University of the Piraeus

Security Services for Small Office Home Office SOHO

Author

Leonidas-Angelos Laliotis

Supervisor

Prof. Christos Xenakis

A thesis submitted in fulfilment of the requirements

for the degree of Master of Digital Systems Security

Department of Digital Systems

February 2018

Abstract

The raid growth of computer networks and the expansion of internet have profoundly changed the way people and organizations communicate and interact. With these developments numerous security threats have been generated in the IT field. As a result Intrusion Detection Systems (IDSs) have received a lot of attention throughout the computer science field, as these technologies offer security mechanisms for protecting computer systems and information assets. Although numerous IDS technologies have been developed for large scale public and private entities, less attention has been given to small and home office (SOHO) environments. This thesis makes the attempt to fill this gap in the information security field by proposing a novel architecture and providing an effective and efficient solution in order to empower SOHO users to deal with security threats.

Greek Abstract

Η ραγδαία ανάπτυξη των δικτύων υπολογιστών και η εξάπλωση του διαδικτύου έχουν αλλάξει δραματικά τον τρόπο επικοινωνίας και αλληλεπίδρασης των ατόμων και των οργανισμών. Αυτές οι εξελίξεις έχουν προκαλέσει πολυάριθμες απειλές ασφάλειας στον τομέα χρήσης των υπηρεσιών πληροφορικής. Ως εκ τούτου, οι τεχνολογίες των συστημάτων ανίχνευσης επισύνδεσης (IDS) τυγχάνουν ολοένα και μεγαλύτερης προσοχής στον τομέα της επιστήμης της Πληροφορικής, καθόσον οι τεχνολογίες αυτές προσφέρουν μηχανισμούς ασφάλειας με σκοπό την προστασία των υπολογιστικών συστημάτων και των πληροφοριακών αγαθών. Αν και έχουν αναπτυχθεί πολυάριθμες τεχνολογίες συστημάτων ανίχνευσης επισύνδεσης ειδικότερα για μεγάλους δημόσιους και ιδιωτικούς οργανισμούς και επιχειρήσεις, λιγότερο ενδιαφέρον έχει επιδειχθεί για κατ' οίκον επιχειρηματικές δραστηριότητες και μικρές επιχειρήσεις που χρησιμοποιούν διαδικτυακές υπηρεσίες. Η παρούσα διπλωματική εργασία επιδιώκει να καλύψει το κενό αυτό στον τομέα της ασφάλειας της πληροφορικής, προσφέροντας μια αποτελεσματική και οικονομική λύση, επί τη βάση μιας νέας αρχιτεκτονικής, προκειμένου οι χρήστες των προαναφερθέντων εργασιακών χώρων να μπορούν να ανταπεξέλθουν στις απειλές που διακυβεύουν την ασφάλεια των πληροφοριακών αγαθών.

Acknowledgements

With the completion of this thesis I would to express my warmest thanks to my supervisor prof. Christos Xenakis and to postdoctoral researcher Mr. Christoforos Ntantogian for kindly helping and advising me in this research. Moreover, I would like to express my deepest appreciation to them for their constant assistance and support during graduate and postgraduate studies.

Finally, I would like to express my deepest gratitude to my parents for supporting me morally and financially during my postgraduate studies.

Contents

List of Figures

List of Tables

Abbreviations

AP	Access Point
ARM	Advanced RISC Machines
BSD	Berkeley Software Distribution
BSSID	Basic Service Set Identifier
CNSS	Committee on National Security Systems
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Server
EAP	Extensible Authentication Protocol
GUI	Graphical User Interface
HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IDPS	Intrusion Detection and Prevention System
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPV4	Internet Protocol version 4
MAC	Media Access Control
MIME	Multipurpose Internet Mail Extensions
N/A	Not Applicable

NIDS	Network Intrusion Detection System
NIST	National Institute of Standards and Technology
NoSQL	Non Structured Query Language
OS	Operating System
PCB	Printed Circuit Board
RADIUS	Remote Authentication Dial-In User Service
RAM	Random Access Memory
SIEM	Security Information and Event Management
SIM	Security Information Management
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
TOR	The Onion Routing
URL	Uniform Resource Locator
USB	Universal Serial Bus
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access II

Chapter 1

1 Introduction

Intense and continuous development of information technology (IT) and the rapid growth of computer networks, as well as the expansion of web services and applications over the internet have profoundly changed the way people and organizations communicate and interact (Wang & Zhanfg, 2015).

This reliance on technology and communication and the interconnectivity of IT applications and web services has exposed computer networks and computer systems to an increasing number of security threats, which have become more and more widespread and sophisticated. Those threats include, among others, web site defacement, network penetrations, denial of service attacks, viruses, Trojans, ransomwares and endless series of new stories that result in loss of control of computational assets, unauthorized disclosure of information, corruption and loss of data, loss of money and reputation, and much more (Ahmad Sharifi, et al., 2014). In essence, any malicious intrusion or attack violates fundamental computer security principles, i.e. confidentiality, integrity and availability (CIA) (Κάτσικας, et al., 2004).

In this framework, security[1] in the IT field has become a critical issue for modern computer systems and networks (Kruegel, et al., 2005). In particular, the increasing pressures from internal and external threats require a consistent and iterative protection approach to identifying, assessing and managing potential vulnerabilities and risks to operations (Karthikeyan & Indra, 2010).

[1] Security in the IT is understood as the protection of computer systems from the theft, alter, damage and corruption to their hardware, software or information, as well as from disruption or misdirection of the services they provide (Committee on National Security Systems, 2015).

In this context advanced technologies such as an Intrusion Detection System (IDS) is an important component of the defense-in-depth security mechanisms in computer network systems (Wang & Battiti, 2006). The IDS refers to the software of hardware system to automate the intrusion detection process (Stavroulakis & Stamp, 2010). While intrusion can be understood as the attempt to compromise CIA or o bypass the security mechanisms of a computer or network, intrusion detection is the process of monitoring the events occurring on a computer system or network, and analyzing them for sings of intrusion (Scarfone & Mell, 2007)[2].

Despite the variety of existing security methods described in the literature in the recent years, including peripheral protection mechanisms and various authentication and access control techniques, integral protection against intrusions cannot be achieved (Wang, et al., 2006).

IDS has proved as an effective defensive mechanism for all public and private IT infrastructures, including computer systems and computer networks especially those connected with the internet. IDS has been developed especially for large organizations and enterprises, as they have increasingly provided critical resources and sensitive data over the internet. Besides, these entities have the necessary financial, technical and scientific resources to maintain and support such technologies in order to protect themselves from security threats. It is equally important, that small offices and home offices (SOHO) have access to sophisticated defensive mechanisms, such as an IDS, against malicious interventions and cyber-attacks. In order for SOHO environments to be able to obtain and maintain this kind of security solutions, it is important to keep the complexity of such defensive mechanisms, as simple and user friendly as possible and at the same time being make these solutions accessible and affordable.

In this framework, this thesis makes the attempt to provide a complete model, based on a novel architecture, as a security solution especially designed for SOHO environments in order to empower these users to mitigate security threats.

[2] The exact definition offered by Scarfone and Mell reads as follows: “An intrusion detection can be defined as the process of monitoring the events occurring on a computer system or network and analyzing them for signs of intrusions defined as attempts to compromise the confidentiality, integrity, availability or to bypass the security mechanisms of a computer or network”

1.1 Purpose and scope

As already pointed out, IT applications and use of internet has penetrated almost every aspect of human life. The internet's explosive growth in power and popularity has dramatically increased the number and impact of malicious activities and the overall cyber criminality. The magnitude of the security threats has prompt it much interest within the security community towards researching technologies that can mitigate these threats. However, less attention has been given to SOHO environments, even though it is important for them to acquire more sophisticated defensive mechanisms against malicious interventions and cyber-attacks. Moreover, it is important to keep the complexity of such defensive mechanisms as simple and user friendly as possible by avoiding special installations of complicated software or hardware to the end devices and at the same time by keeping the cost of such solutions as low as possible using open source software and commodity hardware. In this context, more sophisticated security services such as an Intrusion Detection System (IDS) is deemed necessary.

This thesis has the objective to provide a novel architecture of a complete model regarding the creation and implementation of a SOHO Network Intrusion Detection System (NDIS) for empowering end users to identify, assess and manage potential network attacks and threats, and more to enhance the overall security of their information assets. For this purpose, this thesis presents a plug and play solution built on open source software and commodity hardware.

Regarding the scope of the thesis, the proposed solution concerns SOHO environments. However, development of the proposed model by optimizing and improving the used open source tools could make it applicable for other environments, too.

To be mentioned, this research does not aim to provide a solution model ready for production purposes. The main purpose of this research is to propose a functional architecture and present the various capabilities of the entire solution.

In order to appropriately address the above issues, I have developed and implemented a novel solution, which is a SOHO NIDS.

