UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Javier Richard Quinto Ancieta

IntelliFlow: A Proactive Approach To Add Cyber Threat Intelligence To Software Defined Networking

IntelliFlow: Um Enfoque Proativo para Adicionar Inteligência de Ameaças Cibernéticas a Redes Definidas por Software

CAMPINAS

2015

Javier Richard Quinto Ancieta

IntelliFlow: A Proactive Approach To Add Cyber Threat Intelligence To Software Defined Networking

IntelliFlow: Um Enfoque Proativo para Adicionar Inteligência de Ameaças Cibernéticas a Redes Definidas por Software

Dissertation presented to the Faculty of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Eletrica, na Àrea de Engenharia de Computação.

Supervisor: Prof. Dr. Christian Rodolfo Esteve Rothenberg

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Javier Richard Quinto Ancieta, e orientada pelo Prof. Dr. Christian Rodolfo Esteve Rothenberg

CAMPINAS

2015

Agência(s) de fomento e nº(s) de processo(s): CNPq, 159905/2013-3

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

Q46i

Quinto Ancieta, Javier Richard, 1984-
IntelliFlow : A proactive approach to add cyber threat intelligence to software defined networking / Javier Richard Quinto Ancieta. – Campinas, SP : [s.n.], 2015.

Orientador: Christian Rodolfo Esteve Rothenberg.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes de computadores. 2. Agentes inteligentes (Software). 3. Informática - Medidas de seguranca. I. Esteve Rothenberg, Christian Rodolfo,1982-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: IntelliFlow : Um enfoque proativo para adicionar inteligência de ameaças cibernéticas a redes definidas por software
Palavras-chave em inglês:
Computer networks
Intelligent agents (software)
Computer - Safety Measures
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Christian Rodolfo Esteve Rothenberg [Orientador]
Paulo Licio De Geus
Marco Aurélio Amaral Henriques
Data de defesa: 22-09-2015
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Javier Richard Quinto Ancieta RA: 151566
Data da Defesa: 22 de setembro de 2015
Título da Tese:
"IntelliFlow: A Proactive Approach To Add Cyber Threat Intelligence To Software Defined Networking"
"IntelliFlow: Um Enfoque Proativo para Adicionar Inteligência de Ameaças Cibernéticas a Redes Definidas por Software"

Prof. Dr. Christian Esteve Rothenberg (Presidente, FEEC/UNICAMP)
Prof. Dr. Paulo Licio De Geus (IC/UNICAMP)
Prof. Dr. Marco Aurélio Amaral Henriques (FEEC/UNICAMP)

Ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

I would like to dedicate my whole work to Alexandra, my beautiful daughter, the person that makes me happy everyday of this wonderful life.

Acknowledgements

My first and special thanks to my parents Mr. Gregorio and Mrs. Fortunata, for all their support and constant advice, for trusting me forever, for giving me those words of courage when I needed it. I am also deeply grateful to my whole family for all their moral and financial support, always motivating me to go further.

Thank you so much Christian Rothenberg, my advisor, for all your constant effort and be a great instructor for the development of my dissertation.

Thank you all in the INTRIG research group, for the support, company, friendship and good moments together.

Thank you, Professors Marco Aurelio and Paulo Licio for all the valuable suggestions and reviews of the dissertation.

Finally, thanks to CNPq for the financial support along this journey.

"The only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it."

Steve Jobs

Abstract

Security is a major concern in computer networking which faces increasing threats as the commercial Internet and related economies continue to grow. Virtualization technologies enabling scalable Cloud services pose further challenges to the security of computer infrastructures, demanding novel mechanisms combining the best-of-breed to counter certain types of attacks. Our work aims to explore advances in Cyber Threat Intelligence (CTI) in the context of Software Defined Networking (SDN) architectures. While CTI represents a recent approach to combat threats based on reliable sources, by sharing information and knowledge about computer criminal activities, SDN is a recent trend in architecting computer networks based on modularization and programmability principles. In this dissertation, we propose IntelliFlow, an intelligent detection system for SDN that follows a proactive approach using OpenFlow to deploy countermeasures to the threats learned through a distributed intelligent plane. We show through a proof of concept implementation that the proposed system is capable of delivering a number of benefits in terms of effectiveness and efficiency, altogether contributing to the security of modern computer network designs.

Palavras-chaves: Computer Networks; Software Defined Networking; Cyber Threat Intelligence; Intrusion Detection System; OpenFlow.

Resumo

Segurança tem sido uma das principais preocupações enfrentadas pela computação em rede principalmente, com o aumento das ameaças à medida que a Internet comercial e economias afins crescem rapidamente. Tecnologias de virtualização que permitem serviços em nuvem em escala colocam novos desafios para a segurança das infraestruturas computacionais, exigindo novos mecanismos que combinem o `best-of-breed` para reagir contra as metodologias de ataque emergentes. Nosso trabalho busca explorar os avanços na Cyber Threat Intelligence (CTI) no contexto da arquitetura de redes definidas por software, ou em inglês, Software Defined Networking (SDN). Enquanto a CTI representa uma abordagem recente para o combate de ameaças baseada em fontes confiáveis, a partir do compartihamento de informação e conhecimento sobre atividades criminais virtuais, a SDN é uma tendência recente na arquitetura de redes computacionais baseada em princípios de modulação e programabilidade. Nesta dissertação, nós propomos IntelliFlow, um sistema de detecção de inteligência para SDN que segue a abordagem proativa usando OpenFlow para efetivar contramedidas para as ameaças aprendidas a partir de um plano de inteligência distribuida. Nós mostramos a partir de uma implementação de prova de conceito que o sistema proposto é capaz de trazer uma série de benefícios em termos de efetividade e eficiência, contribuindo no plano geral para a segurança de projetos de computação de rede modernos.

Keywords: Redes de Computadores, Redes Definidas por Software, Inteligência de Ameaça Cibernética, Sistema de Detecção de Intrusos, OpenFlow.

List of Figures

List of Tables

Acronyms

APIs Application Programming Interfaces. 20

CIA Confidentiality, Integrity, and Availability. 36

CIF Collective Intelligence Framework. 20, 44

CLI Command Line Interface. 36

CTI Cyber Threat Intelligence. 18, 20, 21, 29

DD4BC Distributed DoS for Bitcoin. 16

DDoS Distributed Denial of Service. 21, 40

DoS Denial of Service. 16

DPI Deep Packet Inspection. 18, 40

FINE Format for Incident Information Exchange. 33

GPB Google Protocol Buffer. 57

GPG GNU Privacy Guard. 33

IAP Intrusion Alert Protocol. 33

IDPS Intrusion Prevention and Detection System. 17, 18, 21, 22

IDS Intrusion Detection System. 21

IDSs Intrusion Detection Systems. 17

IDXP Intrusion Detection Exchange Protocol. 33

IF Intelligence framework. 45

IOC Indicator of compromise. 32, 50

IOCs Indicators of compromise. 58

IPS Intrusion Prevention System. 21

IPSs Intrusion Prevention Systems. 17

KP Knowledge Plane. 45

LISP Location Identifier Separation Protocol. 33

LXCs Linux Containers. 41

MDBLs Malicious Domain Blacklists. 69

NBA Network Behavior Analysis. 25

OS Operating System. 41

OSSIM Open Source Security Event and Information Management. 35

OVS Open vSwitch. 40

SDN Software-Defined Networking. 19, 36, 44

SMTP Simple Mail Transfer Protocol. 33

TTP Tactics, Techniques and Procedures. 32

VNs Virtual Networks. 55

XSS Cross-Site Scripting. 16

Contents

1 Introduction

The deployment of Internet services has increased immensely during the last years, as well as the access of the users to these services, especially in cloud computing environments. Consequences of these changes are that many adversaries designed new forms of attacks with malicious intents, having as their main goals potential economic gains. Therefore, these threats represent an attractive, new model of business for cyber criminals.

Figure 1 shows the more common data security concerns that cloud users face daily, in where the five highest concerns are data-related with exception of Compliance and legal issues. According to Cloud Security Alliance (2013), some of the top threats are: (i) Data Breach, (ii) Data Loss, (iii) Account Traffic Hijacking, and (iv) Denial of Service (DoS). For example, Snapchat (SNAPCHAT, 2015), the application to share data that are self-destructed in time, suffered a `data breach` of approximately 200 000 user private photos because of a vulnerability in the third-party client application called SnapSaved[1], so that all stolen information was published on Internet forums such as 4chan[2]. Another similar case of data breach was targeted to Play Station network, whose damage was estimated to $4.6 billion (LIBERTY GLOBAL, 2012, p. 27). On the other hand, recently Google announced `data loss` after lightning struck near one of Google's data centers affecting permanently some of their servers (GOOGLE..., 2015), although this only affected a small portion of users, we must consider making backup of our data as well. Regarding the `account hijacking`, we have the cases of the Cross-Site Scripting (XSS) bug found in Amazon that allowed attackers to hijack credentials from the website in 2010, and the leak of 6.5 millions of LinkedIn passwords in 2012 (DARK..., 2014). And with regard to `DoS attacks`, businesses are at an high risk of being extorted by large criminal groups that by using of network flood attacks they extort their victims forcing to pay large amount of money (FBI..., 2015), e.g., the Distributed DoS for Bitcoin (DD4BC) criminal group has been rapidly increasing both the frequency and the scope of its DoS extortion attempts, having as target online casinos, banks, and trading platforms (ARBOR SERT, 2015). These are just some of many threats whereby most organizations face daily.

New container-based technologies with high degrees of isolation and efficiency have emerged and allow to combat some of these security problems. Those new schemes can avoid certain problems around the isolation of the virtual machines, but are helpless in face of other significant threats such as DoS or password guessing attacks. The main difference between containers and traditional virtualization is that the former runs in a single process namespace, sharing the same kernel of the main operating system (SOLTESZ *et al.*, 2007). The best known

[1] First report of the data leak to Snapchat. Source: "http://www.businessinsider.com/snapchat-hacked-the-snappening-2014-10/"

[2] An image-based bulletin board where anyone can post comments and share any kind of on-line images as well as post comments. Source: "http://www.4chan.org/"

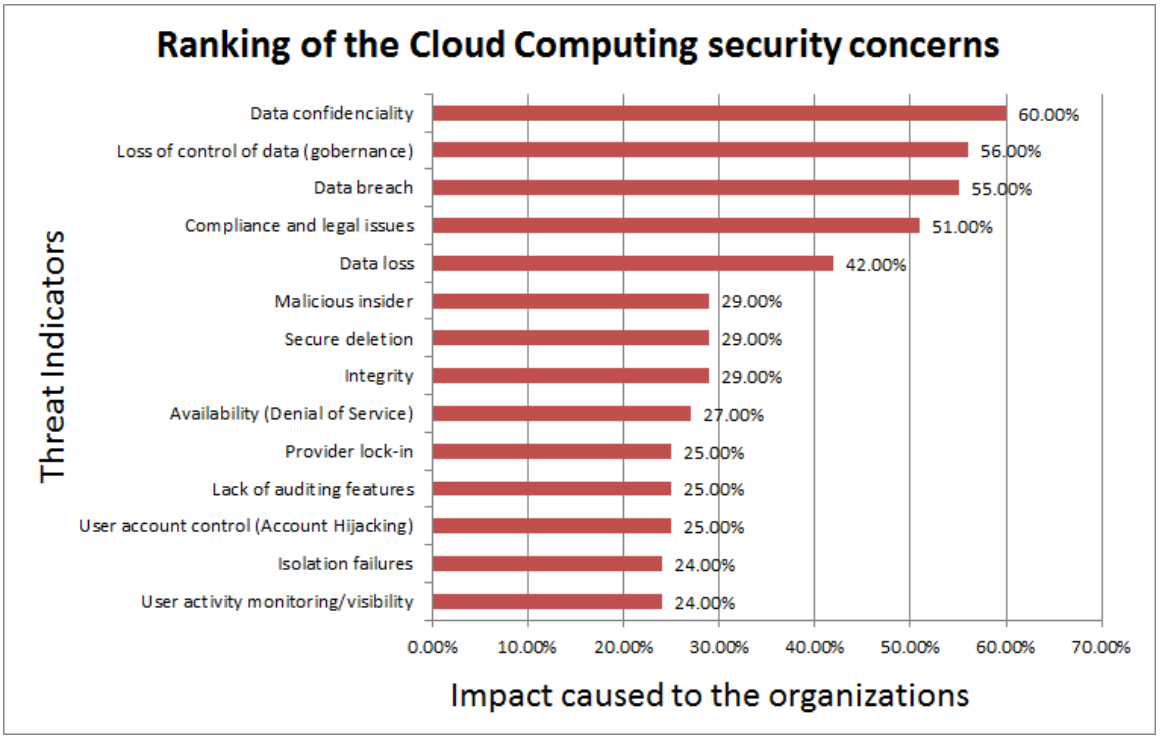


Figure 1 – Cloud computing top threats. Adapted from data available in (CLOUD SECURITY ALLIANCE, 2015, p. 10)

technologies based on containers include CoreOS, OpenVZ, Docker, among others. Docker has become the most popular one and delivers containers to isolate applications with high network throughput when compared to KVM (HUSSAIN, 2014). Figure 2 shows the network performance comparison between a native machine, docker container, and KVM measured using the *iperf* tool.

Despite the improvements in virtualization technologies, new attack techniques continue to appear in virtual environments, so that security devices such as firewall are no longer sufficient, due to new methodologies of the attackers, which have been improved greatly, e.g., the actual threats are mostly directed to the network application layer, being often caused by malicious insiders (INFORMATION SYSTEMS SECURITY, 2007).

Intrusion Detection Systems (IDSs) are security devices whose function is to monitor, analyze and detect anomalous traffic directed to the application and Internet layers. However, they are not able to prevent attacks, but only to alert the existence of them. Intrusion Prevention Systems (IPSs) together with IDSs filter malicious packets in a proactive or reactive way. Both working together are known as Intrusion Prevention and Detection System (IDPS) because of sometimes IPSs only function as IDSs.

For example, Snort (SNORT, 2015) is a typical IDS able to be configured as an IPS to prevent attacks using the method of Deep Packet Inspection (DPI). However, Snort has certain performance limitations in high speed networks such as dropping packets or slowing down the traffic (MEHRA, 2012).

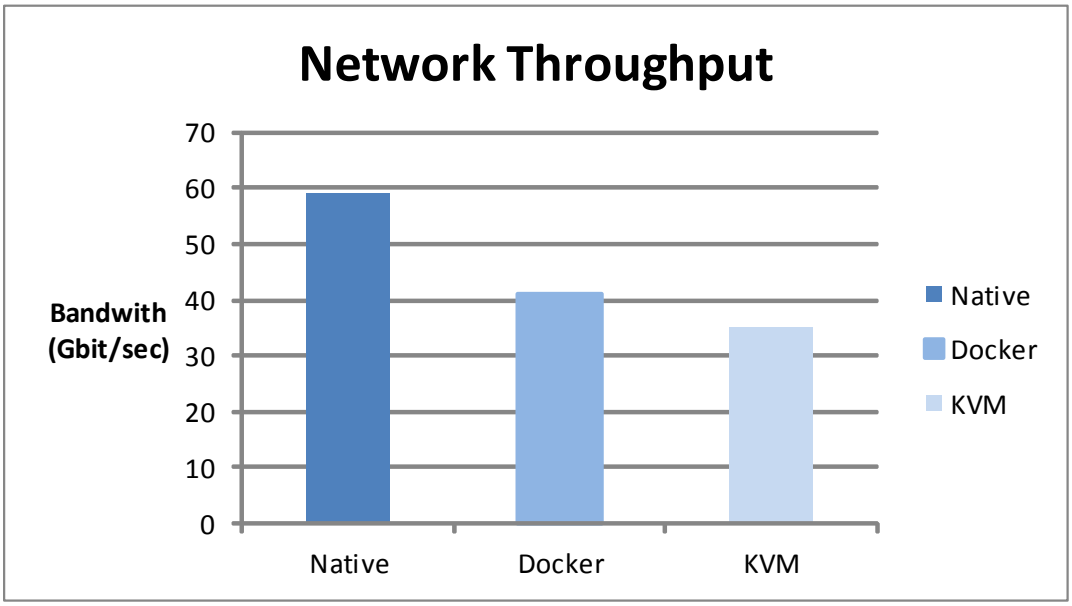


Figure 2 – KVM, Docker, and Native performance comparison. Adapted from data available in (HUSSAIN, 2014)

Organizations, that deploy security devices on their networks, only detect and block attacks which have been seen before, focusing on the event itself. However, they does not perform an analysis in depth of the methodology used by the attackers. These organizations do not known that these attacks have a complex behavior and sometimes variable, in where the attackers conduct a wide range of activities starting with the active and passive reconnaissance, followed by the scanning, initial probes from zombie hosts, send small packets at a higher speed, and when the threats are detected these malicious users simply change the malware used (JOHNSON *et al.*, 2014).

If organizations learn about the behavior of the attackers they could discover the methodology used in the attack, thus the countermeasures would be more effective using the knowing learned in the past by these same organizations. For this reason, the Cyber Threat Intelligence (CTI) methodology emerged as an intelligent way to take advantage of the previously analyzed information by trusted organizations. These data, known as `intelligence`, become an important asset for organizations who want to proactively protect their services and networks against new threats (ISIGHTPARTNERS, 2014, p. 3).

With the goal of improving the IDS analytic skills and to use the `intelligence` provided by CTI, researchers developed Bro IDS (PAXSON, 1999), a powerful data analysis framework that inspects all network traffic in depth in order to find suspicious activities. Bro has the ability to run in high speed to capture data with a higher Gbps; Bro has also more sophisticated signatures, a flexible scripts's policy and, specially, an intelligent framework to

receive data from other security sources.

Recently, Software-Defined Networking (SDN) (KREUTZ *et al.*, 2015) has emerged as a new paradigm in networking, providing a clean and programmatic separation of the control and data planes. The OpenFlow protocol (FOUNDATION, 2014a) is the best known programming interface to allow such separation of planes, enabling direct data plane device programmability, not available before in a standardized, vendor-independent way.

In SDN, network switches become simple forwarding devices and the control logic is implemented in a logically centralized controller. Nevertheless SDN security needs to be built into the architecture to protect the availability, integrity, and privacy of all connected resources and information, as well as to be delivered as a service (SCOTT-HAYWARD *et al.*, 2013).

In the last years, many research projects related to SDN emerged as a result of the flexibility in the SDN architecture, building applications in order to execute instructions to the data plane and to reconfigure the flow tables of the switches that were affected.

1.1 Vision: Towards more secure and collaborative SDN

The methodologies to detect attacks using conventional IDS rules may help to maintain an organization protected once, but not necessarily will show good results. This is due to the fact that the threats change over time. For example, a malware located at the blacklists of security organizations, may change slightly its behavior in order to bypass the IDSs. In this case, the traditional sensors will not be able to properly detect that attack. However, sharing threat information between trusted organizations may help to resolve the weaknesses in the IDSs. Therefore, changing the security model from reactive to proactive with CTI would allow the organizations to understand the behavior of the attackers and develop countermeasures to filter those attacks before they are executed.

Relying only in the blacklists provided by security organizations may be a problem. This is because the shared threat information must be relevant, actionable[3] and valuable for the organization; otherwise it would not be useful (ISIGHTPARTNERS, 2014). The advantages of using the CTI is not only to filter many threats, but also to know who are the adversaries, how they operate, and what are the next steps to protect the networks against those attacks. If the organizations take advantage of the SDN network abstraction layer and the cyber threat intelligence framework, they could program Application Programming Interfaces (APIs) that allow them to generate countermeasures against attacks such as DDoS by interacting with the underlying data plane devices. Our vision is that combining the CTI with SDN would allow the protection of data in new cost-effective ways compared to traditional IDS-based solutions and more efficient detection of attacks using the intelligence provided by reliable organizations.

