 appears quite similar to Figure 4.2. Increasing the number of rules, false positives, and false negatives impacts the

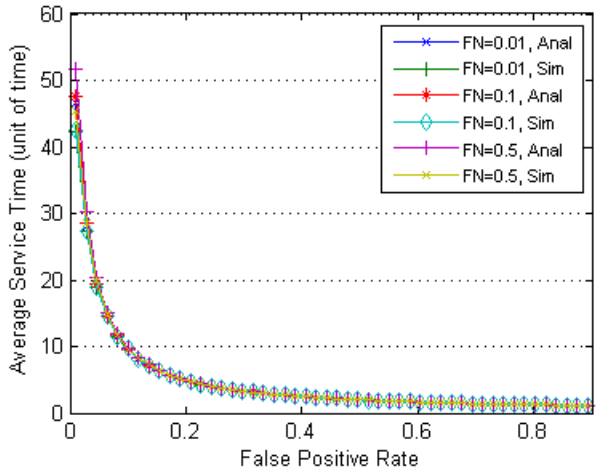


Figure 4.5: Impact of Detection Rates on Average Service Time when IPS=75%

average service time in a way very similar to that illustrated in Figure 4.2. However, the average service time approaches the saturation state more slowly than in the previous case when 75% of the rules are preventive. That is, the improvement in average service time is limited once the saturation point of approximately 130 rules is reached (out of 200 rules).

In Figure 4.4, when 10% of the rules are in preventive mode, we observe a somewhat different impact on the average service time. Figure 4.4 shows that the average service time increases linearly with the number of rules. Improvement in average service time in this case is obviously the most advantageous of the three scenarios, because the saturation point is not reached until approximately 180 rules.

To conclude, the potential for improvement in average service time increases as we reduce the percentage of preventive rules. Of course, the price one has to pay for reduction in the number of preventive rules is a corresponding decrease in enterprise network security.

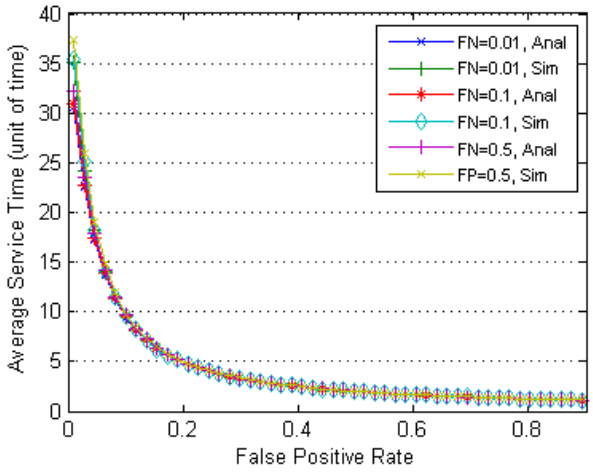


Figure 4.6: Impact of Detection Rates on Average Service Time when IPS=50%

In the following set of experiments, we focus on the average service time as a function of increasing both the false positive and false negative rates. We are interested in understanding the impact of the detection rates on the average time required to completely inspect an arriving event. The number of rules in this case is chosen to be 100. Figures 4.5, 4.6, and 4.7 show the impact of varying the false positive rate while fixing the false negative rate and using different preventive rule percentages.

Figure 4.5 presents the results when 75% of the rules are preventive. We can see that the average service time decreases with an increase in the false positive rate for all false negative rate values. We can see that the average service time is longer when the IDPS becomes accurate in terms of the false positive rate, no matter what the false negative rates are. Indeed, the IDPS consumes more time to correctly distinguish the malicious events from the benign ones rather than just mistakenly identifying a malicious event at an early

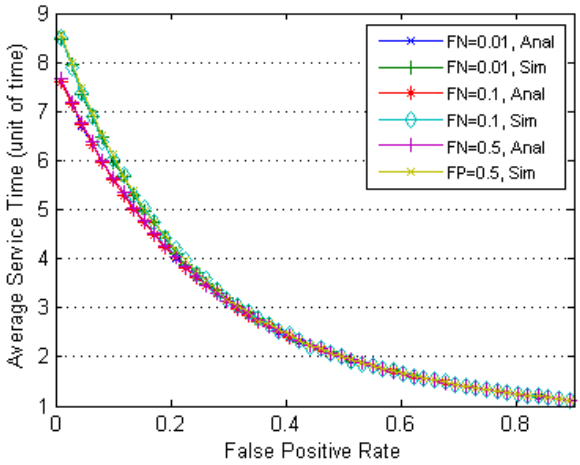


Figure 4.7: Impact of Detection Rates on Average Service Time when IPS=10%

stage.

Figures 4.6 and 4.7 show similar results, except that a dramatic decrease in average service time results from the reduction of the percentage of preventive mode services. However, with the use of different preventive mode percentages, the average service time approaches saturation differently. That is, when the IPS percentage is large, the average service time reaches saturation quickly. In contrast, the saturation state is reached more slowly as the IPS percentage decreases. Clearly, the impact on average service time will be no more than 10 units of time when 10% of the rules are in IPS mode (see Figure 4.7).

Figures 4.8, 4.9, and 4.10 plot the impact of changing the false negative detection rate (FN) for different false positive rates and with the use of different preventive rule percentages. Clearly, a reduction in the percentage of preventive rules results in a significant reduction in average service time. Furthermore, an increase in the rate of false negatives

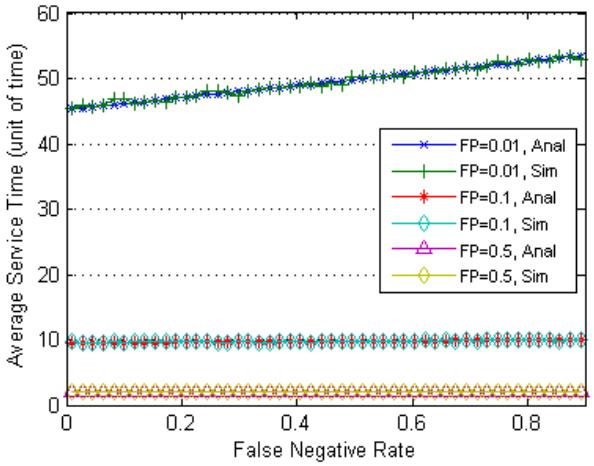


Figure 4.8: Impact of Detection Rates on Average Service Time when IPS=75%

produces very little change in average service time.

4.4.3 Level of Security and Accuracy

In this section, we intend to study the impact of choosing different configuration parameters on the security of the system and on the action and decision accuracy. The results are achieved using both the analytical and the simulation models that we have developed. For all results in this section, we based our study on a total of 100 rules. Table 4.2 presents the impact of varying four configuration parameters, including FP, FN, P_M, and IPS% on security and accuracy of the system. Clearly, an increase in the preventive rule percentage yields a corresponding improvement in system security. Nevertheless, a system still has a good level of security even though the percentage of preventive rules is relatively low. For example, when FP is 0.5, FN is 0.1, and IPS is 10%, the security result reach 79%.

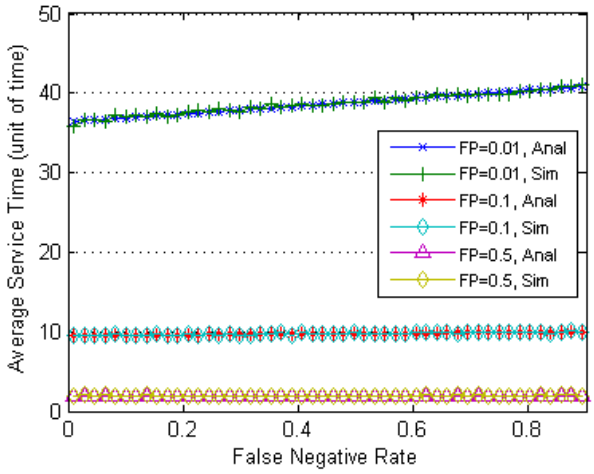


Figure 4.9: Impact of Detection Rates on Average Service Time when IPS=50%

However, the accuracy of action and decision for this case are not satisfactory.

4.5 Conclusion

In this chapter we studied how choosing which rules are preventive or detective has an impact on the security of the system, on the average service time, and on the decision and action accuracy of an IDPS. We developed a new analytical model to investigate the relationship between IDPS performance and its configuration. Simulation was conducted to validate our performance analysis study. Our results show that applying different sets of rules categories and configuration parameters impacts average service time and affects system security. The results demonstrate that it is desirable to strike a balance between system security and network performance in terms of delay.

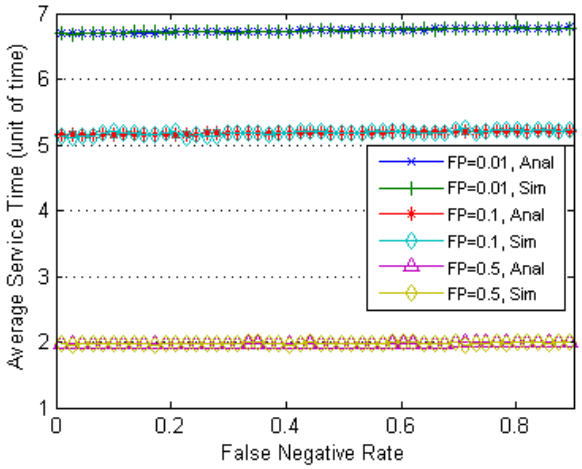


Figure 4.10: Impact of Detection Rates on Average Service Time when IPS=10%

Table 4.2: Selected Results of Security level, Decision Accuracy, and Action Accuracy with Different Configuration Sets

Configuration Parameters			Analytical Results			Simulation Results		
FP	FN	P_M	Security	Action Accuracy	Decision Accuracy	Security	Action Accuracy	Decision Accuracy
10% IPS								
0.1	0.1	0.1	0.3340	0.7624	0.4099	0.3263	0.7611	0.4104
0.1	0.1	0.5	0.3340	0.5720	0.6550	0.3310	0.5687	0.6565
0.5	0.1	0.1	0.7900	0.3040	0.1009	0.7939	0.3056	0.1007
0.1	0.5	0.1	0.2620	0.7552	0.3944	0.2654	0.7548	0.3942
0.1	0.5	0.5	0.2620	0.5360	0.5775	0.2597	0.5354	0.5778
0.5	0.5	0.1	0.7500	0.3000	0.1008	0.7493	0.2991	0.1003
50% IPS								
0.1	0.1	0.1	0.6720	0.5986	0.4099	0.6650	0.5956	0.4094
0.1	0.1	0.5	0.6720	0.6312	0.6550	0.6723	0.6315	0.6555
0.5	0.1	0.1	0.9813	0.1263	0.1009	0.9790	0.1272	0.1028
0.1	0.5	0.1	0.5407	0.5855	0.3944	0.5461	0.5862	0.3959
0.1	0.5	0.5	0.5407	0.5656	0.5775	0.5373	0.5631	0.5774
0.5	0.5	0.1	0.9688	0.1250	0.1008	0.9662	0.1233	0.0995
75% IPS								
0.1	0.1	0.1	0.8193	0.5124	0.4099	0.8196	0.5121	0.4100
0.1	0.1	0.5	0.8193	0.6488	0.6550	0.8197	0.6490	0.6546
0.5	0.1	0.1	0.9966	0.1067	0.1009	0.9959	0.1069	0.1009
0.1	0.5	0.1	0.6705	0.4975	0.3944	0.6780	0.4989	0.3963
0.1	0.5	0.5	0.6705	0.5744	0.5775	0.6760	0.5780	0.5806
0.5	0.5	0.1	0.9922	0.1063	0.1008	0.9931	0.1064	0.1006
100% IPS								
0.1	0.1	0.1	0.9613	0.4099	0.4099	0.9578	0.4113	0.4113
0.1	0.1	0.5	0.9613	0.6550	0.6550	0.9616	0.6559	0.6559
0.5	0.1	0.1	0.9998	0.1009	0.1009	0.9998	0.0993	0.0993
0.1	0.5	0.1	0.8063	0.3944	0.3944	0.8068	0.3952	0.3952
0.1	0.5	0.5	0.8063	0.5775	0.5775	0.8081	0.5771	0.5771
0.5	0.5	0.1	0.9990	0.1008	0.1008	0.9991	0.1006	0.1006

Chapter 5

Optimization of Security Configuration

5.1 Introduction

IDPS design and operation for efficient attack detection and prevention combined with efficient resource usage are desperately needed [52, 99]. Evaluating IDPS performance for a particular security configuration is a key step towards improving their realtime ability [31, 236]. Another concern relates to the impact of safety measures on the performance and usability of an enterprise information system.

In this chapter, we propose a rule mode selection optimization technique that aims at determining an appropriate IDPS configuration set in order to maximize the security enforcement levels while avoiding any unnecessary network performance degradation. The

proposed method demonstrates that different sets of rules categories and configuration parameters have varying impacts on service time and system security. As a result, it is judicious to seek a balance between security and performance tby determining the appropriate IDPS configuration.

This chapter proceeds with addressing the problem of determining the appropriate IDPS configuration set in Section 5.2. We evaluate the efficiency of the Rule Mode Selection Technique ($RMST$) using simulations in Section 5.3. Finally, Section 5.4 concludes the chapter.

5.2 Optimization of IDPS Rule Mode Selection

In this section we address the problem of determining the appropriate IDPS configuration set necessary to balance network security and performance. As explained before, IDPS preventive rules have the capability of blocking attacks once they have been matched. However, this induces a negative impact on network performance in terms of delay, especially when the number of preventive rules increases. Therefore, the main concern is to find the appropriate preventive rule set that maximizes security enforcement levels while avoiding any unnecessary performance degradation in terms of delay. We assume that the security administrator excludes the rules that are supposed to be strictly in preventive or detective modes. The optimal solution for the rule mode selection (RMS) problem is the one that can maximize the prevention level and minimize system delay. The RMS problem is NP-hard due to the maximal minimal matching of its multiple objectives. Indeed, the RMS problem can be reduced to a 0-1 knapsack problem known to be NP-complete [129]

and is therefore NP-complete.

5.2.1 Rule Mode Selection Problem Formulation

The rule mode selection problem is formulated as follows: Given a set of IDPS rules, find a legitimate preventive rule subset that maximizes the level of security, subject to the delay constraint. In our study, we assume a sequential rule checking process where each event passes through a sequence of rules until a decision is made. For an IDPS with N rules associated with weight w_i that resembles the dominance of rule r_i in the expected value metric, we can then formalize the RMS problem as a binary integer program (BIP) as follows:

$$\begin{aligned} &\max \sum_{i=1}^{N} w_i x_i \\ &\text{s.t. (a) } \sum_{i=1}^{N} t_i x_i \leq D_{max} \\ &\text{(b) } x_i \in \{0,1\}, i \in \{i = 1, \ldots, N\}. \end{aligned} \tag{5.1}$$

where x_i is a binary variable such that $x_i = 1$ if rule r_i is a preventive rule and $x_i = 0$ if rule r_i is a detective rule. The rule weight computation is explained in more detail later in this section. Inequality (a) provides an upper bound on the expected response time. Since the expected response time for an event entering the system is proportional to the number of preventive rules, an event has to be served at a rate faster than the arrival rate in order to preserve the stability of the system. The delay constraint is thus translated into an upper bound D_{max} on the number of preventive rules as the mean of the inter-arrival rate.

The RMS problem can be mapped to the 0-1 knapsack problem [129] in order to alleviate

the complexity of its max-min nature. In 0-1 knapsack problem, we are given n items, each associated with a value and weight; the objective is to select a set of items that maximize the total value where the total weight is less than or equal to a given value W. The RMS problem is similar to the 0-1 knapsack problem, where the weight of the rule w_i is similar to the item's value and the upper limit response time is similar to the maximum allowed weight.

5.2.2 RMS Technique

The optimal solution of the RMS problem can be obtained using an exhaustive search in the solution space. However, such a brute force method becomes computationally impractical when the number of rules is large. The use of a heuristic solution allows us to obtain a reasonably good solution in polynomial time without searching the entire solution space. Indeed, using optimization techniques Branch & Bound and Branch & Cut for solving the RMS problem is not practical when the rules number is high due to the size of the search space. On the other hand, the results obtained from a simple selection method, e.g., a Greedy Algorithm, can meet the computational time requirement, but jeopardizes the system security requirements. Therefore, we propose a heuristic-based rules mode selection technique (RMST) which can obtain a solution for the RMS problem in polynomial time, while the system security level remains in good agreement with the results obtained using the optimal solution. RMST is illustrated in Algorithm 1 and the use of the rule weight computation is described in the following section.

Algorithm 1 OptimizeRuleSelection: rule_set, D_{max}

```
1: D_sum ← 0
2: pool_list ← empty
3: for r_i : i = 0 to N do
4:     if t(r_i) + D_sum ≤ D_max then
5:         prevention_list ← r_i
6:         D_sum ← D_sum + t(r_i)
7:     else if prevention_list is not empty then
8:         for r_j : j = 0 to N do
9:             if (r_i ∈ prevention_list; w_j ≥ w_i; t_j ≤ t_i) then
10:                remove r_i from prevention_list;
11:                prevention_list ← r_j;
12:                D_sum ← D_sum + t(r_j) - t(r_j)
13:                pool_list ← r_i
14:            end if
15:        end for
16:        if D_sum < D_max then
17:            Add valid rules from pool_list to prevention_list
18:        end if
19:    end if
20: end for
21: Return prevention_list
```

5.2.3 Rule Weight Computation

In this section, we describe the technique used for computing rule weight with the intention of capturing the importance of every rule in the IDPS. We assign a weight to each rule in the rule configuration set that reflects its value in protecting the network by the IDPS. Two factors can be used to calculate the rule weight: (1) *potential damage* that can be prevented by a true detection, and (2) *operational loss* which is incurred due to the false detection.