1.2 Limitations

One of the main limitations of the proposed solution refers to the BRO limitation regarding the inability of this software to support SSL offload. This means that the encrypted network traffic cannot be inspected with content rules, but it can be analyzed on a network level. Additionally, time was not enough to integrate the appropriate software so as to develop the SSL offload feature. Finally, another important limitation concerns the lack of the adequate resources to create an email infrastructure in order to send automated email notifications to end users notifying them about special network events.

1.3 Target audience

The solution that this thesis presents has been developed for SOHO users with an advanced knowledge and deeper understanding of network traffic and information security, who are looking for more sophisticated methods in order to increase their security levels and resilience of their computer systems and information assets. Especially, this solution can be more attractive for small businesses or sole traders who seek to increase their network security, but they cannot afford to invest in a solution of a large enterprise scale.

1.4 Thesis Structure

The remainder of this thesis is structured in five chapters. More specifically:

- Chapter 2 looks into the technological background and the hardware and software used for implementing the NIDS for SOHO environments.
- Chapter 3 outlines the proposed architecture, the topology of the implementation and the use cases.
- Chapter 4 focuses on the analysis of the methodology followed for implementing the experiments, the measurement tools that were used, as well as the parameters and the scenarios of the conducted experiments.
- In Chapter 5 the results of the experiments are presented and discussed.
- Finally, Chapter 6 presents the conclusions and opportunities for future research

Chapter 2

1 Technological background – Hardware and Software

This chapter describes the technological background of this research and also outlines the software and hardware that have been used. This background helps the reader to get a better understanding of the different information technology concepts and applications that this thesis analyzes and applies.

1.1 Technological background: Fundamental concepts

This section looks into the three main technologies applied for structuring the proposed model, including Intrusion Detection System (IDS), Single Board Computer (SBC) and System Information Management (SIM).

1.1.1 Intrusion Detection System (IDS)

An intrusion detection can be understood as the process of monitoring the events occurring in a computer system or a network and analyzing them for signs of possible incidents. These incidents may refer to violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices (Scarfone & Mell, 2007).

An intrusion detection system (IDS) is a software that has all the capabilities to activate and automate the intrusion detection process and is being used to identify possible incidents that are occurring in real time or at a specific period of time. For example, an IDS could detect an attacker who is trying to compromise a computer system by exploiting a vulnerability in the system, or detect a user who is connecting to a malicious URL (ibid.).

There are many types of IDS technologies mainly characterized by the detection mechanisms they use, and the different protocols and type of events that they detect. The primary classes of detection methodologies are: signature-based, anomaly-based, and stateful protocol analysis.

More, a network intrusion detection system (NIDS) can be defined as a system that monitors network traffic and looks for particular network segments or devices and analyzes network, transport, and application protocols to identify suspicious activity, that can be an attack or unauthorized activity (ibid.).

1.1.2 Application of IDS technology

An IDS inspects and analyzes a network event as soon as occurs. As soon as the IDS detects a violation it logs the incident and reports the violation to the security administrators in order they to prevent or minimize the overall damage caused by the incident. Most of the Intrusion Detection Systems (IDSs) have preconfigured security policies and detection mechanisms that fit to almost all different security standards of a computer system or network. In addition, IDSs can be configured to fit the different security needs of the entities that the IDSs are intended to protect (Wang & Battiti, 2006).

1.1.3 Key functions of an IDS

As already mentioned, the principal operations of an IDS concern, among others, the monitoring and analyzing events, logging information and producing notifications and reports.

In particular, monitoring of events concerns the comprehensive monitoring of connections seen on the wire, the monitoring of application-layer transcripts, such as network protocols. Furthermore, an IDS can extract files from specific sessions, e.g. HTTP and analyze them against malicious code. An IDS can also produce reports regarding a vulnerable version of a software (Liao, et al., 2013).

More precisely, the IDS function of logging information of the monitored events works as follows: information is usually recorded locally or it can be sent to a remote system, such as a centralized logging server, or to be sent a security information and event management (SIEM) solution (Stallings, 2005).

Regarding the notification of the security administrators for specific observed events, the IDS sends a notification message that typically includes only the basic information or a custom explanation about the event. This notification is transmitted to specific recipients. The recipient can either be another computer system, and in this case notification occurs in the form of a simple network management protocol (SNMP), or an end user, such as a security administrator, and in that case notification has the form of an email message (ibid.).

As mentioned above, an IDS can produce reports, which is a summarization of the monitored events providing details for important incidents for a specific period of time.

1.1.4 Detection mechanisms

As mentioned above, IDSs may have different detection mechanisms and methodologies that they use for monitoring and analyzing events. The main classes of detection mechanisms and methodologies are signature-based, anomaly-based, and stateful protocol analysis. Those mechanisms are deployed either separately or all together for most IDSs.

1.1.4.1 Signature-based detection

Signature-based detection usually identifies malicious behavior by matching the observed events against predefined patterns of events of known attacks. Some examples include a Secure Shell (SSH) attempt with a username of “root”, which is a violation of an organization’s security policy; an e-mail message with a subject of “You win the lottery!” and an attachment filename of “lottery.exe”, which are signs of a known form of malware; or, an operating system log entry with a status code value of 645, which indicates that the host’s auditing has been disabled. The signature-based mechanism is able to detect predefined known attacks with high accuracy, because this model just compares the current event to a list of signatures using string comparison operations.

While this mechanism is increasingly effective, simple and easily applicable, besides it is very vulnerable to complex attacks and is required to be constantly updated with signatures of new attacks. It also lacks the ability to remember previous requests when processing the current request. This limitation prevents signature-based detection methods from detecting attacks that comprise multiple events if none of the events contains a clear indication of an attack (Karthikeyan & Indra, 2010). Finally, the signature-based mechanism and other pattern matching algorithms used currently must be able to operate at wire speeds. However, as networking speeds are doubling every year, it is becoming increasingly difficult for software based solutions to keep up with the line rates and this has generated the need for specialized hardware-based solutions.

1.1.4.2 Anomaly-based detection

Anomaly-based detection is the process of comparing profiles of normal behavior with the activities that run in the foreground by identifying the deviations of the defined profile. An IDS is using anomaly-based detection of profiles that represent the normal behavior of things such as users, hosts, network connections or applications that have been developed by monitoring the characteristics of a typical activity over a specific period of time. For example, a profile for an application may show an average of 4% of CPU load during typical workday hours. The IDS then uses statistical methods to compare the characteristics of current activity to thresholds related to the profile, such as detecting when this application compromises significantly more processor usage than expected and alerting an administrator of the anomaly (Scarfone & Mell, 2007).

Profiles can be developed for many behavioral attributes, such as the number of e-mails sent by a user, the number of failed login attempts for a host in a given period of time. Such profiles are generated over a period of time, which may cover days or even weeks, and can either be a static profile or a dynamic one, and stay unchanged until the IDS initiates the creation of a new profile. Systems and networks change over time, so the corresponding measures of normal behavior can change, too. As a result, a static profile will eventually become inaccurate, so it needs to be regenerated periodically. Although dynamic profiles do not have this problem, they are susceptible to evasion attempts from attackers. For example, an attacker can perform small amounts of malicious activity occasionally, and slowly increase the frequency and quantity of activity over time. If the rate of change is sufficiently slow, the IDS might think the malicious activity is normal behavior and include it in its profile. Obviously, a malicious activity as part of a profile is a common problem with anomaly-based IDS products (Garcia-Theodoro, et al., 2009).

Building profiles and making them accurate can be very challenging in some cases, because computing activity can be so complex. Anomaly-based IDSs often produce many false positives, because of benign activity that deviates significantly from profiles, especially in more diverse or dynamic environments (ibid.).

To be stressed, the major benefit of anomaly-based detection methods is that they can be very effective at detecting previously unknown threats.

1.1.4.3 Stateful protocol analysis detection

Likewise the anomaly-based mechanism, the stateful protocol analysis identifies deviations of protocol state but it uses predetermined universal profiles based on "accepted definitions of benign activity" developed by vendors and industry leaders. To provide an example of stateful protocol analysis detection, in the case of monitoring requests with their corresponding response, every request should have a predictable response. The responses that fall outside of the expected results will be flagged and analyzed further.

The stateful protocol detection mechanism presents the same strengths and weaknesses with the anomaly-based mechanism. In particular, the strengths of the stateful protocol refer to identifying unexpected sequences of commands; adding stateful characteristics to regular protocol analysis, and implementing reasonableness checks thresholds for individual commands, e.g. min/max lengths (Scarfone & Mell, 2007).

Regarding weaknesses, the above detection mechanism is a resource intensive one that adds lots of overhead. It cannot detect attacks that do not violate the characteristics of generally accepted protocol behavior.

1.2 Single board computer

A single board computer is a printed circuit board (PCB), which contains integrated circuits (ICs) such as microprocessor, RAM and all of them incorporating a microcomputer. Single board computers can be employed to many diverse applications because of their flexibility. They can be customized to a specific application selecting among different hardware and software options. The preference for a single board computer instead of a custom designed microprocessor system is usually based on economic criteria and the mobility that can be provided by the size of the device. Generally speaking, despite the above positive characteristics, a single board computer cannot be so effective and powerful as a regular computer or server (Isikdag, 2015); (Johnston, et al., 2016).