[3] Data that must be specific enough to trigger some type of response

1.2 Research Objectives and Contributions

In this work, we propose `IntelliFlow`, an intelligent system of intrusion detection and prevention, that acts against different types of known threats in a proactive or reactive mode. IntelliFlow aims at taking advantage of Cyber Threat Intelligence (CTI) systems combining known malicious information and IDS technology to create new security rules that allow blocking malicious traffic by programming OpenFlow-based switches of the SDN architecture. To this end, the objectives of the proposed IntelliFlow architecture include:

- Leverage Collective Intelligence Framework (CIF) to add security service to SDN.
- Integrate the Bro's Intelligence framework to acquire intelligence from reliable sources.
- Evaluate the IntelliFlow architecture for different scenarios, validating it with a proof-of-concept implementation and experiments to assess effectiveness and performance.

1.3 Text Structure

This dissertation is organized as follows: Chapter 2 introduces the literature review and the related works. Chapter 3 gives the overview of the whole IntelliFlow architecture and the intelligence framework. Chapter 4 describes the proof of concept and the experimental evaluations, and Chapter 5 present the conclusions of the dissertation.

2 Literature Review

Initially, security devices were mainly intended to protect the IP layer against a set of well-known attacks. Over time, these devices were not able to combat more advanced threats directed to the transport and application layers. New detection techniques and devices were built to counter these types of attacks, e.g., Intrusion Detection System (IDS). These security devices often work together with firewalls to block a lot of threats found on the Internet, acting as Intrusion Prevention and Detection System (IDPS).

Recently, more sophisticated ways to combat several threats have emerged, e.g., intelligence frameworks, stateful protocol analysis, statistical-based detection, and so on. In the following sections, we describe the Cyber Threat Intelligence (CTI), and how these security intelligence data can be used to filter many attacks. Finally, we discuss various advantages related to SDN, and a brief description of the mode of operation of SDN applications.

This section covers the main relevant concepts behind our research proposal. The first part deals with network monitoring by using Intruder Detection System such as Bro-IDS. The second discusses the use of Cyber Threat Intelligence to share valuable information between the organizations. Then, finally, we talk about the concepts related to software defined network as well as the different types of SDN applications.

2.1 Intruder Detection System

Intrusion Detection System (IDS) is a network device that allows to monitor, analyze and detect possible attacks on the network infrastructure in terms of confidentiality, integrity and availability (LIAO *et al.*, 2013). These devices can be implemented in hardware or software. However, IDS only reports suspected events, without guaranteeing an immediate response to a specific attack.

Intrusion Prevention System (IPS) not only automates the intrusion detection process, but also filters malicious packets to block the source of the attack (SANS, 2004). Thus, an IPS extends the IDS capabilities, so both work as a single system known as Intrusion Prevention and Detection System (IDPS). Besides detecting threats, IDPS also fix with a vulnerability, however they are unable to fix Distributed Denial of Service (DDoS) attacks. Those may only be tackled with a complex set of measures involving the end-user provider's set of routers. These DDoS attacks are attempts to make an online service unavailable to its intended users.

2.1.1 Intrusion Detection Methodologies

According to (LIAO *et al.*, 2013), the Intrusion Prevention and Detection System (IDPS) methodologies can be classified into three different types: anomaly-based detection, signature-based detection and stateful protocol analysis. Most of these IDPS devices combine these methodologies, in order to provide a more accurate analysis.

Anomaly-based Detection (AD)

This type of detection identifies events which do not agree with an expected pattern or that are unusual, e.g., DoS or brute-force attacks. It usually works through the comparison of profiles that represent a normal behavior against observed events, in order to identify some type of anomaly. This normal behavior refers to an initial profile performed through an analysis of the applications, network connections, computers or system users. By monitoring of the network in short periods of time, profiles of some applications are developed to give evidences of any unusual activity occurred in a specific time, e.g., tries of access to a malicious portal from the internal network during a certain time of day. To counter anomalous threats, administrators often generate statistical reports comparing the actual activity with the threshold of a normal profile.

Anomalous detection works with two types of profiles: static and dynamic. The static profiles are not usually changed, unless they are manually modified to generate news profiles; therefore, sometimes they may become inaccurate, so they'll need to be updated eventually. Unlike the static ones, the dynamic profiles are updated constantly as required; however, they are exposed to attacks from malicious users. For example, an attacker may perform a passive scanning on different time points, slowly increasing the frequency of its attack. As a result, the IDPS relies on this activity and it could be included on its profile. Another disadvantage is the generation of many false positives because of a complex network environment where the profiles were created incorrectly.

Signature-based Detection (SD)

This other type of detection compares each network's event with patterns or strings located at a local database of threats, thus avoiding attacks from known sources, e.g., a malicious e-mail with an attachment file called `money.exe` and a subject "Get easy Money!" may be tentative for users downloading and executing this virus; another example consists in receiving brute-force attacks from scanner networks to a sever's port 22. However, this type of detection is ineffective at detecting unknown threats if the methodology is not properly used. For example, if a malicious file was recognized as `explore.exe` by many signatures, an attacker may change the name of the file to `explore1.exe`, causing the signature to search incorrectly for the file, since it would be looking for its former name and not the current one. On the other hand, signatures also lack the ability to understand why some application service stopped working. This

is also known as knowledge-based detection. However, despite the disadvantages mentioned above, the signature-based detection is greatly improved when used together with cyber threat intelligence.

Stateful Protocol Analysis (SP)

This methodology compares profiles previously created for each protocol with observed events, identifying some types of unusual activity on networks. It also natively decodes any application-layer protocols, tracking the state of the whole network, e.g., pairing requests with replies.

Unlike anomaly-based detection, it depends on vendor-developed global profiles to specify protocols being used; For instance, when an user tries to access a TELNET service, the session initially provides helping information or only asks for username and password; then, depending on the code type that appears after entering the credentials, the IDPS may know if the access was successful or not. Therefore, in the case of access refused repeatedly, it would be considered suspicious.

Table 1 makes a comparison based on advantages and disadvantages of each intrusion detection methodology mentioned above.

In (SCARFONE; MELL, 2007), the authors present an advancing guide on IDS, evaluate the different capabilities of the intrusion detector and provide recommendations for designing, implementing, configuring, securing, monitoring, and maintaining of the IDSs.

2.1.2 Intrusion Detection Types

The IDSs are divided into different types of technologies, depending on the event they are monitoring and on how they are deployed. In the next sections, we go through the four most well known IDS types:

Network IDS (NIDS)

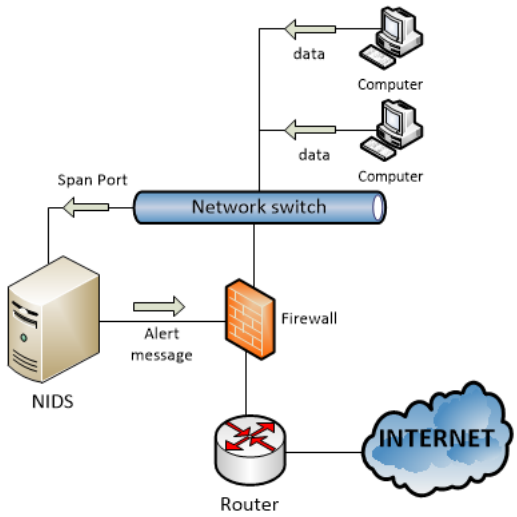
The NIDS is a network security system that monitors and analyzes network traffic in order to identify suspicious activities. This security device captures packets from a mirror interface using the `Libpcap` library and examine packets in real time. Figure 3a shows how NIDS works together with the firewall to block attack attempts from the Internet.

Host IDS (HDIS)

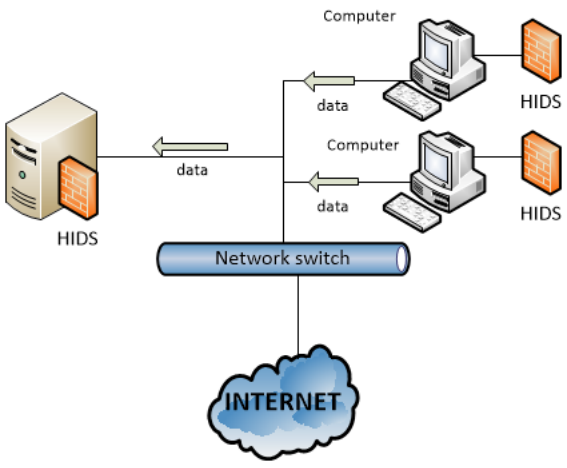
Unlike NIDS, HIDS only monitors individual hosts. It is commonly deployed on critical hosts, e.g., public servers containing sensitive information. Figure 3b shows how IDS works on particular hosts inside of the internal network.

Detection methodologies	Advantages	Disadvantages
Signature-based (Knowledge)	- Detect known attacks by using patterns or strings. - Simple method of detection.	- New attacks are not detected in time. - Little understanding of the network or applications used.
Anomaly-based (Behavior)	- Identify events which do not agree to some expected pattern. - Detecting previously unknown threats.	- New rules are difficult to be created. - False positives.
Stateful Protocol Analysis (Specification-based)	- Identify and distinguish unexpected sequences of commands. - Keep tracking of the authenticator used of each session and record it for suspicious activity.	- Complex analysis and an overhead involved in performing a continuous state tracking for many simultaneous sessions. - Incompatible of versions against specific applications and operating systems.

Table 1 – Intrusion Detection Methodologies

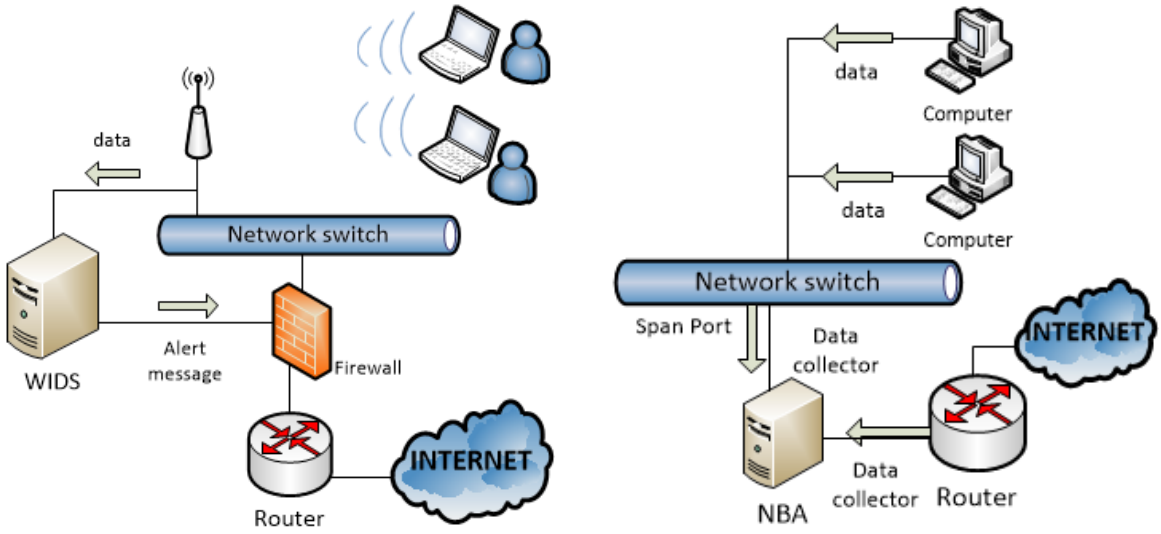


(a) Network Based IDS



(b) Host Based IDS

Figure 3 – Network and Local IDPSs Types



(a) Wireless IDS (b) Network behavior analysis

Figure 4 – Wireless and NBA IDPSs Types

Wireless IDS (WIDS)

WIDS monitors, analyzes and identifies certain malicious traffic on the wireless network by capturing anomalous data coming from Access Points. However, they can not identify malicious events on the application layer or protocols such as TCP or UDP, e.g., Packet injection Aireplay-ng. Figure 4a shows a scenario of intrusion detection system with two mobile devices connected to the wireless network. We note that the IDS device receives data from the access point and then sends an alert message in the case of any anomalous threat being detected.

Network Behavior Analysis (NBA)

This system inspects monitored traffic in order to identify threats generating anomalous traffic flows, such as DDoS, malwares, policy violations and so on. Its scope is on the organizational's internal networks and sometimes also on the external networks. In Figure 4b, we note that Network Behavior Analysis (NBA) provides capabilities to monitor the whole network by analyzing data from switches or routers.

2.1.3 Detection Approaches

As the detection methodologies are becoming more complex, they are divided in three subcategories including: computational approach, artificial intelligence and biological concepts. However, the analysis of these approaches is of great complexity. Therefore, (LIAO *et al.*, 2013) presents a classification of five subclasses with an in-depth perspective on their characteristics.

Statistical-based Detection

This approach works by analyzing data traffic in real time and processing the information with machine learning algorithms, in order to look for anomalies in established traffic patterns. For instance, each event of the network has a particular anomaly's level. If in a specific moment, a certain anomaly is higher than the allowable threshold, then the IDS device would generate an alert (FARSHCHI, 2010).

One of the advantages of this detection approach is the possibility to detect `zero-day` attacks by sending alerts when there is an unusual activity in the network. Another noticeable advantage is the detection of passive attacks, such as porn scanner attacks. Additionally, there is no need to update signatures. However, the approach completely relies in learning patterns generated by its own algorithms.

Pattern-based Detection

This approach monitors packets on the network and compares them with a database of known attack patterns of threats. It consists of five modules: capture module, decode module, detection module, known attack pattern module and action module (KSHIRSAGAR *et al.*, 2011). The first captures raw data of the internal network, the second decodes that data, the third detects attacks from known sources, the fourth is a database in which the information on found attacks created by the capture module is stored. Finally the fifth performs appropriate actions to filter these attacks.

Rule-based Detection

This approach uses the anomaly and signature-based methodologies to decide if a given behavior is considered an intrusion on host or network scenarios. The detection approach detect unknown or known threats with a high performance of detection. The approach works through the observation of events in the system, applying rules leading to a decision with regards to a given pattern of activity suspicious (YANG *et al.*, 2013).

State-based Detection

With the state-based detection, anomaly scores are computed to define the current state of an associated event to the network. This approach only uses the anomaly detection to determine an intrusion on host or network environments with a high detection performance. The original idea behind state-based anomaly detection was that the detector would return a large anomaly value when it found a missing transition, and return zero otherwise. But this is not necessarily the best way to present the results of the detection, since users may want to adjust the detector's sensitivity. The detector should be able to return a number between zero and one that somehow measures its confidence whether an intrusion is happening (CARCANO *et al.*, 2010).

Heuristic-based Detection

This type of heuristic detection is also known as anomaly-based. It consists in building a model of acceptable behavior with certain exceptions. Its approach is used for efficient detection of intrusion. For instance, when an administrator designs a determined behavior as acceptable, the heuristic IDS will will also consider that behavior acceptable. Heuristics work similarly to a common IDS. The IDS learns over time what types of traffic patterns are considered normal for your network. The heuristics feature then watches for anomalies in the traffic pattern (SHAH, 2001).

In Table 2, we have a comparison between each detection approach according to its methodology, detection type and detection of attack.

Approach	**Detection meth.**			**Detection type**	**Detection of attack**
	AD	SD	SP		
Statistic-based	✓	✓	-	HIDS and NIDS	Known and Unknown
Pattern-based	-	✓	-	HIDS and NIDS	Known
Rule-based	-	✓	-	NIDS and WIDS	Known
State-based	✓	✓	✓	HIDS	Known and Unknown
Heuristic-based	✓	-	-	NIDS	Unknown

Table 2 – Intrusion Detection Approaches

2.1.4 Open Source IDPS tools

Throughout the years, attacks have been improved with the use of more sophisticated strategies, becoming more dangerous. Therefore, many efforts have been made by companies in order to improve threat combat technology, by developing security devices with evolving strategies to block attacks. Currently, most companies use commercial hardware as security devices, e.g., Cisco (CISCO. . . , 2015) offers an anomaly detection module against DDoS attacks, IBM (IBM. . . , 2015) monitors and detects intrusions by extra sensor modules for the prevention of attacks and blocking of packets.

On the other hand, the most famous open source IDSs are: snort (SNORT, 2015), suricata (SURICATA, 2015) and bro (PAXSON, 1999). We will explain each of these open source security devices in detail.

Snort

Snort is an open and free intrusion detection system capable of performing real-time analysis and suspicious events logging. The packets are analyzed using signature databases or anomalous detection methodologies.

This sensor works in three modes: sniffer, packet logger, and network intrusion detection. The sniffer mode reads all allowed network packets to identify troubleshooting prob-

lems, then displays them on a console. The packet logger mode stores analyzed packets into a local directory. The last mode monitors the network traffic in real time, analyzing it regarding rules previously defined by the user. Therefore, Snort will perform actions based on what anomalous events it has detected before.

The architecture of Snort is based on four main components: Packet decoder, Preprocessors, Detection engine, Alert generation. The packet decoder module filters packets from any network for preprocessing by using AF_Packet module, and then sends the preprocessed packets to the detection engine that, by using external rule set, generates alerts that are exported to the Alert database module. Recently, Snort replaced its packet capture framework from Libpcap to AF_Packet, thus the performance was increased to 500MBit/sec (KHALIL, 2015). One advantage is that Snort is supported by hardware platforms and operating systems. However, Snort does not support load balancing across multiple CPUs, and it drops packets exceeding the maximum capacity of a single CPU.

Suricata

Suricata is another open source, high performance intrusion detection system. This technology also works as IPS and network security monitoring engine, similar to Snort. Unlike Snort, Suricata supports both AF_Packet and PF_Ring for high-performance packet capture and it also uses the standard capture PCAP. One advantage is that it is highly scalable, supporting multi threated. Therefore, Suricata can run one instance and it will balance the load of processing across all processors used. However, similar to Snort, Suricata only works with signature-based rules.

Bro

Bro is a powerful network analysis framework with more advanced features than other intrusion detection systems. For instance, it offers a policy script interpreter layer that executes a set of event handlers written in Bro's own language; This permits taking actions when attacks are found by the event engine, applying different policies such as dropping or redirecting malicious packets. However, Bro still isn't widely used by companies, because the learning curve for implementing Bro can be steep for most users. The new version Bro 2.3 uses PF_Ring for the line rate packet processing of 1 Gbps of transmission to 10Gbps of reception. Bro can be configured for both NIDS or HIDS, inspecting all packets with its event engine. One of its main advantages is that Bro supports intelligence from external sources using its intelligence framework. The main goal of the framework is consuming data processed by security's organizations. The intelligence data can be loaded by an input framework that allows reading text files with a defined format.

In Table 3 we make a comparison between these three most well known IDS. However, we chose Bro as our IDS device for testing, due to its flexibility in the description of its

policies and supporting data intelligence.

Description	**Snort**	**Suricata**	**Bro**
Multi-thread	✓	✓	
Native IPS	✓		
Own Language			✓
Operating system compatibility	✓	✓	
High-speed monitoring			✓
Large user community			✓
Intelligence			✓

Table 3 – Comparison between IDPS solutions

2.2 Cyber Threat Intelligence

There are various concepts related to Cyber Threat Intelligence (CTI) because of the emerging space in which many organizations work to offer reliable data processing. According to Gartner (MCMILLAN, 2013), CTI is a recent methodology of evidence-based knowledge that organizations employ to identify and successfully respond to a cyber attack. It includes context, mechanisms, indicators, implications and actionable advice on emerging threats to assets that can be used to inform decisions related to the subject's response to the threat. In addition, the FBI (FEDERAL. . . , 2015) considers that intelligence is the information that has been analyzed and refined, so that it is useful to decision making on potential threats to the information. Waltz (WALTZ, 1998) defines the intelligence as the information and knowledge about an adversary, obtained through observation, investigation, analysis, or understanding the same.

However, the security informations published on the Internet often are prone to false positives because they are equating CTI with raw data information. For instance: a bad IP address coming from an unreliable blacklist could result in an indicator wrongly employed by organizations. Therefore, we believe `intelligence` is not simply any raw data: it is information that has been analyzed and must be actionable; otherwise the information would no be useful for organizations. In Table 4, we compare intelligence and information.

Intelligence	**Information**
- Processed, sorted data.	- Any raw and unfiltered data.
- Carefully analyzed by security analyst.	- Analyzed without a clear understanding of security.
- Data are found from reliable sources.	- Data are found from any unreliable source.
- Accurate, relevant, actionable and valuable.	- Data doubtfully reliable, incomplete or irrelevant.