Potential damage $D(r_i)$: the damage prevented by rule r_i can be measured by using the severity of an attack and the accuracy of rule r_i. The attack severity measures the

risk level posed by a particular attack. We let $Sev(r_i)$ denote the severity score for rule r_i that is responsible for attack a_i. There are several knowledge base sources which provide severity scores for known attacks, including MITRE-CVE, NIST-NVD, Secunia, as well as software developer specific severity score databases. For example, the FileZilla unspecified format string vulnerability has been reported in NIST-NVD to be scored as 7.5 out of 10, where `SNORT` includes a rule accountable for this attack with multiple score references. The potential damage $D(r_i)$ can be expressed as follows:

$$D(r_i) = (1 - FN_{r_i}) \times Sev_{r_i} \tag{5.2}$$

where FN_{r_i} is the false negative rate for rule r_i and $Sev(r_i)$ is the severity score for rule r_i.

Operational loss $L(r_i)$: The operational loss incurred due to rule r_i can be measured using the cost associated to the response triggered by false detection of rule r_i. For example, cost of blocking legitimate traffic or analyzing false alarm. This cost can be estimated based on the business impact specific to the site as follow.

$$L(r_i) = FP_{r_i} \times Cost_{r_i} \tag{5.3}$$

where FP_{r_i} is the false positive rate for rule r_i and $Cost_{r_i}$ is the cost score for rule r_i.

The rule weight $w(r_i)$ can be computed using the potential damage and operational

loss as follows:

$$w(r_i) = \big(\alpha \times D(r_i) - (1-\alpha) \times L(r_i)\big) \times H(r_i) \times P_M \tag{5.4}$$

where α is a configurable parameter indicating how much of the rule weight should rely on the potential damage and operational loss, $H(r_i)$ is a vector indicating the proportion of malicious events for rule r_i among all the malicious events for all rules $i = 1, \ldots, N$, and P_M is the prior probability of attack occurrence. In Algorithm 1, the rule weight $w(r_i)$ function is leveraged to select preventive rules that maximize the system security level within the maximum allowed delay D_{max} (line 7-18). The factors used to calculate the rule weight can be obtained during the site-specific risk analysis. The goal is to benefit from information gathered by security administrators during the site-specific risk analysis process about activities that were encountered in the past. Among these are false positive rate, false negative rate, and H and P_M.

5.3 Performance Evaluation and Results

In this section, we evaluate the efficiency of RMST. A number of simulation experiments were conducted to evaluate RMST. We also present the optimal solution produced by solving the Binary Integer Program (BIP) of the RMS problem using matlab [293].

In the first experiment, we examine the accuracy of our technique in selecting the preventive rules subset. We analyze the impact of varying the maximum delay constraint (D_{max}) on the resulting security and on the preventive rules selection. In this scenario,

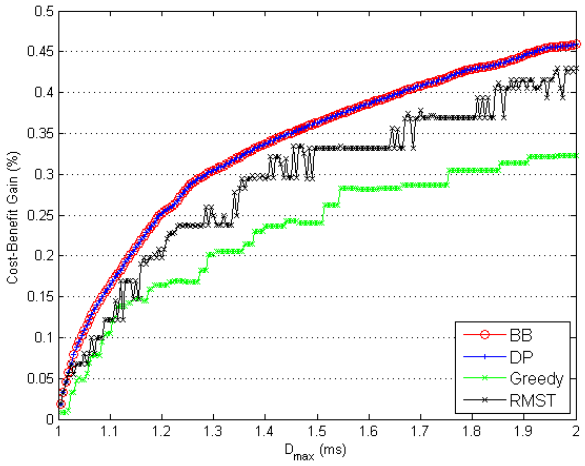


Figure 5.1: Gain percentage when varying delay constraint using different methods

the total number of rules is 200. The weight and processing time of the rules are assigned based on zipf distribution [299] similar to [69].

Three techniques beside RMST are evaluated and all techniques' results are compared in Figure 5.1. A simple solution for the RMS problem can be obtained by a greedy algorithm, where rules are sorted by their processing time in decreasing order and chosen sequentially until the maximum allowed delay is achieved. However, the cost-benefit gain of the greedy algorithm is not desirable in terms of system security performance. Although the cost-benefit gain obtained using branch and bound (BB) and dynamic programming (DP) for the RMS problem is better than our proposed technique, when the number of rules increases the computational time of (BB) or (DP) becomes unacceptable for our application. In turn the proposed RMST solves the problem in polynomial time, and the obtained rule set has an acceptable security level.

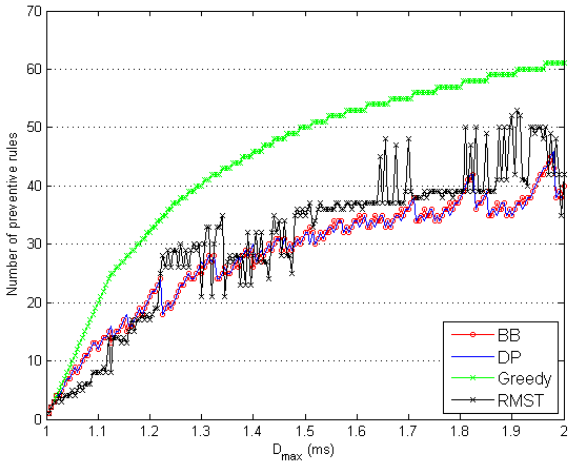


Figure 5.2: Number of selected rules versus D_{max} using different methods

The qualitative and quantitative comparison of selected rules is shown in Figure 5.2. The comparison shows that the number of preventive rules selected in (BB) or (DP) is lower than in others; however, the cost-benefit gain is maximum. On the other hand, the quality of the rules selected by the greedy algorithm is the lowest in terms of the cost-benefit gain, so the number of the preventive rules is high in order to satisfy the system security performance, whereas RMST selects prevention rules with a more balanced cost-benefit gain.

The scale of the IDPS system affects the required computational time to find an optimal solution. Specifically, 2^N combinations have to be computed in order to obtain the optimal solution. When the rules number is high, the search space is very large; therefore, the computational time needed is high, too. Figure 5.3 shows the relationship between the number of rules and the computational time. The Branch and Bound method takes a

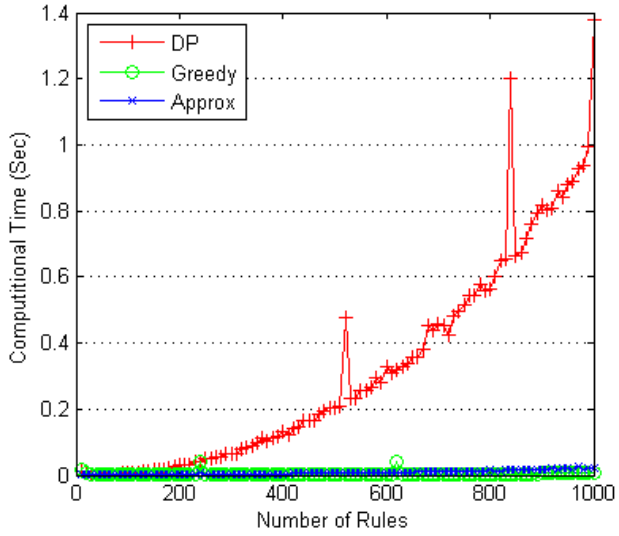


Figure 5.3: Computational time when increasing preventive rules

longer time so it is not included in Figure 5.3. The greedy algorithm and RMST do not suffer from scalability issue. While the number of preventive rules and the computational time are related exponentially in Dynamic Programming (DP), limiting the applicability of DP to a small number of IDPS rules.

The second set of experiments studies the impact of the accuracy of the rules set in terms of FP and FN on the average response time and the total number of preventives rules selected by BB and RMST techniques. The average response time is measured by the performance analysis model presented in Section 4.4 for any configuration set chosen by BB and RMST. The total number of rules is chosen to be 100 in this experiment. Also, we set the probability of maliciousness to be $P_M = 0.5$ and $\alpha = 0.7$.

Figures 5.4, 5.5, and 5.6 plot the average response time as a function of increasing

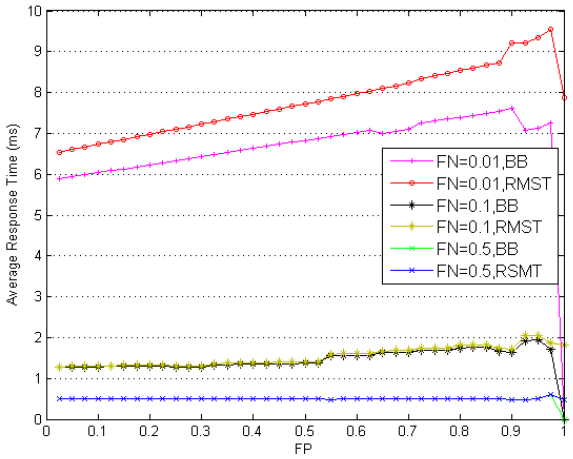


Figure 5.4: Impact of detection rates on average response time when D_{max}=high

both the false positive (FP) and false negative (FN) rates. Figure 5.4 presents the results when D_{max} is relatively high. We can see that the average response time decreases with an increase in the FN for all FP values. We can see that the average response time is longer when the IDPS becomes accurate in terms of the FN rate, no matter what the FP rates are. We can see that the BB technique adapts its selection criteria according to the change in the accuracy values while the RMST remains the same. This happens because assigning a high value to the D_{max} constraint is similar as if we are relaxing it.

Figure 5.5 illustrates the impact of the accuracy parameters when D_{max} is set to be low. The figures share similar results to the previous case, except that the gap between BB and RMST in average response time is reduced. This is because the low value of the D_{max} restriction makes the BB adapt its selection criteria. Overall, with a high FN rate, the average response time is similar for both techniques, although the BB has better

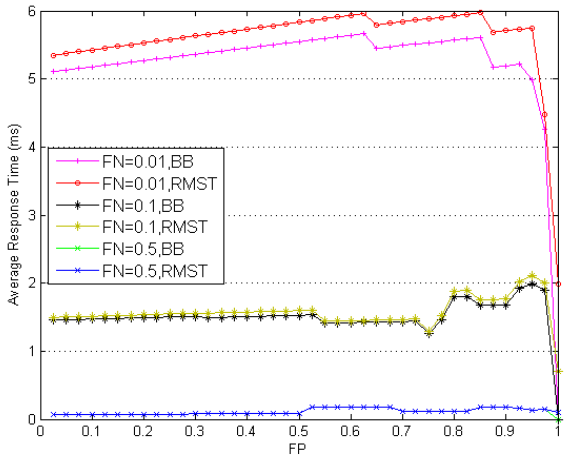


Figure 5.5: Impact of detection rates on average response time when D_{max}=low

average response time when the FN rate is very low. In other words, FN rate affects the optimization results, and thus system performance will be affected.

Finally, 5.6 shows the sharp relation between FN rate and system performance in terms of number of preventive rules and average response time.

5.4 Conclusion

We proposed a rule mode selection optimization technique called RMST that aims at determining an appropriate IDPS configuration set that maximizes security enforcement levels while avoiding unnecessary network performance degradation. With a small number of IDPS rules, a configuration set can be completed to achieve optimal results; however, when the number of IDPS rules is large, the computation time becomes unacceptable from

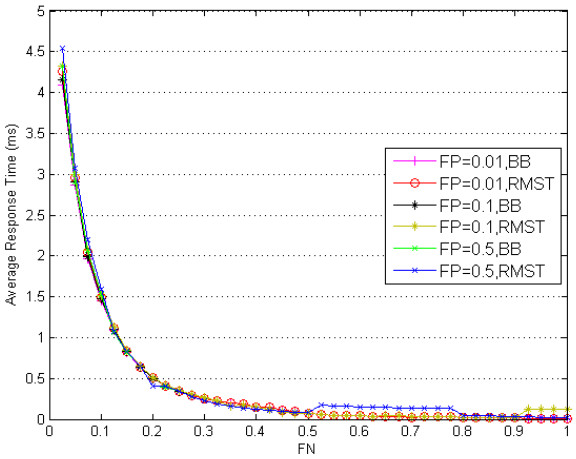


Figure 5.6: Impact of detection rates on average response time when D_{max}=low

a network performance point of view. RMST was shown to reconcile these two conflicting goals of guaranteeing network security and network performance.

Chapter 6

Markovian based Queuing Analysis Model

6.1 Introduction

The design of an IDPS poses many challenges including how to identify attack signatures, how to improve the detection engine, and how to manage the system. A large body of work has addressed these issues; however, little work focused on the impact of deploying an IDPS on network performance in terms of processing delay, throughput, and packet loss [31, 52, 99, 236]. Ironically, while the IDPS can efficiently detect and prevent many attacks that can compromise network performance, it may cause performance degradation itself. For instance, the processing delay of the IDPS can increase significantly as the size of the signature database grows. In addition, the IDPS may not be able to cope with the

increase in the amount of traffic, mainly because of limited resources in terms of CPU and memory. This particular case usually leads to an increase in packet queue length, resulting in higher waiting times for packets, and eventually packets to be dropped. Here, the IDPS becomes the network bottleneck, and its intervention no longer appropriate. To decrease queuing delay, processing time, and packet loss due to the IDPS, its configuration should be adjusted such that the complexity of the detection engine is reduced by decreasing IDPS attack coverage. As a consequence, the operator faces a trade-off between guaranteeing required security enforcement levels on one hand and Quality of Service (QoS) requirements on the other. Thus, it is crucial to study the impact of different IDPS configurations on network performance and select the one that satisfies both security objectives and QoS requirements.

In this chapter we analyze the performance of the IDPS for different configurations and under different traffic characteristics. Different from existing works on IDPS performance analysis, we develop an analytical model for the system based on an embedded Markov chain, to predict the impact of IDPS operations on network performance. We then leverage the model to provide mathematical derivations of key performance metrics; namely: the throughput, queuing delay, system utilization, and packet loss at the IDPS level. We define many configurations for the IDPS that reflect security enforcement levels and we study their effect on the network performance metrics under different traffic intensities. The analytical model is then validated through extensive simulations. Ultimatelly, our model allows security administrators to strike a balance between security level and network performance.

The remainder of the chapter is organized as follows. Section 6.2 presents our analytical

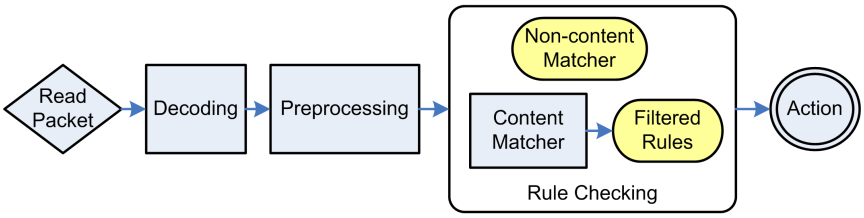


Figure 6.1: Analysis Tasks for Intrusion Detection and Prevention Systems

model for the IDPS based on a finite queuing system. In Section 6.3, we provide the performance evaluation of the IDPS analysis model. Finally, we draw our conclusions in Section 6.4.

6.2 Queuing Analysis Model

In this study, we consider that the IDPS operation involves mainly two stages: header analysis and content analysis, as shown in Figure 6.1. The header analysis stage examines packets for many types of malicious activity. For instance, this first analysis can detect attacks that exploit fragmentation vulnerabilities, such as the Ping of Death attack where attackers use many small fragmented ICMP packets such that their assembling results in a huge packet that exceeds the maximum allowable size for an IP datagram [230]. If a particular packet is identified as belonging to an attack, there is no need to do any further analysis. Thus, the first stage processing result allows the IDPS to decide to either release the packet to the network without any further analysis or to forward it to the content analysis for further checking. The content analysis stage is usually a rule-checking engine that uses a signature database to determine whether the examined packet belongs to any malicious traffic. Generally, this second stage is more time-consuming than the first one.

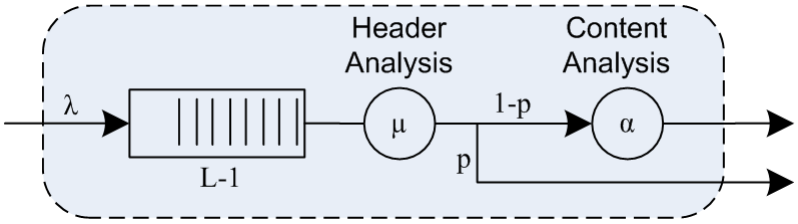


Figure 6.2: Tandem Queue Model with Blocking

In this study, we use a finite queuing model to analyze the performance of the IDPS.

6.2.1 Finite Queuing Model

In order to model the IDPS, we propose a two-stage embedded Markov chain [142, 174, 213, 257], as shown in Figure 6.2. The model consists of a FIFO queue and two servers. The first and second servers correspond respectively to the header analysis and content analysis stages. We assume that packets arrive at the IDPS according to a Poisson process with an arrival rate λ. We assume a fixed packet size in order to simplify the analysis. Packets join a finite queue of a maximum size $L-1$ ($L \in \mathbb{N}^*$). Only one packet at a time is passed to the header analysis stage, which has an average service time of $1/\mu$. When the header analysis is completed, the packet either leaves the system with an *early decision probability* p or moves to the second stage, with a probability $(1-p)$. The content header process has an average service time of $1/\alpha$. Service times for both stages are exponentially distributed. Packets are considered lost only when the buffer becomes full and they cannot join the IDPS system. In order to make the analytical solution feasible, we consider Poisson arrivals, exponential services, and fixed packet sizes. This applies only for certain types of traffic, as reported in [166]. Considering other distributions and variable packet sizes is

part of our future work.

On the other hand, we assume that the header analysis stage can not accept any new packets if the content analysis process is already busy. Thus, the execution of the two stages is mutually exclusive, i.e., if one of the stages is running, the other is idle. This assumption is realistic for two reasons. The first reason is that the CPU is usually executing only one task at a time. The second reason is that if we allow the pre-processor stage to accept packets while the rule-checking stage is busy, we need to introduce another queue at the second server. However, this situation can result in out-of-order packets, as the service times of the two stages are different and some packets can directly leave the system from the first stage. In practice, the service time of packet header analysis is much lower than that of the content checking ($\alpha < \mu$), since the underlying idea behind the header analysis stage is to avoid detailed rule-checking if an anomaly is detected by looking at the header alone.