1.3 Security Information Management (SIM) background

Security Information Management (SIM)[3] is a process mining approach. Process mining is the research area focused on the analysis of event log that aims to get knowledge about business processes (Kent & Souppaya, 2006).

In the field of Information Systems, especially within Information Security, SIM process mining can be applied in order to analyze log datasets generated by security sources such as Operating Systems, Firewalls, Intrusion Detection System (IDS), Intrusion Prevention Systems (IPS), communication devices, etc.

The Security Information Management (SIM) provides log management capabilities and in general is focused on real-time analysis and long-term storage respectively. According to Kent and Souppaya (2006), the main key functions of SIM include the following:

- Indexing: Uses special techniques to facilitate indexing and searching information on aggregated data. This is a vital task since data aggregation and retention imply huge amount of data to analyze.

[3] Security Information Management (SIM) is one of the components of the Security Information and Event Management (SIEM) software products and services, the other one being the Security Event Management (SEM).

- Retention: Historical data is stored in order to feed datasets for data correlation over the time. This is especially useful for forensics and incident response purposes.
- Correlation: Based on different data sources, a meaningful interpretation can be generated by linking events taking into account common parameters and attributes. Forensics analysis: Analysis of different datasets are correlated based on a timeline in order to reconstruct events that happened as part of security incidents.
- Compliance: Auditing techniques can be applied in order to assess whether specific procedures are followed within a system or network environment.
- Intelligence: By combining the aforementioned capabilities, a better, detailed and meaningful context can be described in order to feed a knowledge database for high-level interpretations suitable for decision making processes.

2 Hardware and Software

In order to build the proposed model, this research has been based on the hardware and software that are presented below. To be clarified, that the software used give emphasizes on the main packages that were installed for building the project, and does not analyze the prerequisites of the software packages.

2.1.1 Hardware

The hardware that has been used refers to the Raspberry Pi 3 Model B that has been selected to host the SOHO NIDS and the cloud server, which hosts the data analysis software tools.

2.1.1.1 Raspberry Pi

The Raspberry Pi is a credit card size fully featured computer capable to run different Linux operating systems, which has been evolved through several versions with feature variations in memory capacity and peripheral-device support. More precisely, the version that has been used in this research is the Raspberry Pi 3 Model B, which uses a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 Cache and 1GB RAM on board memory (Raspberry Pi Foundation, n.d.).

Among the different hardware features that the Raspberry Pi comes with, the most important ones are the Ethernet network adapter and the Wi-Fi 802.11n network adapter.

Concerning the disk capacity of Raspberry Pi 3 Model B, an 8GB class 10 SD network adapter has been selected, because there was no need for more disk capacity.

2.1.1.2 Cloud Server

For the storage, analysis and visualization of the generated data from the SOHO NIDS, a specific software, which is later analyzed, has been installed on a cloud server provided by Okeanos, which is a GRNET's cloud service provided for the Greek research and academic community (Okeanos GRNET SA, n.d.). The cloud server has been built with the following specifications: 1 CPU, 4GB RAM, 40GB HDD and 1 Public IPV4 address.

The abovementioned server and the specifications used have been selected not only because they are provided by Okeanos GRNET for free, but also because they can be used as a service for a wide range of home and small offices end users, and more, because they help to prototype the proposed model.

2.1.2 Software

In this research work, the software that has been used refers to the open source packages that were installed on Raspberry Pi 3 Model B and the cloud server.

2.1.2.1 BRO Network IDS

BRO (version 2.4.1) is a passive, open-source Unix-based network traffic analyzer. In essence, it is a powerful network analysis framework that can be used as a Network Intrusion Detection System (NIDS) for monitoring and analyzing network traffic looking for suspicious activity. BRO can also be used for collecting network measurements, conducting forensic investigations, traffic base lining and more. So, BRO supports a wide range of traffic analysis mechanisms even outside of the security domain, including performance measurements (The Bro Project, n.d.).

BRO is capable for anomaly-based, signature-based and stateful-protocol detection. It has an extensive set of log files that record a network's activity in high-level terms. These logs include not only a comprehensive record of every connection seen on the wire, but also application-layer transcripts such as, all HTTP sessions with their requested URLs, key headers, MIME types, and server responses, DNS requests with replies, SSL certificates, key content of SMTP sessions, and many more. By default, BRO writes all this information into well-structured tab-separated log files suitable for post-processing with external software (ibid.).

In addition to the logs, BRO comes with built-in functionality for a range of analysis and detection tasks, including extracting files from HTTP sessions, detecting malware by interfacing to external registries, reporting vulnerable versions of software seen on the network, identifying popular web applications, detecting SSH brute-forcing, validating SSL certificate chains, and much more. Moreover, BRO has its own specialized policy language that allows the creation of custom policies (ibid.).

Finally, BRO runs on commodity hardware and hence provides a low-cost alternative to expensive proprietary solutions and supports such standard functionality as well. BRO's scripting language facilitates a much broader spectrum of very different approaches to finding malicious activity, including semantic misuse detection, anomaly detection, and behavioral analysis.

BRO presents some other advantages, too. First, there are no particular minimum system requirements for BRO. However, the fastest memory and CPU core speed that can be afforded is recommended since all of the protocol parsing and most analysis will take place there. According to BRO developers, the rule of thumb that they have expressed is that BRO allocates approximately 1 core for every 250Mbps of traffic that is being analyzed. However, this estimate could be extremely traffic mix-specific. It has generally worked for mixed traffic with many users and servers. For example, if a network traffic peaks around 2Gbps (combined) and BRO needs to handle traffic at peak load, the host machine should have 8 cores available (2048 / 250 == 8.2). In the case of home and small office networks, which are the target audience of this project, a safe consumption would be between 50Mbps-1Gbps traffic peeks, which would need less than 1 core up to 4 cores (ibid.).

For all the above reasons, BRO is a perfectly suitable solution to run in high speed networks, without losing packets or slowing down the traffic, while running on commodity hardware with few resources.

2.1.2.1.1 BRO Architecture

First and foremost, while the analysis of other IDSs is packet oriented, BRO approach is more connection based. BRO is mainly separated into two major components, the event engine and policy script interpreter. The event engine narrows down the observed traffic into a series of higher-level events. These events describe what has been observed, but without interpreting them. Such semantics derive from BRO's second main element, the script interpreter, which executes a set of event handlers written in BRO's custom scripting language. These scripts can express an entity's security policy, for example what kind of actions to be taken when different types of activity are being detected, and then log any desired properties and statistics from the observed traffic (ibid.).

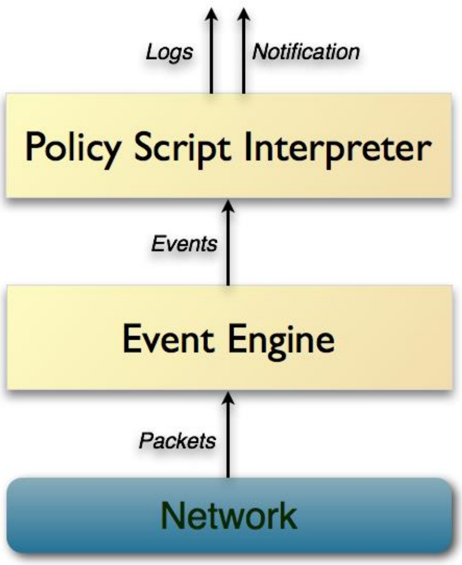


Figure 1: BRO architecture (The Bro Project, n.d.)

2.1.2.2 Critical Stack – Intel Feed

Intel Feed by Critical Stack (version 0.5.3), (hence Intel feature) is a free Intel marketplace for BRO providing online free feeds and blacklists for a variety of security parameters, such as blacklisted domains and IPs. This is delivered by Critical Stack for the BRO network security monitoring platform in order to add threat intelligence to the NIDS by continually updating the different online feeds (Critical Stack Team, n.d.). This added threat intelligence enhances BRO’s efficiency against internet threats, such as malicious URLs.

2.1.2.3 Raspbian Jessie Light

The operating system (OS) that has been installed in the Raspberry Pi 3is the Raspbian Jessie Light version 8.0 (hence Rasbian). The Raspian is a Debian-based computer operating system specifically developed for the Raspberry Pi and the Banana Pi.

The reason for selecting the Raspian OS is that the Raspberry Pi foundation officially provides the Raspian as the primary operating system for the family of the Raspberry Pi single-board computers and it is highly optimized for the Raspberry Pi line’s low performance ARM CPUs (Raspberry Pi Foundation, n.d.).

2.1.2.4 Ubuntu server

Ubuntu is a complete Linux distribution operating system based on Debian architecture, freely available with both community and professional support. The Ubuntu community is built on the ideas enshrined in the Ubuntu Manifesto (Ubuntu Team, n.d.): software should be available free of charge, software tools should be usable by people in their local language and despite any disabilities, as well as people should have the freedom to customize and alter their software in whatever way they see fit to them.