Table 4 – Information versus Intelligence

The techniques of cyber attackers have evolved during the last years, by using new resources and developing new attack methods (FARNHAM; LEUNE, 2013). Unfortunately, many intrusions remain undetected and, even in the case of them being detected, they will often remain unpublished. This facilitates for malicious users to continue performing the same attack patterns without disruption.

On the other hand, some organizations offer different types of intelligence, or even most of them do not offer the required intelligence value; instead, they only provide some types of raw information. Thus, a bad analysis of this information would generate any data but not necessarily intelligence. Consequently, many false positives emerge caused by this misunderstanding of the differences between raw information and intelligence.

To produce intelligence, a certain correlation is performed on the gathered data, identifying malicious users and determining a confidence rating for each one of them; For instance, a high confidence level, nearly 95% for a given IP address guarantees that such value comes from a malicious user, and a low level confidence, close to 45%, can not assure a value actually comes from a malign source.

Because of the rapid spread of threats, the value of CTI defined should be immediately used to be analyzed at the time that the source of the attack was found. If CTI is not used in time, its value can converge to zero within of few hours; For example: When an institution identifies some type of attack and successfully respond using the correct mitigation procedure, it acquires an useful value, and this information can be shared within a set of trusted organizations. Thus, another institution facing a similar problem would be able to rapidly deploy countermeasures based on the experience acquired by another, thus intelligently preventing attacks (JOHNSON *et al.*, 2014).

2.2.1 Threat Intelligence Features

As of today, most organizations still rely on indicators found on the Internet. Though this kind of data may be useful, it often does not represent a significant information value for the organization. So a few years ago, the concept of `intelligence` emerged as a new security space in which organizations could use data shared by other trusted organizations. Accordingly, intelligence has seen a significant growth of its relevance in the last years. However, many definitions recently appeared to define cyber threat intelligence, creating misunderstandings between network administrators, whereby they often use raw data instead of intelligence. For instance, security information provided by social network could be considered useful for some, but not for others, though if the information has a considerable value for the organization, it is named as intelligence. We base our definitions of intelligence through their five main features:

Timely

The intelligence value is very high when it is used quickly. If it is used after some days or even hours, its value could assume zero rapidly. Thus, if not employed in a reasonable period of time, intelligence may become obsolete. On the other hand, if intelligence is used within a acceptable time-span, it provides enough opportunity for an organization to anticipate threats or prepare for an effective response.

Relevant

The intelligence should be applicable or useful for the organization. If it is not relevant, it wont be considered intelligence, despite any other factors.

Accurate

The intelligence as value should be correct, complete to avoid false positives, false negatives or irrelevant data. If these features are not fulfilled, the result could be a inappropriate response or a false sense of security.

Specific

The level of understanding of the intelligence must be optimal, allowing the recipient a good understanding about how the threat was detected and how the process was carried out.

Actionable

The intelligence must be sufficiently clear to stimulate some response. It should identify actions so organizations can counter threats or develop a suitable response..

2.2.2 CTI Methodologies

According to (FARNHAM; LEUNE, 2013), CTI can be classified in `Strategic` or `Tactical`. Strategic focuses on the adversaries' motivations in executing attacks. Tactical is the ability to represent the intelligence in understandable terms. Its components are Tactics, Techniques and Procedures (TTP) and Indicator of compromise (IOC). TTP represents the behavior of the attackers including an understanding of such behavior along a time-frame. The IOC (GRAGIDO, 2012) is an actionable data type observed on a network or computer, categorized by a certain confidence level. These indicators are classified in various types of CTI, e.g., IP Address, Domain, URL, Files Hashes, Certificate Hashes, EMAIL, among others. We describe the most commonly used IOC by adversaries, which are indicators we will use for our experiments. For example: the IP indicators are malicious servers from the Internet that are

represented by IP address; the Domain indicators are websites with malicious code such as malware, web pages with exploit code, web pages with driveby downloads (XU *et al.*, 2014); and finally the URL indicator, unlike of Domain indicators, it contains malwares on the allocated website, however this indicator's type blocks the URLs with malwares without needing to block the entire domain itself.

2.2.3 Cyber Threat Intelligence Sources

SANS (FARNHAM; LEUNE, 2013) divides CTI into three categories: internal, community and external.

Internal

This CTI refers to the internal process realized within the organization, which includes network servers such as firewalls, IPS as well as softwares like anti-viruses. These reports comes from forensic analysis and are not often publicly available.

Community

The community category is often a closed group of organizations that share processed and analyzed information through their trusted relationship. There are several of these groups, e.g., REN-ISAC[1] that promotes cybersecurity operation protection and response within the research and higher education (R&E) communities.

External

This last category consists in external intelligence sources provided by private and public organizations. The public sources are available to anyone interested in using it as input source for their security devices. For instance: malware domains provide a list of malicious domains found on the Internet, with lists regularly updated. The private sources are only available through payment, so it is important to subscribe to a reliable organization. For instance, Emerging Threats offers subscription services for IDS rules and IP reputation.

2.2.4 Share Threat Intelligence

Currently there is not a standard mechanism to share `intelligence` between organizations, which makes the communication more complex between them. Most of these organizations still share their data using CSV files and web servers with basic authentication (YOUNG, 2013). Others organizations have been using SMTP with GPG, which requires previous authentication to validate messages sent to a secure channel. For this reason, different

[1] Research and Education Networking Information Sharing and Analysis Center. Source:"http://www.ren-isac.net/"

format exchange mechanisms appeared as a standard for that the organizations can share data between themselves.

Common Intrusion Detection Framework (CIDF)

This methodology provides a structure similar to Location Identifier Separation Protocol (LISP) format to express information about events, attacks, and responses. It is used like protocols to interchange intrusion detection information. However, it has become obsolete(IETF, 1998).

Incident Object Description Exchange Format (IODEF)

This methodology provides a way to share information between CSIRTs. The language is based on XML structures. However, this framework has been interrupted and then recontinued by Format for Incident Information Exchange (FINE) sponsored by IETF. The interchange of information after the incidents are used to prevent future attacks (DANYLIW *et al.*, 2007).

Intrusion Detection Message Exchange Format (IDMEF)

Unlike IODEF, this methodology defines the data format for the interchange of information between IDSs. There are two protocols used to transfer the data between them. The first Intrusion Alert Protocol (IAP) based on HTTP, and the second Intrusion Detection Exchange Protocol (IDXP) (DEVARM *et al.*, 2007).

Extended Abuse Report Format (X-ARF)

This methodology reports incidents via e-mail about network abuses. However, X-ARF is not able to add multiple security events in a single message, and it is only allowed messages with a specific target (SHAFRANOVICH *et al.*, 2010).

2.2.5 Shared Threat Frameworks

There are multiples CTIs frameworks as well as security tools in order to combat the cyber threats. However, they would not be useful if the inteligence is not relevant, actionable and valuable for the goals of the organizations. In the next section we present the most important frameworks that offer intelligence through a shared channel.

Open Indicators of Compromise (OpenIOC)

Initially introduced by Mandiant (MANDIANT, 2015), then released as an open standard. OpenIOC(OPENIOC, 2013) is a open framework for sharing threat intelligence. OpenIOC is also composed by a community that provides advanced threat detection capability.

One of the advantages of OpenIOC is its capability to offer the creation of its own custom sets of indicators, so that organizations can describe technical characteristics to identify either known threats, or an attacker's methodology, or other evidences of compromise. Because sophisticated threats require sophisticated indicators, OpenIOC also provides advanced threat detection by using extensible XML schema.

Vocabulary for Event Recording and Incident Sharing (VERIS)

The VERIS framework (VERIS. . . , 2015) provides a common language for defining and sharing incident information. One of the advantages of VERIS is that other organizations can contribute data in a standard format to be used as a larger data set for deeper analysis. According to (FARNHAM; LEUNE, 2013), VERIS can be divided into five sections: victim demographics, incident description, discover & response, and impact assessment. Each one of them has multiple elements with specific data types and variable names.

One of the advantages of VERIS is to be part of a community that adds intelligence into a database available from Verizon (VERIZON, 2015). VERIS also is able to store data in a custom format.

MITRE Standards

MITRE (MITRE. . . , 2015) makes use of three standards to complement the needs of the CTI. The first is CybOX (CYBOX, 2015), a standardized schema for the specification, capture, characterization, and communication of events that are observable in the operational domain. The second is STIX (STIX, 2015), a structured language to represent the full range of cyber threat information, which consists of 9 key constructs and the relationships between them: observables, indicators, incidents, adversary TTP, exploit targets, courses of action, campaigns, and threat actors. The latest is TAXII (CONNOLLY *et al.*, 2014), a Trusted Automated eXchange of Indicator Information, which defines a set of services and messages exchanges for exchanging cyber threat information; In addition it also supports multiple sharing models to push or pull transfer of CTI data.

Open Threat Exchange (OTX)

OTX (OTX, 2015) is widely considered the world's first truly open threat intelligence community that enables collaborative defense with actionable, community-powered threat data. OTX is a centralized system for collecting intelligence, that is available at (ALIENVAULT, 2015). The intelligence is provided by AlienVault and interoperates with Open Source Security Event and Information Management (OSSIM) system to update the intelligence to OTX.

Collective Intelligence Framework (CIF)

CIF is a cyber threat intelligence management system (YOUNG, 2013) that combines multiple known malicious threat informations from trusted organizations. CIF was developed by REN-ISAC[2], and supported by a community where many developers contribute offering improvements in the system. One of the most advantageous features of CIF is its capability for sharing threat intelligence data between multiple reliable organizations and general users, through a client/server system. The IOC used by CIF are IP addresses, domains and URLs. CIF is also able to export intelligence for specific security tools, such as Snort or Bro IDS. Finally, CIF supports an intelligence framework to add more reliable data from trusted organizations.

Therefore, we can conclude that the intelligence frameworks aforementioned offer interesting features to be used in CTI. In Table 5, we make a comparison between the multiple frameworks, taking into consideration several factors such as the capacity to import and export IOC, structured incident data in a standard format, among others features of the intelligence framework.

Intelligence Framework	**OpenIOC**	**VERIS**	**MITRE**	**OTX**	**CIF**
Import and export IOC from/to other systems in a standard format	✓		✓		✓
Import and Export structured incident data from/to other systems in a standard format	✓	✓	✓		
Import, Export, Query and Manage CTI through CLI					✓
Enforce data sharing based attributes attached to CTI data					✓
Support CIA when sharing data			✓		✓
Export and select data based on creation dates of CTI data					✓
Automatize to import and export CTI data				✓	

Table 5 – Comparison between intelligence frameworks

2.3 Software Defined Networking (SDN)

Network architectures and devices are experiencing a significant growth in their complexity due to requirements of modern deployments, cost pressures at buying and operation time, and new demands of Cloud-scale virtualized data center infrastructures (NADEAU; GRAY, 2013). Recently, Software-Defined Networking (SDN) (KREUTZ *et al.*, 2015) has entered the networking scene as an architectural effort based on four main principles. Firstly, control and data planes are decoupled. Secondly, forwarding decisions are no longer based on

[2] Research and Education Networking Information Sharing and Analysis Center

destination only but on multi-layer flow abstractions, allowing the unification of the forwarding behavior of networking devices. Thirdly, the control logic is moved to external device(s) called SDN controller(s), which build and maintain a global network view to facilitate the programming of forwarding devices via standardized APIs (e.g. OpenFlow) to the forwarding devices. Lastly, network applications interact with the data plane devices through an abstract network view and higher-level APIs provided byt the SDN controller, easing the network programmability. Figure 5 shows a generic SDN architecture decoupling the control and data planes, the northbound and southbound interfaces, a global network view, and the abstract network view exposed to the control applications. More information on the OpenFlow protocol specification and the SDN architecture definition can be found in the Open Networking Foundation (ONF) organization in charge of related standardization and development work (FOUNDATION, 2014b). In the present document, our focus is on the SDN control application.

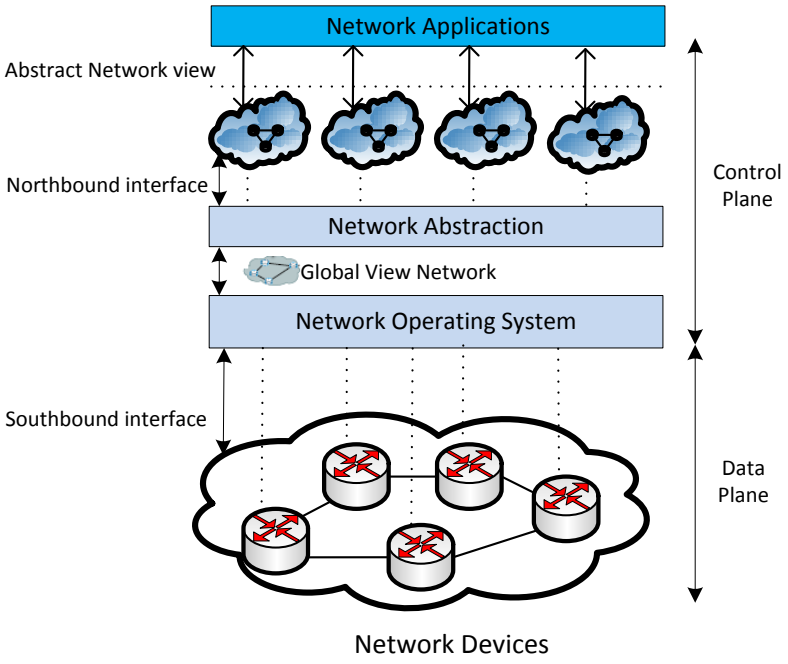


Figure 5 – SDN Architecture

The SDN application is becoming increasingly relevant. Developers and software engineers may build applications running on the abstract network view by using the northbound interface, and execute instructions on the forwarding devices by the southbound interface. One of the advantages of SDN is that the Open SDN controller provides a network-wide view and abstraction layer. We use OpenDaylight (OPENDAYLIGHT, 2015) as SDN controller because it allows applications to control non-OpenFlow-enabled switches as well as OpenFlow-enabled switches (FOUNDATION, 2014b).

SDN can work under both a reactive and a proactive application mode, depending

on how the controller works to communicate with the data plane and the network applications. Some SDN applications use both modes, reactive and proactive simultaneously. This application type is often known as a hybrid application.

2.3.1 Reactive SDN applications

The reactive application works as follows: When a switch receives packets that are not found in the flow table, these packets are encapsulated in `PacketIN` messages and then forwarded to the appropriate controller. Then, a software application, running atop the controller, resolves the requests made by the end-user node installing a rule on the switch, and indicating the action taken when the same flows are sent again. The application usually executes instructions to multiple switches at the same time, so that each switch will have a large set of flow entries for each particular query.

According to (NADEAU; GRAY, 2013), reactive applications need to be asynchronously notified of incoming packets that have been forwarded to the controller from switches. Because of the asynchronous characteristic and of the fact the controller need to notify the application of the events, the controller would act as a requester and the application as a responder. Thus, it needs an interface to register a listener and then receive callbacks from the controller when packets arrive. These listeners are able to receive notifications from the controller when certain events occur. When the application is informed of an event, e.g., a change of port state, the application has a chance to take some type of action. Figure 6 shows the reactive application design, where the controller has a listener interface that is used by the application to provide executing instructions on received packets.

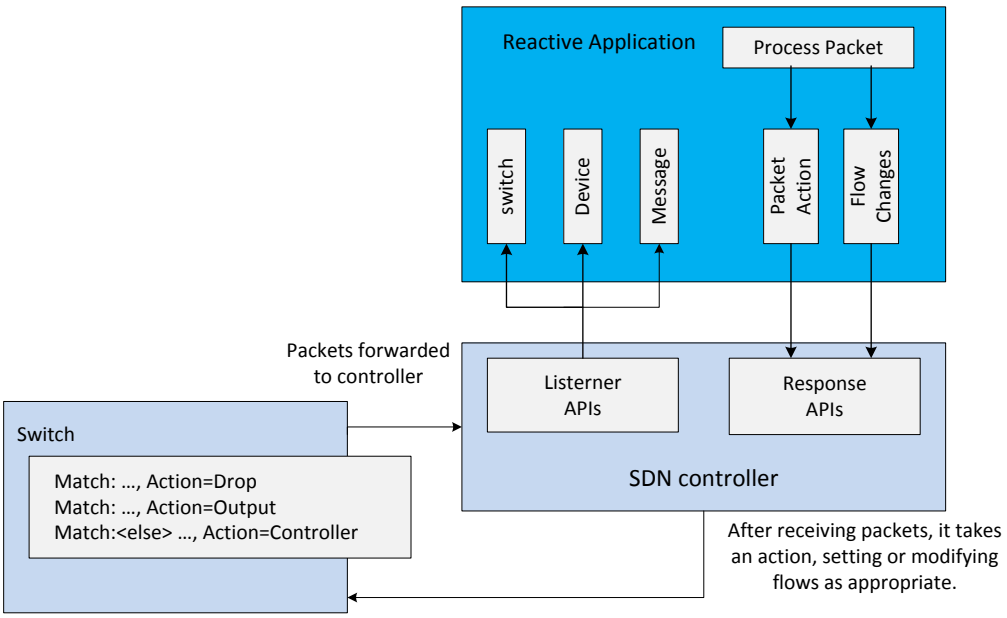


Figure 6 – Reactive application design. Adapted from (AZODOLMOLKY, 2013).

2.3.2 Proactive SDN applications

Unlike reactive, proactive SDN applications have the controller as responder and the application as requester. These proactive applications are implemented using RESTful APIs instead of using a listening interface. They also operate at different levels, depending on the northbound-interface type being used, and are placed on Open SDN controller, and called by the application.

In Figure 7, we show the proactive application design. As we can note, there is no listener interface between the SDN controller and the application , but an application from the controller that receives events about switches. These applications rely on queries from external network events, e.g., SNMP, sflow, among others. One of the advantages of these applications is the ability to retrieve the network information such as domains, switches, and hosts. The interface offered by REST API is known as `flow pusher`, which allows the application to set flow on switches (NADEAU; GRAY, 2013).

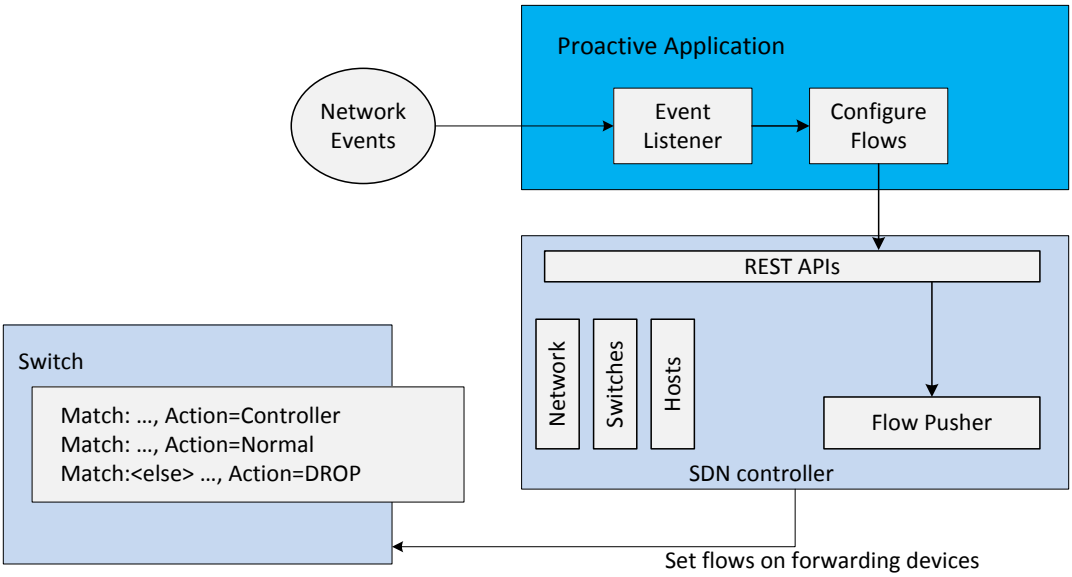


Figure 7 – Proactive application design. Adapted from (AZODOLMOLKY, 2013).