We represent the behavior of the multi-stage service queuing system by a finite queuing model based on the embedded Markov chain process with a state space $S = \{(n, m), 0 < n \leq L, m = \{1, 2\}\}$, where n denotes the number of packets in the system and m denotes the stage which the IDPS is performing. In particular, in the first stage ($m = 1$), the IDPS is performing the packet processing task, and when $m = 2$, the IDPS is performing the rule-checking process. The queuing system has a buffer size of $L-1$. State $(0, 0)$ represents the special case when the IDPS is idle.

From the state transition diagram depicted in Figure 6.3, we can infer steady-state

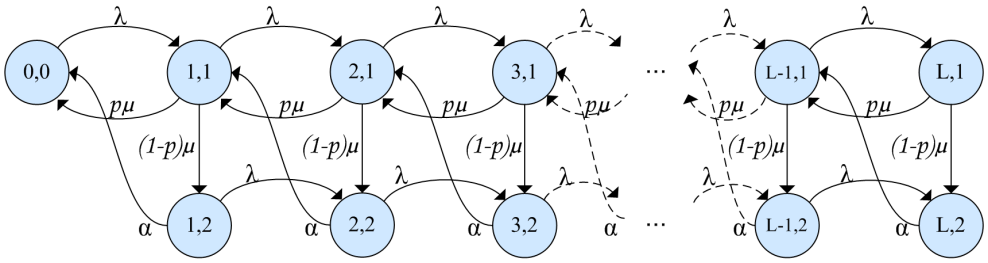


Figure 6.3: State Transition Diagram for Finite Queuing Model

equations. Thus, for the initial states (0,0), (1,1), and (1,2), we have:

$$0 = -\lambda q_{0,0} + p\mu q_{1,1} + \alpha q_{1,2} \qquad \text{at state } (0,0)$$

$$0 = -(\lambda + \mu) q_{1,1} + \lambda q_{0,0} + p\mu q_{2,1} + \alpha q_{2,2} \qquad \text{at state } (1,1)$$

$$0 = -(\lambda + \alpha) q_{1,2} + (1-p)\mu q_{1,1} \qquad \text{at state } (1,2)$$

For intermediate states $(n,1)$ and $(n,2)$, where $n \in [2, L-1]$, we have:

$$0 = -(\lambda + \mu) q_{\text{n,1}} + \lambda q_{\text{n-1,1}} + p\,\mu\, q_{n+1,1} + \alpha q_{\text{n+1,2}} \qquad \text{at state } (n,1)$$

$$0 = -(\lambda + \alpha) q_{\text{n,2}} + \lambda q_{\text{n-1,2}} + (1-p)\mu q_{\text{n,1}} \qquad \text{at state } (n,2)$$

At the boundary states $(L,1)$ and $(L,2)$, steady-state equations are expressed as follows:

$$0 = -\mu q_{\text{L,1}} + \lambda q_{\text{L-1,1}} \qquad \text{at state } (L,1)$$

$$0 = -\alpha q_{\text{L,2}} + \lambda q_{\text{L-1,2}} + (1-p)\mu q_{\text{L,1}} \qquad \text{at state } (L,2)$$

Based on these equations, we would like to express $q_{\text{n,1}}$ and $q_{\text{n,2}}$ in terms of $q_{0,0}$. It

is straightforward to calculate the probabilities of the initial and boundary states. Thus, from state equations $(0,0)$ and $(1,2)$, the probabilities $q_{1,1}$ and $q_{1,2}$ can be expressed in terms of $q_{0,0}$ as follows:

$$q_{1,1} = \frac{(\alpha+\lambda)\lambda}{(\alpha+p)\mu} q_{0,0}$$
$$q_{1,2} = \frac{\lambda - p\lambda}{\alpha + p\lambda} q_{0,0}$$

In order to calculate the state probabilities $q_{\mathrm{n},1}$ and $q_{\mathrm{n},2}$ in terms of $q_{0,0}$ where $n \in [1, L-1]$, we define $(w_{\mathrm{n},1})_{n\in[1,L-1]}$ and $(w_{\mathrm{n},2})_{n\in[1,L-1]}$ such as:

$$
\begin{cases}
w_{0,1} = w_{0,2} = 1 & \\
w_{1,1} = \frac{(\alpha+\lambda)\lambda}{(\alpha+p)\mu} & \\
w_{1,2} = \frac{\lambda - p\lambda}{\alpha + p\lambda} & \\
w_{\text{n,1}} = \frac{(\alpha+\lambda)(\lambda+\mu)}{(p\lambda+\alpha)\mu} w_{\text{n-1,1}} - \frac{(\lambda+\alpha)}{(\lambda p+\alpha)} w_{\text{n-2,1}} & 2 \leq n < L \\
\qquad - \frac{(\lambda\alpha)}{(\lambda p+\alpha)\mu} w_{\text{n-1,2}} & \\
w_{\text{n,2}} = \frac{\lambda}{\lambda+\alpha} w_{\text{n-1,2}} + \frac{(1-p)\mu}{\lambda+\alpha} w_{\text{n,1}} & 2 \leq n < L
\end{cases}
\tag{6.1}
$$

Using state equations $(n, 1)$ and $(n, 2)$ along with the definition of $(w_{\text{n,1}})_{n\in[1,L-1]}$ and $(w_{\text{n,2}})_{n\in[1,L-1]}$, we can express $q_{\text{n,1}}$ and $q_{\text{n,2}}$ in terms of $q_{0,0}$ as follows [1] :

$$
\begin{aligned}
q_{\text{n,1}} &= w_{n,1} q_{0,0} \\
q_{\text{n,2}} &= w_{n,2} q_{0,0}
\end{aligned}
\tag{6.2}
$$

[1] These equations can be easily demonstrated using mathematical induction, where steady-state equations $(2, 1)$, $(2, 2)$ and $(n-1, 1)$, $(n, 2)$ can be used respectively to check the base case and validate the inductive step.

Furthermore, the probabilities at the boundaries are calculated in terms of $q_{0,0}$ based on equations $(L,1)$, $(L,2)$ and (6.1) as follows:

$$q_{\mathrm{L},1} = \begin{cases} \left(\frac{\lambda}{\mu}\right) w_{\mathrm{L\text{-}1,1}} q_{0,0} & L > 1 \\ \left(\frac{\lambda}{\mu}\right) q_{0,0} & L = 1 \end{cases}$$

$$q_{\mathrm{L},2} = \begin{cases} \frac{\lambda}{\alpha}(w_{\mathrm{L\text{-}1,2}} + (1-p) w_{\mathrm{L\text{-}1,1}}) q_{0,0} & L > 1 \\ \left(\frac{(1-p)\lambda}{\alpha}\right) q_{0,0} & L = 1 \end{cases}$$

The next step is to determine $q_{0,0}$. For this purpose we use the normalization condition, which is expressed as follows:

$$q_{0,0} + \sum_{n=1}^{L} (q_{\mathrm{n},1} + q_{\mathrm{n},2}) = 1 \tag{6.3}$$

Using equation (6.2), we obtain:

$$\begin{aligned} q_{0,0} + \sum_{n=1}^{L} (w_{\mathrm{n},1} + w_{\mathrm{n},2}) q_{0,0} &= 1 \\ \Rightarrow q_{0,0} &= \frac{1}{1 + \sum_{n=1}^{L} (w_{\mathrm{n},1} + w_{\mathrm{n},2})} \end{aligned} \tag{6.4}$$

Based on equations (6.1), (6.2), and (6.4), it is possible to calculate the steady-state probabilities, and then we are able to determine the different performance metrics of the system at the steady state. The next subsection discusses those metrics.

Security level	Service time	
	Stage one($1/\mu$)	Stage two($1/\alpha$)
1	0.5 $\boldsymbol{\mu s}$	4 $\boldsymbol{\mu s}$
2	0.5 $\boldsymbol{\mu s}$	8 $\boldsymbol{\mu s}$
3	0.5 $\boldsymbol{\mu s}$	12 $\boldsymbol{\mu s}$
4	5 $\boldsymbol{\mu s}$	12 $\boldsymbol{\mu s}$

Table 6.1: Security enforcement levels and their corresponding processing time

6.2.2 Performance metrics

In this section, we identify the metrics that should be measured at the IDPS system level and which have an impact on the network performance. Particularly, end-to-end delay and packet loss ratio can be directly affected respectively by the average time spent in the IDPS per packet and the packet loss ratio at the IDPS level. Furthermore, other important metrics are also considered in our study, such as the mean system throughput, the average number of packets in the system, and packet average waiting delay. In the following, we provide the equation of each of those metrics as a function of the steady-state probabilities [257].

The average of the IDPS throughput γ is the average number of packets (per second) leaving the IDPS system either from the first stage or the second one and regardless of the decision of the IDPS with respect to packets. It is expressed as follows:

$$\gamma = p\mu \sum_{n=1}^{L} q_{\mathrm{n},1} + \alpha \sum_{n=1}^{L} q_{\mathrm{n},2} \tag{6.5}$$

The packet loss probability q_{lost} is the probability of being in the state $(L, 1)$ or $(L, 2)$. This means that the queue is full, and as a consequence incoming packets will not be admitted.

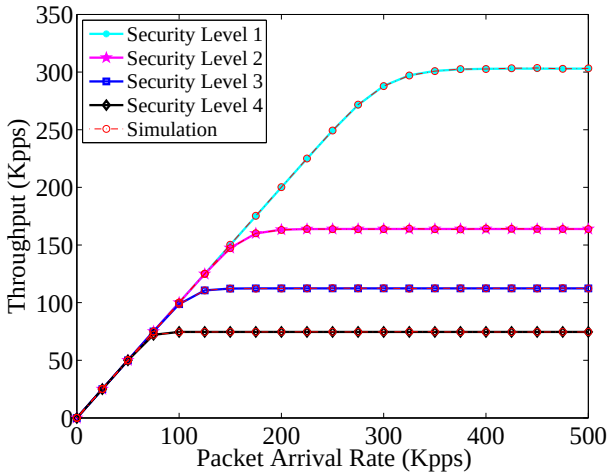


Figure 6.4: Throughput versus packet arrival rate for different Security Enforcement Levels ($L = 25$, $p = 0.3$).

It is given by:

$$q_{\text{lost}} = q_{\text{L},1} + q_{\text{L},2} \tag{6.6}$$

The average number of packets $\bar{X}$ in the system can be expressed as follows:

$$\bar{X} = \sum_{n=1}^{L} n(q_{\text{n},1} + q_{\text{n},2}) \tag{6.7}$$

The average time that a packet spends in the system W_s is then expressed using $\bar{X}$ and γ as:

$$W_s = \frac{\bar{X}}{\gamma} \tag{6.8}$$

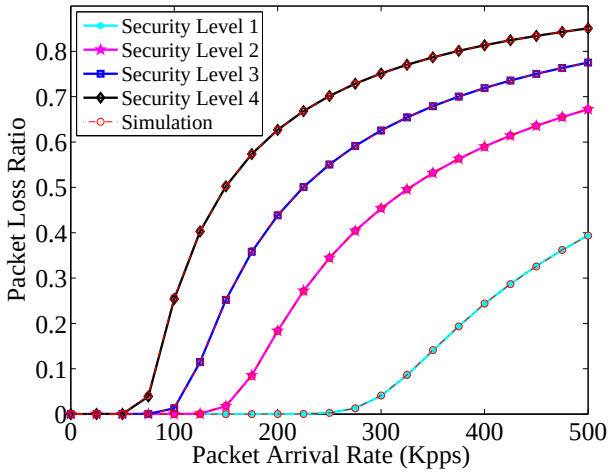


Figure 6.5: Packet loss ratio versus packet arrival rate for different Security Enforcement Levels ($L = 25$, $p = 0.3$).

The average service time of the two stages denoted W_a is given by:

$$W_a = \frac{1}{\mu} + \frac{1-p}{\alpha}$$

The average time spent by a packet in the queue W_q can be measured as follows:

$$W_q = W_s - W_a \tag{6.9}$$

Based on those equations, we can compute the values of all the performance metrics. The next section is dedicated to performance evaluation and the validation of the analytical model using simulation .

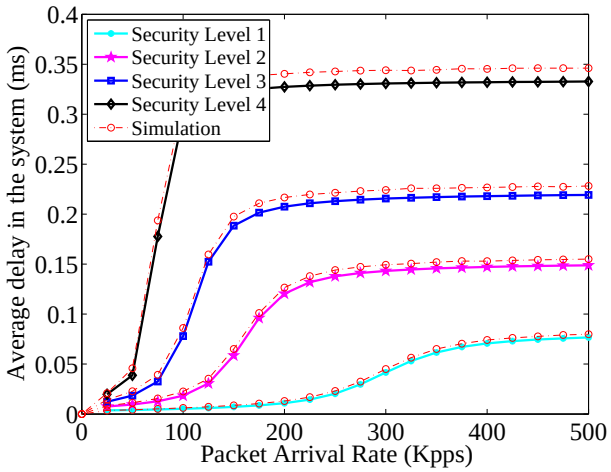


Figure 6.6: Average time spent in the system per packet versus packet arrival rate for different Security Enforcement Levels ($L = 25$, $p = 0.3$).

6.3 Performance Evaluation and Results

The goal here is to validate our analytical model through simulations and to analyze the effect of different security levels on network performance. We describe the settings of the different security enforcement levels and then analyze the effect of the packet arrival rates, the queue size, and the early decision probability p on the studied performance metrics. For all experiments, we provide the results determined from the analytical model and from simulations.

6.3.1 Experiments Settings

For the sake of our experiment, we defined four configurations that reflect different levels of detection capabilities of the IDPS, and Table 6.2.1 illustrates the selected configuration for each. While setting those values, we consider that the average service time at stage 2 is

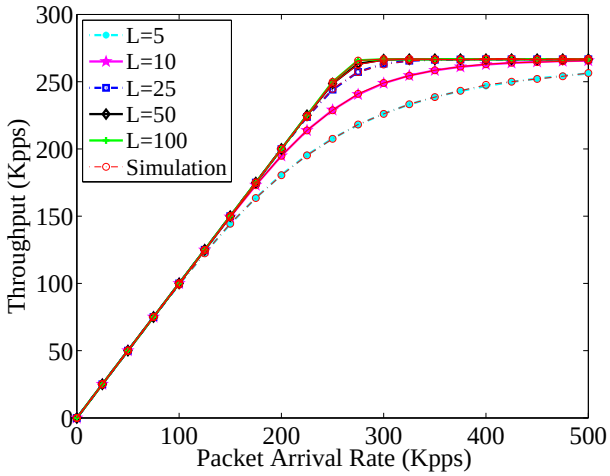


Figure 6.7: Throughput versus packet arrival rate for different queue size values (*Security Level* 1, $p = 0.3$).

an increasing function of the number of selected rules. This is motivated by our previous work, where we provide an analytical model that relates the service time to the number of rules [31]. This was also shown in other works using real data [265].

At the first three levels, we fixed the processing time at the first stage. As the security level is increased, the IDPS enlarges the selection of checking rules to improve the security coverage, and therefore the server processing time at stage 2 increases. For instance, level 1 corresponds to the minimum detection level where a small set of rules is checked at stage 2, whereas levels 2 and 3 increase the size of the rule database to provide better detection of malicious traffic. In addition, security level 4 is a particular case where both stages have to check a larger number of rules in order to increase the protection capability of the IDPS, especially at stage 1.

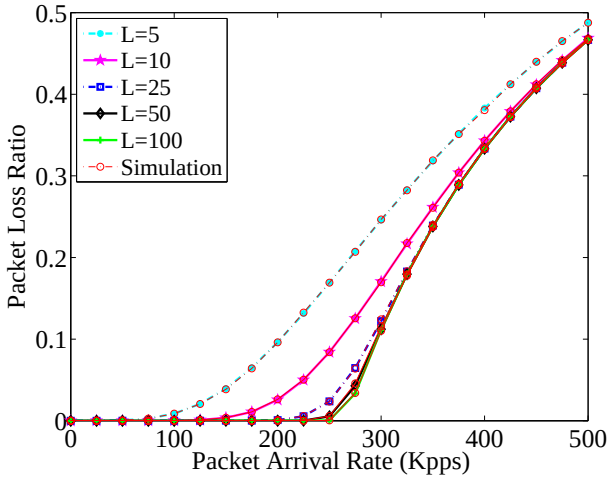


Figure 6.8: Packet loss ratio versus packet arrival rate for different queue size values (*Security Level* 1, $p = 0.3$).

6.3.2 Results

We conducted several experiments in order to validate the analytical results. Using Matlab [293], we implemented a discrete-event simulation of finite queueing for the system. Every simulation goes through independent sub-runs with different initial seeds (after discarding the transient part), and is finished when the confidence interval of 95% is achieved [208].

In our first experiment, we evaluate the accuracy of the proposed queuing model for a fixed queue size ($L = 25$), a fixed early decision probability ($p = 0.3$) and for different security enforcement levels. The results are depicted in Figures 6.4, 6.5, and 6.6. Each figure shows a performance metric calculated, while varying the arrival rate, based on the analytical model compared with the corresponding value obtained by simulation. We can observe that the values obtained by simulation are almost identical to the analytical ones (notice that in all figures, circles, which represent values obtained by simulations,

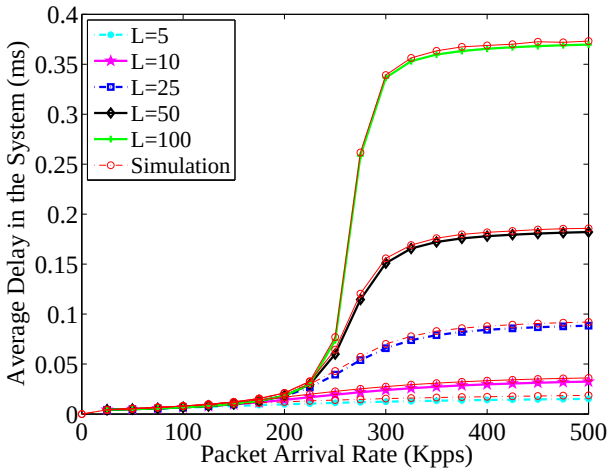


Figure 6.9: Average time spent in the system per packet versus packet arrival rate for different queue size values (*Security Level* 1, $p = 0.3$).

are overlapping with the points which represent the values obtained using the analytical model).