Ubuntu is suitable for both desktop and server use. More particularly, Ubuntu Server 16.04.3 Long Term Support (LTS) has been chosen to host the data analysis tools (Elastic stack) of this project is an operating system designed for dedicated servers (Ubuntu Team, n.d.).

2.1.2.5 Elastic stack

The Elastic stack is a collection of three open-source products: Elasticsearch, Logstash and Kibana, powered by Elastic.

- The Elasticsearch (version 5.6.2) is a NoSQL database based on the Lucene search engine capable to power extremely fast searches.
- Logstash (version 5.2.1) is a log pipeline tool that accepts inputs from various sources, executes different transformations and exports the data to various targets. To be added, Grok plug-in has been used for parsing the unstructured log data into structured and queryable ones. This tool is perfect for any log format that is generally written for humans and not computer consumption.
- Kibana (version 5.6.2) is a visualization layer that works on top of Elasticsearch.

All together, in sum, the above all three open source products are combined in order to provide a centralized logging and monitoring system. Logstash collects and parses the logs, which then are being sent to Elasticsearch that indexes and stores the information. After this step has been completed successfully, Kibana presents the data in visualizations and provides actionable insights into the network environment (Elastic Team, n.d.).

2.1.2.6 Host access point daemon (Hostapd)

The Hostapd (version 2.3) is a user space software capable of turning normal network interface network adapters into access points and authentication servers. Hostapd is designed to be a "daemon"[4] program that runs in the background and acts as the backend component controlling authentication. It implements IEEE 802.11 access point management, IEEE 802.1X/WPA/WPA2/EAP Authenticators and much more (Malinen, n.d.). In this thesis, Hostapd has been used with the specific aim to turn the normal network interface Wi-Fi network adapter into a Wi-Fi Access Point (AP).

[4] A daemon is a computer program that runs as a background process, rather than being under the direct control of an interactive user.

2.1.2.7 Dnsmasq

Dnsmasq (version 2.76) provides network infrastructure for small networks: DNS, DHCP, router advertisement and network boot. It is designed to be lightweight and have a small footprint, suitable for resource constrained routers and firewalls. It is also widely used for tethering on smartphones and portable hotspots, and to support virtual networking in virtualization frameworks. Dnsmasq is included in most Linux distributions and provides full IPv6 support (Kelley, n.d.). This research has used the Dnsmasq in order for Raspberry Pi 3 to provide DNS and DHCP services to the network devices connected to the Wi-Fi AP.

2.1.2.8 Iptables

Iptables (version 1.4.21) is a user-space utility program that allows a system administrator to configure the tables provided by the Linux kernel firewall and the chains and rules it stores (Imes, n.d.). The iptables have been used in this research work for postrouting and masquerade of IPv4 addresses.

Chapter 3

1 Architecture

1.1 Introduction

This chapter outlines the overall architecture of the proposed security model SOHO NIDS, and more specifically the topology of the implementation, as well as some key functions of the solution.

1.2 Scientific Approach

As already said, the objective of the present research work is the creation of a SOHO NIDS solution. In order to build this SOHO NIDS device, it is necessary all network traffic to be routed through the Raspberry Pi 3. In this way, the BRO can inspect the network packets of the incoming and outgoing network traffic and analyze the different network events.

To route all network traffic through the Raspberry Pi, I am taking advantage of the hardware capabilities of the Raspberry Pi 3 model B by utilizing the two pre-installed network adapters and avoiding any extra equipment, e.g. TAP[5]. The first of the two pre-installed network adapters, that is the Ethernet network adapter is used to connect Raspberry Pi 3 with the router with an Ethernet cable. The second pre-installed

[5]The network TAP is a mechanism that splits the packet stream in order to make a copy available for inspection. Examples include the monitoring port on a switch and an optical splitter on fiber networks.

network adapter, that is the Wi-Fi network adapter, is configured as a Wi-Fi Access Point (AP). The users connect their devices to the Wi-Fi AP of Raspberry Pi where BRO listens to this interface. BRO inspects the network traffic while this is being routed through the Ethernet network adapter to the router.

The analysis of BRO's created data takes place on the Elastic server which is a centralized logging and monitoring tool, capable to prepare, transform, evaluate and model the data by discovering useful information and present it in visual graphs in real time.

To be stressed, in this model the collaboration of the end user is mandatory. The end user needs to connect the network device to the Raspberry Pi 3 Wi-Fi AP by selecting the Pi's Basic Service Set Identifier (BSSID) and providing the correct passphrase. Additionally, the collaboration of the end user is deemed necessary in order to see the analyzed data through the browser. To be added, power to the Raspberry Pi can be supplied either by an adaptor plugged into a power supply or by a USB cable plugged into the USB port of the router.

1.3 Topology

The figure below (Figure 2) shows the topology of the scientific approach.

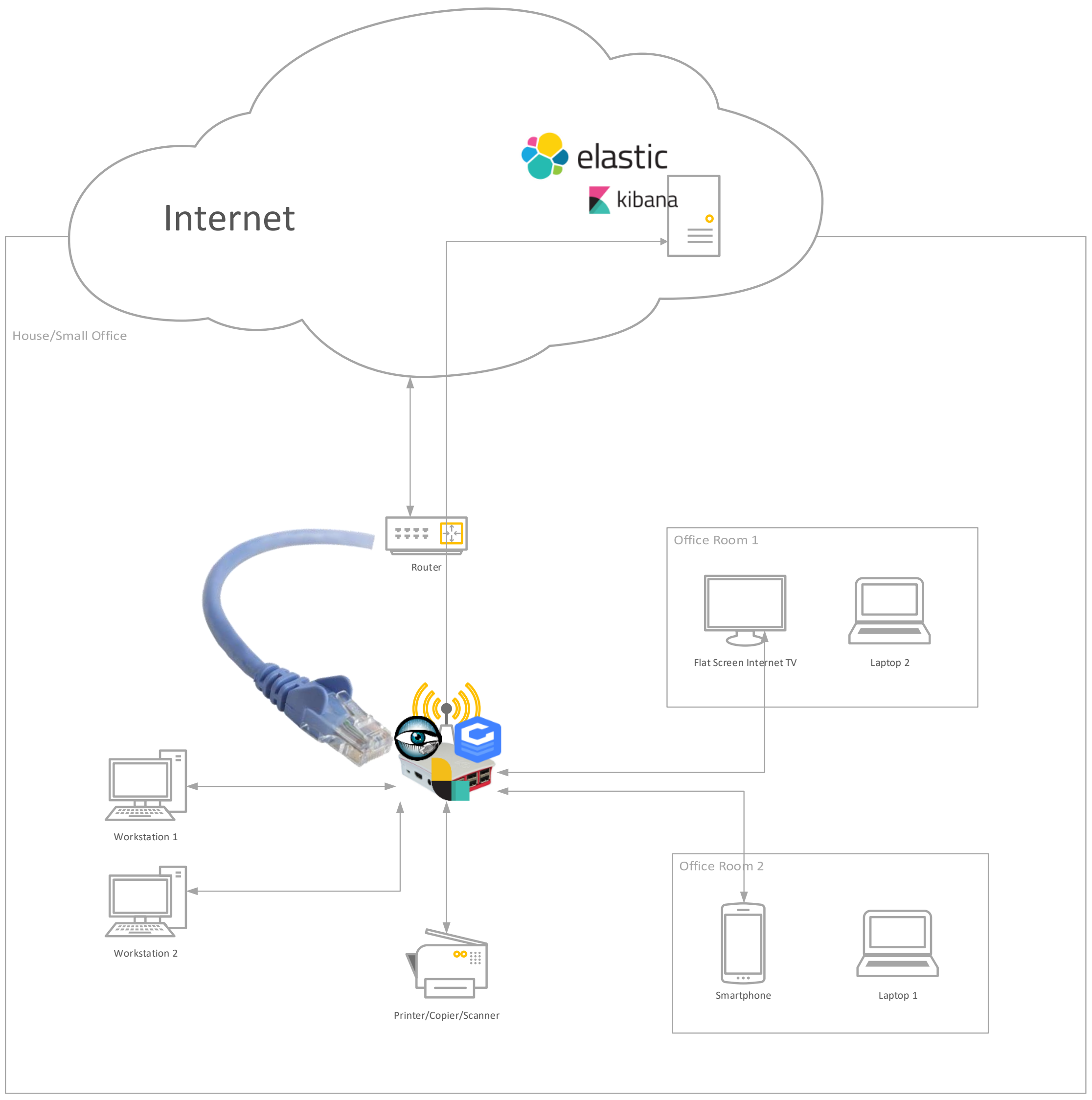


Figure 2: Topology of the proposed solution

For a better analysis and understanding, I have divided topology into two separate parts. The first part depicts how the inspection of the network traffic is being achieved, and the second part shows the process of parsing, forwarding and analysis of the logged data.