2.3.3 Hybrid SDN applications

Hybrid applications work for both traditional networking and SDN protocols, operating them in the same environment. The OpenDaylight works with hybrid applications, allowing to control legacy switches as well as OpenFlow switches. These applications allow users to introduce new SDN technologies, such as OpenFlow, to their legacy environments. According to (KREUTZ *et al.*, 2015), switches that do not support OpenFlow require an understanding of the level of SDN-via-APIs support that is available.

2.4 Related Work

In (LOPEZ *et al.*, 2014), the authors of BroFlow propose a system capable of reacting against DoS attacks in real time, combining an intruder detection system and an OpenFlow application programming interface. BroFlow is an extension of the Bro architecture (PAXSON, 1999) with two additional modules, one for the security policies and the other for message countermeasure. If a threat is found, POX (NOX, 2015) application either drops packets to eliminate malicious events or uses an output to forward packets to a specific target. According to (FELTER *et al.*, 2014), one of the disadvantages of BroFlow is to use hardware virtualization based on XEN instead of an operating system-level virtualization. Another is the use of reactive applications to counter DoS attacks, so it does not ensure effective response against that type of attack.

Our work is different because we propose an intelligent mode to drop known threats using proactive applications in the SDN environment using multiple isolated Linux systems (FELTER *et al.*, 2014).

In (LOBATO *et al.*, 2014), the authors propose an elastic architecture for intrusion detection and prevention. It uses certain mechanisms to detect anomalies on an intra-domain network with multiple virtual networks. They employ Deep Packet Inspection (DPI) for analyzing packets as well as the balance of traffic directed to virtual machines. These virtual machines perform intrusion detection and take action in case of malicious activity. By mirroring switch ports, traffic is forwarded both to the original destination and to the IDS which is not directly in the flow path. When the IDS discovers an attack, an alarm is sent to the controller indicating the reconfiguration of the flow table. Similar to (LOPEZ *et al.*, 2014), the work environment is based on hardware virtualization instead of containers and does not focus on preventing the Distributed Denial of Service (DDoS) attack. Our work scenario is not only intended to an intra-domain environment, but also to an inter-domain, and makes effective use of cyber threat intelligence as an important tool for defense against malicious users on the Internet.

SnortFlow (XING *et al.*, 2013) is one of the early works on IDS and SDN, in order to build a flexible IPS system in cloud virtual networking environments. SnortFlow is based on the performance evaluation of the virtual machines, reconfiguring the network in case of any abnormal activity. However, it only focuses on an intra-domain environment, with a snort agent acting on the domain of the XEN virtualization platform.

Our proposal uses Bro (PAXSON, 1999) instead of Snort (SNORT, 2015) because, Bro has the ability to run in high-speed environments and allows exchanging information in real-time with other security applications (MEHRA, 2012), e.g., threats intelligence sources. Thus, IntelliFlow takes advantage of Bro design's ability to integrate with the intelligence framework, in order to increase the database with sophisticated security rules.

SDNIPS (XING *et al.*, 2014) is an IPS based on Snort and Open vSwitch (OVS).

SDNIPS's authors compare a SDN solution with the traditional IPS approach and shows that, using SDN, the network reconfiguration designed by the controller enhances the prevention flexibility. The authors demonstrated the SDN feasibility and efficiency over traditional approaches. However, different from our proposal, the authors use Snort as IDS. We believe that IntelliFlow is a better alternative because it works in a proactive way, achieving optimal results in terms of feasibility and efficiency over other similar proposals.

IPSFlow (NAGAHAMA *et al.*, 2012) is a solution of IPS based on SDN/OpenFlow with automatic blocking of malicious traffic. One of its advantages is the selective and distributed capture of traffic in the switches for data analysis by one or more IDSs. IPSFlow uses an application named IPSFlowApp that allows communication with the controller of actions to be taken. The main drawback of the proposal is the time taken to detect some attacks in the network, since each IDS waits the confirmation of the controller to continue sending packets, as it does not mirror interfaces. Thus, some mechanisms are required to send a copy to the IDS for further analysis. IntelliFlow uses mirror interfaces to capture packages of the network in search of malicious activity, then these packages are analyzed by Bro IDS and sent to the SDN application.

Radware (RADWARE, 2015) is a commercial SDN application, which improves the security, performance and availability by optimally forwarding traffic to deliver network services. Their applications are dedicated to specific hardware platforms and support network throughputs up to 40Gbps. The main disadvantage of this system is the high cost of purchasing one of these products, e.g., Defense Flow and Defense Protection. Another drawback is the time taken for learning to properly use the tool. Our proposal uses a simple network architecture based on open source and proactive applications using SDN.

In (FELTER *et al.*, 2014), the authors propose a performance comparison between the traditional virtualization technology (Virtual Machines) and Linux Containers (LXCs). The main difference is that the VMs run a full Operating System on a virtual hardware. Unlike VMs, the containers modify an existing Operating System (OS) to provide extra isolation. The authors conclude that containers such as docker (DOCKER. . . , 2015) have equal or better performance than VMs. For that reason, we use Docker in our work environments.

In (BRAGA *et al.*, 2010), the authors propose a lightweight procedure to extract feature information needed for traffic analysis in order to detect DDoS attacks. They use SDN with OpenFlow to achieve low overhead. IntelliFlow offers better methods for DDoS attack detection and a lower rate of false positives compared to other proposals.

SciPass (BALAS; RAGUSA, 2014) proposes an OpenFlow application to transport "clean" data such as large scientific data and send it to its destination without going through firewalls and other security devices that introduce performance hits. SciPass improves the data transfer and reduces the load on network infrastructure. Evaluations are performed taking into account proactive and reactive modes. The authors compare the throughput of the proposal with

the conventional firewall method in the proactive mode. The results were such that the reactive bypass performance doubled throughput of firewall in 250ms, while with an equivalent throughput the transfer performance achieved 1.5s without a firewall. The disadvantage of SciPass is that proactive rules are manually created, according to the attacks on the network. IntelliFlow uses an intelligent system to create security rules automatically.

The table 6 comparison each proposal in terms of mode of action, main contribution, countermeasures, and inter-domain.

Name	Operation Mode	Inter domain	Controller	Countermeasure
SnortFlow Xing *et al.* (2013)	Reactive	No	POX	Performance evaluation about SnortFlow agent deployed at Dom 0 is better than at Dom U for about 40 %
BroFlow (LOPEZ *et al.*, 2014)	Reactive	No	POX	Effective detecting DoS attacks caused by flooding and blocking attacks from its origin. It reduces delay at 10 times on the networks under the attack and ensures the delivery of useful packets in the maximum rate of the link.
Elastic (LOBATO *et al.*, 2014)	Reactive	No	POX	Blocking a malicious flow; evaluation of resources consumed for packet analysis and elasticity overload and discharge in Detecting Module intrusion.
IPSFlow (NAGAHAMA *et al.*, 2012)	Proactive	No	Undefined	Automatic blocking of malicious traffic close to the orign
DefenseFlow (RADWARE, 2015)	Proactive	No	ODL, Cisco	DDoS protection as a native network service and collecting statistics
SciPass (BALAS; RAGUSA, 2014)	Reactive & Proactive	No	Owner	Improve transfer performance and reducing load on network infrastructure. Load balancing, bypass rules to avoid forwarding good data through firewalls of good data
IntelliFlow	Proactive	yes	any	Detect and prevent certain threats on networks by a proactive mode and deploying countermeasures to the threats learned through the CTI which lead to the networking infrastructure layer being reconfigured through flow table updates to the data plane switches

Table 6 – Comparison between different approaches

3 IntelliFlow Architecture

This chapter introduces IntelliFlow, the proposed intelligence system that aims to detect, prevent and/or react against security incidents in Software-Defined Networking (SDN) environments. IntelliFlow leverages mechanisms of Cyber Threat Intelligence (CTI) technologies and BroIDS to proactively drop different types of Internet threats. IntelliFlow also works in a reactive way against threats that have not been proactively blocked but are stored in the so-called knowledge plane.

3.1 Architecture

As shown in Figure 8, the IntelliFlow architecture is composed of the following modules: Process Connectors, Decider, False Positive, Intelligence Framework, Knowledge Plane, Flow Mapping, and IntelliFlow application. The remaining components present in the figure are either part of the original Bro IDS or the SDN controller architectures. IntelliFlow reads intelligence feeds by using both the Collective Intelligence Framework (CIF)[1] and Bro IDS input framework.

- **Process Connectors** is the module responsible for receiving notifications from Bro IDS about a possible threat. In addition, it removes certain inconsistencies in the notifications and sends them to the Decider module.
- **Decider** is the module which takes a decision according to the type of analyzed information, deciding if the resulting notification comes from a known or unknown source. In the case of a known source, this data is sent to the Intelligence Framework, otherwise it is sent to the Notice Framework.
- **False Positive** is the module responsible for deciding whether an unknown source is a false positive or not. It is based on the logic of a detection algorithm, which defines the number of times (threshold = 10) that our system can resist to take no corrective actions against a supposed attack. For example, if the number of times a threat executes attacks exceeds the threshold, countermeasures are executed using the Bro detection algorithm together with the Notice Framework to drop those attacks immediately.
- **Intelligence framework (IF)** processes and handles all the intelligence from known sources and the resulting information is represented in terms of Bro internal data structures that we refer to Knowledge Plane (KP). Figure 9 shows how the intelligence Framework works.

[1] Collective Intelligence Framework. Source: http://csirtgadgets.org/collective-intelligence-framework/

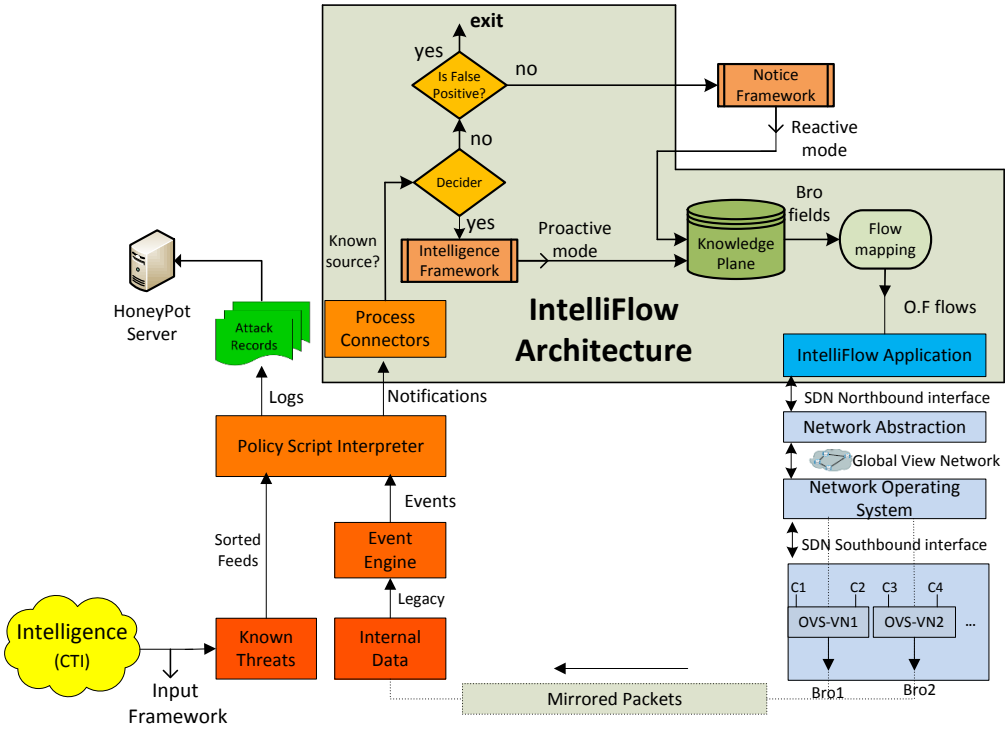


Figure 8 – IntelliFlow Architecture

- **Knowledge Plane (KP)** is a module that collects and stores information of known or unknown threats previously processed and analyzed by the Intelligence and Notice Frameworks respectively. The KP's information is used by the flow mapping module to translate the Bro fields into OpenFlow flow rules. Figure 10 shows how the KP works: after receiving the `intelligence` obtained by CIF as `input`, exported OpenFlow flows and sent as `output`. In addition, the KP allows Bro-IDS queries of analyzed data.

- **Flow Mapping** is a module that performs the flow mapping of the stored data by KP and then sends its information translated to the IntelliFlow API.

- **IntelliFlow API** is an API of the network application layer. It receives values from the flow mapping module to create the flow reconfiguration actions on the affected switches.

3.2 Mode of Operation

As discussed in Chapter 2.3, SDN applications can be broadly divided into three types of depending in the mode of operation they work. In addition, our requirement is to allow SDN applications to interact with IDS detection approaches as discussed in Chapter 2.1.3. Hence, IntelliFlow uses RESTful APIs for SDN proactive mode of application together with IDS that can be either based in reactive and/or proactive detection approaches. The resulting mode of operation allows to create proactive security rules based on the learned intelligence,

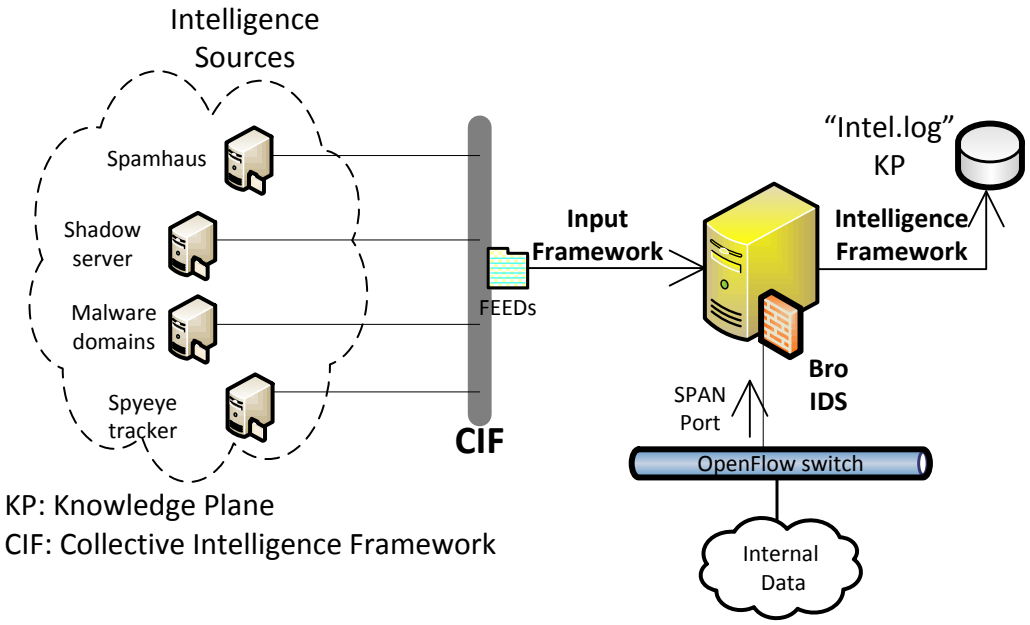


Figure 9 – IntelliFlow Framework

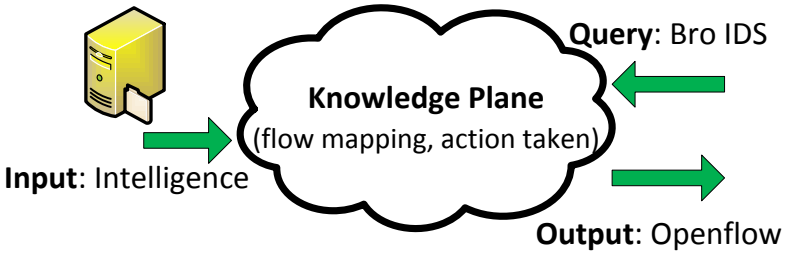


Figure 10 – Knowledge Plane

and reactive rules based on some anomalies found in the network. Below, we define both modes of operations and discuss their main advantages and disadvantages.

3.2.1 Reactive

When a new flow comes into the switch, the OpenFlow does a lookup in its flow table. If no match is found for the packet, the switch creates a `PACKET-IN` packet and sends it to the controller for further processing. This mode of operation is used by most IDSs, because they often work using the IDS's methods based on reacting an instant after an attack is executed. The main advantages is that new attacks can be filtered as soon as an anomaly is found in the network. The disadvantage is that this mode of operation may generates many false positives.

3.2.2 Proactive

The proactive approach is commonly used by the CTI communities to proactively block all threats considered dangerous. The advantage of this approach is that, if we understand the adversaries, we can develop tactics to counter attacks proactively. The disadvantage is that we need to constantly update the KP based on the security feeds.

3.3 Input Framework

Bro provides an interface between its event engine and the CIF intelligence data, known as Input Framework. This interface is able to read intelligence files formatted into a bro table, and then send it to the Intelligence framework for processing. To accomplish reading the file, we add the absolute file path in the Bro local site policy as shown below:

```
redef Intel::read_files += { “/home/bro-ids/INTELLIGENCE/intelligence-data.intel”, };
```

3.3.1 Intelligence Sources

As shown in Section 2.2, there is a notable difference between intelligence and information. However, there are many websites on the Internet that still publish security information without a deep analysis of the data. Here, we only show a list of three reliable intelligence sources, which have been properly analyzed, and that are used by the CIF server for loading in its configuration file to generate reliable security feeds. Then, these feeds are used as input for our KP.

- **Malware Domain List (MDL)** is a non-commercial community project, that publishes malicious domain information containing malwares.
- **Alienvault** is a commercial organization focused on providing network security to large companies. This reliable organization owns a threat intelligence laboratory containing an unified security management platform to identify the last malicious hosts from Internet.
- **Spamhaus** is a trusted organization that provides cyber threat informations to security devices in order to prevent different type of attacks from Internet. For example: Spamhaus Block List (SBL) is a database of IP addresses that acts as spam. Exploit Block List (XBL) is a realtime database of IP addresses that were infected by malwares, and that are being controlled by a botnet master. Policy Block List (PBL) is a DNS black list database that have no reason to be delivering unauthenticated SMTP emails to any internet mail server. PBL provides a central repository of dynamic IP’s and that are not maintained directly by Spamhaus.

3.3.2 Intelligence types

These intelligence types are threat indicators where malicious users use to be effective their attacks. There are a lot of types of intelligence, but we only show three of them because they are more used by organizations, and that also are the indicators used in our experiments on Chapter 4.

IP Address

The IP-based threats often come from botnet networks, or from scanner networks and malwares. The way to detect these threats using CIF is:

```
$ cif -q infrastructure/botnet -c 85 -p bro
$ cif -q infrastructure/scan -c 85 -p bro
$ cif -q infrastructure/malware -c 85 -p bro
```

Domain

The Domain-based threats are located in remote bot servers infected with malwares. Many times, the domain names are difficult to detect because these modern bots talk to Command&Control servers (C&Cs) whose domain names are generated randomly and might be valid for a single day only or maybe a few hours only, rendering a centralized source utterly useless. However, several of the reliable sources update their threat informations each hour or day. The way to detect these threats using CIF is:

```
$ cif -q domain/botnet -c 85 -p bro
$ cif -q domain/malware -c 85 -p bro
```

URL

Similarly to Domain, the Url-based threats are located in remote servers infected with malwares or phishing. These servers contains different types of malwares within a predefined path of the URL. Often they are used to infect the computers of their victims with malwares. The way to detect these threats using CIF is:

```
$ cif -q url/malware -c 85 -p bro
$ cif -q url/phishing -c 85 -p bro
```

3.4 IntelliFlow Implementation

In order to implement our proposed architecture, we developed an API in Python that acts on the SDN application layer. This implementation can be well used for both local

environment as well as different domain's environments. In Chapter 4, we will explain in more detail where the IntelliFlow API can be implemented. In the next sections, we describe the procedures used for flow mapping and reconfiguration of OpenFlow flows.