On the other hand, the figures show the effect of the security level on the performance metrics. As the security level is set higher, we can observe that the throughput (expressed in Kilo packets per second (Kpps)) is decreasing (Figure 6.4) while the packet loss ratio (Figure 6.5) and the average time spent in the system per packet (Figure 6.6) are increasing. This shows clearly that improving security coverage comes at the expense of performance. The degradation of performance is due to the extra analysis carried by the IDPS in order to cover more attacks (by checking more rules).

The second set of experiments was conducted to evaluate the effect of the queue size (L). Therefore, we fixed other parameters like the security level (set to 1) and probability (set to 0.3). Figures 6.7, 6.8, and 6.9 show the different metrics. It shows that although increasing the size of the queue can slightly improve the throughput (Figure 6.7), the

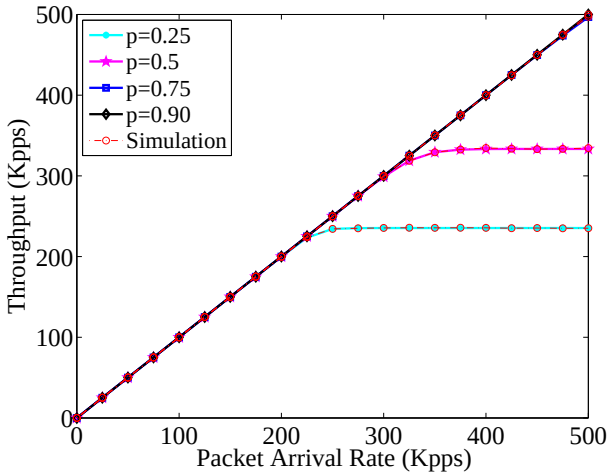


Figure 6.10: Throughput versus packet arrival rate for different early decision probabilities (*Security Level* 1, $L = 25$).

packet loss ratio does not change so much when the traffic rate gets higher (Figure 6.8). More importantly, a high queue size can cause a significant delay per packet, as shown in Figure 6.9. We can leverage such results in practice in order to configure the queue size of an IDPS that uses security level 1 and a probability of $p = 0.3$. As a practical example, assuming that to achieve the QoS target, we require a certain performance at the IDPS level, for example, a packet loss ratio of no more than 0.1, a processing time that does not exceed 0.03 ms, and a throughput higher than 225 Kpps. Knowing that the incoming traffic is fluctuating between 200 and 300 Kpps, we can infer from Figures 6.7, 6.8 and 6.9 that setting the queue size to 10 packets achieves the required QoS.

The final set of experiments investigates the effect of the early decision probability (p) on the different performance metrics. From Figure 6.10, it can be seen that for a high packet arrival rate, a high value of p can increase the IDPS throughput. This can be explained by the fact that the header analysis is able to take the decision upon the receival of the packet

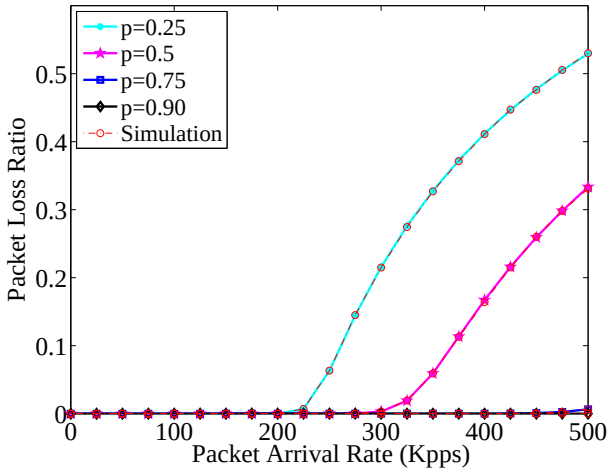


Figure 6.11: Packet loss ratio versus packet arrival rate for different early decision probabilities (*Security Level* 1, $L = 25$).

without the need for a content analysis process. Thus, the throughput increases while the average packet delay in the system is reduced. As a consequence, there are fewer packets in the queue and the packet loss ratio is almost zero, as shown in Figure 6.11. On the other hand, a small value of p can result in more packets being directed to the second stage, and thereby incurring a higher time spent in the system per packet. In this case, there are more packets waiting in the queue and the loss ratio can easily increase (Figure 6.12). These experiments demonstrate that if the header analysis can efficiently take a decision on the malicious traffic, thereby avoiding the second stage, it can significantly reduce packet loss as well as the average time spent in the system. Furthermore, these results can also help to choose the appropriate p value that can make the IPDS satisfy QoS requirements. However, this depends on whether the administrator can control the complexity and the accuracy of the header analysis stage.

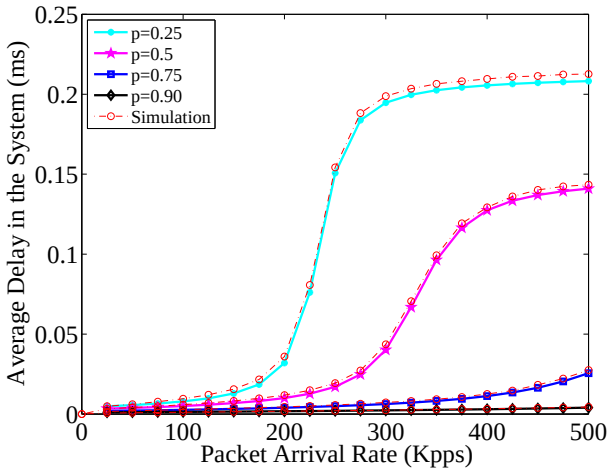


Figure 6.12: Average time spent in the system per packet versus packet arrival rate for different early decision probabilities (*Security Level* 1, $L = 25$).

6.4 Conclusion

Although intrusion detection systems can shield the network from various attacks and malicious traffic, they can have drawbacks, i.e., they can introduce significant delay and packet loss due to their large processing stages and eventually their inappropriate configuration. In this chapter, we have tried to address this particular problem by evaluating the impact of such systems on the key performance metrics. To this end, we modelled the IDPS as an analytical queuing model based on embedded Markov chain. We also performed extensive simulations that demonstrated the accuracy of the model. In addition, we analyzed through the results the effect of the different configuration parameters of the IDPS on network performance. This study provides concrete examples of how to tune those parameters in order to control the impact on network performance. As such, this model not only allows the analysis of various performance metrics at the IPDS level, but

it can be considered a valuable tool in setting up an appropriate configuration capable to strike a balance between guaranteeing a high security enforcement level and achieving network performance objectives.

Chapter 7

Elastic Cloud-based IDPS Placement

7.1 Introduction

Cloud computing has become a very cost-effective model for sharing large-scale services over the Internet in recent years. It's success is due to the attractive features offered by the underlying virtualization concept, including portability, isolation, live migration, and customizability of virtual machines [39]. Popular examples of cloud-based services are Microsoft Azure [5], Google AppEngine [110], and Amazon Elastic Computing Cloud (EC2) [244]. Cloud services are generally categorized into three areas: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). IaaS refers to on-demand provisioning of infrastructural resources, usually in terms of VMs. The cloud owner who offers IaaS is called an IaaS provider. Examples of IaaS providers include Amazon EC2 [244], GoGrid [87] and Flexiscale [89]. PaaS refers to providing

platform layer resources, including operating system support and software development frameworks. Examples of PaaS providers include Google App Engine [110], Microsoft Windows Azure [5] and Salesforce [90]. SaaS refers to providing on demand applications over the Internet. Examples of SaaS providers include Rackspace [100] and SAP Business ByDesign [68].

As cloud services are distributed in nature and provisioned through the Internet, they are prone to security threats [44]. IDPSs can be employed as a countermeasure. Traditionally, IDPSs are deployed as middleboxes at strategic places (such as the network edges) where traffic is forced to traverse through these middleboxes. This conventional approach is not scalable and flexible enough to cope up with increasing data traffic. Since all the communication traffic should be routed to these middle boxes, they could be become a bottleneck. One way to tackle this issue is to upgrade the network with more powerful middleboxes, yet it is very costly and time consuming. Moreover, this approach is rigid to topology changes due to tightly coupling between security and topology. Many Cloud Providers (CPs) such as Amazon AWS, Microsoft Azure and IBM Bluemix offer Security Function as a service (SFaaS). An Enterprise Client (EC) can deploy IDPS to cloud as Security Functions (NFs) and leverage the pay-per-use pricing model. Thus, an EC can eliminate the cost of provisioning NFs for peak-load on its own premises and only pay for the actually used resources in cloud. This can greatly reduce an EC's capital and operational expenditures.

From a CP standpoint, a core management problem to offer SFaaS is placing IDPS instances in cloud infrastructure, and allocate resources to these instances elastically according to IDPS service requests and workloads. The gaol is to utilize available bandwidth

and host resources optimally without violating Service Level Agreements (SLAs). However, elastically allocating and releasing resources incur costs, e.g., necessary VM migrations and corresponding service disruption [246]. An IDPS placement algorithm should consider these overheads, as well as the overall consumption of bandwidth and host resources.

Existing works in VM placement are not suitable for the placement of IDPSs for the reasons described by Bouet et al. [62]. Moreover, most of these works consider only a part of the problem by optimizing either host or bandwidth resource [53]. Additionally, the elastic IDPS placement area has not been explored sufficiently, and the few recent works [62, 86, 226, 249] do not address the challenges that may arise due to *conflicting objectives* and *elasticity*, as discussed below.

Conflicting Objectives: An optimal IDPS placement algorithm should minimize bandwidth and host resource consumption. Host resource consumption can be minimized by serving the IDPS request using minimum number of IDPS instances, which may be many hops away from the source or target of the request resulting into high bandwidth consumption. Bandwidth consumption, on the other hand, can be minimized by placing dedicated IDPS instances at the source or target of each request, but in this case host resource consumption will be high. Optimizing bandwidth resource consumption causes more host resource consumption, and vice versa. A naïve solution to optimize bandwidth consumption is to install an IDPS instance for each service request at the source or target of service traffic (if the source or target and IDPS instance reside in a same CP data-center). Besides, to optimize host resources, a solution is to install a minimum number of IDPS instances and routing traffic to these instances. This traffic routing consumes more bandwidth resources. In other words, both solutions overlook one aspect on the other and optimize the

utilization of one resource at the expense of the other. Therefore, the challenge is to find a trade-off between host and bandwidth resources consumption.

Elasticity: Horizontal and *vertical* scaling are the prominent mechanisms for achieving elasticity. Horizontal scaling is the *installation/removal* of IDPS instances, whereas vertical scaling is the*allocation/release* of host and bandwidth resources to/from an IDPS instance. In addition, live *migration* of IDPS instances [85] and *reassignment* of partial workload to another IDPS instance [132] can be employed to scale bandwidth and host resources. Employing these mechanisms can reduce operational cost, but each mechanism has its own overhead. The challenge is to determine which of these elasticity mechanisms is the most appropriate for a given workload.

To address the above challenges, we introduce the Elastic IDPS problem and propose an efficient solution called *Simple Lazy Facility Location (SLFL)*. Specifically:

1. We formulate the Elastic IDPS placement problem considering the challenges of conflicting objectives and elasticity.

2. We propose SLFL a a solution approach that optimizes placement of IDPS instances in response to new service requests, and workload variation.

3. The effectiveness of our solution is examined through realistic experiments. The results suggest that SLFL can accept two times more workload with 5–8% less operational cost compared to first-fit and random algorithms.

The rest of this chapter is organized as follows. The target problem and our solution are presented in Sections 7.2 and 7.3, respectively. The proposed solution is evaluated in

Section 7.4. Finally, the chapter is concluded in Section 7.5.

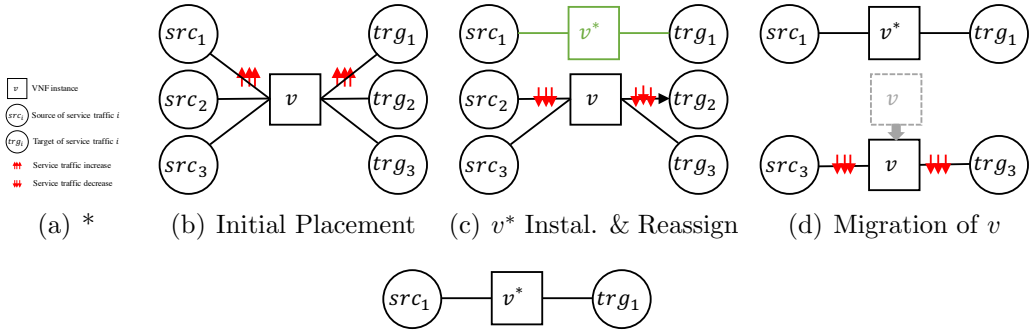
7.2 Elastic IDPS Placement

From the cloud provider perspective, an IDPS request can be modelled as follows: traffic (stream of packets) originating from a source is routed to an IDPS instance, where the packets are processed and forwarded to a target. If the source or target is outside the datacenter, we can assume a border router as the source or target of the traffic. IDPS instances reside at hosts within datacenter.

We are interested in an optimal cost-aware elastic placement of IDPS instances involving (i) selecting an optimal set of hosts on which IDPS instances are placed (ii) optimally allocating bandwidth resources to route service traffic, and (iii) applying the most cost-effective elasticity mechanism.

When to elastically scale? Resources must be scaled in response to two events: i) *Demand arrival*: when workload increases or a new service request is received; and ii) *Demand departure*: when one unit of service traffic drops. In addition, for the sake of simplicity, we assume that each demand consists of one unit of traffic transmitted from its source to an IDPS instance and delivered to its target, and requires a unit of IDPS resource to be served.

How to elastically scale? Due to the inflexibility in vertical scaling, we opt for horizontal scaling. We also assume one IDPS instance-type. A small instance-type is lightweight, can be easily distributed over the network and instantiated on-demand. More-



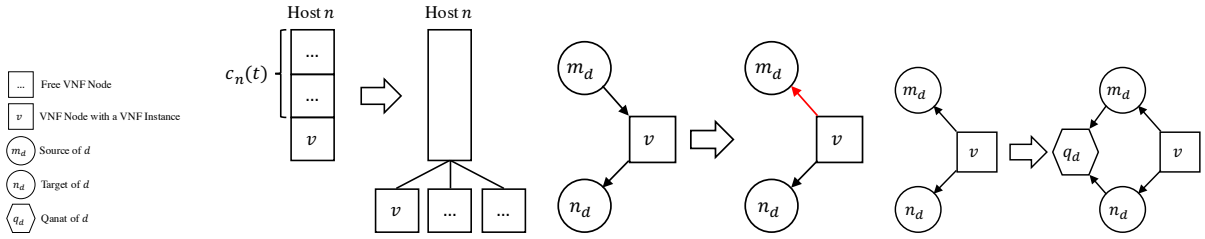
(a) * (b) Initial Placement (c) v^* Instal. & Reassign (d) Migration of v

(e) Removing v

Figure 7.1: Elasticity Mechanisms

over, having multiple instance-types can unnecessary complicate IDPS instance management tasks. In summary, we consider following elasticity mechanisms: *Installing* a new IDPS instance, *Removing* an existing IDPS instance, *Migrating* an IDPS instance, and *Reassignment* of a demand to another IDPS instance.

Figure 7.1 depicts the above elasticity mechanisms. Initially in 7.1(b), traffic from 3 requests are served by a single IDPS instance v. After sometime the traffic of the first request increases. To accommodate this new workload, a new IDPS v^* is instantiated and the first request is reassigned to v^* (Figure 7.1(c)). Then, traffic of the second request terminates, and allocated resources for this request are released. Because, v is still serving the third request, it is migrated to a more optimal location to reduce bandwidth usage (Figure 7.1(d)). Next, third request terminates, and IDPS instance v is removed to save host resources (Figure 7.1(e)).



(a) * (b) VNF Nodes (c) Traffic Flow Direction (d) Simplifying Demands

Figure 7.2: Simplified Model

7.2.1 Notation

Data Center Network

The data center network is denoted as a graph $G = (N, A)$, where N is a set of switches and hosts, while A is a set of links (arcs). We identify host nodes by N_H. Capacity of host $n \in N_H$ is denoted by the maximum number of IDPS instances that can be installed on n. Let $c_n(t)$ denote the available capacity of $n \in N_H$ at time t. Let w_{mn} and $c_{mn}(t)$ represent weight and capacity at time t of arc $(m, n) \in A$, respectively.

Demand

$D(t)$ is the set of demands at time t. A demand $d \in D(t)$ is identified by its two endpoints, source $m_d \in N$ and target $n_d \in N$. Let *demand nodes* refer to sources and targets. A demand needs a unit of traffic b and a unit IDPS resource to be served.

IDPS instance

$V(t)$ is the set of installed IDPS instances at time t. $c_v(t)$ denotes the number of demands that IDPS instance v can serve at time t. As we use one instance-type, we assume the maximum capacity is C. To show assignments of demands to IDPS instances, we use two maps at time t: $D_v(t)$ denoting a set of demands assigned to v and $v_d(t)$ representing the IDPS instance to which demand d is assigned.

7.2.2 Simplified Model

We refine the above model from two aspects in order to simplify our formulation. First, host and IDPS instance constraints are transformed to arc bandwidth capacity constraints; second, demands are simplified:

Constraints Transformation

For each host $n \in N_H$, $c_n(0)$ nodes are added, and we assume that IDPS instances are installed on these nodes. These nodes are called *IDPS nodes*. Imagine an IDPS instance v is installed on n at time t, and n still has capacity of $c_n(t) = 2$. As shown in Figure 7.2(b), three nodes are added. $F_n(t)$ denotes these nodes, and $F(t) = \bigcup_{n \in N_H}(F_n(t))$. Each $m \in F_n(t)$ is connected to n via arc (m, n). The capacity of arcs initially are set to $2 \times b \times C$, as at most, traffic of C number of demands enters and leaves m. These arcs force that no traffic can enter an IDPS instance node; however, this does not change the problem, because we can assume that the demand traffic is sent to a demand source

from the IDPS instance node, instead of the opposite direction (Figure 7.2(c)). Let $A_n^F(t)$ represent the arcs connecting IDPS nodes to the node n, and $A_F(t) = \bigcup_{n \in N_H}(A_n^F(t))$. Finally, let $n_v(t) \in F(t)$ be the IDPS instance node hosting IDPS instance v at time t, and $N_V(t) = \bigcup_{v \in V(t)}\{n_v(t)\}$.