1.3.1 Inspection of network traffic

According to the displayed figure (Figure 2), the end user connects the network device to the Wi-Fi AP of Raspberry Pi. This done, an IP address of the subnet

172.24.1.0/24 is being automatically assigned to the connected network device. The network traffic is then routed to the Ethernet network adapter, which is connected through an Ethernet cable with the router, with the use of iptables in order to achieve IPV4 forwarding. The IP address of the subnet 172.24.1.0/24 is remapped using network address translation (NAT) to the private IP address that has been provided by the default gateway, that is the home router/access point. In this way, the Raspberry Pi acts like a bridge, a so called “man-in-the-middle” device. While the network packets are being transferred between those two interfaces, the BRO has been configured to listen to the Wi-Fi interface. As a result, BRO inspects and analyzes the network packets in real time and creates logs according to the deployed policies.

The above description demonstrates the route path of the network traffic and how BRO inspects it.

1.3.2 Parsing, forwarding and analysis of logs

The second part of the proposed solution provided in this thesis regards the reporting process, which refers to parsing, forwarding and analysis of logs and the ability of the administrator or the end user to have a full and clear overview of the occurred events that the BRO has detected.

As soon as the logs have been created by BRO and they have been stored locally on the system, they are parsed in real time by the Logstash tool. This tool is responsible to collect, parse, transform and send the logs to the cloud Elastic server.

BRO logs contain more information than it is really necessary for a specific event to be described. Moreover, this information may appear in the form of numerical codes which are not easily inteligible and interpretable by the end user. For this reason, the log entries are being dynamically transformed by Logstash either by removing completely specific fields or by replacing specific strings with more understandable ones and being transmitted to the cloud Elastic server. Moreover, a listener has been configured on the Elastic server capable to receive log entries from Raspberry Pi’s Logstash tool.

By the time logs reach their destination, the Elasticsearch tool indexes them and stores them on the server for further analysis. Finally, having successfully completed the previous step, the stored logs can be analyzed and displayed with Kibana in a variety of visual graphs that support multiple filters.

2 Demonstration of solution

To demonstrate the capabilities and the functionality of the proposed solution three use cases have been examined. To this end, the predefined rule sets that come with BRO were used with some extra ones (Critical Stack - Intel Feeds), since the main aim of the feasibility study was to define the functionality of the solution as a whole from the architecture point of view.

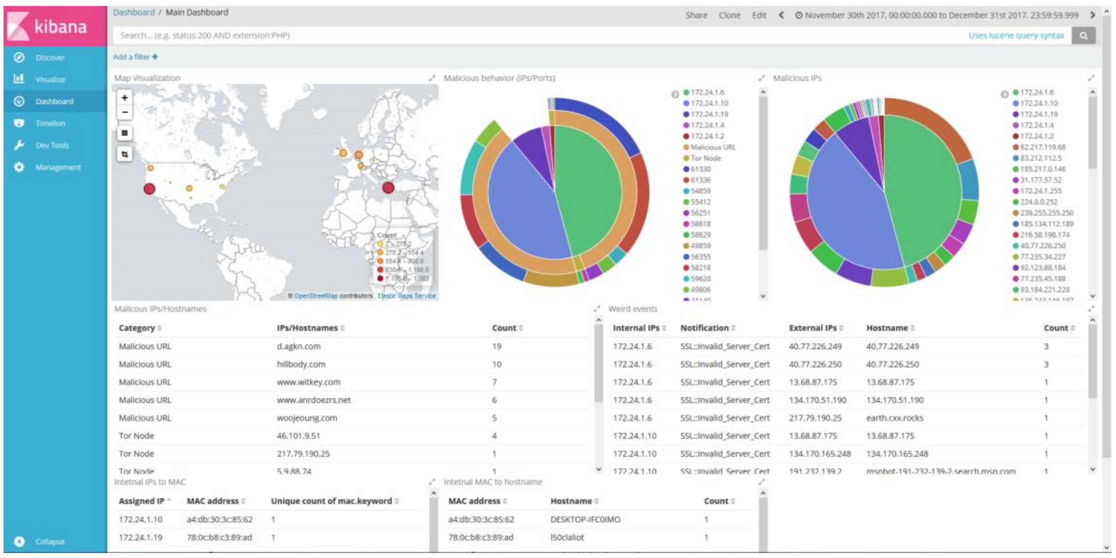
2.1 Demonstration of the proposed solution

The end user is able to view preconfigured dashboards and graphs of the analyzed network events, or create new ones, by connecting to the Elastic management web console. The management console is hosted to the Elastic server and it is made accessible to the end user by using a browser through the HTTP protocol.

For all three use case scenarios multiple network devices have been connected to Raspberry Pi 3, including workstations, laptops and mobile devices.

2.1.1 Main dashboard

The main dashboard of the Elastic infrastructure is presented below (Figure 3) that manages all the recorded and uploaded logs from Raspberry Pi 3. The analysis and presentation of the events take place in real time with minor delay.



Malicous IPs/Hostnames

Category	IPs/Hostnames	Count
Malicious URL	d.agkn.com	19
Malicious URL	hillbody.com	10
Malicious URL	www.witkey.com	7
Malicious URL	www.anrdoezrs.net	6
Malicious URL	woojeoung.com	5
Tor Node	46.101.9.51	4
Tor Node	217.79.190.25	1
Tor Node	5.9.88.74	1

Weird events

Internal IPs	Notification	External IPs	Hostname	Count
172.24.1.6	SSL::Invalid_Server_Cert	40.77.226.249	40.77.226.249	3
172.24.1.6	SSL::Invalid_Server_Cert	40.77.226.250	40.77.226.250	3
172.24.1.6	SSL::Invalid_Server_Cert	13.68.87.175	13.68.87.175	1
172.24.1.6	SSL::Invalid_Server_Cert	134.170.51.190	134.170.51.190	1
172.24.1.6	SSL::Invalid_Server_Cert	217.79.190.25	earth.cxx.rocks	1
172.24.1.10	SSL::Invalid_Server_Cert	13.68.87.175	13.68.87.175	1
172.24.1.10	SSL::Invalid_Server_Cert	134.170.165.248	134.170.165.248	1
172.24.1.10	SSL::Invalid_Server_Cert	191.232.139.2	msnbot-191-232-139-2.search.msn.com	1

Intetnal IPs to MAC

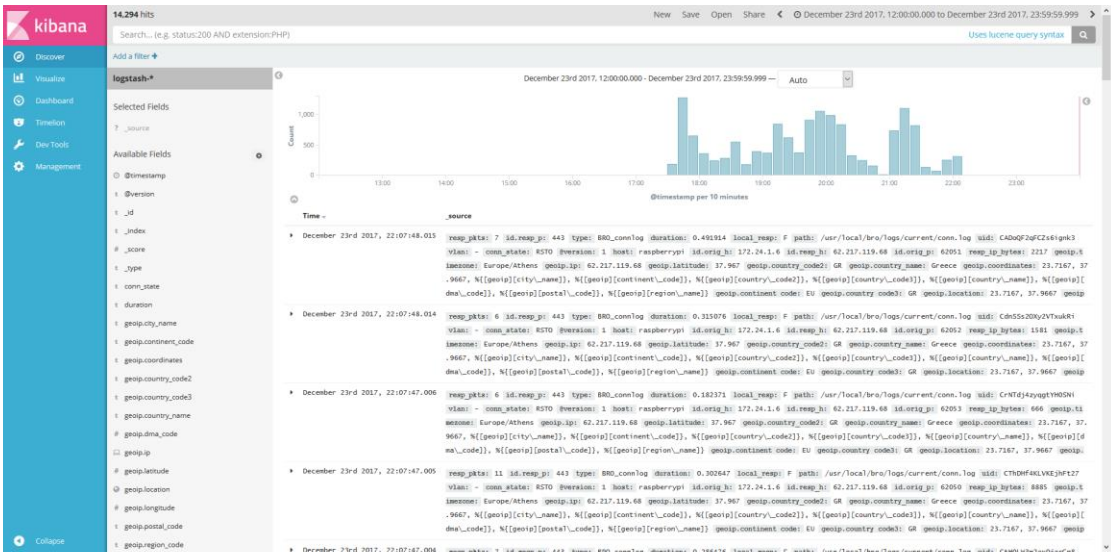
Assigned IP	MAC address	Unique count of mac.keyword
172.24.1.10	a4:db:30:3c:85:62	1
172.24.1.19	78:0c:b8:c3:89:ad	1

Intetnal MAC to hostname

MAC address	Hostname	Count
a4:db:30:3c:85:62	DESKTOP-IFC0IMO	1
78:0c:b8:c3:89:ad	l50claliot	1