3.4.1 Bro IDS Input Fields

Once Bro executes the monitoring of live traffic on a mirrored network, it produces several types of human-readable ASCII log files, properly classified according to the event being triggered. Each different event produces a file with a defined output format; these are controlled by the logging framework. This allows fine-grained control of what situation generated which event and the reasons behind how it was produced (PAXSON, 1999). Bro also warns anomalous or potentially interesting situations produced by the Notice Framework. These events are shown in a `Notice.log` file.

On the other hand, the `intelligence` data are handled by the Intelligence framework in conjunction with the Input framework, in order to log hits seen on a specific network and to send the results of the analysis to a formatted file called `intel.log`. The intelligence data are read off disk by calling the constant `read_files` located in the Bro's `input` framework. In addition, Bro needs to load scripts located within the intelligence framework default directory. The fields shown in the intelligence file are shown in Table 7.

Field	**Description**
id.orig_h	Connection originator's endpoint IP address
id.orig_p	Connection originator's endpoint TCP/UDP or ICMP code
id.resp_h	Connection responder's endpoint IP address
id.resp_p	Connection responder's endpoint TCP/UDP or ICMP mode
seen.indicator	The indicator that triggered the match, e.g., a malicious IP.
seen.indicator_type	The indicator type that represents a threat, e.g., DOMAIN
seen.where	Location where the event was triggered, e.g., DNS:request

Table 7 – Bro fields

We emphasize the use of the `seen.indicator` field because it give us clues of where the attack was originated. In addition, we also emphasize the `seen.indicator_type` and `seen.where` fields because these generate the indicator type and location's service in which the event was triggered respectively.

The seen indicator, also known as Indicator of compromise (IOC), can be one or more hosts or networks that are identified as actors suspected of an attack. Those are determined through a process of rigorous analysis realized by security experts.

The types of indicators supported by Bro are shown in (PROJECT, 2015). In the present work, we only use the following types of IoC: Intel::ADDR, Intel::DOMAIN, Intel::URL, because we consider that they are the indicators more used by organizations that share `intelligence` on the Internet. With regard to where the data were discovered, we use

certain parameters to determine the origin of the attack, e.g., Conn::IN_ORIG, Conn::IN_RESP, HTTP::IN_HOST_HEADER, and HTTP::IN_URL. In Table 8, we show the relation between the Bro indictator types with their respective localizations where the threat is found:

Indicator Type	**Localization**
Intel::ADDR	Conn::IN_ORIG and Conn::IN_RESP
Intel::DOMAIN	HTTP::IN_HOST_HEADER
Intel::URL	HTTP::IN_URL

Table 8 – Indicator types used by our proposal

3.4.2 OpenFlow Output Flows

This section describes the entries of the OpenFlow flow tables used by the IntelliFlow application. As this table varies depending of the OpenFlow version, we use the latest OpenFlow 1.3, which contains seven main components shown in (FOUNDATION, 2014a). Two of them identify a unique flow in a specific flow table: a `match field` entry to match against packets, and a `priority` entry to match precedence of the flow entry. The `instructions` flow entry contains actions to be performed on the packet, such as dropping packets or forwarding them to another target. In Table 9 we have a description of each flow used by our API. We can see that there are some values used by default that are necessaries to build the new OpenFlow flows.

Flow	**Value used**	**Description**
dl_type	0x800	Matches ethernet protocol type ethertype
nw_proto	6	Matches IP protocol type proto
nodeid	any	Bridge's mac address
priority	0-65535	The order that an entry will match in comparison to other
nw_src	any	Matches the source IP address
nw_dst	any	Matches the destination IP address
tp_src	any	Matches the TCP source port
tp_dst	any	Matches the TCP destination port
actions	any	Represents a list of actions done on a packet when its entry has been matched

Table 9 – OpenFlow flows

3.5 IntelliFlow Countermeasures

The countermeasures sent by the IntelliFlow API are based on the sending of flow data to the SDN controller using the HTTP PUT method and the module `Requests` of Python. The execution of the API automatically corrects certain malicious activity sent to the network, by adding new entries with action field `drop` for dropping malicious flows, and `output` for forwarding the bad flows to a specific output interface, belonging to a HoneyPot server.

In the frame below, we note that the sending of flow data to the controller passes through four parameters: url, data, headers, authentication. The URL indicates the path where the controller flow administration is found. The data parameter contains the new flow, without encryption, that will be passed to the controller for processing. The header gives more specific information on the request. And, the `auth` parameter gives us the option to include the credentials of access from OpenDaylight server.

```
r = requests.put(url, json.dumps(d), headers='Content-Type' : 'application/json', auth=('username', '******'))
```

One essential part of our work consists in defining new flows according to the event triggered. For example, the new flow created for a Denial of Service attack is different from the one created for blocking malicious websites. Therefore, we create four different functions for each experiment being analyzed. Each function reads the last `intel.log` alarm file created by Bro.

Regarding the flow mapping, it is performed by translating the intelligence values from Bro to OpenFlow parameters, in order to create new OpenFlow flows. To differentiate a flow from another, we use a different priority value for each one. Thus, flows created will not be duplicated and can be interpreted by OpenFlow switches.

Finally, these flows created are sent to the controller by the RESTful application and the Northbound interface. The pseudocode of IntelliFlow details how the application is built to detect and drop attacks from known sources.

Pseudocódigo 3.1 Detection of different threats by using of the proactive approach

α = The indicator that triggered the match "*seen.indicator*"
β = The type of indicator "seen.indicator_type"
γ = Location in Bro's where the event triggered "*seen.where*"
δ = Connection originator's endpoint IP address "*id_orig*"
θ = Connection responder's endpoint IP address "*id_resp*"
ψ = Array of the set of indicators already used
η = Honeypot Server to study the features of the attack and to replicate the victim server.

if β == Intel::ADDR **and** γ == Conn::IN_ORIG **then**
  $\alpha \leftarrow \delta$
  if $(\alpha, \theta) \notin$ in ψ **then**
    Dropping "α" of the flow tables and forwarding it to "η"
    Including (α, θ) in ψ
  else
    Nothing to do
  end if
else if β == Intel::ADDR **and** γ == Conn::IN_RESP **then**
  $\alpha \leftarrow \theta$
  if $(\delta, \alpha) \notin$ in ψ **then**
    Denying access to the remote service "α" and forwarding it to "η"
    Including (δ, α) in ψ
  else
    Nothing to do
  end if
else if β == Intel::DOMAIN **or** Intel::URL **then**
  $\alpha \leftarrow$ is a malicious website
  if γ == HTTP::IN_HOST_HEADER **or** γ == HTTP::IN_URL **then**
    if $(\delta, \alpha) \notin$ in ψ **then**
      Denying access to the malicious site "α" and forwarding it to "η"
      Including (δ, α) in ψ
    else
      Nothing to do
    end if
  else
    Nothing to do
  end if
else
  Nothing to do
end if

4 Prototype and Experimental Evaluation

In this chapter, we present the proof-of-concept implementation as well as a series of experiments to validate and evaluate the IntelliFlow architecture considering both reactive and proactive modes of operation.

4.1 Proof of Concept Implementation

To validate the proof of concept, we designed two types of work scenarios implemented on our INTRIG[1] laboratory infrastructure, which are: Intra-domain and Inter-domain Ssenarios. These are composed ofscentainers (DOCKER..., 2015), virtual switches (PFAFF *et al.*, 2009) and virtual machines on KVM. This implementation also allows the coexistence of multiple parallel virtual networks, each one executing several services, based on the ODL SDN controller, that sends instructions to each virtual network in order to modify the behavior, by using of OpenFlow protocol, on the vSwitches (OVS) to which the containers are attached to. Application scripts and other codes related to the present work is available at "https://github.com/richardqa/IntelliFlow". We refer to the organizations that publish threats found on the Internet as just "organizations", container-based virtualized systems that are attacked by malicious users as just "victim", known threats as just "threats" and security filters as "intelligence" in this text.

4.2 Testbed

Figure 11 shows the testbed for KVM servers whose components are CIF server, a master Bro-IDS and an OpenDaylight Controller. With regard to the testbed for Linux containers, we have the scenario shown in Figure 12, that simulates Bro IDS sensors, victim servers and several malicious hosts. Both the KVMs and containers use Ubuntu 14.04 as operational system, including the machine that hosts all these services that we call as `main server`. Regarding virtual bridges, we use Open vSwitch 2.3 because it allows the programming the data plane by using the OpenFlow protocol. Moreover, Open vSwitch also provides the interaction with docker through pipework[2]. We also use the OpenDaylight controller, because it supports an extensive collaborative community based on open source and because also provide REST interfaces which makes possible to program flows from external applications. And finally a HoneyPot server for data analysis in deep of the anomalies found on the network.

[1] A network laboratory interested in the state-of-the-art of Internet technologies and innovation. Source:"http://intrig.dca.fee.unicamp.br"

[2] A lightweight tool that allows you connect together containers in arbitrarily complex scenarios through `cgroups` and `namespace`. Source: "https://github.com/jpetazzo/pipework"

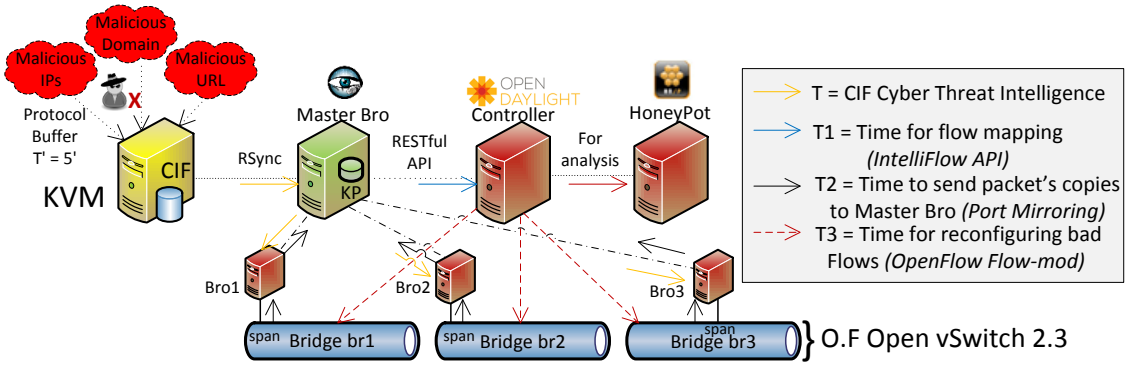


Figure 11 – Testbed for KVM servers

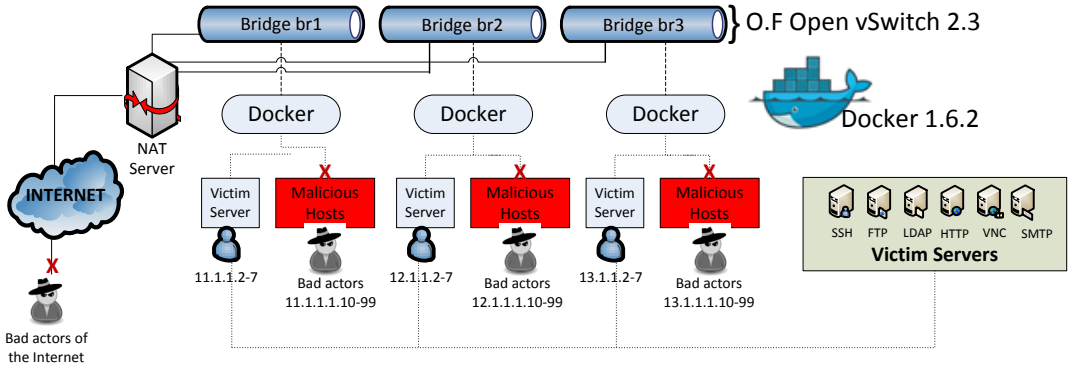


Figure 12 – Testbed for containers

Both hard disk and memory size together with their IP addresses for each virtual KVM are shown in Table 10. This figure also shows the capacity of the machine that hosts all virtual machines.

We simulate different services and software applications for each LXC, e.g., Scanners, Botnet and Web Services. The malicious containers use different range of addresses because we simulate a set of attacks executed at the same time.

Containers	**CIF**	**Master Bro**	**Controller**	**Main Server**
IP Address	192.168.122.2	192.168.122.3	192.168.122.4	192.168.122.1
Memory	8GRam	2GRam	4GRam	16GRam
Hard Disk	100GBytes	20GBytes	20GBytes	1TBytes

Table 10 – Values used for the servers virtualized on KVM

4.2.1 Intra-Domain Scenario

Figure 13 shows the intra-domain scenario composed of three different Virtual Networks (VNs), each of them VN-1,VN-2 and VN-3 contains a local Bro IDS that monitors the

incoming and outgoing traffic from its own network, reporting alert messages to the KP at different times T11,T12,T13 respectively.

On the other hand, the information of external threats is provided by a CIF server through its intelligence mechanism and sent to the KP. Hence, both alert messages `m1`,`m2`,`m3` and threats coming from external sources compose the `Input` source of the KP. The communication channel between CIF and master Bro is performed by RSync[3] protocol. Subsequently, Master Bro feeds our KP with different informations of malicious events that are harmful to the networks.

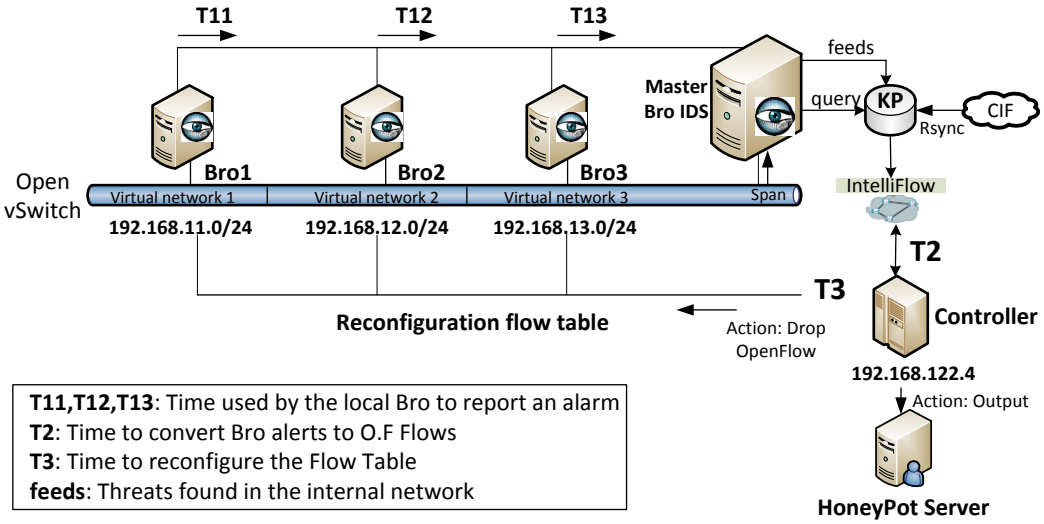


Figure 13 – Intra-domain Scenario

We assume that the time taken for translating Bro packets to OpenFlow flows is expressed as `T2` and the reconfiguration of malicious flows on bridges affected is `T3`. Thus, the total time spent by our proposed system for dropping one of those threats, is calculated by the sum of `T1` + `T2` + `T3`, being `T1` either `T11` or `T12` or `T13`. When working with threats, the time T1 is less in the proactive mode, because these threats have already been previously configured to be dropped before they are executed. Unlike the proactive mode, the reactive analyzes the threats looking for some anomaly or signature that can be used for dropping malicious flows; therefore, it spends more time detecting some particular attack.

We classify the reactive methodologies in two modes: with and without intelligence. The first one reacts in a intelligent way, using the `Input` framework to read the intelligence from a text file, and the `Intelligence` framework for processing, so known sources are dropped rapidly. The second one offers the capacity of reacting against attacks based on the analysis of its event engine and script interpreter, inspecting all the network traffic looking for any subnormal activity. Instead, the intelligent methodology monitors all its data traffic, making a comparison

[3] File transfer program capable of efficient remote update via a fast differencing algorithm. Source: "http://rsync.samba.org"

of the received information against the known attacks database and previously processed by reliable security organizations.

4.2.2 Inter-Domain Scenario

Our system can also be applied to an inter-domain scenario shown in Figure 14, which includes three virtual networks configured from different administrative domains, and a network management that receives security alerts from reliable organizations in conjunction with a Honeypot for analyzing of unknown threats. In this scenario, the network management entity uses CIF as an intelligence framework, which is responsible for parsing, normalizing, storing, post processing, querying, sharing and producing data sets of threat intelligence. The other three domains receive the data processed by CIF over a period of time of five minutes.

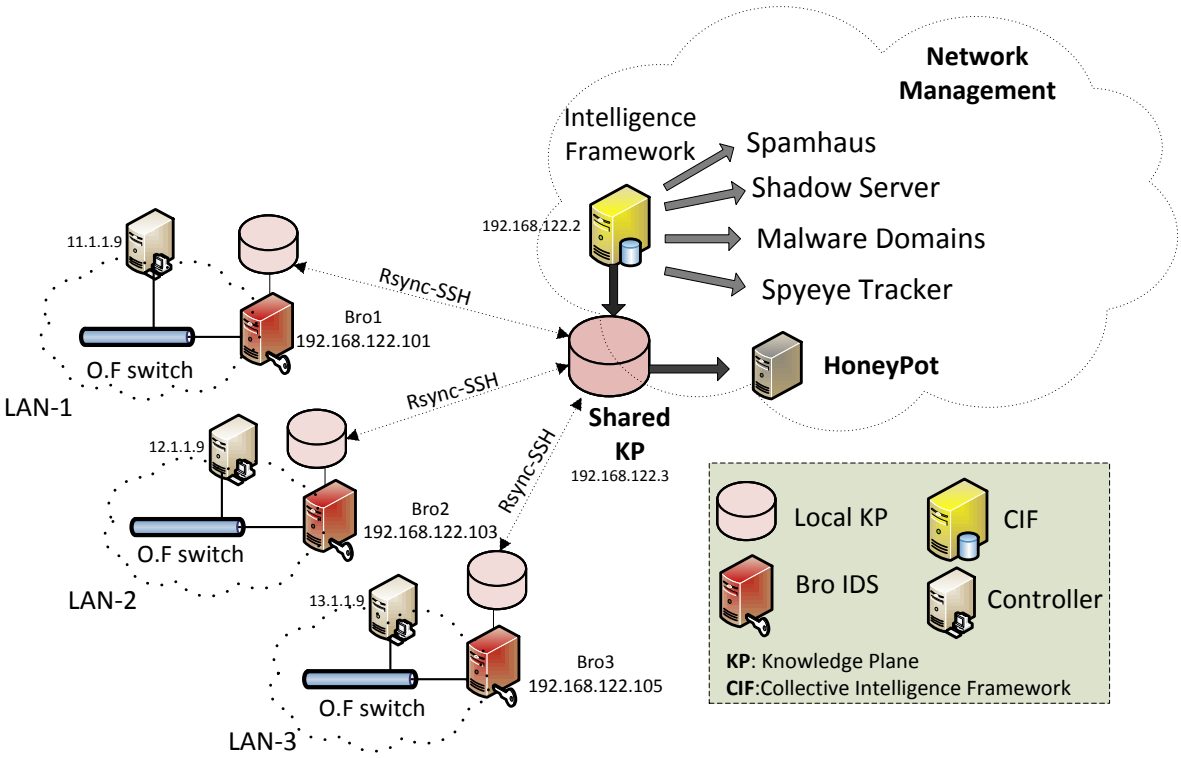


Figure 14 – Inter-domain Scenario

The idea behind centralized feeds, is to automate the sending of actionable data to a single centralized Bro server that hosts to our KP. This server receives all threat information, normalizes and produces feeds to be shared to the other networks such as LAN-1, LAN-2, and LAN-3. We use the Rsync file-copying tool together with the SSH protocol to keep the intelligence synchronized among these multiples domains. Thus, each domain will update its database with the latest source of intelligence available on the Internet.