Simplifying Demands

By previous transformation, we assume that IDPS instances send the traffic to demand nodes. We add a node q_d called Qanat to the graph for each demand $d \in D(t)$. Traffic is now received in q_d instead of m_d and n_d (Figure 7.2(d)). Let $Q(t) = \bigcup_{d \in D(t)}\{q_d\}$ denotes all Qanat nodes at time t. A q_d is connected to m_d and n_d via arcs (m_d, q_d) and (n_d, q_d). The capacity of these arcs initially are set to b, and their weights are set to 0. These two arcs ensure that if traffic reaches q_d, it has met m_d and n_d earlier.

7.2.3 Mathematical Model

A discrete-time system is considered to model the problem in which time is divided into equal slots $0 \leq t \leq T$.

Decision Variables

Let $x_{mn}^d(t) \in \mathbb{R}$ denotes the amount of traffic for demand d on arc $(m, n) \in A$ at time t. We derive two variables $y_n^d(t)$ and $z_n(t)$ from $x_{mn}^d(t)$. $y_n^d(t)$ denotes if demand d gets traffic from IDPS instance node n. $z_n(t)$ represents if an IDPS instance is installed in node n.

They are defined as follows.

$$\forall (n,m) \in A_F(t) : y_n^d(t) = \begin{cases} 1 & \textbf{if } x_{nm}^d(t) > 0 \\ 0 & \textbf{otherwise} \end{cases}$$

$$\forall n \in F(t) : z_n(t) = \begin{cases} 1 & \textbf{if } \sum_{d \in D(t)} y_n^d(t) > 0 \\ 0 & \textbf{otherwise} \end{cases}$$

Capacity Constraint

Eq. 7.1 ensures that arcs capacities are not violated.

$$0 \leq \sum_{d \in D(t)} x_{mn}^d(t) \leq c_{mn}(t) \text{ for } \forall (m,n) \in A \tag{7.1}$$

Flow Conservation Constraint

For each node $n \in N$, Eq. 7.2 guarantees that the amount of traffic entering and leaving n are equal, where n is not an IDPS node or a Qanat.

$$\sum_{(m,n) \in A} x_{mn}^d(t) - \sum_{(n,m) \in A} x_{nm}^d(t) = \begin{cases} -2b y_n^d(t) & \textbf{if } n \in F(t) \\ 2b & \textbf{if } n \in Q(t) \\ 0 & \textbf{otherwise} \end{cases} \tag{7.2}$$

Pair Connectivity Constraint

Eq. 7.3 ensures that a Qanat receives traffic from an IDPS instance, so both source and target of a demand are connected to the same IDPS instance.

$$\forall d \in D(t) : \sum_{n \in F(t)} y_n^d(t) = 1 \tag{7.3}$$

Installations Cost ($\mathcal{C}_{ins}(t)$)

The cost of installed IDPS instances at time t is defined by Eq. 7.4. f is the cost of host resources consumed by an IDPS instance for each time slot.

$$\mathcal{C}_{ins}(t) = f \sum_{n \in F(t)} z_n(t) \tag{7.4}$$

Transportation Cost ($\mathcal{C}_{tr}(t)$)

The cost of delivering demands traffic at time t is denoted by Eq. 7.5. g is the cost of a unit of bandwidth usage for each time slot.

$$\mathcal{C}_{tr}(t) = g \sum_{d \in D(t)} \sum_{(m,n) \in A} x_{mn}^d(t) w_{mn} \tag{7.5}$$

Reassignment Cost ($\mathcal{C}_{re}(t)$)

Eq. 7.6 is the cost of reassigning a set of demands at time t. $h_d(t)$ is the cost of moving state of demand d from original IDPS instance $v_d(t-1)$ to another IDPS instance $v_d(t)$. This cost depends on available bandwidth between these instances at time t. Here, $|y_n^d(t-1) \neq y_n^d(t)|$ is 1 if $y_n^d(t-1) \neq y_n^d(t)$ otherwise is 0.

$$\mathcal{C}_{re}(t) = \sum_{n \in F(t) \cap F(t-1)} \sum_{d \in D(t) \cap D(t-1)} \big(h_d(t) \; |y_n^d(t-1) \neq y_n^d(t)|\big) \tag{7.6}$$

Migration Cost ($\mathcal{C}_{mig}(t)$)

This is the cost of migrating a set of IDPS instances at time t. In Eq. 7.7, $k_v(t)$ is the cost of migrating IDPS instance v from $n_v(t-1)$ to $n_v(t)$ and depends on the available bandwidth between these two IDPS nodes. Here, $|z_n(t-1) \neq z_n(t)|$ is 1 if $z_n(t-1) \neq z_n(t)$, otherwise is 0.

$$\mathcal{C}_{mig}(t) = \sum_{n \in F(t) \cap F(t-1)} \big(k_v(t) | z_n(t-1) \neq z_n(t)|\big) \tag{7.7}$$

Objective Function

The objective is to minimize Eq. 7.8.

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T} \big(\mathcal{C}_{ins}(t) + \mathcal{C}_{tr}(t) + \mathcal{C}_{re}(t) + \mathcal{C}_{mig}(t)\big) \tag{7.8}$$

The static version of the problem at hand generalizes to the NP-Hard location routing problem (LRP) [25]. An optimal solution needs solving a dynamic version of LRP for each time slot. Due to its intractability, it is not possible to solve the problem for large datacenters. Thus, we break down the problem and solve it independently for each demand arrival or departure. Let $\hat{T}$ be a time system, and at each $t \in \hat{T}$, an arrival or departure occurs. We rewrite the objective function as Eq. 7.9. Here, λ_t is a weight factor to balance the transportation and installation costs with the migration and reassignment costs.

$$\min \Big(\sum_{t \in \hat{T}} (\mathcal{C}_{ins}(t) + \mathcal{C}_{tr}(t)) + \sum_{t \in \hat{T}} \lambda_t (\mathcal{C}_{re}(t) + \mathcal{C}_{mig}(t)) \Big) \tag{7.9}$$

Although, this problem is easier than the original one, it still generalizes to LRP. Hence, we propose a heuristic algorithm for solving this problem in the next section.

7.3 Simple Lazy Facility Location

In this section, we describe our solution, Simple Lazy Facility Location (SLFL), including two novel heuristics to handle arrival and departure events. For simplicity, we omit time variable t. Also, we assume function $flow(n, D^*, R^*)$ that finds an optimum way of routing traffic between a node n and demands D^*, and stores the routes in R^*. This function can find R^* in polynomial time by solving the single-commodity min-cost flow problem [136].

7.3.1 Demand Arrival

Upon new demand d arrival, a combination of installation, migrations and reassignments can be applied to optimize the placement. Because in most cases the arrival affects its locality, an algorithm that locally optimizes the placement is developed. This algorithm considers three possible actions (i) assignment to an existing IDPS instance, (ii) migration and (iii) creating a new IDPS instance followed by a set of reassignments. The first action assigns d to an existing IDPS instance with the minimum transportation cost. For the two other actions, *migration potential* and *installation potential* metrics are defined as follows:

Migration potential is the difference between current transportation cost of D_v and transportation cost of $D_v \cup \{d\}$ after possible migration of v to node n. Function pot_{mig}, defined in algorithm 2, finds this potential and stores routes in R^*.

Algorithm 2 Migration Potential

1: **function** $pot_{mig}(v, n, R^*)$
2: $\quad \mathcal{C} \leftarrow$ Transportation cost of D_v;
3: $\quad \mathcal{C}^* \leftarrow g \times flow(n, D_v \cup \{d\}, R^*) + \lambda \times k_v$;
4: $\quad$ return $(\mathcal{C} - \mathcal{C}^*)$ if ($\mathcal{C}^*$ is not ∞), otherwise $-\infty$;
5: **end function**

Installation potential is the difference between the operational cost before and after installing an IDPS instance in node n. We consider a set of reassignments during installation. Function pot_{ins}, as defined in Algorithm 3, computes the installation potential for a node n, finds candidate demands D^* for reassignment, and stores routes in R^*.

Algorithm 4 shows how SLFL handles a demand arrival. The cost of the best assignment is found (lines 3-4). The best migration and installation potential are computed (lines 4-8). If the best installation potential is greater than the best migration potential, a new

Algorithm 3 Installation Potential

1: **function** $pot_{ins}(n, R^*)$
2: $e^* \leftarrow -g \times flow(n, \{d\}, R^*)$; $D^* \leftarrow \emptyset$;
3: **for** $j = 1$ to $C - 1$ **do**
4: $d^* \leftarrow$ Find best next demand to reassign;
5: $e \leftarrow C - (g \times flow(n, D^* \cup \{d, d^*\}, \mathcal{R}) + \lambda \times h_{d^*})$;
6: break if ($e \leq 0$ or $e \leq e^*$);
7: $e^* \leftarrow e$; $D^* \leftarrow D^* \cup \{d^*\}$; $R^* \leftarrow \mathcal{R}$;
8: **end for**
9: return ($e^* - f$, D^*);
10: **end function**

IDPS v is instantiated and demands D_{re} are reassigned to v (lines 9-14). Otherwise, IDPS instance v_{mig} is migrated to n_{mig} (lines 14-18). Finally, in the lack of potential to change, d is assigned to IDPS instance v_{asn}.

7.3.2 Demand Departure

Similar to an arrival event, SLFL locally optimizes the placement of IDPS instances upon departure of demand d. Assume that d was assigned to IDPS instance v. Two actions are considered: (i) migration of v, and (ii) removal of v. We define *migration potential* and *removal potential* metrics for migration and removal of v as follows:

migration potential, as defined in Algorithm 5, is the difference in transportation cost of D_v before and after migration of v to node n.

Removal potential is the difference in operational-cost before and after the removal of v. Similar to installation potential, a set of reassignments are considered. Function pot_{rmv}, defined in Algorithm 6, computes the removal potential of v, finds candidate IDPS instances V_{re} for reassignment of D_v, and stores routes in R^*.

Algorithm 4 SLFL-Demand Arrival

function DEMANDARRIVAL(d)
  $D \leftarrow D \cup \{d\}$;
  $v_{asn} \leftarrow \arg\min_{v \in V}\{flow(n_v, \{d\}, \emptyset)\}$
  $p_{asn} \leftarrow g \times flow(n_{v_{asg}}, \{d\}, R^*_{asg})$;
  $(v_{mig}, n_{mig}) \leftarrow \arg\max_{v \in V: n \in F}\{pot_{mig}(v, n, \emptyset)\}$;
  $e_{mig} \leftarrow pot_{mig}(v_{mig}, R^*_{mig})$;
  $n_{ins} \leftarrow \arg\max_{n \in F/N_V}\{pot_{ins}(n, \emptyset)\}$;
  $(e_{ins}, D_{re}) \leftarrow pot_{ins}(n_{ins}, R^*_{ins})$;
  if $(e_{ins} > -p_{asn})$ and $(e_{ins} \geq e_{mig})$ **then**
    $u \leftarrow$ install a facility at n_{ins};
    Reassign $\forall d \in D_{re}$ and assign d to u;
    Route related traffic based on R^*_{ins};
    $V \leftarrow V \cup \{u\}$;
  else if $e_{mig} > -p_{asn}$ **then**
    Migrate v_{mig} to node n_{mig};
    Assign d to v_{mig};
    Route traffic based on R^*_{mig};
  else
    Assign d to v_{asn};
    Route traffic based on R^*_{asg};
  end if
end function

Algorithm 5 Emigration Potential

function $pot_{emg}(v, n, R^*)$
  $\mathcal{C} \leftarrow$ Transportation cost of D_v;
  $\mathcal{C}^* \leftarrow g \times flow(n, D_v, R^*) + \lambda \times k_v$;
  return $(\mathcal{C} - \mathcal{C}^*)$ if $\mathcal{C}^*$ is not ∞, otherwise $-\infty$;
end function

Algorithm 6 Removing Potential

1: **function** $pot_{rmv}(v, R^*)$
2: $\{D_{re}, V_{re}\} \leftarrow \{\emptyset, \emptyset\}$; $e^* \leftarrow f$;
3: $\mathcal{C} \leftarrow$ Transportation cost of D_v;
4: $\mathcal{C}^* \leftarrow 0$; $U \leftarrow V/\{v\}$;
5: **for all** $d_v \in D_v$ **do**
6: $v_{re} \leftarrow$ best IDPS instance for reassignment of d_v;
7: $\mathcal{C}^* \leftarrow \mathcal{C}^* + g \times flow(n_{v_{re}}, \{i\}, R^*_{re}) + \lambda \times h_{d_v}$;
8: **if** $\mathcal{C}^*$ is ∞ **then**
9: $e^* \leftarrow 0$; $\{D_{re}, V_{re}\} \leftarrow \{\emptyset, \emptyset\}$;
10: break;
11: **end if**
12: $\{V_{re}, D_{re}\} \leftarrow \{V_{re} \cup v_{re}, D_{re} \cup d\}$;
13: $R^* \leftarrow R^* \cup R^*_{re}$;
14: **end for**
15: return $\big(e^* + (\mathcal{C} - \mathcal{C}^*), \{D_{re}, V_{re}\}\big)$;
16: **end function**

Finally, algorithm 7 defines how SLFL handles a demand departure. First, v's resources assigned to d are released (line 2). Then, the best node to migrate and migration potential are computed (lines 3-4). The removal potential and possible reassignments are determined (line 5). If removal potential is positive and greater than migration potential, v is removed and its demands are reassigned to other IDPS instances (lines 6-12). Otherwise, if migration potential is positive, v is migrated to a more optimal node (lines 13-16).

Algorithm 7 SLFL-Demand Departure

function DEMANDDEPARTURE(d, v)
  Release v's resources assigned to d;
  $n_{emg} \leftarrow \arg\max_{n \in F/N_V} \{pot_{emg}(v, n, \emptyset)\}$;
  $e_{emg} \leftarrow pot_{emg}(v, n_{emg}, R^*_{emg})$;
  $(e_{rmv}, \{D_{re}, V_{re}\}) \leftarrow pot_{rmv}(v, R^*_{rmv})$;
  if $(e_{rmv} > 0)$ and $(e_{rmv} > e_{emg})$ **then**
    $V \leftarrow V/\{v\}$;
    Remove facility v;
    for all $\{d_{re}, v_{re}\} \in \{D_{re}, V_{re}\}$ **do**
      Reassign d_{re} to v_{re};
    end for
    Route traffic based to R^*_{rmv};
  else if $e_{emg} > 0$ **then**
    Migrate v to the node n_{emg};
    Route traffic based to R^*_{emg};
  end if
end function

7.4 Evaluation

7.4.1 Experimental Setup

We have implemented SLFL and evaluated its performance by simulations on a data center topology with 54 hosts. We used a 6-ary Fat-tree topology [23] providing full bisection bandwidth. Each host has 8 CPU cores, 8GB of memory, and contains a 1Gbps network adapter. A host CPU consumes 140W of power at electricity cost of ¢11 per kWh. We select Bro IDS [223] as a representative IDPS providing a capacity of 80 Mbps. We assume that Bro can be installed on a VM which requires 1 vCPU and, 1GB of RAM. In regards to bandwidth, we set the cost of using a unit of bandwidth for a link to 20% of the power consumption cost. Regrading live migration, the full contents of IDPS instance's memory

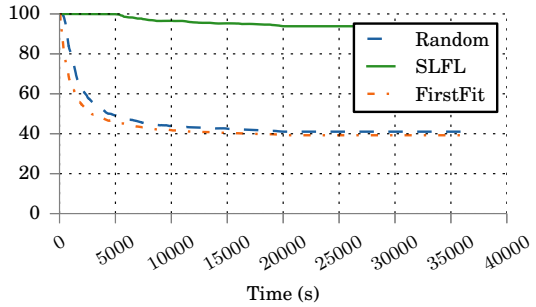


Figure 7.3: Workload Acceptance

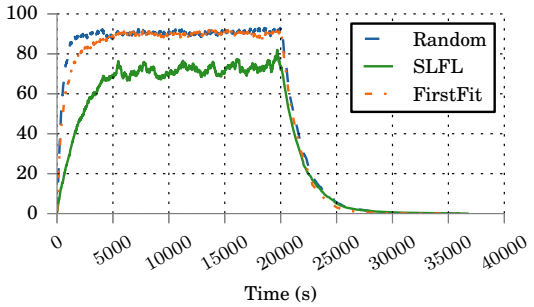


Figure 7.4: Bandwidth Resource Utilization

(1GB) is transported. Reassignment involves transporting a fraction of an IDPS instance's memory. This fraction is relative to the IDPS instance's maximum capacity.

We model demand arrival using Poisson distribution with an average rate of 1 demand per second. The lifetime of a demand follows an exponential distribution with an average of 1800 seconds. Demand nodes are uniformly distributed in the data-center network. We set $b = 20$Mbps and $\lambda = 1$.

We compare SLFL with *random* and *first fit* algorithms. Upon a demand arrival, random algorithm randomly selects an IDPS instance with sufficient available bandwidth. Otherwise, an IDPS instance is installed in a not-saturated host. First-fit algorithm selects the first not-saturated IDPS instance with adequate available bandwidth. If not, an IDPS instance is installed in the first not-saturated host. Upon demand departure, both algorithms remove an IDPS instance if this IDPS instance has no assigned demand.