Figure 3: Main dashboard #1

As said, the above figure (Figure 3) presents one of the main dashboards that has been created in order to display an adequate number of data. At the top left corner a world map is presented that depicts the geographical location of the recorded external IPs of the related events. The central pie depicts the internal IPs, the network ports of malicious behavior per IP, including the type of event. At the top right corner another pie chart is depicted, which indicates the internal and external IPs in pair for malicious or suspicious incidents. Further information can be extracted from the bottom datagrams that present the MAC addresses, hostnames and internal IPs of the connected devices. Finally, from the central datagrams the hostnames, if they exist, including malicious sites, of the external IPs can be identified or other suspicious activities that have been detected, such as an invalid server certificate and much more.



```
December 23rd 2017, 22:07:48.015   resp_pkts: 7 id.resp_p: 443 type: BRO_connlog duration: 0.491914 local_resp: F path: /usr/local/bro/logs/current/conn.log uid: CADoQF2qFCZs6ignk3
                                   vlan: - conn_state: RSTO @version: 1 host: raspberrypi id.orig_h: 172.24.1.6 id.resp_h: 62.217.119.68 id.orig_p: 62051 resp_ip_bytes: 2217 geoip.t
                                   imezone: Europe/Athens geoip.ip: 62.217.119.68 geoip.latitude: 37.967 geoip.country_code2: GR geoip.country_name: Greece geoip.coordinates: 23.7167, 37
                                   .9667, %{[geoip][city\_name]}, %{[geoip][continent\_code]}, %{[geoip][country\_code2]}, %{[geoip][country\_code3]}, %{[geoip][country\_name]}, %{[geoip][
                                   dma\_code]}, %{[geoip][postal\_code]}, %{[geoip][region\_name]} geoip.continent code: EU geoip.country code3: GR geoip.location: 23.7167, 37.9667 geoip

December 23rd 2017, 22:07:48.014   resp_pkts: 6 id.resp_p: 443 type: BRO_connlog duration: 0.315076 local_resp: F path: /usr/local/bro/logs/current/conn.log uid: Cdn5Ss20Xy2VTxukRi
                                   vlan: - conn_state: RSTO @version: 1 host: raspberrypi id.orig_h: 172.24.1.6 id.resp_h: 62.217.119.68 id.orig_p: 62052 resp_ip_bytes: 1581 geoip.t
                                   imezone: Europe/Athens geoip.ip: 62.217.119.68 geoip.latitude: 37.967 geoip.country_code2: GR geoip.country_name: Greece geoip.coordinates: 23.7167, 37
                                   .9667, %{[geoip][city\_name]}, %{[geoip][continent\_code]}, %{[geoip][country\_code2]}, %{[geoip][country\_code3]}, %{[geoip][country\_name]}, %{[geoip][
                                   dma\_code]}, %{[geoip][postal\_code]}, %{[geoip][region\_name]} geoip.continent code: EU geoip.country code3: GR geoip.location: 23.7167, 37.9667 geoip

December 23rd 2017, 22:07:47.006   resp_pkts: 6 id.resp_p: 443 type: BRO_connlog duration: 0.182371 local_resp: F path: /usr/local/bro/logs/current/conn.log uid: CrNTdj4zyqgtYHOSNi
                                   vlan: - conn_state: RSTO @version: 1 host: raspberrypi id.orig_h: 172.24.1.6 id.resp_h: 62.217.119.68 id.orig_p: 62053 resp_ip_bytes: 666 geoip.ti
                                   mezone: Europe/Athens geoip.ip: 62.217.119.68 geoip.latitude: 37.967 geoip.country_code2: GR geoip.country_name: Greece geoip.coordinates: 23.7167, 37.
                                   9667, %{[geoip][city\_name]}, %{[geoip][continent\_code]}, %{[geoip][country\_code2]}, %{[geoip][country\_code3]}, %{[geoip][country\_name]}, %{[geoip][d
                                   ma\_code]}, %{[geoip][postal\_code]}, %{[geoip][region\_name]} geoip.continent code: EU geoip.country code3: GR geoip.location: 23.7167, 37.9667 geoip.

December 23rd 2017, 22:07:47.005   resp_pkts: 11 id.resp_p: 443 type: BRO_connlog duration: 0.302647 local_resp: F path: /usr/local/bro/logs/current/conn.log uid: CThDHf4KLVKEjhFt27
                                   vlan: - conn_state: RSTO @version: 1 host: raspberrypi id.orig_h: 172.24.1.6 id.resp_h: 62.217.119.68 id.orig_p: 62050 resp_ip_bytes: 8885 geoip.t
                                   imezone: Europe/Athens geoip.ip: 62.217.119.68 geoip.latitude: 37.967 geoip.country_code2: GR geoip.country_name: Greece geoip.coordinates: 23.7167, 37
                                   .9667, %{[geoip][city\_name]}, %{[geoip][continent\_code]}, %{[geoip][country\_code2]}, %{[geoip][country\_code3]}, %{[geoip][country\_name]}, %{[geoip][
                                   dma\_code]}, %{[geoip][postal\_code]}, %{[geoip][region\_name]} geoip.continent code: EU geoip.country code3: GR geoip.location: 23.7167, 37.9667 geoip

December 23rd 2017, 22:07:47.004
```

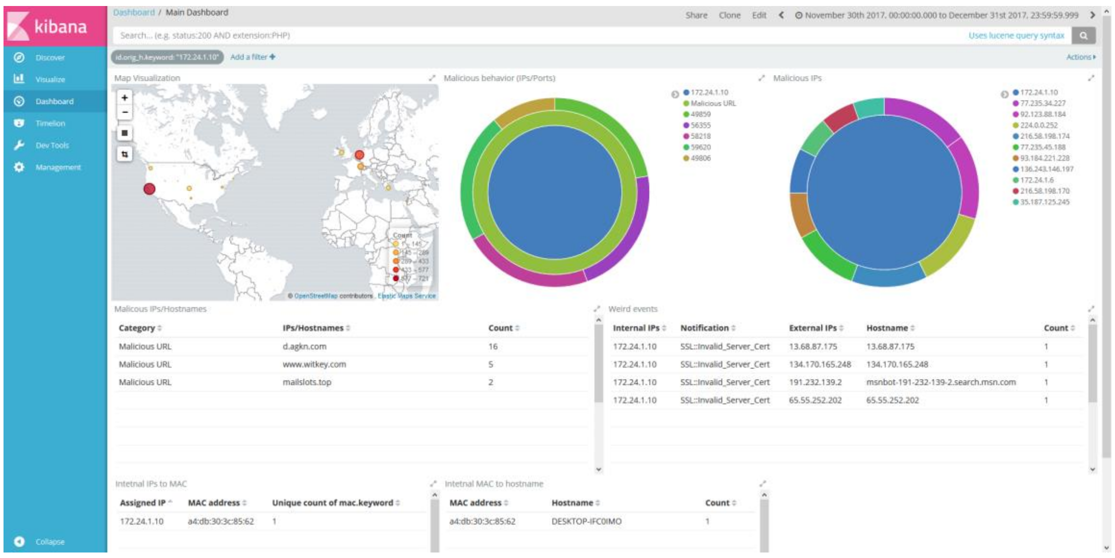
Figure 4: Main dashboard #2

Figure 4 shows another main dashboard, which displays the histogram of the uploaded logs on the top. Under this histogram, the entire information of each log can be seen.

To be noted, that many other features and dashboards can be studied and presented regarding all the different populated BRO logs, but they are out of the scope of the present thesis.

2.1.2 1st use case

The end user received a malicious URL link via email and the user connected to the malicious site. BRO detected immediately the specific incident and created the necessary logs, which were sent to the hosted Elastic server on the cloud.



Malicous IPs/Hostnames

Category	IPs/Hostnames	Count
Malicious URL	d.agkn.com	16
Malicious URL	www.witkey.com	5
Malicious URL	mailslots.top	2

Weird events

Internal IPs	Notification	External IPs	Hostname	Count
172.24.1.10	SSL::Invalid_Server_Cert	13.68.87.175	13.68.87.175	1
172.24.1.10	SSL::Invalid_Server_Cert	134.170.165.248	134.170.165.248	1
172.24.1.10	SSL::Invalid_Server_Cert	191.232.139.2	msnbot-191-232-139-2.search.msn.com	1
172.24.1.10	SSL::Invalid_Server_Cert	65.55.252.202	65.55.252.202	1

Intetnal IPs to MAC

Assigned IP	MAC address	Unique count of mac.keyword
172.24.1.10	a4:db:30:3c:85:62	1