Most of organizations often publish their reputation-database lists in a period of time of one hour approximately, each one containing enough information about a particular threat, so users may query this intelligence for their own further analysis. In this work, we use the CIF Server, an intelligence framework that allows to make queries against different reputation's databases by using Google Protocol Buffer (GPB). This framework is also able to maintain different sources of data synchronized, marking each one with a particular confidence rating. The GPB allows better performance compared with XML or JSON. The organizations that contribute with intelligence to CIF update their data using a confidence rate from 0 to 100% and in interval of one hour. However, our CIF server loads intelligence data each five minutes and enables to generate processed data to the KP in a period of 45 minutes. In the frame below, we show the procedure used to automate the intelligence feeds used by CIF server, also it shows the frequency to pull feed data is each 5 minutes, and to generate the feeds to be sent to the KP is of 45 minutes.

```
# Pulling the feed data
05 * * * * /opt/cif/bin/cif_crontool -p hourly -P -d -A root »/var/log/cif/crontool_hourly.log
2>&1
# Feeding the data generation in the CIF server
45 * * * * /opt/cif/bin/cif_feed -d »/var/log/cif/cif_feed.log 2>&1
NOTE:
#cif_crontool: Tool used for importing external data.
# The configure lines is located in the file '/etc/crontab'.
```

Then, these feeds are gathered at the KP as `intelligence`, and by using of the Rsync synchronization tool and the SSH authentication protocol we can maintain updated the local KP of each VN. Below, we show the procedure used to generate SSH cryptographic keys and how to transfer fast intelligence by using of Rsync. This tool also copies remote files from the CIF server to the main Bro IDS using the SSH authentication agent together with the Rsync and SSH.

Step 1: Both CIF and main Bro generate GPG keys in order to create a relationship between them.
ssh-keygen -t rsa -b 4096 -C "This is my new KEY"
Step 2: Each one shares its own public key through the SCP command.
CIF$ scp ∼/cif/.ssh/**id_rsa.pub** bro@192.168.122.3:/home/bro/.ssh/**authorized_keys**
BRO$ scp ∼/bro/.ssh/**id_rsa.pub** cif@192.168.122.2:/home/cif/.ssh/**authorized_keys**
Step 3: Configure SSH-Agent in order does not have to enter the passphrase each time you make a ssh or scp connection.
exec /usr/bin/ssh-agent $SHELL
ssh-add (enter the passphrase)
Enter passphrase for /root/.ssh/id_rsa:
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
Step 4: As both were identified like trusted servers, we launch directly the Rsync command on CIF Server
rsync -az --delete /home/cif/INTEL/ 192.168.122.3:/home/bro/IntelliFlow/
NOTE:
#The path **"/home/bro/IntelliFlow"** contains the list of Indicators of compromise (IOCs) obtained from CIF Server.

4.3 Experimental Evaluation

We evaluate six different types of attacks: brute-force, dictionary, botnets, scanners, malicious domains, and URLs. The first four attacks use `ADDR` as IoC, the fifth attack (malicious domains) use `DOMAIN` as IoC and the sixth attack (malicious URLs) use `URL` as IoC. These experiments are based on the methodology presented in Chapter 3.

Our proposal aims to protect the network against threats mentioned above by using both methodologies: proactive and reactive with intelligence. The first methodology uses new security filters to be pro-actively added to each OpenFlow virtual switch before a new threat, detected by our intelligence framework, is executed against victim servers. For example, when one of these attacks is launched, the victim or victims do not receive malicious packets, because these are immediately blocked by the corresponding bridge before they can affect any of these servers. What is then reported is only the packets going to Bro IDS sensor through the mirror interface and its event engine. The second methodology intelligently reacts against threats that are not found at the Knowledge plane, responding to an anomaly at the detection engine. For example, at the time an attack is launched, the local Bro-IDS sensor looks to match threats in the local KP. If no attack is found, Bro queries to the shared KP about the attack. And, if the threat is found, Bro automatically sends a security alarm generating an Intel.log file to the KP,

then the IntelliFlow API executes countermeasures, reconfiguring the flow table of the affected switch dropping any next malicious activity coming from the same source.

The reliable organizations provide the types of threat indicators, that will be periodically added to the shared KP, and therefore to the local KP of each VN in a period of time of five minutes, hence IntelliFlow constantly will read the information contained in its local KP, dropping malicious flows in the bridges. As a benefit, the bridges will be prepared to drop any malicious flow coming from these known sources. Despite the methodology with intelligence has an extensive information database, this methodology consumes less computing processing evaluated for different amount of malicious hosts.

Both automate tools, preventing or reacting against different adversaries, and dropping the majority of malicious packets directed to vulnerable servers within the shortest time possible. By the time we block these packets, our system will also have forwarded all the traffic to a HoneyPot Server, in order to understand how they operate and what specific steps we should do to protect our data. Alike, we are making a more efficient use of the time, e.g., to counter a small cyber-crime group, we would require a different time compared with a big group of dangerous criminals. Therefore, our work joins the cyber threat intelligence with the reactive methodology to drop the most of packets that were previously analyzed by other reliable organizations. As shown in Figure 12, we enabled six test servers: SSH, FTP, LDAP, HTTP, VNC and SMTP which are exploited by sophisticated hacking tools such as Hydra[4] for brute-force and dictionary attacks, Hping3[5] for DDoS attacks and Nmap[6] for port scanning attacks.

The communication between each Bro IDS, controller, and CIF is through a dedicated channel by using of the SSL protocol, ensuring an secure and isolated communication between them. The countermeasure messages are in the JSON format, and these contain flow informations such as ‘‘`destination and source IPs`’’, ‘‘`destination port`’’, ‘‘`ethertype`’’, ‘‘`priority`’’, and the ‘‘`protocol field`’’ that indicate the actions to match against a specific flow. To simulate the behavior of these attackers, we designed a script in bash that executes instructions from master docker through the ‘‘`docker exec`’’ command. For more reliable results, each experiment is executed ten rounds, varying the number of containers involved at the attack from 1, 5, 10, 20, and 40 for password-guessing attacks, and from 1, 5, 10 bots for DoS attacks.

In order to explicitly evaluate the proposed IntelliFlow architecture, we propose four study cases that include both methodologies. In the next sections, we demonstrate the use of the methodology reactive with intelligence applied to counter each type of threat previously mentioned, also we make a comparison between each one of them with the conventional

[4] A very fast network logon cracker which support many different services. Source: “https://www.thc.org/thc-hydra”

[5] Network tool able to send custom TCP/IP packets and to display replies like ping program does with ICMP replies. Source: “http://www.hping.org”

[6] Open source tool for network discovery and security auditing. Source: “https://nmap.org”

methodology. Throughout these proposed experiments, we intended to find answers to questions such as: How fast can attacks be detected?. How effective are the proposed methods?. Our proposal also can be applied of a hybrid mode, using functionalities of the proactive mode, automatically dropping known threats, and of the reactive mode with intelligence executing scripts for reconfiguring the flow tables as soon as that a threat is detected on the network.

4.3.1 Mitigation of Brute-force and Dictionary attacks

To perform this first experiment, we evaluate two types of password-guessing attacks, which are brute-force and dictionary-based, and that are launched from the main server. They work in an automatized and distributed way, launching their attacks to authentication servers such as SSH, FTP, LDAP, and so on, in order to gain illegitimate access to user accounts of the companies or personal user accounts such as Gmail, Facebook, or Yahoo.

These types of attacks do not act by directly looking for a flaw or bypass, but rather trying to guess passwords by combinations of different characters or using a word list. These attacks can either be on-line or off-line attack. The first allows that the attacker automates routines against any open authentication protocol, without looking at the creation of an exploit, but to abuse of some weaknesses in the networks. The second attempts to emulate the encryption together with the hash in order to gain access to a system account. We evaluate on-line attacks because we consider that they can be blocked in time.

To execute a brute-force attack, `hydra` generates random passwords of variable size by using of characters that contains `1` for numbers, `a` for lowcase, and `A` for upcase characters. Any other added may be put into the list. In the example above, the passwords generated have a length of 3 to 4 characters and contain lowercase letters, numbers, a percentage sign and a dot. Unlike the brute-force, the dictionary attack only needs of two text files: one for usernames and another for passwords.

The behavior of the attacker was simulated by designing a script that invokes different docker containers at the same time, being executed against a SSH authentication server. In our simulations, we use an extensive list of words for both usernames and password, that are obtained from the Internet.

```
#!/usr/bin/bash
for (( i=1; i<=10; i++ ))
do
docker exec -d bruteforce$i /usr/bin/hydra  -L user.txt -x 3:4:a1%. victim proto
docker exec -d dictionary$i /usr/bin/hydra  -L user.txt -P pass.txt victim proto
done
```

NOTE:

victim: IP Address of the victim server. E.g., SSH, FTP, LDAP.

proto: Authentication protocol of the victim server. E.g., 22,21,389.

user.txt: List of usernames. It contains an extensive list of the more known usernames.

pass.txt: List of passwords. It contains an extensive list of the more known passwords.

To counter these types of threats, we use the proactive and reactive methodologies, and the `intelligence` stored in the Knowledge Plane module presented in the Chapter 3. For example, many password-guessing hosts are proactively dropped in the OpenFlow flow table, therefore the attacks coming from one of them will be automatically blocked. The malicious hosts that are not located in the local KP, could be found in the shared KP as long as there is a match with the indicator of the threat.

The procedure to detect how many authentication tries crossed the network without being detected by the local Bro IDS is performed by counting the number of `Keys Exchange` events that received the server from the attacker. Therefore, we could measure the amount of authentication tries, and the maximum time that the victim server stops receiving the malicious events. Figure 15 show how our proposed system reduces the amount of access tries against a SSH server. The same results could also be obtained using other servers such as FTP, HTTP, VNC, SMTP as well as any another type of authentication server, included Gmail, Facebook, or Yahoo accounts that are exposed to these types of password-guessing attacks, because the access of the users to these social networks use authentication methodologies that also could be corrupted by brute-force or dictionary attacks.

We also defined the number of concurrent tasks and the maximum wait time in seconds for responses, because most of IDSs have predefined signatures to detect password cracking attacks; however, they are based on a defined rates.

Table 11 shows the response times that a victim server and the Bro IDS delay to drop the next malicious packets sent by the attacker, and the amount of authentication events that spanned the network and reached the victim server before countermeasures were installed on the network. These values are obtained by using of both methodologies with intelligence ($+$I) or without intelligence ($-$I), and the comparison of the two values obtained ($\triangle$I). We can note that for 1 and 5 malicious hosts, the methodology with `intelligence` consumes much less time to drop tries of authentication. However from 10 to more malicious hosts, the time

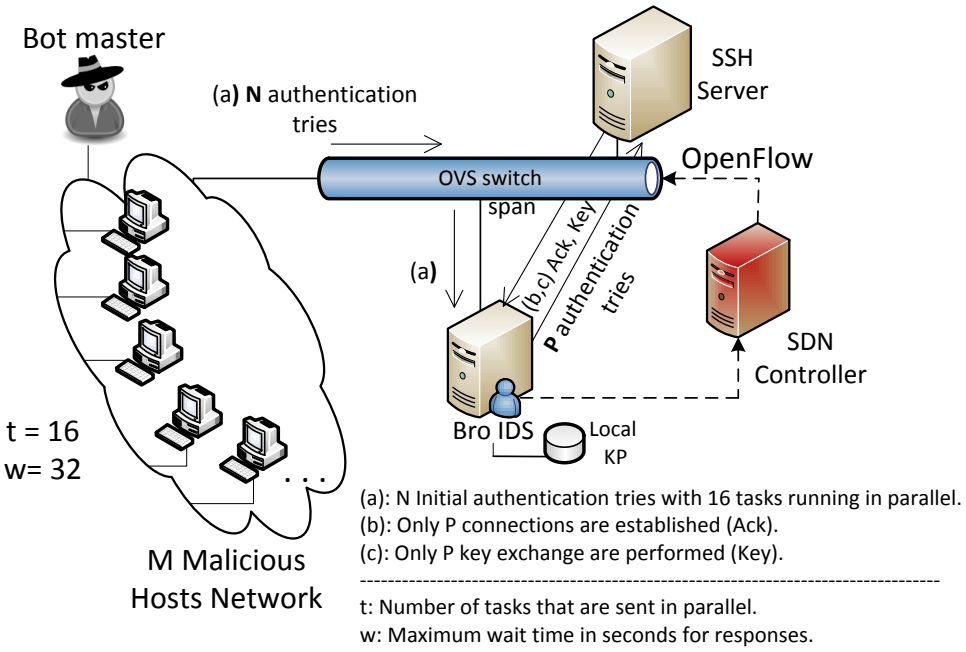


Figure 15 – Methodology to counter password guessing-based attacks

consumed by the intelligence is greater than the another methodology, but the processing to mitigate all the received authentication events is always of the 100% for the intelligence system, compared with the another methodology, e.g., for 10 malicious hosts the system without intelligence only would drop 8 of them, for 20 hosts only drops 16, and for 40 hosts only in 33. It is because of the anomaly detection algorithm does not work well for big amount of threats acting the same time, while our proposal get dropping all attacks that are found in the KP database.

Figure 16 and Figure 17 compare the response times and the not analyzed authentications evaluated for different amounts of malicious hosts. For example, for 5 hosts, the response time to drop authentication tries is of 0.48 seconds and the number of received events is of 51 both using the intelligence compared to the response time of 10.77 seconds and the received authentications tries of 55 using the another methodology. Both are totally effectives to drop threats. However, for number of hosts longer than 10, the response time and the received events of both are modified. For example, for 60, maximum number of malicious hosts, the response time using the intelligence is 82.59 seconds compared to 41.89 seconds of the another methodology, and the number of received authentication tries is of 185 less than 139 of the methodology without intelligence. Nevertheless, for all cases, using the intelligence we get dropping all the received malicious events differently of the another methodology that only get dropping some malicious events. We also conclude that it is better to drop all malicious packets that to let some of them to pass to the network, because between one of his attempts they could access to the user account of the victim server.

#Malicious Hosts	**M**	**Brute-force & dictionary**			
		Response time(s)		**Unanalyzed events**	
		Victim	Bro	Victim	Bro
	+I	0.15	0.01	16	18
	−I	9,80	11,74	19	124
1 malicious host	△I	9.65	11.67	3	106
	+I	0.48	0.10	51	34
	−I	10.77	32.71	55	314
5 malicious host	△I	10.29	32.61	4	280
	+I	16.09	11.36	99	86
	−I	13.75	46.74	108	679
10 malicious host	△I	11.19	3 5.38	9	619
	+I	59.53	45.49	179	283
	−I	35.01	144.58	117	2019
20 malicious host	△I	24.52	99.09	62	1736
	+I	82.59	82.54	185	311
	−I	41.89	230.51	139	3044
40 malicious host	△I	40.70	147.97	46	2733

M: Methodology used for each experiment.
+**I**: With Intelligence. −**I**: Without intelligence. △**I**: Difference between +**I** and −**I**

Table 11 – Comparison of the response time and received events by the victim server

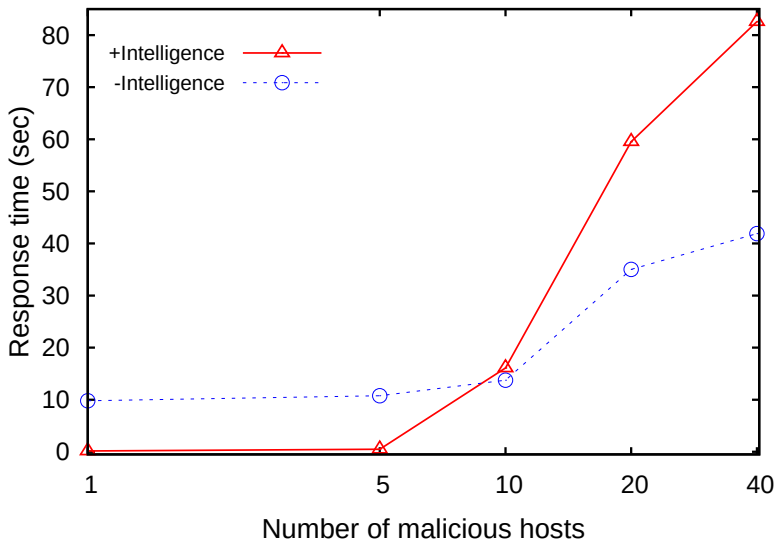


Figure 16 – Comparison of the response time varying the amount of malicious hosts

4.3.2 Mitigation of Scanners

TCP port scanners are computer programs that scans network services looking for connections with open TCP/UDP ports, in order to discover vulnerable services that can be corrupted by using of exploits and thereby access to the data network. These threats are multi-threaded, because they scan multiple hosts for a specific listening port, and they also execute port sweeping to remote networks.

To avoid being detected by firewalls or IDSs, scanners use advanced techniques such as `TCP SYN` or `TCP ACK`, in order to discover hosts running a particular service and also

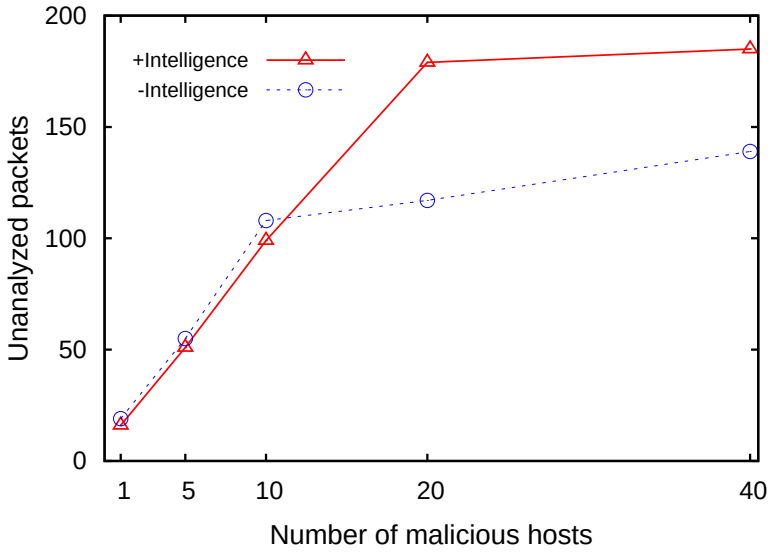


Figure 17 – Comparison of the unanalyzed packets varying the amount of malicious hosts

some other features of the target host. These malicious users use half-open TCP connections to trick the victim server. These methods of scanning are difficult to be detected by security devices.

As port scanning is often used by attackers, it is important to know how they work and what possible targets are used to exploit their tools. IntelliFlow leverages the relationships of organizations to advance threat analysis and security intelligence. For example, (ATLAS. . . , 2015), a reliable organization that uses a distributed network of sensors to capture and analyze data, it also provides malicious scanner lists that IntelliFlow may use to store these data into the knowledge plane.

By using `intelligence`, we prevent many of these scanners that were previously identified as threat indicators. Therefore, knowing their existence allows the security administrators maintain the servers free to be scanned by these malicious users. We simulate the behavior of these threats by designing a script based in `nmap`, that executes `port sweeping` against one of the test servers. This script performs a slight scanning by setting of the `TCP SYN` requests asking for TCP ports of the victim server.

```
#!/usr/bin/bash
for (( i=1; i<=10; i++ ))
do
docker exec -d scanner$i /usr/bin/nmap  -PS -p 1-100 victim
done
```

NOTE:

victim: IP Address of the victim server. E.g., SSH, FTP, LDAP.

Figure 18 describe the way how port scanners work by sending client requests to a range of ports of the victim server. We use TCP SYN to discovery server ports through of half-open connections. Our results demonstrate that despite the attackers use TCP SYN or TCP ACK flags, IntelliFlow proactively would block any try of port scanning, before they get any type of information of the network or server. But if the threat indicator is not found in the local KP, our system would ask to the shared KP. Each time new threat indicators are registered in the KP, IntelliFlow immediately executes actions to block the communications between these new indicators and the destination port of the whole network. However, whether the indicator is not found in the shared KP, then the Bro reactive methodology is used to counter a possible threat based on the anomalous behavior of the suspect. The technique to detect anomaly is based on the number of tries of port scanning performed to an specific server. By default, Bro IDS sends alarms `Scan:Port_Scan` when a host has been scanned in 10 unique ports. However, we consider a threshold of 5 port sweeping per malicious host to determine if the port scanner is really a threat and not a false positive. If it is considered as a threat, automatically a flow is created on the bridge blocking the source of the attack, and to the same time the indicator is forwarded to the HoneyPot server. Our results also demonstrate that the response time for dropping port scanning attacks is nearly 1.81 seconds by using of the intelligence, and longer time for the methodology without intelligence, because the system waits that the port sweeping attacks are repeated more than 5 times.