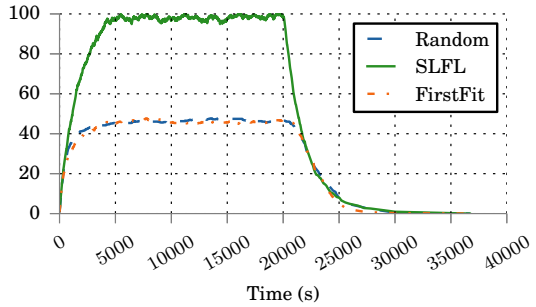


Figure 7.5: Host Resource Utilization

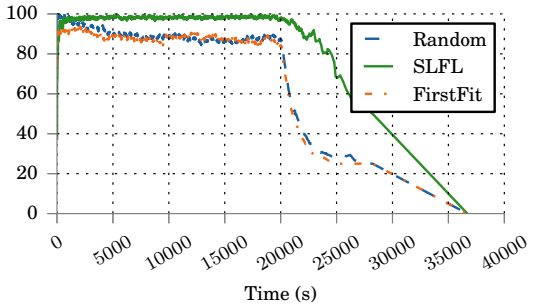


Figure 7.6: IDPS Resource Utilization

7.4.2 Acceptance and Utilization

Figures 7.3, 7.4, 7.5, and 7.6 depict workload acceptance and resources utilization. As shown in 7.3, SLFL accepts 97% of workload whereas random and first-fit accept 48% and 45% of workload, respectively. Bandwidth, host, and IDPS resource utilization are depicted in 7.4, 7.5 and 7.6, respectively. Random and first-fit approaches quickly exhaust bandwidth resources (utilization of 94% and 92%, respectively) resulting into low host resource utilization (45% and 44%, respectively). Moreover, these algorithms utilize 88% and 87% of IDPS resources, respectively. SLFL achieves bandwidth, host and IDPS resource utilization of 82%, 91% and 98%, respectively.

Operational costs are reported in Figures 7.7, 7.8, 7.9, and 7.10. Compared to random and first-fit algorithms, SLFL incurs 9% and 4% less operational cost (7.7), and pays 22% and 19% less bandwidth cost (7.8), respectively. However, SLFL incurs two times more installation cost (7.9) compared to the random and first-fit algorithms. The reason is that SLFL accepts two times more workload than the other approaches. Finally, 7.10 shows the overhead of SLFL. SLFL does reassignments more frequently than migrations. The

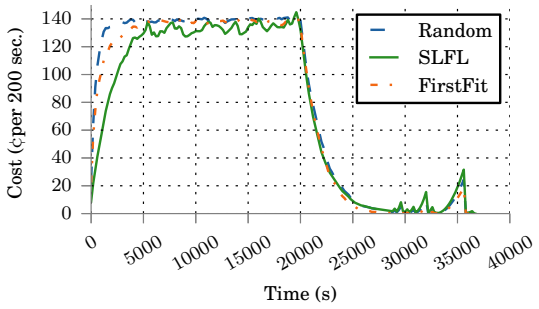


Figure 7.7: Total Operational Cost

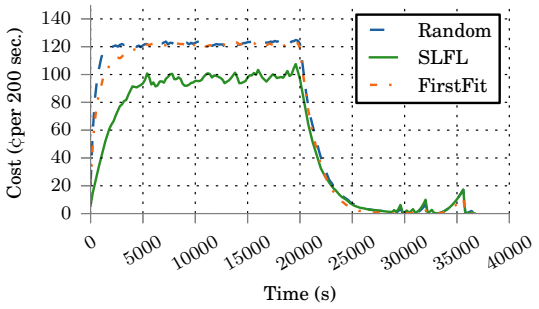


Figure 7.8: Transportation Cost

reason is that reassignment requires transmission of a part of IDPS instance state, whereas migration requires transmission of the entire memory of an IDPS instance.

7.5 Conclusion

In this chapter, we have introduced Elastic IDPS Placement problem and presented a model for minimizing operational costs in providing IDPS as a service. We considered the elasticity overhead and the trade-off between bandwidth and host resource consumption. We developed and evaluated an algorithm, named SLFL, to solve this problem in polynomial time. Our experiments suggest that by taking both bandwidth and host resources into consideration, and by carefully selecting the right elasticity mechanism, SLFL can accept $\sim 2\times$ more workload in comparison to first-fit and random algorithms. Additionally, SLFL incurs 5–8% less operational cost.

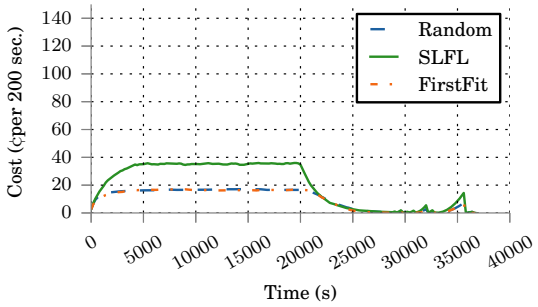


Figure 7.9: Installation Cost

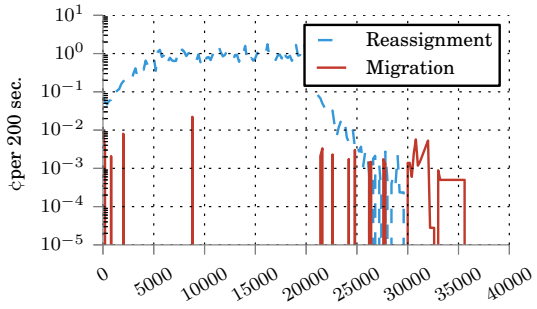


Figure 7.10: SLFL Overhead

Chapter 8

Conclusion and Future work

Intrusion detection and prevention systems (IDPSs) are software/hardware systems designed to monitor network traffic or computer activities and generate alerts when suspicious intrusions are detected. IDPSs can prevent intrusions to be successful attacks by blocking them using various methods. Although many IDPS systems have been proposed in the past, their configuration and management for effective detection and prevention of attacks are still challenging tasks. This is mainly because of the significant degradation of system performance when maximum security is applied; hence arises the need to trade off between security enforcement levels on one hand and the performance and usability of an enterprise information system on the other. In this thesis, we discussed the challenges underlying the design and management of intrusion detection and prevention systems. Specifically, we highlighted the open research issues raised by the conflict between security and performance, and the importance of dynamic adaptation and reconfiguration to reconcile them. In this perspective, we proposed several mechanisms to address these key issues, including

policy-based adaptation, performance analysis, trade-offs in configuration management, resource management, and elastic resource allocation.

For policy-based adaptation, we proposed a policy-based framework (AdapSec) in Chapter 3 for adaptive security configuration management of IDPSs. AdapSec provides policy-based dynamic adaptation and reconfiguration of the deployed levels of security measures. Policies are used to define and dynamically maintain the right balance between effective security and system usability/performance. We evaluated AdapSec through experiments on a testbed including Snort and *FireCol* .

For performance analysis (Chapter 4), we proposed a probabilistic model to investigate the relationship between the IDPS performance and its configuration. We showed that applying different sets of rules categories and configuration parameters impacts the average service time and affects system security. In addition, we proposed in Chapter 5 a rule mode selection optimization technique that determines the IDPS configuration set that maximizes the security enforcement levels while avoiding unnecessary network performance degradations. The proposed technique analyses the impact of the detection rates (i.e., FP and FN) on the performance of the IDPS.

For resource management, we developed an embedded Markov chain model (Chapter 6) to analyze the performance of the IDPS for different configurations and under different traffic characteristics. This study showed how to predict the impact of the IDPS parameters on network performance and provided concrete examples of how to tune those parameters. The evaluation of our models (Chapter 4, 5, and 6) was performed using a discrete-event simulation tool developed under Matlab.

Finally, we leveraged the elasticity of resource allocation in cloud computing environments to optimize the placement of IDPS appliances in response to new service requests, and workload variation (Chapter 7) and thereby achieve dynamic IDPS scaling. We believe the later capability is helpful in dynamically balancing security and performance objectives.

The contribution of this thesis is four folds. First, we applied probabilistic performance and queueing theoretical modelling to solve intrusion detection problems. Second, we introduced the concept of dynamic IDPS to achieve a balance between a high security enforcement level and network performance objectives. Third, we presented a model for minimizing operational costs in providing IDPS as a service in the Cloud. Fourth, we proposed a rule mode selection optimization technique to determine appropriate IDPS configurations.

Possible future work includes: (1) Develop the architecture components not addressed in this thesis, namely the alert management component and the monitoring module; (2) Extend AdapSec to consider attack graphs, attack statistical relationships, and learning mechanisms to define effective adaptation policies and security levels; (3) Evaluate the impact of potential damage $D(r_i)$ and operational loss $L(r_i)$ parameters on performance; (4) Investigate the performance of different resource management solutions to mitigate DoS attacks targeting worst case scenario in the IDPS ruleset; (5) Consider the use of feedback control-theoretic approaches for dynamic adjustment of IDPS configurations. This way, AdapSec will be able to cope with network traffic dynamics while balancing the security-performance trade-off in an autonomic manner; (6) Investigate the use of software-defined networking (SDN) to enhance IDPS dynamic scaling, for example by steering demand request to IDPS instances using SDN.

References

[1] Barracuda Web Application Firewall. https://goo.gl/p3OVvX.

[2] Cisco Cloud Services Router 1000V. https://goo.gl/xCr84r.

[3] CloudStack. http://cloudstack.apache.org/.

[4] ETSI Network Functions Virtualisation Introductory White Paper. https://portal.etsi.org/nfv/nfv_white_paper.pdf.

[5] http://www.microsoft.com/azure/default.mspx.

[6] libvirt. http://libvirt.org/.

[7] Network Functions Virtualisation. https://goo.gl/eDXR9N.

[8] Network Functions Virtualisation (NFV); Use Cases. https://goo.gl/1HrZlw.

[9] Network Functions Virtualisation Introductory White Paper. https://goo.gl/7RG3YM.

[10] OpenStack. http://www.openstack.org/.

[11] OpNFV - White Paper.

[12] RightScale Cloud Management. http://www.rightscale.com.

[13] Riverbed SteelHead CX. https://goo.gl/bJCE9v.

[14] Sophos UTM. https://goo.gl/Jyitnw.

[15] Trend Micro Deep Security. https://goo.gl/wvtpTZ.

[16] VMware vSphere. http://www.vmware.com/ca/en/products/vsphere.

[17] Hussam Abu-Libdeh, Paolo Costa, Antony Rowstron, Greg O'Shea, and Austin Donnelly. Symbiotic routing in future data centers, 2010.

[18] G Acosta-Marum and MA Ingram. Georgia Institute of Technology. *IEEE Vehicular Technology Magazine*, 2007.

[19] Sara Ahmadian, Z Friggstad, and C Swamy. Local-search based approximation algorithms for mobile facility location problems. *... Symposium on Discrete Algorithms*, pages 1607–1621, 2013.

[20] Sara Ahmadian and Chaitanya Swamy. Improved approximation guarantees for lower-bounded facility location. *Approximation and Online Algorithms*, 2013.

[21] A.V. Aho and M.J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[22] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications*. 1993.

[23] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture.

[24] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture, 2008.

[25] M. Albareda-Sambola, J. A. Dıaz, and E. Fernández. A compact model and tight bounds for a combined location-routing problem. *Computers & Operations Research*, 32(3):407–428, 2005.

[26] T. Alharkan and P. Martin. Idsaas: Intrusion detection system as a service in public clouds. In *Proc. of IEEE/ACM ccgrid '12*.

[27] K. Alsubhi, F. Zhani, and R. Boutaba. Embedded markov process based model for performance analysis of intrusion detection and prevention system. In *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE Press, 2012.

[28] Khalid Alsubhi, Issam Aib, and Raouf Boutaba. A policy-based approach to risk management, a snort configuration use case. In *In Proc. of the 15th HP Software University Association (HP-SUA), HP LABS*, page 309 314, 2008.

[29] Khalid Alsubhi, Issam Aib, Jérôme François, and Raouf Boutaba. Policy-based security configuration management application to intrusion detection and prevention. In *Proceedings of the 2009 IEEE international conference on Communications (ICC)*, ICC'09, 2009.

[30] Khalid Alsubhi, Yassir Alhazmi, Nizar Bouabdallah, and Raouf Boutaba. Rule mode selection in intrusion detection and prevention systems. In *2011 IEEE Global Telecommunications Conference*, GLOBECOM'11, pages 1–6. IEEE Press, 2011.

[31] Khalid Alsubhi, Nizar Bouabdallah, and Raouf Boutaba. Performance analysis in intrusion detection and prevention systems. In *IFIP/IEEE Integrated Network Management Symposium (IM)*, 2011.

[32] HC An, A Bhaskara, and C Chekuri. Centrality of trees for capacitated k-center. *Integer Programming . . .*, 2014.

[33] K.G. Anagnostakis, EP Markatos, S. Antonatos, and M. Polychronakis. E2xb: A domain specific string matching algorithm for intrusion detection. In *Proc. of the 18th IFIP International Information Security Conference (SEC2003)*, pages 217–228, 2003.

[34] Aris Anagnostopoulos, Russell Bent, Eli Upfal, and Pascal Van Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Information and Computation*, 194(2):175–202, November 2004.

[35] James W Anderson, Ryan Braud, Rishi Kapoor, George Porter, and Amin Vahdat. xomb: Extensible open middleboxes with commodity servers. In *Proc. of ACM/IEEE ANCS '12*.

[36] Konstantin Andreev, Charles Garrod, Daniel Golovin, Bruce Maggs, and Adam Meyerson. Simultaneous source location. *ACM Trans. Algorithms*, 6(1):16:1–16:17, December 2009.

[37] Bilal Anwer, Theophilus Benson, Nick Feamster, Dave Levin, and Jennifer Rexford. A slick control plane for network middleboxes. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 147–148. ACM, 2013.

[38] A Boloori Arabani and RZ Farahani. Facility Location Dynamics: An overview of classifications and applications. *Computers & Industrial Engineering*, 62(1):408–420, 2012.

[39] Junaid Arshad, Paul Townend, and Jie Xu. An automatic intrusion diagnosis approach for clouds. *International Journal of Automation and Computing*, 8(3):286–296, 2011.

[40] N.S. Artan, R. Ghosh, Y. Guo, and H.J. Chao. A 10-gbps high-speed single-chip network intrusion detection and prevention system. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 343–348. IEEE, 2007.

[41] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual {ACM} symposium on Theory of computing*, pages 21–29, 2001.

[42] Windows Azure. www.microsoft.com/azurem.

[43] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. The price is right: Towards location-independent costs in datacenters. In *In Hotnets '11*.

[44] Md Faizul Bari, Raouf Boutaba, Rafael Esteves, Lisandro Zambenedetti Granville, Maxim Podlesny, Md Golam Rabbani, Qi Zhang, and Mohamed Faten Zhani. Data center network virtualization: A survey. *Communications Surveys & Tutorials, IEEE*, 15(2):909–928, 2013.

[45] Md. Faizul Bari, Raouf Boutaba, Rafael Esteves, Lisandro Zambenedetti Granville, Maxim Podlesny, Md Golam Rabbani, Qi Zhang, and Mohamed Faten Zhani. Data Center Network Virtualization: A Survey. *IEEE Commun. Surv. Tutorials*, 15:909–928, 2013.

[46] Md Faizul Bari, Arup Raton Roy, Shihabur Rahman Chowdhury, Qi Zhang, Mohamed Faten Zhani, Reaz Ahmed, and Raouf Boutaba. Dynamic controller provisioning in software defined networks. In *CNSM*, pages 18–25, 2013.

[47] Cataldo Basile, Antonio Lioy, Gregorio Martinez Perez, Felix J. Garcia Clemente, and Antonio F. Gomez Skarmeta. Positif: A policy-based security management system. In *POLICY '07: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks*, page 280, Washington, DC, USA, 2007. IEEE Computer Society.

[48] Mohammadhossein Bateni and Mohammadtaghi Hajiaghayi. Assignment problem in content distribution networks. *ACM Transactions on Algorithms*, 8(3):1–19, July 2012.

[49] Mohammadhossein Bateni and Mohammadtaghi Hajiaghayi. Assignment problem in content distribution networks. *ACM Transactions on Algorithms*, 8(3):1–19, July 2012.

[50] Hendrik Baumann and Werner Sandmann. Markovian modeling and security measure analysis for networks under flooding dos attacks. In *Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on*, pages 298–302. IEEE, 2012.

[51] Matthew Baxter, Tarek Elgindy, Andreas T. Ernst, Thomas Kalinowski, and Martin W.P. Savelsbergh. Incremental network design with shortest paths. *European Journal of Operational Research*, 238(3):675–684, November 2014.

[52] S.M. Bellovin and R. Bush. Configuration management and security. *IEEE Journal on Selected Areas in Communications JSAC*, 27(3), 2009.

[53] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in Computers, M. Zelkowitz (ed.)*, 82:47–111, 2011.

[54] Robert Benkoczi and Binay Bhattacharya. Faster Algorithms for k-Medians in Trees. *... of Computer Science ...*, 09077:218–227, 2003.

[55] et al. Benson, Theophilus. Microte: Fine grained traffic engineering for data centers. In *Proc. of ACM CoNEXT '11.*

[56] O. Biran et al. A stable network-aware vm placement for cloud systems. In *CCGRID*, pages 498–506, 2012.

[57] Alireza Boloori Arabani and Reza Zanjirani Farahani. Facility location dynamics: An overview of classifications and applications. *Computers & Industrial Engineering*, 62(1):408–420, February 2012.

[58] Alireza Boloori Arabani and Reza Zanjirani Farahani. Facility location dynamics: An overview of classifications and applications. *Computers & Industrial Engineering*, 62(1):408–420, 2012.

[59] D. Bolzoni, S. Etalle, and P. Hartel. Poseidon: a 2-tier anomaly-based network intrusion detection system. In *Information Assurance, 2006. IWIA 2006. Fourth IEEE International Workshop on*, pages 10–pp. Ieee, 2006.

[60] Josh Borglund. Top 5 most costly viruses of all time. In *http://anti-virus-software-review.toptenreviews.com/top-5-most-costly-viruses-of-all-time.html*. Visited: January 2015, 2009.

[61] Allan Borodin. *Online computation and competitive analysis*, volume 2.

[62] M. Bouet, J. Leguay, and V. Conan. Cost-based placement of vdpi functions in nfv infrastructures.

[63] Raouf Boutaba and Issam Aib. Policy-based Management: A Historical Perspective. *Journal of Network and System Management*, 15(4):447–480, 2007.

[64] R.S. Boyer and J.S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977.

[65] A. Bremler-Barr, Y. Harchol, D. Hay, and Y. Koral. Deep packet inspection as a service. In *Proc. of ACM CoNEXT '14*.