Intetnal MAC to hostname

MAC address	Hostname	Count
a4:db:30:3c:85:62	DESKTOP-IFC0IMO	1

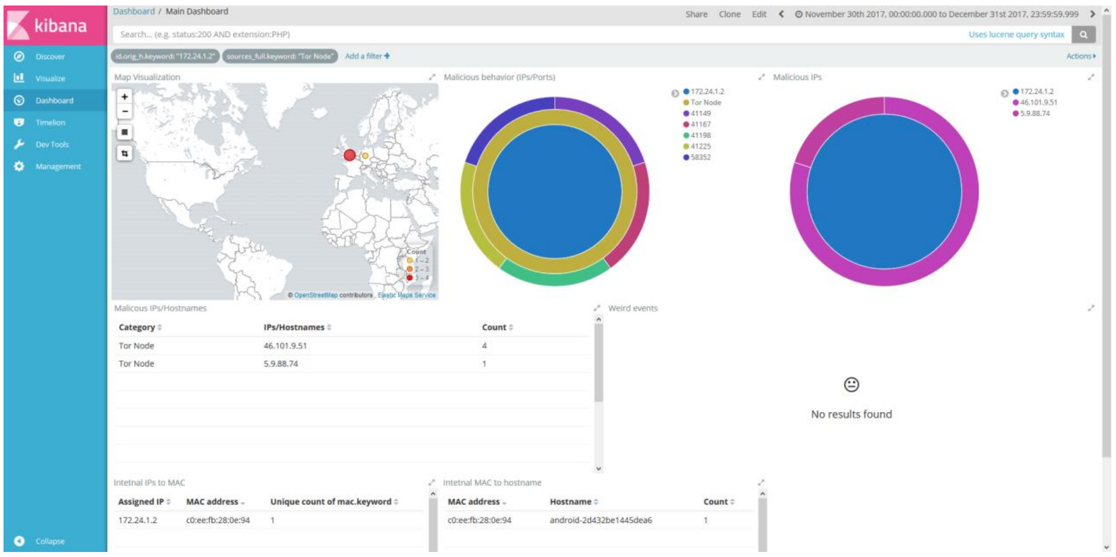
Figure 5: 1st use case dashboard

The administrator can immediately identify the threat for the internal IP address 172.24.1.10 with MAC address a4:db:30:3c:85:62 and hostname DESKTOP-IFC0IMO. The administrator can see that the end user has been connected to the malicious website “www.witkey.com”, because a “Malicious URL” incident has been populated for this IP address. Furthermore, as the above figure (Figure 5) shows, the IP address 172.24.1.10 has been isolated and, therefore, multiple incidents can be seen and further information can be extracted beyond the above described.

As already mentioned, the above figure (Figure 5) displays incidents related to the IP 172.24.1.10. To be more precise, at the top left corner a world map is depicted, which specifies the geographical location of the recorded external IPs of the related events. The central pie depicts the network ports of the malicious behavior for this IP. At the top right corner another pie chart exists, which depicts the external IPs that were recorded for malicious or suspicious incidents. Further information can be extracted from the datagrams that present the MAC address and the hostname of the connected device. Finally, from the central datagrams the hostnames, if they exist, including malicious sites, of the external IPs can be identified or other detected suspicious activities, such as an invalid server certificate (SSL: Invalid Server Certificate), and much more.

2.1.3 2nd use case

In this use case, the end user used the TOR[6] protocol. BRO detected this event including the TOR nodes that were used with the help of online feeds.


kibana
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
Collapse
Dashboard / Main Dashboard
Share
Clone
Edit
November 30th 2017, 00:00:00.000 to December 31st 2017, 23:59:59.999
Search... (e.g. status:200 AND extension:PHP)
Uses lucene query syntax
id.orig_h.keyword: "172.24.1.2"
sources_full.keyword: "Tor Node"
Add a filter
Actions
Map Visualization
Count
1 – 2
2 – 3
3 – 4
© OpenStreetMap contributors, Elastic Maps Service
Malicious behavior (IPs/Ports)
172.24.1.2
Tor Node
41149
41167
41198
41225
58352
Malicious IPs
172.24.1.2
46.101.9.51
5.9.88.74
Weird events
No results found

Malicous IPs/Hostnames

Category	IPs/Hostnames	Count
Tor Node	46.101.9.51	4
Tor Node	5.9.88.74	1

Intetnal IPs to MAC

Assigned IP	MAC address	Unique count of mac.keyword
172.24.1.2	c0:ee:fb:28:0e:94	1

Intetnal MAC to hostname

MAC address	Hostname	Count
c0:ee:fb:28:0e:94	android-2d432be1445dea6	1

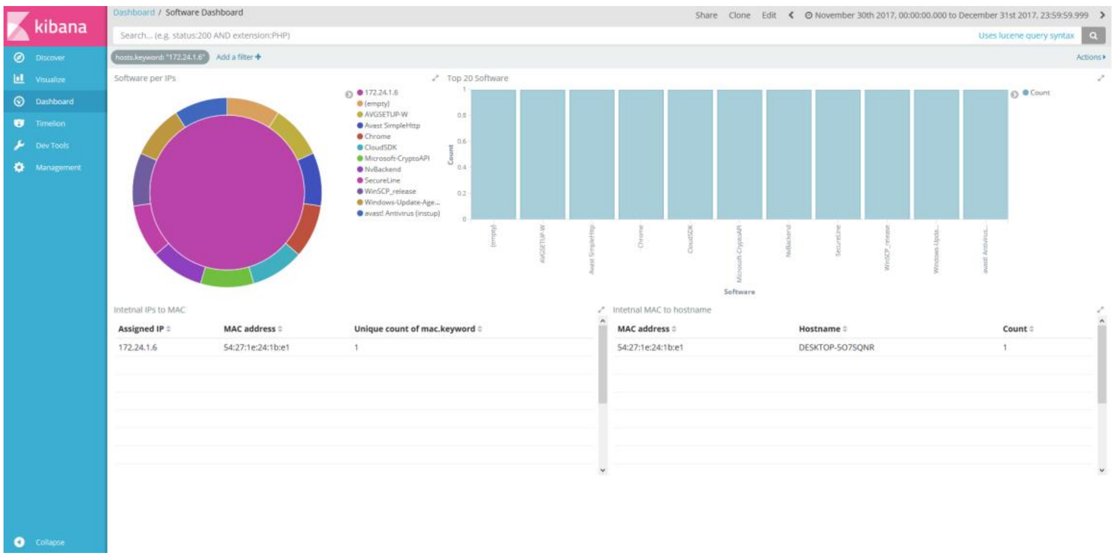
Figure 6: 2nd use case dashoard

The administrator can immediately identify the TOR protocol being used for the internal IP address 172.24.1.2 with MAC address c0:ee:fb:28:0e:94 and hostname android-2d432be1445dea6. The administrator sees that the end user used the TOR protocol for browsing the internet and hiding personal activities, because a "TOR Node" incident has been populated for this IP address. Furthermore, as the Figure 6 shows, the administrator can see the network ports, the TOR nodes with IPs 46.101.9.51 and 5.9.88.74 that have been used by the protocol as well as the geographical location of those IPs.

2.1.4 3rd use case

This use case presents the software that was detected by BRO while it was used by the end users or the connected devices for a specific period of time over the network.

[6] TOR is free software and an open network that helps users defend against traffic analysis, a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security.



Intetnal IPs to MAC

Assigned IP	MAC address	Unique count of mac.keyword
172.24.1.6	54:27:1e:24:1b:e1	1

Intetnal MAC to hostname

MAC address	Hostname	Count
54:27:1e:24:1b:e1	DESKTOP-5O75QNR	1

Figure 7: 3rd use case dashboard

As the above figure shows (Figure 7) demonstrates, the administrator can identify the software that was detected for the internal IP address 172.24.1.6 with MAC address 54:27:1e:24:1b:e1 and hostname DESKTOP-5O75QNR for one month period. The administrator sees among others that Chrome, AVG Antivirus, WinSCP_release, Avast! Antivirus and Windows-Update service were used by the end user or the connected device.

Chapter 4

1 Methodology

This chapter outlines the methodology used for conducting the experiments, the measuring tools deployed and the parameters and conditions set for these experiments.

This research focuses on the functionality of the proposed architecture and the solution as a whole covering system performance and network overhead of the SOHO NIDS. To be specified, that this thesis does not aim to analyze and optimize the open source software and the functions used nor to examine the range of operations that BRO attains.

1.1 Analysis of methodology

As already mentioned, this research has used the BRO software, because it offers a plethora of predefined rules capable to analyze the network activity in high-level terms. More specifically, BRO has predefined set of rules for a variety of different event analysis, as well as its own script language for creating even more custom rules. One of the strongest benefits of deploying BRO is the extensive set of log files that are being generated from analyzing the network traffic.

In order to verify the functionality of the proposed solution, it has been necessary to generate network traffic and observe, on the one hand, BRO’s impact on detecting and analyzing it, and on the other hand, the proposed solution’s impact on it. Moreover, it has been necessary to test if the Raspberry Pi 3 is able to run all the required software without problem.

In addition, the proposed architecture has been designed without the use of a TAP port. Instead, it utilizes the Wi-Fi network adapter as Access Point for enabling BRO to inspect the network traffic. Since the Raspberry becomes an extra node to the default routing path off the network traffic, it has been necessary to test the impact of the solution on network throughput.

To be made clear, focus have been placed on CPU, Memory load and Network throughput utilization for different scenarios of BRO and Logstash usage, which are the two main and most resource intensive software installed on Raspberry Pi 3. Regarding BRO, all default set of rules, including Critical Stack Intel Feeds, have been activated. During the experiments up to 5 network devices were connected while their end users were browsing the internet intensively.