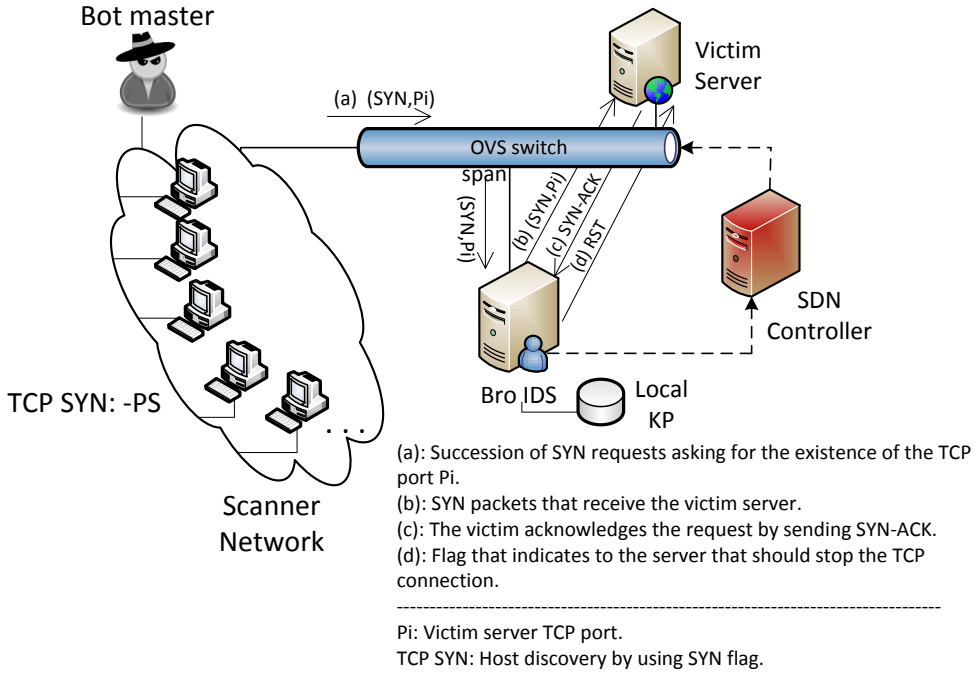


Figure 18 – Methodology to counter port scanning-based attacks

4.3.3 Mitigation of Botnet Networks

To accomplish the mitigation of these types of threats, we simulate a DDoS attack launched from a internal botnet network and directed to a victim server. With regard to real DDoS attacks, we does not consider the problem of the external link's bandwidth, because our work only focus attacks performed by our test servers.

Below, we show a script that executes this type of attack by using of `hping3` together with the configuration options such as the SYN flag (−−syn option), the destination port (−−`destport` option) and a variable rate of packets per second (−−`interval` option). We also note the use of −−`verbose` to enable verbose output showing more detailed information about the attack and the parameters −− *baseport* together with `victim` for the source port and the specific target respectively.

```
#!/usr/bin/bash
for (( i=1; i<=10; i++ ))
do
docker exec -d botnet$i /usr/sbin/hping3  --verbose --syn --destport 80 --interval
rate --baseport sport victim
done
NOTE:
rate: Packets sent per second (e.g. u10000 for 100 pps).
sport: Source port used by the botnet network .
victim: IP Address of the victim server. E.g., SSH, FTP, LDAP.
```

Figure 19 describes how a DDoS attack works, and how our proposal, based on SDN, get avoiding the SYN malicious packets flooding directed to a victim server. For example, if we have `M` bots that form a botnet network and that each one sends 400 SYN packets using a constant rate of 10 packets per second (pps), then `400*M` packets would be sent to the victim server per attack. However, this server only would receive `P` packets, that is a lesser amount compared to the total number of packets initially sent by the attacker, due to some of them are dropped by IntelliFlow, which executes countermeasures to reconfigure the flow tables by adding drop rules to the affected bridge. Concerning the Bro IDS, it receives 400*M raw packets including one packet of TCP retransmission (TCP rtns) for each bot. The TCP retransmissions occur because there is a lost packet for each new indicator that is mirrored to the IDS. Thus, the amount of TCP retransmissions is equivalent to the size of the botnet network. On the other hand, as initial packets are sent with the SYN flag enabled, Bro does not know how to respond to the victim with ACK flag, hence it only responds by sending RST packets indicating to stop the TCP connection.

Figure 20 compares the response times against each different rate of packets per

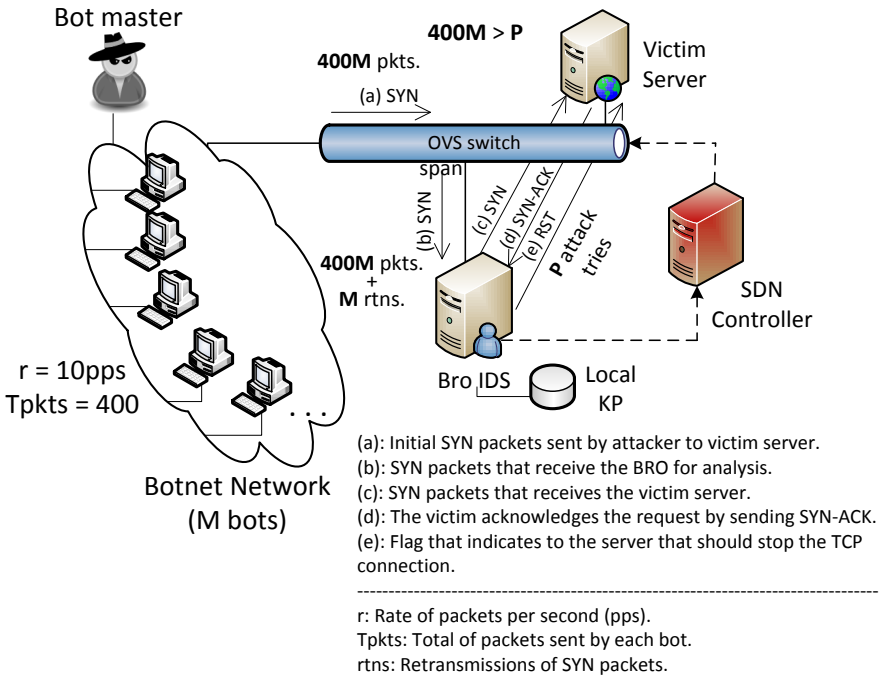


Figure 19 – Methodology to counter SYN flood-based attacks

second, observing a favorable trend for the intelligence that for higher rates the time to counter all packets is much less than the methodology without intelligence. We also observe a significant difference between the response time for the rates 4000 pps and 6000 pps in the without intelligence system, thus we deduce that for higher rates, the conventional without intelligence system does not guarantee an optimal response time, unlike of the intelligence provided by IntelliFlow proposed system.

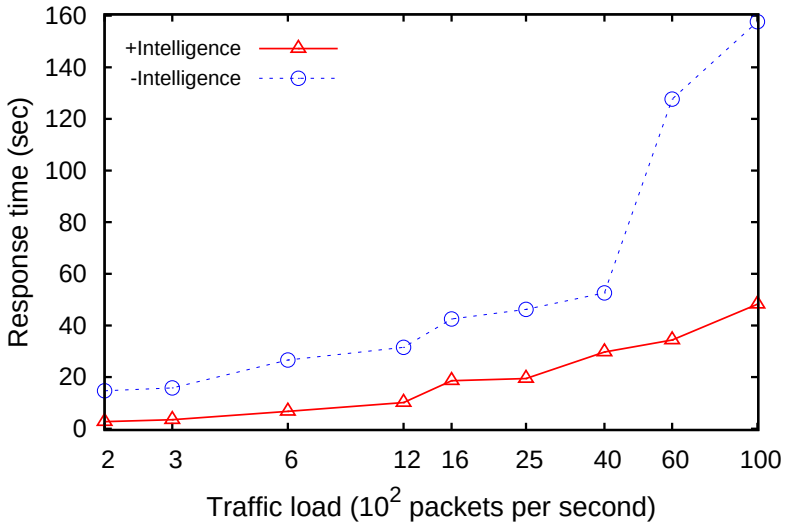


Figure 20 – Comparison of the response time varying the rate of packets per second sent by the attacker

In turn, Figure 21 compares unanalyzed packets against the same range of traffic load, noting that for a rate of 2500 pps, the amount of unanalyzed packets increased slightly faster in the without intelligence system, being more noticeable for the rate of 6000 pps, in where these unanalyzed packets for the conventional system is more than twice compared with our proposed system.

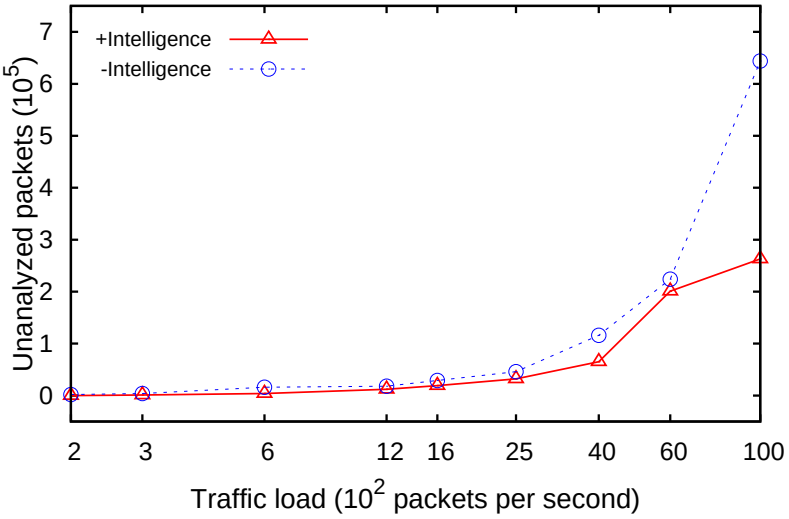


Figure 21 – Comparison of the not analyzed packets varying the rate of packets per second sent by the attacker

Regarding the memory and processor usage percentage, Figure 22 and Figure 23 compare the consume of their resources for each variation of the rate of malicious packets. We also note a similar trend where the intelligence system consume less resources than the other methodology, nevertheless both they reach their maximum value for rates nearing to 10^4 packets per second.

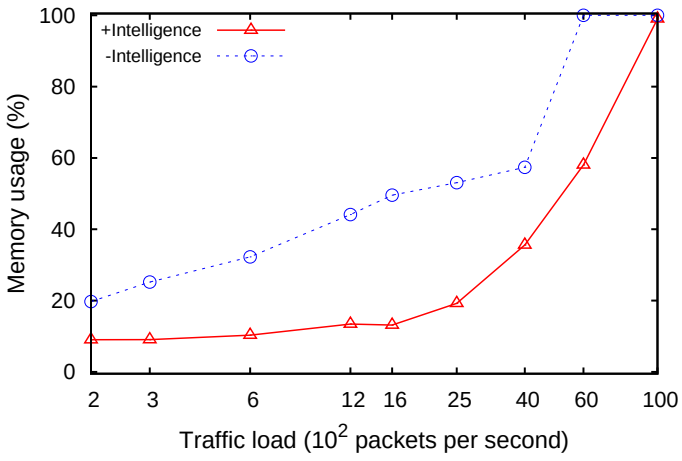


Figure 22 – Comparison of the memory usage performance varying the rate of packets per second sent by the attacker

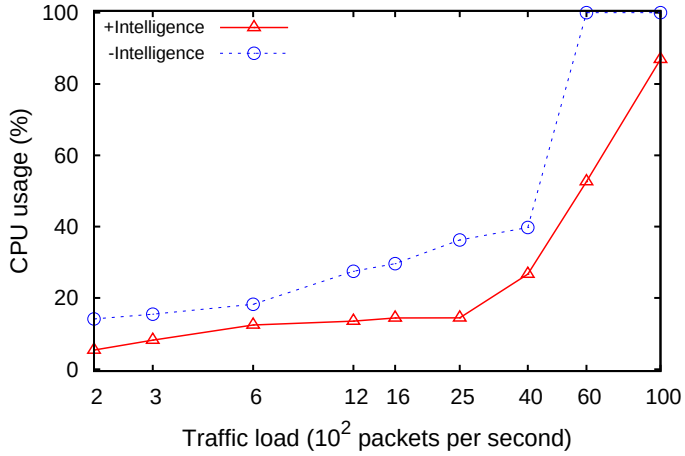


Figure 23 – Comparison of the CPU usage performance varying the rate of packets per second sent by the attacker

Table 12 shows the values of the response time, the amount of not analyzed events, memory and CPU usage percentages that were obtained in our experiments. These values vary depending of the size of the botnet network, the amount of packets per second sent by the attacker, and the methodology used. For example, in the case of a network with 3 bots, each one would send a succession of SYN requests to a target's server using a constant rate of 100 pps, so all the entire network would send the same attack to 300 pps, therefore the methodology with intelligence would drop the most of the syn flood events leaving only to pass almost a quarter compared to the other methodology, and in a time near the one-fifth to drop all the packets. Regarding the memory allocation, the victim server consumed nearly the third part compared to the used in the another methodology, and with regard to the CPU usage, near the half is consumed using the intelligence compared the another methodology. We also note that for all the cases, the amount of packets dropped is more effective using our proposed methodology than the conventional methodology. We also note in the table that for the worst case of 10^4pps, the CPU usage percentage is nearly 100%, so we assume that this rate of packets is the capacity threshold of each docker container. Therefore, for this worst case, in the proposed methodology the response time is less that the third part of the memory consumed, and less that half of the not analyzed packets both being compared with the without intelligence methodology.

4.3.4 Mitigation of malicious Domains and URLs

In this experiment, we work with Malicious Domain Blacklists (MDBLs), in where reliable organizations, such as `Malware Domains`, detect and publish malicious domains found on the Internet. Therefore, any user connected to the Internet, may to know whether its web domain is in the blacklist database. However, in order to perform a successful attack, the attackers would only use a very short period of time to execute their malwares on the infected computer, so they would avoid being detected and blocked by these domain reputation systems.

r(pps)	**M**	**#Bots**	**Response Time(s)**		**Not analyzed events**		**%Memory**	**%CPU**
			Victim	Bro	Victim	Bro	Victim	Victim
5 pps.	+I	1 bot	0.58	0.59	4	4	8.62%	3.17%
	−I		11.61	23.60	54	102	12.50%	4.27%
	△I	5 pps.	11.03	23.01	50	98	3.88	1.10
10 pps.	+I	1 bot	0.87	0.87	10	10	8.65%	3.65%
	−I		12.61	26.32	107	202	13.88%	4.51%
	△I	10 pps.	11.74	25.45	97	192	5.23%	0.86
100 pps.	+I	1 bot	1.63	1.59	165	24	8.72%	4.51
	−I		14.01	32.01	1,071	1,993	17.01%	12.71%
	△I	100 pps.	12.38	30.42	906	1,969	8.29%	8.20%
300 pps.	+I	3 bots	3.50	3.15	1,045	50	9.07%	8.17%
	−I		15.79	44.02	3,806	3,973	25.21%	15.44%
	△I	100 pps.	12.29	40.87	2,761	3,923	16.14%	7.27%
600 pps.	+I	3 bots	6.70	17.64	3,992	145	10.31%	12.42%
	−I		26.59	48.76	15,813	11,717	32.27%	18.24%
	△I	200 pps.	19.89	31.12	11,821	11,572	21.96%	5.82%
1,200 pps.	+I	3 bots	10.13	27.51	11,986	145	13.41%	13.51%
	−I		31.51	58.76	18,222	23,512	44.12%	27.48%
	△I	400pps.	21.40	31.25	6,236	23,367	30.71%	13.97%
2,500 pps.	+I	5 bots	19.43	14.89	31,805	230	19.24%	14.42%
	−I		46.21	63.15	45,942	49,069	53.08%	36.26%
	△I	500 pps.	26.78	46.26	14,137	48,839	33.84%	21.84%
4,000 pps.	+I	5 bots	29.69	29.16	65,493	339	35.60%	26.67%
	−I		52.62	71.41	116,083	78,306	57.37%	39.75%
	△I	800 pps.	22.93	42.25	50,590	77,967	21.77%	17.08%
6,000 pps.	+I	6 bots	34.37	33.63	200,116	418	58.04%	52.62%
	−I		127.69	138.33	223,668	117,185	100%	100%
	△I	1,000 pps.	93.32	104.70	23,552	116,767	41.96%	44.38%
10,000 pps.	+I	10 bots	48.23	47.45	263,596	427	99.00%	86.92%
	−I		157.65	163.13	644,759	124,150	100%	100%
	△I	1,000 pps.	109.42	115.68	381,163	123,723	1.00%	13.08%

r(pps): Total of rate of packets per second.
M: Methodology used for each experiment.
+**I**: With Intelligence. −**I**: Without intelligence. △**I**: The difference between +**I** and −**I**.

Table 12 – Comparison of the response time, received events, memory, and CPU usage percentage

To avoid that these malicious domains infect computers, we propose the same two methodologies discussed above: reactive and proactive with intelligence. By using the proactive methodology, we avoid any access to these websites, due to OpenFlow rules would immediately being created to block any remote access from the users. Unlike the proactive, the reactive methodology with intelligence would consist in reactively dropping any access to these domains.

Figure 24 shows an example of how the malicious websites work. By the moment an user accesses to a malicious website, the local Bro IDS looks for the domain indicator in the KP. If this is found, Bro immediately sends a security alarm to the network indicating that one threat has been found. Then, one countermeasure is executed disconnecting the access from the user to the website in a very short time.

On the other hand, we simulate the behavior of these websites by setting up malicious web servers and URLs. We use "example.com" as main domain name and generate different sub domains having as base to example.com. All these are located at the knowledge plane of our system.

Each server contains a certain malicious code that may exploit vulnerabilities against victims at the time that they access to one of these malicious websites. However, according to our experiments, the time to block the access to these websites is nearly 0.07 seconds. Thus, our system would intelligently block these websites before or at the beginning of the attack, avoiding the victim getting infected with malwares.

These methods are effective because we consider that people do not usually immediately stop browsing on websites and are often attracted by misleading advertisements containing malicious code. As most websites have to complete multiple actions in order to activate their malwares, our proposed methodology is valid for any attempt of access to malicious websites.

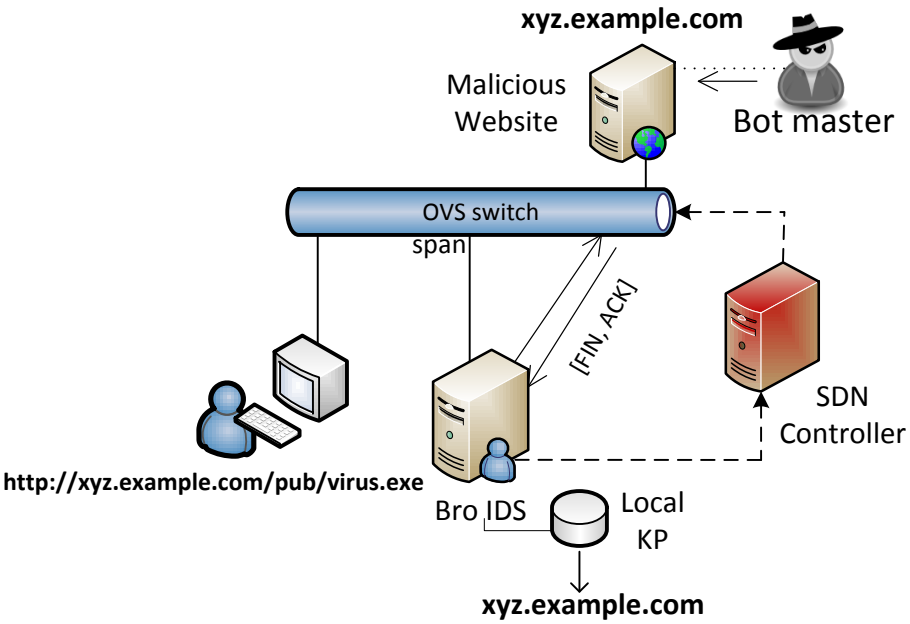


Figure 24 – Methodology to counter malicious website-based attacks

4.3.5 Considerations on the cost of proactive approaches

A final noteworthy consideration on all the experiments shall be made with regard to the costs of a proactive approach in terms of OpenFlow state. In case of software-based implementations of OpenFlow switches (e.g., OVS), while the size of the flow table is not a limitation, the amount of OpenFlow rules cause a linear increase in the memory consumption of the switch instance and may also add delay to flow table operations such as rule updates, packet lookups, statistic gathering, and so on. Some user experiences have pointed to certain unexpected

behavior in OpenFlow switches when adding a large amount of flow entries. [7] In case that a hardware-based OpenFlow switch solution is considered, without further considerations on the flow table sizes, a simple proactive approach may be infeasible since hardware switches have strict limitations on the flow table sizes – typically in the range of 10s of thousands of entries in today's commercial equipments (KREUTZ *et al.*, 2015). Therefore, future work may consider state optimization strategies to add only the more relevant flows indicating malicious activities to the data plane while keeping the remaining flows stored in the SDN controller and/or local or shared KP.