[66] Susan M Bridges and Rayford B Vaughn. Fuzzy data mining and genetic algorithms applied to intrusion detection. In *Proceedings of 12th Annual Canadian Information Technology Security Symposium*, pages 109–122, 2000.

[67] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[68] SAP Business ByDesign. www.sap.com/sme/solutions/businessmanagement/businessbydesign/inde

[69] J.B.D. Cabrera, J. Gosar, W. Lee, and R.K. Mehra. On the statistical distribution of processing times in network intrusion detection. In *43rd IEEE Conference on Decision and Control (CDC)*. IEEE, 2004.

[70] JBD Cabrera, J. Gosar, W. Lee, and RK Mehra. On the statistical distribution of processing times in network intrusion detection. *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, 1, 2004.

[71] C Carver, JM Hill, John R Surdu, and Udo W Pooch. A methodology for using intelligent agents to provide automated intrusion response. In *Proceedings of the IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, West Point, NY*, pages 110–116, 2000.

[72] Curtis A Carver and Udo W Pooch. An intrusion response taxonomy and its role in automatic intrusion response. In *Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, pages 129–135, 2000.

[73] Curtis A Carver Jr. Adaptive agent-based intrusion response. Technical report, DTIC Document, 2001.

[74] M Casassa Mont, Adrian Baldwin, and Cheh Goh. Power prototype: Towards integrated policy-based management. In *Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP*, pages 789–802. IEEE, 2000.

[75] Rocky KC Chang. Defending against flooding-based distributed denial-of-service attacks: a tutorial. *Communications Magazine, IEEE*, 40(10):42–51, 2002.

[76] Suresh N. Chari and Pau-Chen Cheng. Bluebox: A policy-driven, host-based intrusion detection system. *ACM Trans. Inf. Syst. Secur.*, 6(2):173–200, 2003.

[77] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 626–635. ACM, 1997.

[78] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 1–10. ACM, 2001.

[79] Huoping Chen and Salim Hariri. An evaluation scheme of adaptive configuration techniques. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering (ASE)*, pages 493–496, New York, NY, USA, 2007. ACM.

[80] Jun Chen, Weidong Liu, and Jiaxing Song. Network performance-aware virtual machine migration in data centers. In *CLOUD COMPUTING 2012, The Third*

International Conference on Cloud Computing, GRIDs, and Virtualization, pages 65–71, 2012.

[81] Kai Chen, Ankit Singla, Atul Singh, Kishore Ramachandran, Lei Xu, Yueping Zhang, Xitao Wen, and Yan Chen. OSA: An optical switching architecture for data center networks with unprecedented flexibility. *IEEE/ACM Trans. Netw.*, 22:498–511, 2014.

[82] Y.M. Chen and Y. Yang. Policy management for network-based intrusion detection and prevention. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 219–239, 2004.

[83] Marek Chrobak. Sigact news online algorithms column 17. *ACM SIGACT News*, 42(1):97–131, 2010.

[84] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[85] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. of USENIX NSDI '05*.

[86] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca. The dynamic placement of virtual network functions. In *14th IEEE/IFIP Network Operations and Management Symposium 2014*, Krakow, Poland, May 2014.

[87] Cloud Computing Cloud Hosting and Hybrid Infrastructure from GoGrid. http://www.gogrid.com.

[88] B. Commentz-Walter. A string matching algorithm fast on the average. *Automata, Languages and Programming*, pages 118–132, 1979.

[89] FlexiScale Cloud Comp and Hosting. http://www.flexiscale.com.

[90] Salesforce CRM. http://www.salesforce.com/platform.

[91] T. Crothers. Implementing intrusion detection systems. *Recherche*, 67:02, 2002.

[92] Tim Crothers. *Implementing intrusion detection systems*. Wiley, 2003.

[93] András Császár, Wolfgang John, Mario Kind, Catalin Meirosu, Gergely Pongrácz, Dimitri Staessens, Attila Takács, and J Westphal. Unifying cloud and carrier network. *Proceedings of. DCC, Dresden, Germany, to appear Dec*, 2013.

[94] János Csirik, Leah Epstein, Csanád Imreh, and Asaf Levin. Online Clustering with Variable Sized Clusters. *Algorithmica*, 65(2):251–274, November 2011.

[95] William H Cunningham. A network simplex method. *Mathematical Programming*, 11(1):105–116, 1976.

[96] Marek Cygan. LP Rounding for k-Centers with Non-uniform Hard Capacities. *Foundations of Computer ...*, 2012.

[97] Amir Vahid Dastjerdi, Kamalrulnizam Abu Bakar, and Sayed Gholam Hassan Tabatabaei. Distributed intrusion detection in clouds using mobile agents. In *Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP'09. Third International Conference on*, pages 175–180. IEEE, 2009.

[98] Joao Porto de Albuquerque, Heiko Krumm, Paulo Licio de Geus, and René Jeruschkat. Scalable model-based configuration management of security services in complex enterprise networks. *Software: Practice and Experience*, 41(3):307–338, 2011.

[99] H. Debar and et. al. Towards a taxonomy of intrusion-detection systems. *COMPUT. NETWORKS*, 31(8):805–822, 1999.

[100] Web Hosting by Rackspace Hosting Dedicated Server, Managed Hosting. http://www.rackspace.com.

[101] Gabriella Divéki and Csanád Imreh. Online facility location with facility movements. *Central European Journal of Operations Research*, 19(2):191–200, July 2010.

[102] Gabriella Divéki and Csanád Imreh. An online 2-dimensional clustering problem with variable sized clusters. *Optimization and Engineering*, 14(4):575–593, October 2013.

[103] Pawan Prakash Dixit, Advait and Ramana Rao Kompella. On the efficacy of fine-grained traffic splitting protocolsin data center networks. In *ACM SIGCOMM CCR. Vol. 41. No. 4. ACM, 2011.*

[104] T. Dörges, K.P. Kossakowski, and P.C. GmbH. Proactive Security Monitoring in a Policy Managed Network. *18th Annual FIRST Conference*, 2006.

[105] Bharat Doshi. Analysis of a two phase queueing system with general service times. *Operations Research Letters*, 10(5):265 – 272, 1991.

[106] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer. Operational experiences with high-volume network intrusion detection. In *Proceedings of the 11th ACM conference on Computer and communications security*. ACM, 2004.

[107] H. Dreger, A. Feldmann, V. Paxson, and R. and Sommer. Predicting the resource consumption of network intrusion detection systems. In *Recent Advances in Intrusion Detection (RAID)*. Springer, 2008.

[108] David Eisenstat, Claire Mathieu, and N Schabanel. Facility location in evolving metrics. *arXiv preprint arXiv:1403.6758*, pages 1–12, 2014.

[109] A. El Rheddane. *Elasticity in the Cloud*. Theses, Université de Grenoble, February 2015.

[110] Google App Engine. http://code.google.com/appengine.

[111] Shimon Even, Alon Itai, and Adi Shamir. On the complexity of time table and multi-commodity flow problems. In *Foundations of Computer Science, 1975., 16th Annual Symposium on*, pages 184–193. IEEE, 1975.

[112] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. Approximating Metric Spaces by Tree Metrics. In *Encyclopedia of Algorithms*, pages 51–53. Springer, 2008.

[113] Wei Fan, Wenke Lee, Salvatore Stolfo, and Matthew Miller. A multiple model cost-sensitive approach for intrusion detection. *Machine Learning: ECML 2000*, pages 142–154, 2000.

[114] Reza Zanjirani Farahani and Masoud Hekmatfar. *Facility location: concepts, models, algorithms and case studies*. 2009.

[115] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, and Amin Vahdat. Helios : A Hybrid Electrical / Optical Switch Architecture for Modular Data Centers. *ACM SIGCOMM Comput. Commun. Rev.*, 40:339–350, 2010.

[116] San Felice, David P Williamson, and Orlando Lee. The Online Connected Facility Location Problem. (2012):574–585, 2014.

[117] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 14186:637–652, 2003.

[118] Dimitris Fotakis. Incremental Algorithms for Facility Location and k-Median. *Algorithms, ESA 2004*, 14186, 2004.

[119] Dimitris Fotakis. Incremental algorithms for Facility Location and k-Median. *Theoretical Computer Science*, 361(2-3):275–313, 2006.

[120] Dimitris Fotakis. Memoryless facility location in one pass. *STACS 2006*, pages 608–620, 2006.

[121] Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *Journal of Discrete Algorithms*, 5(1):141–148, March 2007.

[122] Dimitris Fotakis. On the Competitive Ratio for Online Facility Location. *Algorithmica*, 50(1):1–57, October 2007.

[123] Dimitris Fotakis. Online and incremental algorithms for facility location. *ACM SIGACT News*, 42(1):97–131, 2011.

[124] J. Francois, I. Aib, and R. Boutaba. Firecol: A collaborative protection network for the detection of flooding ddos attacks. *Networking, IEEE/ACM Transactions on*, 20(6):1828–1841, December 2012.

[125] Jérôme François, Adel El Atawy, Ehab Al Shaer, and Raouf Boutaba. A collaborative approach for proactive detection of distributed denial of service attacks. In *IEEE Workshop on Monitoring, Attack Detection and Mitigation - MonAM2007*, Toulouse France, 11 2007.

[126] Delbert R Fulkerson. An out-of-kilter method for minimal-cost flow problems. *Journal of the Society for Industrial & Applied Mathematics*, 9(1):18–27, 1961.

[127] Guilherme Galante and Luis Carlos E. de Bona. A survey on cloud computing elasticity. In *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, UCC '12, pages 263–270, Washington, DC, USA, 2012. IEEE Computer Society.

[128] Joaquin Garcia-Alfaro, Frédéric Cuppens, Nora Cuppens-Boulahia, and Stere Preda. Mirage: a management tool for the analysis and deployment of network security policies. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 203–215. Springer, 2011.

[129] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[130] A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella. Stratos: Virtual middleboxes as first-class entities. *UW-Madison TR1771*, 2012.

[131] Aaron Gember and Saul St John Anand Krishnamurthy. Virtual middleboxes as first-class entities in the cloud.

[132] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella. Opennf: Enabling innovation in network function control. In *Proc. of ACM SIGCOMM '14*.

[133] Abdolhamid Ghodselahi and Fabian Kuhn. Serving Online Demands with Movable Centers. *arXiv preprint arXiv:1404.5510*, pages 1–20, 2014.

[134] Abdolhamid Ghodselahi and Fabian Kuhn. Serving Online Demands with Movable Centers. page 20, April 2014.

[135] AliA. Ghorbani and Iosif-Viorel Onut. Y-means: An autonomous clustering algorithm. 6076:1–13, 2010.

[136] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM (JACM)*, 36(4):873–886, 1989.

[137] Andrew V Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of algorithms*, 22(1):1–29, 1997.

[138] Z. Gong, X. Gu, and J. Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *IEEE CNSM '10*.

[139] Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

[140] Albert Greenberg, James Hamilton, David a Maltz, and Parveen Patel. The Cost of a Cloud : Research Problems in Data Center Networks. *ACM SIGCOMM Comput. Commun. Rev.*, 39:68–73, 2009.

[141] By Albert Greenberg, James R Hamilton, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, A Maltz, Parveen Patel, Sudipta Sengupta, Albert Greenberg, Navendu Jain, and David A. Maltz. VL2: a scalable and flexible data center network. In *Proc. ACM SIGCOMM 2009 Conf. Data Commun.*, volume 09, pages 51–62, 2009.

[142] D. Gross and C.M. Harris. Fundamentals of queueing theory. 1998. *ISBN: 0-471-17083-6*, pages 244–247.

[143] G. Gu, P. Fogla, D. Dagon, W. Lee, and B. Skorić. Measuring intrusion detection capability: An information-theoretic approach. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*. ACM, 2006.

[144] Chuanxiong Guo, G Lu, Dan Li, Haitao Wu, and X Zhang. BCube: a high performance, server-centric network architecture for modular data centers. In *SIGCOMM '09 Proc. ACM SIGCOMM 2009 Conf. Data Commun.*, pages 63–74, 2009.

[145] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: a scalable and fault-tolerant network structure for data centers, 2008.

[146] B. Haagdorens, T. Vermeiren, and M. Goossens. Improving the performance of signature-based network intrusion detection sensors by multi-threading. *Information Security Applications*, pages 188–203, 2005.

[147] S. L. Hakimi. Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph, 1964.

[148] S. L. Hakimi. Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph, 1964.

[149] Ali Hammadi and Lotfi Mhamdi. A survey on architectures and energy efficiency in Data Center Networks, 2014.

[150] Rui Han, Li Guo, Moustafa M Ghanem, and Yike Guo. Lightweight resource scaling for cloud applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 644–651. IEEE, 2012.

[151] D.L. Hancock and G.B. Lamont. Multi agent system for network attack classification using flow-based intrusion detection. In *IEEE Congress on Evolutionary Computation (CEC)*, 2011.

[152] Ahmed Hassan and Waleed Bahgat. A framework for translating a high level security policy into low level security mechanisms. *Journal of Electrical Engineering*, 61(1):20–28, 2010.

[153] A. Hay, D. Cid, and R. Bray. *OSSEC host-based intrusion detection guide*. Syngress, 2008.

[154] Brandon Heller, Rob Sherwood, and Nick McKeown. The controller placement problem. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 7–12. ACM, 2012.

[155] N. R. Herbst, S. Kounev, and R. Reussner. Elasticity in cloud computing: What it is, and what it is not. In *Proc. of USENIX ICAC '13*.

[156] N. R. Herbst, S. Kounev, and R. Reussner. Elasticity in cloud computing: What it is, and what it is not. In *Proc. of USENIX ICAC '13*.

[157] F. Hermenier, X. Lorca, J. M. Menaud, G. Muller, and J. Lawall. Entropy: A consolidation manager for clusters. In *Proc. of ACM SIGPLAN/SIGOPS VEE '09*.

[158] A. Hess, H.-F. Geerdes, and R. Wessaly. Intelligent distribution of intrusion prevention services on programmable routers. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–11, 2006.

[159] Qiang Huang, Hisashi Kobayashi, and Bede Liu. Analysis of a new form of distributed denial of service attack. In *Proceedings of CISS03, the 37th Annual Conference on Information Science and Systems*, 2003.

[160] Qiang Huang, Hisashi Kobayashi, and Bede Liu. Modeling of distributed denial of service attacks in wireless networks. In *Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on*, volume 1, pages 41–44. IEEE, 2003.

[161] Jinho Hwang, KK Ramakrishnan, and Timothy Wood. Netvm: high performance and flexible networking using virtualization on commodity platforms. In *11th USENIX*

Symposium on Networked Systems Design and Implementation (NSDI 14). Seattle, WA: USENIX Association, pages 445–458, 2014.

[162] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 731–740, New York, NY, USA, 2002. ACM.

[163] SD Jena, JF Cordeau, and Bernard Gendron. Lagrangian Heuristics for Large-Scale Dynamic Facility Location with Generalized Modular Capacities. *INFORMS Journal on Computing*, (April), 2014.

[164] Hai Jin, Guofu Xiang, Deqing Zou, Song Wu, Feng Zhao, Min Li, and Weide Zheng. A vmm-based intrusion prevention system in cloud computing environment. *The Journal of Supercomputing*, 66(3):1133–1151, 2013.

[165] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *ACM SIGCOMM ACM SIGCOMM CCR. Vol 38. No. 4. ACM, 2008.*

[166] Mansour J. Karam and Fouad A. Tobagi. Analysis of delay and delay jitter of voice traffic in the Internet. *Computer Networks*, 40(6):711–726, dec 2002.

[167] Suraiya Khan and Issa Traore. Queue-based analysis of dos attacks. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 266–273. IEEE, 2005.

[168] Md Tanzim Khorshed, ABM Ali, and Saleh A Wasimi. A survey on gaps, threat remediation challenges and some thoughts for proactive attack detection in cloud computing. *Future Generation Computer Systems*, 28(6):833–851, 2012.

[169] Dong Seong Kim, Ha-Nam Nguyen, and Jong Sou Park. Genetic algorithm to improve svm based network intrusion detection system. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 2, pages 155–158. IEEE, 2005.

[170] G.H. Kim and E.H. Spafford. Experiences with tripwire: Using integrity checkers for intrusion detection. 1994.

[171] H.J. Kim, H.S. Kim, and S. Kang. A memory-efficient bit-split parallel string matching using pattern dividing for intrusion detection systems. *Parallel and Distributed Systems, IEEE Transactions on*, 22(11):1904–1911, 2011.

[172] Kyong Hoon Kim, Anton Beloglazov, and Rajkumar Buyya. Power-aware provisioning of virtual machines for real-time cloud services. *Concurr. Comput. : Pract. Exper.*, 23(13):1491–1505, September 2011.

[173] Zoltán Király and P. Kovács. Efficient implementations of minimum-cost flow algorithms. *CoRR*, abs/1207.6381, 2012.

[174] L. Kleinrock. Queueing systems. volume 1: Theory. 1975.

[175] D.E. Knuth, J.H. Morris Jr, and V.R. Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350, 1977.

[176] C.M. Krishna and Y.-H. Lee. A study of two-phase service. *Operations Research Letters*, 9(2):91 – 97, 1990.

[177] MIT Lincoln Lab. 2000 DARPA intrusion detection scenario specific datasets, 2000.

[178] Hoang Le and Viktor K. Prasanna. A memory-efficient and modular approach for large-scale string pattern matching. *Computers, IEEE Transactions on*, 62(5):844–857, 2013.

[179] T.H. Lee. Hardware architecture for high-performance regular expression matching. *Computers, IEEE Transactions on*, 58(7):984–993, 2009.

[180] W. Lee, J. Cabrera, A. Thomas, N. Balwalli, S. Saluja, and Y. Zhang. Performance adaptation in real-time intrusion detection systems. In *Recent Advances in Intrusion Detection (RAID)*. Springer, RAID, 2002.

[181] Wenke Lee, Wei Fan, Matthew Miller, Salvatore J. Stolfo, Erez Zadok, et al. Toward cost-sensitive modeling for intrusion detection and response. *Journal of Computer Security*, 10(1/2):5–22, 2002.

[182] Markus Leitner and Günther R. Raidl. Branch-and-Cut-and-Price for Capacitated Connected Facility Location. *Journal of Mathematical Modelling and Algorithms*, 10(3):245–267, March 2011.

[183] Dan Li, Chuanxiong Guo, Haitao Wu, Kun Tan, Yongguang Zhang, and Songwu Lu. FiConn: Using backup port for server interconnection in data centers. In *Proc. - IEEE INFOCOM*, pages 2276–2285, 2009.

[184] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.

[185] Shi Li and Ola Svensson. Approximating k-Median via Pseudo-Approximation. page 18, November 2012.

[186] Wei Li. Using genetic algorithm for network intrusion detection. *Proceedings of the United States Department of Energy Cyber Security Group*, pages 1–8, 2004.

[187] C.H. Lin, C.H. Liu, S.C. Chang, and W.K. Hon. Memory-efficient pattern matching architectures using perfect hashing on graphic processing units. In *INFOCOM 2012. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1978–1986. IEEE, 2012.

[188] P.C. Lin, Y.D. Lin, T.H. Lee, and Y.C. Lai. Using string matching for deep packet inspection. *Computer*, 41(4):23–28, 2008.

[189] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

[190] Yang Liu, Jogesh K Muppala, and Malathi Veeraraghavan. A Survey of Data Center Network Architectures.

[191] Men Long, Chwan-Hwa John Wu, and John Y Hung. Denial of service attacks on network-based control systems: impact and mitigation. *Industrial Informatics, IEEE Transactions on*, 1(2):85–96, 2005.

[192] T. Lorido-Botrán, J. Miguel-Alonso, and J. A. Lozano. Auto-scaling techniques for elastic applications in cloud environments. 2012.

[193] D. Luchaup, R. Smith, C. Estan, and S. Jha. Multi-byte regular expression matching with speculation. In *Recent Advances in Intrusion Detection (RAID)*, pages 284–303. Springer, 2009.

[194] Gabriel Maciá-Fernández, Jesús E Díaz-Verdejo, and Pedro García-Teodoro. Mathematical model for low-rate dos attacks against application servers. *Information Forensics and Security, IEEE Transactions on*, 4(3):519–529, 2009.

[195] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation Algorithms for Metric Facility Location Problems. *SIAM Journal on Computing*, 36(2):411–432, January 2006.

[196] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman. Vmflow: Leveraging vm mobility to reduce network power costs in data centers. In *IFIP NETWORKING, 2011*.

[197] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[198] Joao Martins, Mohamed Ahmed, Costin Raiciu, Vladimir Olteanu, Michio Honda, Roberto Bifulco, and Felipe Huici. Clickos and the art of network function virtualization. In *Proc. USENIX NSDI*, pages 459–473, 2014.

[199] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling inno-

vation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[200] Saeed Mehrabidavoodabadi. Online Problems in Facility Location. 2012.

[201] S. Mehraghdam, M. Keller, and H. Karl. Specifying and placing chains of virtual network functions. In *IEEE CloudNet '14*.

[202] C.R. Meiners, J. Patel, E. Norige, E. Torng, and A.X. Liu. Fast regular expression matching using small tcams for network intrusion detection and prevention systems. In *Proceedings of the 19th USENIX conference on Security*, pages 8–8. USENIX Association, 2010.

[203] P. Mell and T. Grance. The nist definition of cloud computing. 2011.

[204] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *IEEE INFOCOM, 2010*.

[205] Adam Meyerson. Online facility location. *... of Computer Science, 2001. Proceedings. 42nd ...*, pages 0–5, 2001.

[206] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and S. Davy. Design and evaluation of algorithms for mapping and scheduling of virtual network functions.

[207] Zhou Mingqiang, Huang Hui, and Wang Qian. A graph-based clustering algorithm for anomaly intrusion detection. In *Computer Science & Education (ICCSE), 2012 7th International Conference on*, pages 1311–1314. IEEE, 2012.

[208] I. Mitrani. *Simulation techniques for discrete event systems*. Cambridge Univ Pr, 1982.

[209] Nenad Mladenovi, Jack Brimberg, and Pierre Hansen. The p -median problem : A survey of metaheuristic approaches. (2):1–22.

[210] H. Moens and F. D. Turck. Vnf-p: A model for efficient placement of virtualized network functions. In *IEEE CNSM '14*.

[211] M Moorthy and S Sathiyabama. A study of intrusion detection using data mining. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pages 8–15. IEEE, 2012.

[212] Amuthan Prabakar Muniyandi, R Rajeswari, and R Rajaram. Network anomaly detection by cascading k-means clustering and c4. 5 decision tree algorithm. *Procedia Engineering*, 30:174–182, 2012.

[213] M.F. Neuts. *Matrix-geometric solutions in stochastic models: an algorithmic approach*. Dover Pubns, 1981.

[214] P. Ning, Y. Cui, D.S. Reeves, and D. Xu. Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):274–318, 2004.

[215] P. Ning, Y. Cui, D.S. Reeves, and D. Xu. Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security (TISSEC)*, 7(2), 2004.

[216] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, Amin (University of California) Vahdat, and Radhika Niranjan Mysore. PortLand: a scalable fault-tolerant layer 2 data center network fabric. In *SIGCOMM '09 Proc. ACM SIGCOMM 2009 Conf. Data Commun.*, pages 39–50, 2009.

[217] M. Norton. Optimizing pattern matching for intrusion detection. *white paper, Sourcefire Inc*, 2004.

[218] James B Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations research*, 41(2):338–350, 1993.

[219] Pradeep Padala, Kai-Yuan Hou, Kang G Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated control of multiple virtualized resources. In *Proc. of ACM EuroSys '09*, pages 13–26.

[220] Trend Micro White Paper. Best practices for security and compliance with amazon web services.

[221] VT Paschos. *Paradigms of Combinatorial Optimization: Problems and New Approaches*. 2013.

[222] V. Paxson, R. Sommer, and N. Weaver. An architecture for exploiting multi-core processors to parallelize network intrusion prevention. In *Sarnoff Symposium, 2007 IEEE*, pages 1–7. IEEE, 2007.

[223] Vern Paxson. Bro: a system for detecting network intruders in real-time. In *Proceedings of the 7th conference on USENIX Security Symposium (SSYM)*, Berkeley, CA, USA, 1998. USENIX.

[224] Lucian Popa. A Cost Comparison of Data Center Network Architectures. *ACM Conex.*, 6:1, 2010.

[225] Niels Provos. A virtual honeypot framework. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 1–1, Berkeley, CA, USA, 2004. USENIX.

[226] Z. A. Qazi, C. C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu. Simple-fying middlebox policy enforcement using sdn. In *Proc. of ACM SIGCOMM '13*.

[227] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield. Split/merge: System support for elastic execution in virtual middleboxes. In *Proc. of USENIX NSDI '13*.

[228] Shriram Rajagopalan, Dan Williams, and Hani Jamjoom. Pico replication: a high availability framework for middleboxes. In *Proc. of ACM SoCC '13*.

[229] J. Rao, X. Bu, C. Z. Xu, L. Wang, and G. Yin. Vconf: a reinforcement learning approach to virtual machines auto-configuration. In *Proc. of ACM ICAC '09*.

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

[231] Sebastian Roschke, Feng Cheng, and Christoph Meinel. Intrusion detection in the cloud. In *Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on*, pages 729–734. IEEE, 2009.

[232] E Rozenshine-Kemelmakher, R Puzis, A Felner, and Y Elovici. Cost benefit deployment of dnips. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010.

[233] Khaled Salah, Khalid Elbadawi, and Raouf Boutaba. Performance modeling and analysis of network firewalls. *IEEE Transactions on Network and Service Management*, 9(1):12–21, 2012.

[234] Hamed Salehi, Hossein Shirazi, and Reza Askari Moghadam. Increasing overall network security by integrating signature-based nids with packet filtering firewall. In *Artificial Intelligence, 2009. JCAI'09. International Joint Conference on*, pages 357–362. IEEE, 2009.

[235] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). *National Institute of Standards and Technology (NIST)*, (CSRC special publication SP 800-94), Feb 2007.

[236] L. Schaelicke, T. Slabach, B. Moore, and C. Freeland. Characterizing the Performance of Network Intrusion Detection Sensors. *Recent Advances in Intrusion Detection: 6th International Symposium, Raid 2003, Pittsburgh, Pa, Usa, September 8-10, 2003: Proceedings*, 2003.

[237] N. Schear, D. Albrecht, and N. Borisov. High-speed matching of vulnerability signatures. In *Recent Advances in Intrusion Detection*, pages 155–174. Springer, 2008.

[238] D.L. Schuff and V.S. Pai. Design alternatives for a high-performance self-securing ethernet network interface. In *IEEE International Parallel and Distributed Processing Symposium, IPDPS 2007*.

[239] M. Sedaghat, F. Hernandez-Rodriguez, and E. Elmroth. A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling. In *Proc. of ACM CAC '13*.

[240] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. In *Proc. of USENIX NSDI '12*.

[241] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture.

[242] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi. Design and implementation of a consolidated middlebox architecture. Technical Report UCB/EECS-2011-110, EECS Department, University of California, Berkeley, Oct 2011.

[243] et al. Sen, Siddhartha. Scalable, optimal flow routing in datacenters via local link balancing. In *Proc. of ACM CoNEXT '13*.

[244] Amazon Web Services. http://aws.amazon.com.

[245] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev, and Chanan Glezer. Google android: A comprehensive security assessment. *IEEE Security and Privacy*, 8(2):35–44, 2010.

[246] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh. A cost-aware elasticity provisioning system for the cloud. In *IEEE ICDCS '11*.

[247] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. Cloudscale: Elastic resource scaling for multi-tenant cloud systems. In *Proc. of ACM SoCC '11*.

[248] Z. Shen, Z. Zhang, A. Kochut, A. Karve, H. Chen, M. Kim, H. Lei, and N. Fuller. Vmar: Optimizing i/o performance and resource utilization in the cloud. In *Proc. of ACM/IFIP/USENIX Middleware '13*.

[249] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: network processing as a cloud service. *ACM SIGCOMM CCR. Vol. 42. No. 4. ACM, 2012.*

[250] David B Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274. ACM, 1997.

[251] V. Shrivastava, P. Zerfos, K. W. Lee, H. Jamjoom, Y. H. Liu, and S. Banerjee. Application-aware virtual machine migration in data centers. In *Proc. of IEEE INFOCOM '11*.

[252] S. Sinha, F. Jahanian, and J. Patel. Wind: Workload-aware intrusion detection. In *Recent Advances in Intrusion Detection*, pages 290–310. Springer, 2006.

[253] Lawrence V. Snyder. Facility location under uncertainty: a review. *IIE Transactions*, 38(7):547–564, June 2006.

[254] A. Sperotto and A. Pras. Flow-based intrusion detection. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2011.

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

[256] Chris Strasburg, Natalia Stakhanova, Samik Basu, and Johnny S Wong. Intrusion response cost assessment methodology. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 388–391. ACM, 2009.

[257] H. Takagi. *Queueing analysis*. North-Holland Amsterdam, 1991.

[258] Arie Tamir. An O (pn 2) algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.

[259] S. Tanachaiwiwat, K. Hwang, and Y. Chen. Adaptive Intrusion Response to Minimize Risk over Multiple Network Attacks. *ACM Trans on Information and System Security*, 19, August 2002.

[260] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[261] Lawrence Teo and Gail-Joon Ahn. Managing heterogeneous network environments using an extensible policy framework. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 362–364, New York, NY, USA, 2007. ACM.

[262] Lawrence Teo, Gail-Joon Ahn, and Yuliang Zheng. Dynamic and risk-aware network access management. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 217–230, New York, NY, USA, 2003. ACM.

[263] Ashley Thomas. Adaptive real time intrusion detection systems. 2003.

[264] JA Tomlin. Minimum-cost multicommodity network flows. *Operations Research*, 14(1):45–51, 1966.

[265] Alok Tongaonkar, Sreenaath Vasudevan, and R. Sekar. Fast packet classification for snort by native compilation of rules. In *Proceedings of the conference on Large installation system administration conference (LISA)*, pages 159–165, 2008.

[266] Joaquin E. Torres-Soto and Halit Üster. Dynamic-demand capacitated facility location problems with and without relocation. *International Journal of Production Research*, 49(13):3979–4005, July 2011.

[267] Udaya Tupakula, Vijay Varadharajan, and Naveen Akku. Intrusion detection techniques for infrastructure as a service cloud. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 744–751. IEEE, 2011.

[268] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Trans. Auton. Adapt. Syst.*, 3(1):1:1–1:39, 2008.

[269] A. Valdes and K. Skinner. Probabilistic Alert Correlation. *Recent Advances in Intrusion Detection: 4th International Symposium, RAID 2001, Davis, CA, USA, October 10-12, 2001: Proceedings*, 2001.

[270] J. van Lunteren and A. Guanella. Hardware-accelerated regular expression matching at multiple tens of gb/s. In *INFOCOM, 2012 Proceedings IEEE*, pages 1737–1745. IEEE, 2012.

[271] R Vanathi and S Gunasekaran. Comparison of network intrusion detection systems in cloud computing environment. In *Computer Communication and Informatics (ICCCI), 2012 International Conference on*, pages 1–6. IEEE, 2012.

[272] L. M. Vaquero, L. Rodero-Merino, and R. Buyya. Dynamically scaling applications in the cloud. *ACM SIGCOMM CCR. Vol. 41. No. 1. ACM, 2011.*

[273] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. Markatos, and S. Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In *Recent Advances in Intrusion Detection*, pages 116–134. Springer, 2008.

[274] A. Verma, P. Ahuja, and A. Neogi. pmapper: Power and migration cost aware application placement in virtualized systems. In *Proc. of ACM/IFIP/USENIX Middleware '08*.

[275] Vedat Verter. Foundations of Location Analysis. 155, 2011.

[276] M. Walfish, J. Stribling, M. N. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes no longer considered harmful. In *OSDI*, volume 4.

[277] Michael Walfish, Jeremy Stribling, Maxwell Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker. Middleboxes no longer considered harmful.

[278] Gang Wang, Jinxing Hao, Jian Ma, and Lihua Huang. A new approach to intrusion detection using artificial neural networks and fuzzy clustering. *Expert Systems with Applications*, 37(9):6225–6232, 2010.

[279] Guohui Wang, David G Andersen, Michael Kaminsky, Konstantina Papagiannaki, T S Eugene Ng, Michael Kozuch, and Michael Ryan. c-Through : Part-time Optics in Data Centers. *ACM SIGCOMM Comput. Commun. Rev.*, 40:327–338, 2010.

[280] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection. In *Recent Advances in Intrusion Detection*, pages 203–222. Springer, 2004.

[281] Limin Wang, KyoungSoo Park, Ruoming Pang, Vivek S Pai, and Larry L Peterson. Reliability and security in the codeen content distribution network. In *USENIX Annual Technical Conference, General Track*, pages 171–184, 2004.

[282] Meng Wang, Xiaoqiao Meng, and Li Zhang. Consolidating virtual machines with dynamic bandwidth demand in data centers. In *Proc. of IEEE INFOCOM '11*.

[283] Yang Wang, Chuang Lin, Quan-Lin Li, and Yuguang Fang. A queueing analysis for the denial of service (dos) attacks in computer networks. *Computer Networks*, 51(12):3564–3573, 2007.

[284] Huaqiang Wei, Deb Frinke, Olivia Carter, and Chris Ritter. Cost-benefit analysis for network intrusion detection systems. In *CSI 28th Annual Computer Security Conference*, pages 29–31, 2001.

[285] George O. Wesolowsky and William G. Truscott. The Multiperiod Location-Allocation Problem with Relocation of Facilities. *Management Science*, 22(1):57–65, September 1975.

[286] P. Winter, E. Hermann, and M. Zeilinger. Inductive intrusion detection in flow-based network data using one-class support vector machines. In *IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2011.

[287] C. Wu, J. Yin, Z. Cai, E. Zhu, and J. Chen. A hybrid parallel signature matching model for network security applications using simd GPU. pages 191–204. Springer, 2009.

[288] Di Wu, Yupeng Zeng, Jian He, Yi Liang, and Yonggang Wen. On p2p mechanisms for vm image distribution in cloud data centers: Modeling, analysis and improvement. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 50–57, Dec 2012.

[289] Haitao Wu, Guohan Lu, Dan Li, Chuanxiong Guo, and Yongguang Zhang. MDCube: a high performance network structure for modular data center interconnection. In *Proc. 5th Int. Conf. Emerg. Netw. Exp. Technol.*, pages 25–36, 2009.

[290] Jan-Jan Wu, Shu-Fan Shih, Pangfeng Liu, and Yi-Min Chung. Optimizing server placement in distributed systems in the presence of competition. *Journal of Parallel and Distributed Computing*, 71(1):62–76, January 2011.

[291] Linlin Wu, Saurabh Kumar Garg, and Rajkumar Buyya. Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID '11, pages 195–204, Washington, DC, USA, 2011. IEEE Computer Society.

[292] S. Wu, U. Manber, et al. A fast algorithm for multi-pattern searching. Technical report, Technical Report TR-94-17, University of Arizona, 1994.

[293] www.mathworks.com.

[294] Xiaokui Xiao, Bin Yao, and Feifei Li. Optimal location queries in road network databases. *2011 IEEE 27th International Conference on Data Engineering*, pages 804–815, April 2011.

[295] Li Yan and Marek Chrobak. LP-rounding Algorithms for the Fault-Tolerant Facility Location Problem. *Algorithms and Complexity*, pages 1–36, 2013.

[296] Shui Yu, Yonghong Tian, Song Guo, and D.O. Wu. Can we beat ddos attacks in clouds? *Parallel and Distributed Systems, IEEE Transactions on*, 25(9):2245–2254, Sept 2014.

[297] Saman Taghavi Zargar, Hassan Takabi, and James BD Joshi. Dcdidp: A distributed, collaborative, and data-driven intrusion detection and prevention framework for cloud computing environments. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, pages 332–341. IEEE, 2011.

[298] M. F. Zhani, Q. Zhang, G. Simon, and R. Boutaba. Vdc planner: Dynamic migration-aware virtual data center embedding for clouds. In *IFIP/IEEE IM '13*.

[299] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Reading MA (USA), 1949.