[7] Discuss mailing list OpenVSwitch. Topic: Able to add more than maximum number of flow entries in Flow table. Source: "http://openvswitch.org/pipermail/discuss/2012-May/007276.html"

5 Conclusions

Malicious users keep innovating their attack techniques much faster than defenders are finding ways to avoid them. Conventional defense and mitigation approaches such as anomaly-based or signature-based detection methods are not enough to counter the increasing amount and diversity of security threats. Security devices such as firewalls and IDS/IPS commonly rely on pre-defined rules added by humans. Many times security teams only focus on the event itself and do not perform a proper deep analysis on the root security problem.

In order to understand how adversaries work, organizations are resorting to new security approaches based on jointly processed information, actionable by security experts of organizations, forming the so-called Cyber Threat Intelligence (CTI). This classified information is known as `intelligence`, and it allows other organizations to updated their security devices with up-to-date configuration based on the intelligence data.

In this thesis, we have proposed IntelliFlow as a security architecture that leverages CTI in the context of Software-Defined Networking (SDN). More specifically, we use three CTI threat indicators: IP address, Domain, and URL, available in the Collective Intelligence Framework (CIF). CIF uses the `IODEF` format to exchange events or incident information, and the GPB protocol to make queries against different reputation's databases. Thus, CIF improves the communication between the security team and the reliable organizations. After processing the intelligence data and importing feeds from different reliable sources, we populate the knowledge plane proposed in the IntelliFlow system.

Taking advantage of `CTI` feeds, the Bro IDS intelligence framework was used to read the data processed by the CIF and to intelligently alert the network about possible threats. Then, the OpenFlow protocol was used to reconfigure the flow tables of the affected switches either reactively or proactively. When using the proactive methodology, our knowledge plane (KP) was updated every five minutes with the data shared by reliable organizations participating in CIF. The data was immediately transformed into suitable OpenFlow rules to drop all traffic matching the malicious indicators. In the reactive mode of operation using the intelligence feeds, the system is able to react to different types of threats.

Our experiments demonstrated that, for brute-force or dictionary attacks, we can mitigate long authentication attempts using the intelligence provided by CTI, unlike conventional methodologies that would only manage to drop part of them. In the case of distributed denial of service (DDoS) attacks, our system manages to drop packets in less time, receiving a smaller amount of packets, using less memory and CPU. Therefore, the IntelliFlow architecture allows better ways to intelligently prevent or react against some type of known threats when present in our KP or in the KPs shared by different organizations participating in the threat

intelligence layer.

One of the limitations of our proposal is updating all available intelligence periodically. Another limitation is having explored only the three most known indicators provided by trusted organizations. Other indicators could also be used –provided proper methods to convert indicators to OpenFlow rules is available.

As future work, we intend to explore with more detail the correlation between the information obtained from reliable sources and from IDS sensors strategically located in different public networks, in order to collect real data of attacks received in real time. By looking into OpenFlow statistics we may be able to identify zero-day attacks and detect different threats. Then, by leveraging machine learning techniques, we could provide capabilities to analyze the behavior patterns from attackers.

Future efforts shall also be devoted to a better understanding of the costs in terms of required state and its implications in software- and hardware-based OpenFlow switches. In turn, state optimization opportunities shall be pursued. Another avenue of future work would be generating automatic countermeasures to protect the data against newly attacks that are not found in the intelligence database of the reliable sources. When applied to inter-domain scenarios with real attacks from the Internet, developing new attack models based on machine learning techniques would allow further levels of collaborative approaches towards a more secure Internet.

Bibliography

ALIENVAULT. *Public data reputation*. 2015. Disponível em: <http://reputation.alienvault.com/reputation.data>. Acesso em: 28 aug. 2015. Citado na página 34.

ARBOR SERT. *ASERT Threat Intelligence Report*: Dd4bc ddos extortion threat activity. [S.l.], 2015. 48 p. Disponível em: <http://pages.arbornetworks.com/rs/082-KNA-087/images/ATIB2015-04DD4BC.pdf>. Citado na página 16.

ATLAS: Summary Report. 2015. Disponível em: <https://atlas.arbor.net/summary/scans>. Acesso em: 22 aug. 2015. Citado na página 63.

AZODOLMOLKY, S. *Software Defined Networking with OpenFlow*. [S.l.]: Packt Publishing, 2013. ISBN 1849698724, 9781849698726. Citado 3 vezes nas páginas , 37, and 38.

BALAS, E.; RAGUSA, A. Scipass: a 100gbps capable secure science dmz using openflow and bro. In: . [S.l.: s.n.], 2014. Citado 2 vezes nas páginas 40 and 42.

BRAGA, R.; MOTA, E.; PASSITO, A. Lightweight ddos flooding attack detection using nox/openflow. In: *Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2010. (LCN '10), p. 408–415. ISBN 978-1-4244-8387-7. Disponível em: <http://dx.doi.org/10.1109/LCN.2010.5735752>. Citado na página 40.

CARCANO, A.; FOVINO, I.; MASERA, M.; TROMBETTA, A. State-based network intrusion detection systems for scada protocols: A proof of concept. *Critical Information Infrastructures Security*, v. 6027, p. 138–150, 2010. Citado na página 26.

CISCO. Cisco Anomaly Guard Module. 2015. Disponível em: <http://www.cisco.com/c/en/us/products/interfaces-modules/catalyst-6500-cisco-7600-router-anomaly-guard-module/index.html>. Acesso em: 28 aug. 2015. Citado na página 27.

CLOUD SECURITY ALLIANCE. *The Notorious Nine Cloud Computing Top Threats in 2013*. [S.l.], 2013. 21 p. Citado na página 16.

CLOUD SECURITY ALLIANCE. *How Cloud is Being Used in the Financial Sector: Survey Report*. [S.l.], 2015. 13 p. Disponível em: <https://downloads.cloudsecurityalliance.org/initiatives/surveys/financial-services/Cloud_Adoption_In_The_Financial_Services_Sector_Survey_March2015_FINAL.pdf>. Citado 2 vezes nas páginas and 17.

CONNOLLY, J.; DAVIDSON, M.; SCHMIDT, C. *The Trusted Automated eXchange of Indicator Information (TAXII)*. [S.l.], 2014. Citado na página 34.

CYBOX. 2015. Disponível em: <http://cybox.mitre.org>. Acesso em: 28 jun. 2015. Citado na página 34.

DANYLIW, R.; MEIJER, J.; DEMCHENKO, Y. *The Incident Object Description Exchange Format*. [S.l.], 2007. Citado na página 33.

DARK Side of the Cloud. 2014. Disponível em: <https://molescope.com/blog-post/The-Dark-Side-of-the-Cloud>. Acesso em: 22 aug. 2015. Citado na página 16.

DEVARM, R.; CURRY, D.; FEINSTEIN, B. *The Intrusion Detection Message Exchange Format (IDMEF)*. [S.l.], 2007. Citado na página 33.

DOCKER Inc. 2015. Disponível em: <https://www.docker.com/>. Acesso em: 10 aug. 2015. Citado 2 vezes nas páginas 40 and 52.

FARNHAM, G.; LEUNE, K. *Tools and Standars for Cyber Threat Intelligence Projects*. [S.l.], 2013. Citado 4 vezes nas páginas 30, 31, 32, and 34.

FARSHCHI, J. *Statistical-Based Intrusion Detection*. 2010. Disponível em: <https://www.sans.org/security-resources/idfaq/statistic_ids.php>. Acesso em: 28 aug. 2015. Citado na página 26.

FBI warns of rise in DDoS extortion cases. 2015. Disponível em: <http://www.zdnet.com/article/fbi-warns-of-rise-in-ddos-extortion-cases/>. Acesso em: 22 aug. 2015. Citado na página 16.

FEDERAL Bureau of Investigation. 2015. Disponível em: <www.fbi.gov>. Acesso em: 30 aug. 2015. Citado na página 29.

FELTER, W.; FERREIRA, A.; RAJAMONY, R.; RUBIO, J. *An Updated Performance Comparison of Virtual Machines and Linux Containers*. Austin, TX 78758, 2014. Citado 2 vezes nas páginas 39 and 40.

FOUNDATION, O. N. *OpenFlow Switch Specification*. [S.l.], 2014. Citado 2 vezes nas páginas 19 and 49.

FOUNDATION, O. N. *SDN Architecture*. [S.l.], 2014. Citado na página 36.

GOOGLE loses data after lightning strikes. 2015. Disponível em: <http://money.cnn.com/2015/08/19/technology/google-data-loss-lightning/>. Acesso em: 22 aug. 2015. Citado na página 16.

GRAGIDO, W. *Understanding Indicators of Compromise (IOC) Part I*. 2012. Disponível em: <https://blogs.rsa.com/understanding-indicators-of-compromise-ioc-part-i/>. Acesso em: 13 jan. 2015. Citado na página 31.

HUSSAIN, A. Benchmarking performance of docker and traditional vms. In: CLOUDOPEN NORTH AMERICA. Chicago, 2014. Citado 3 vezes nas páginas , 17, and 18.

IBM: Real Secure Sensor Server. 2015. Disponível em: <http://www-935.ibm.com/services/th/en/it-services/realsecure-server-sensor.html>. Acesso em: 26 aug. 2015. Citado na página 27.

IETF. *Common Intrusion Detection Framework*. 1998. Disponível em: <https://www.ietf.org/proceedings/43/43rd-ietf-98dec-110.html>. Acesso em: 13 jan. 2015. Citado na página 33.

INFORMATION SYSTEMS SECURITY. *Securing Against Insider Attacks*. [S.l.], 2007. 48 p. Citado na página 17.

ISIGHTPARTNERS. *What is Cyber Threat Intelligence and why do I need it?* [S.l.], 2014. 9 p. Citado 2 vezes nas páginas 18 and 19.

JOHNSON, C.; BADGER, L.; WALTERMIRE, D. *Guide to Cyber Threat Information Sharing (DRAFT)*. U.S Departament of Commerce, 2014. 73 p. Citado 2 vezes nas páginas 18 and 30.

KHALIL, G. *Open Source IDS High Performance Shootout*. [S.l.], 2015. Citado na página 28.

Software-Defined Networking: A Comprehensive Survey, v. 103, n. 1. 63 p. Disponível em: <http://arxiv.org/abs/1406.0440>. Citado 4 vezes nas páginas 19, 35, 38, and 71.

KSHIRSAGAR, D.; TAGAD, D.; SALE, S.; KHANDAGALE, G. Network intrusion detection based on attack pattern. In: *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*. [S.l.: s.n.], 2011. v. 5, p. 283–286. Citado na página 26.

LIAO, H.-J.; LIN, C.-H. R.; LIN, Y.-C.; TUNG, K.-Y. Review: Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.*, Academic Press Ltd., London, UK, UK, v. 36, n. 1, p. 16–24, jan. 2013. ISSN 1084-8045. Disponível em: <http://dx.doi.org/10.1016/j.jnca.2012.09.004>. Citado 3 vezes nas páginas 21, 22, and 25.

LIBERTY GLOBAL. *The value of our digital identity*. [S.l.], 2012. 122 p. Disponível em: <http://www.libertyglobal.com/PDF/public-policy/The-Value-of-Our-Digital-Identity.pdf>. Citado na página 16.

LOBATO, A. P.; FIGUEIREDO, U.; LOPEZ, M. A.; DUARTE, O. C. M. B. Uma arquitetura elástica para prevenção de intrusão em redes virtuais usando redes definidas por software. In: SBRC 2014. *Anais do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos – SBRC 2014*. Florianópolis, SC, Brazil, 2014. Citado 2 vezes nas páginas 39 and 42.

LOPEZ, M. A.; FIGUEIREDO, U.; LOBATO, A. P.; DUARTE, O. C. M. B. Broflow: Um sistema eficiente de detecção e prevenção de intrusão em redes definidas por software. In: CSBC2014. *XXXIV Congresso da Sociedade Brasileira de Computação – CSBC 2014*. Centro de Convenções Brasil 21, 2014. Citado 2 vezes nas páginas 39 and 42.

MANDIANT. *MANDIANT*. 2015. Disponível em: <https://www.mandiant.com>. Acesso em: 13 jan. 2015. Citado na página 33.

MCMILLAN, R. *Definition: Threat Intelligence*. [S.l.], 2013. Citado na página 29.

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER AND COMMUNICATION ENGINEERING. *A brief study and comparison of Snort and Bro Open Source Network Intrusion Detection Systems*, v. 1 de *ISSN : 2278 – 1021*, (ISSN : 2278 – 1021, v. 1). Disponível em: <http://www.ijarcce.com>. Citado 2 vezes nas páginas 17 and 39.

MITRE Coorporation. 2015. Disponível em: <http://www.mitre.org/>. Acesso em: 21 jun. 2015. Citado na página 34.

NADEAU, T. D.; GRAY, K. *SDN: Software Defined Networks*. [S.l.]: O'Reilly Media, 2013. Citado 3 vezes nas páginas 35, 37, and 38.

NAGAHAMA, F. Y.; FARIAS, F.; AGUIAR, E.; LUCIANO, G.; GRANVILLE, L.; CERQUEIRA, E.; ANTôNIO, A. Ipsflow–uma proposta de sistema de prevençao de intrusao baseado no framework openflow. In: *III WPEIF-SBRC*. [S.l.: s.n.], 2012. v. 12, p. 42–47. Citado 2 vezes nas páginas 40 and 42.

NOX. 2015. Disponível em: <http://www.noxrepo.org/pox/about-pox/>. Acesso em: 22 jul. 2015. Citado na página 39.

OPENDAYLIGHT. 2015. Disponível em: <https://wiki.opendaylight.org/>. Acesso em: 29 mar. 2015. Citado na página 36.

OPENIOC. *Sophisticated Indicators for the Modern Threat Landscape: An Introduction to OpenIOC*. [S.l.], 2013. Citado na página 33.

OTX. *Open Threat Exchange*. 2015. Disponível em: <https://www.alienvault.com/open-threat-exchange>. Acesso em: 28 jun. 2015. Citado na página 34.

PAXSON, V. Bro: a System for Detecting Network Intruders in Real-Time. *Computer Networks*, v. 31, n. 23-24, p. 2435–2463, 1999. Disponível em: <http://www.icir.org/vern/papers/bro-CN99.pdf>. Citado 4 vezes nas páginas 18, 27, 39, and 48.

PFAFF, B.; PETTT, J.; T., K.; AMIDON, K.; CASADO, M.; SHENKER, S. Extending networking into the virtualization layer. In: . [s.n.], 2009. Disponível em: <http://openvswitch.org>. Citado na página 52.

PROJECT, T. B. *Documentation and Training*. 2015. Disponível em: <https://www.bro.org/documentation>. Acesso em: 19 may. 2014. Citado na página 48.

RADWARE. *DefenseFLow: The SDN Application that Programs Networks for DoS Security*. [S.l.], 2015. Disponível em: <http://www.radware.com/Products/DefenseFlow/>. Citado 2 vezes nas páginas 40 and 42.

SANS. *Understanding IPS and IDS: Using IPS and IDS together for Defense in Depth*. [S.l.], 2004. 14 p. Citado na página 21.

SCARFONE, K. A.; MELL, P. M. *SP 800-94. Guide to Intrusion Detection and Prevention Systems (IDPS)*. Gaithersburg, MD, United States, 2007. 127 p. Citado na página 23.

SCOTT-HAYWARD, S.; O'CALLAGHAN, G.; SEZER, S. Sdn security: A survey. In: *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*. [S.l.: s.n.], 2013. p. 1–7. Citado na página 19.

SHAFRANOVICH, Y.; LEVINE, J.; KUCHERAWY, M. *An Extensible Format for Email Feedback Reports*. [S.l.], 2010. Citado na página 33.

SHAH, B. *How to choose Intrusion Detection System*. [S.l.], 2001. 10 p. Citado na página 27.

SNAPCHAT. 2015. Disponível em: <https://www.snapchat.com/ads/>. Acesso em: 10 jul. 2015. Citado na página 16.

SNORT: Get-started. 2015. Disponível em: <http://www.snort.org>. Acesso em: 22 aug. 2015. Citado 3 vezes nas páginas 17, 27, and 39.

SOLTESZ, S.; PöTZL, H.; FIUCZYNSKI, M. E.; BAVIER, A.; PETERSON, L. Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev.*, ACM, New York, NY, USA, v. 41, n. 3, p. 275–287, mar. 2007. ISSN 0163-5980. Disponível em: <http://doi.acm.org/10.1145/1272998.1273025>. Citado na página 16.

STIX. *Structured Threat Information Expression*. 2015. Disponível em: <http://stix.mitre.org/language/usercases.html>. Acesso em: 11 dec. 2013. Citado na página 34.

SURICATA. 2015. Disponível em: <"http://suricata-ids.org">. Acesso em: 26 aug. 2015. Citado na página 27.

VERIS Community. 2015. Disponível em: <http://veriscommunity.net>. Acesso em: 21 apr. 2014. Citado na página 34.

VERIZON. 2015. Disponível em: <http://www.verizon.com>. Acesso em: 29 mar. 2015. Citado na página 34.

WALTZ, E. L. *Information Warfare Principles and Operations*. 1st. ed. Norwood, MA, USA: Artech House, Inc., 1998. ISBN 089006511X. Citado na página 29.

XING, T.; HUANG, D.; XU, L.; CHUNG, C.-J.; KHATKAR, P. Snortflow: A openflow-based intrusion prevention system in cloud environment. In: *Proceedings of the 2013 Second GENI Research and Educational Experiment Workshop*. Washington, DC, USA: IEEE Computer Society, 2013. (GREE '13), p. 89–92. ISBN 978-0-7695-5003-9. Disponível em: <http://dx.doi.org/10.1109/GREE.2013.25>. Citado 2 vezes nas páginas 39 and 42.

XING, T.; X., Z.; HUANG, D.; M., D. Sdnips: Enabling software-defined networking based intrusion prevention system in clouds. *10th International Conference on Network and Service Management*, 2014. Citado na página 39.

XU, W.; SANDERS, K.; YANXIN, Z. We know it before you do: Predicting malicious domains. In: *Virus Bulletin Conference September 2014*. [s.n.], 2014. Disponível em: <https://www.virusbtn.com/conference/vb2014/>. Citado na página 32.

YANG, Y.; MCLAUGHLIN, K.; LITTLER, T.; SEZER, S.; WANG, H. Rule-based intrusion detection system for scada networks. In: *Renewable Power Generation Conference (RPG 2013), 2nd IET*. [S.l.: s.n.], 2013. p. 1–4. Citado na página 26.

YOUNG, W. *Collective Intelligence Framework*. 2013. Disponível em: <https://code.google.com/p/collective-intelligence-framework/>. Acesso em: 14 mar. 2014. Citado 2 vezes nas páginas 32 and 35.