In order to measure the experiments, I used certain measurement tools, as they are described in the following section. These tools have been selected because they are widely accepted by the scientific community considering them reliable. At the same time, these measurement tools have been considered as the most appropriate for the hardware and software I have used. Measurements have been made for the CPU and Memory utilization of BRO and Logstash; the CPU load of all processes; the Memory utilization of all processes; the idle CPU and the idle Memory usage. Memory includes RAM and SWAP area. To be noted, that the end results have derived from the combination of the above mentioned measurements.

It must be reminded that this thesis, does not examine specific network attacks, but attempts to build a solution that fully utilizes BRO capabilities, especially focused on default set of rules. So, measurements reflect only the average values of the results and they do not present the top and end values that were recorded, because those values are increasingly dynamic since they are directly depended on the type and size of traffic, the type and size of files, as well as the rules that are enabled and the type of the malicious behavior/attack under investigation.

Additionally, the Critical Stack Intel Feeds is utilizing 209242 indicators. According to Intel Critical Stack, this number of indicators may cause performance issues. So, the experiments have been performed in both modes, firstly, with this feature activated and secondly, deactivated.

To be added, that certain measurements were not possible. In particular, it was not possible to measure how many connected devices can be simultaneously supported by the Raspberry Pi 3 and by the proposed solution as a whole.

1.2 Measurement tools

For this research the following measurement tools have been used:

- The top tool, which displays processor and memory activity of tasks managed by kernel in real-time (Kirkland, n.d.). This tool was used in order to identify the CPU and Memory load of the running tasks.
- The mpstat tool, which is used to monitor specifically the CPU utilization on the system (Kirkland, n.d.). This tool was used in order to find out the CPU utilization for all running process and the idle CPU.
- The free tool, which displays the overall used and free memory (Kirkland, n.d.).
- The IPERF3 tool, which performs active measurements to determine the maximum achievable bandwidth on IP networks (iPerf3 Team, n.d.).

1.2.1 Measurement parameters

In this section, the parameters set for the commands are summarized.

1.2.1.1 Top

top –b –p pid >> filename.txt

- –b: Starts top in 'Batch mode', which could be useful for sending output from top to other programs or to a file
- –p pid: Monitor only processes with specified process ID(s).
- >> filename.txt: Appends output to a txt file with the specified name.

1.2.1.2 Mpstat

mpstat 1 >> filename.txt

- 1: Iteration of results every 1 second.
- >> filename.txt: Appends output to a txt file with the specified name.

1.2.1.3 Free

free –t –h –m –s 5 >> filename.txt

- –t: Display a line showing the column totals.
- –h: Show all output fields automatically scaled to shortest three digit unit and display the units of print out.
- –m: Display the amount of memory in megabytes.
- –s 5: Continuously display the result in 5 seconds apart.
- >> filename.txt: Appends output to a txt file with the specified name.

1.2.1.4 IPERF3

At this point has to be stressed, that good conventional speed tests are multi-threaded and create multiple connections to the speed test server using the UDP protocol, thus maxing the Internet Service Provider (ISP) connection to its full potential. The IPERF3 tool using the default options appears to only create two connections, which may not be enough to max out the bandwidth of an established connection, especially when congestion comes into play. For this reason, the IPERF3 results reflect the bandwidth of the established connection for the specific parameters and conditions that have been used for the experiments. Regarding bandwidth since it does not reflects reality, it is treated as throughput, as well. Below, the parameters used to run the IPEFR3 tool are presented:

(i) TCP

- Server: iperf3 –s
    - -s: Run in server mode.
- Client: iperf3 –c x.x.x.x –t 100 --logfile filename.txt
    - –c x.x.x.x: Run in client mode, connecting to an iperf3 server running on the defined IP.
    - –t 100: Transmit data for 100 seconds.
    - --logfile filename.txt: Send output to a log file.

(ii) UDP

- Server: iperf3 –s
    - –s: Run in server mode.
- Client: iperf3 –c x.x.x.x –u –t 100 –b 1G --logfile filename.txt
    - –c x.x.x.x: Run in client mode, connecting to an iperf3 server running on the defined IP.
    - –u: Use UDP rather than UDP.
    - –t 100: Transmit data for 100 seconds.
    - –b 1G: Set target bandwith to 1Gbps. Unlimited for TCP.
    - –t 100: Transmit data for 100 seconds.
    - --logfile filename.txt: Send output to a log file.

For the rest of the parameters that are not specified above the default values have been set.

1.3 Experiments

Experiments refer to system performance and network throughput utilization. All experiments which are described below have been conducted for an extensive period of time with each experiment lasting for 1-10 hours, and was repeated several times. For some experiments 5 network devices were connected while the end users were browsing the internet intensively.

1.3.1 System performance

System performance experiments have been conducted in three stages with the aim to determine if the Raspberry Pi 3 is able to run all the necessary software packages, as well as to determine if it is able to cope with the hardware resources of CPU and Memory utilization, specifically for BRO and Logstash, which are the more intensive software packages. To this end, the Linux top, free and mpstat tools were used for the measurements.

The first stage consists of a number of measurements where the Raspberry Pi 3 was up and running but without enabling any of the added services, such as the BRO. The objective of this stage has been to determine the result of CPU and Memory load while Raspberry Pi 3 is idle, (Figure 8).

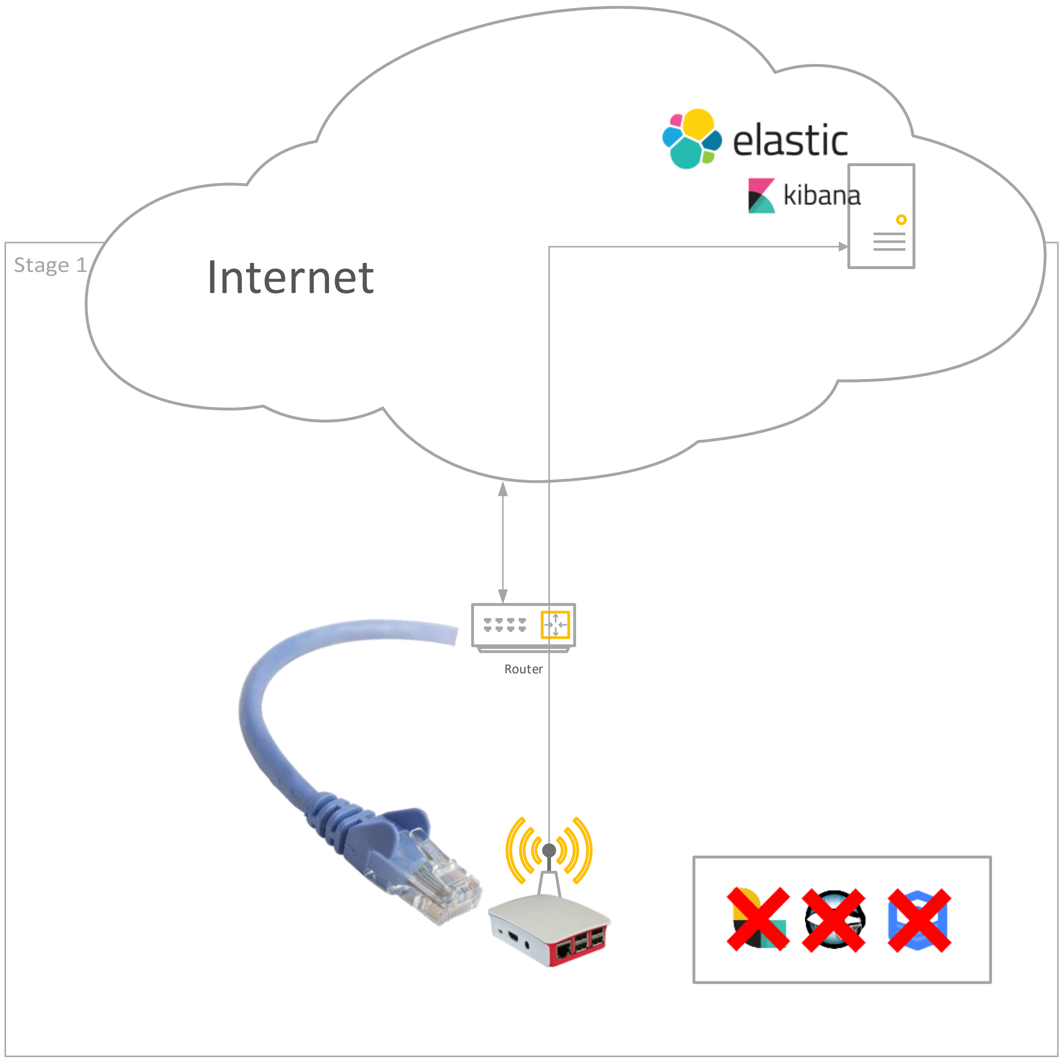


Figure 8: System performance experiment - stage 1

The second stage consists of a number of measurements split into two scenarios with the Raspberry Pi 3 up and running. In the first scenario, BRO and Logstash were enabled while Intel feature was disabled. In the second scenario, all features were enabled. During the measurements for both scenarios, none of the existed network devices were connected. The objective of this stage has been to determine the result of CPU and Memory load while the whole solution was enabled but idle, (Figure 9).

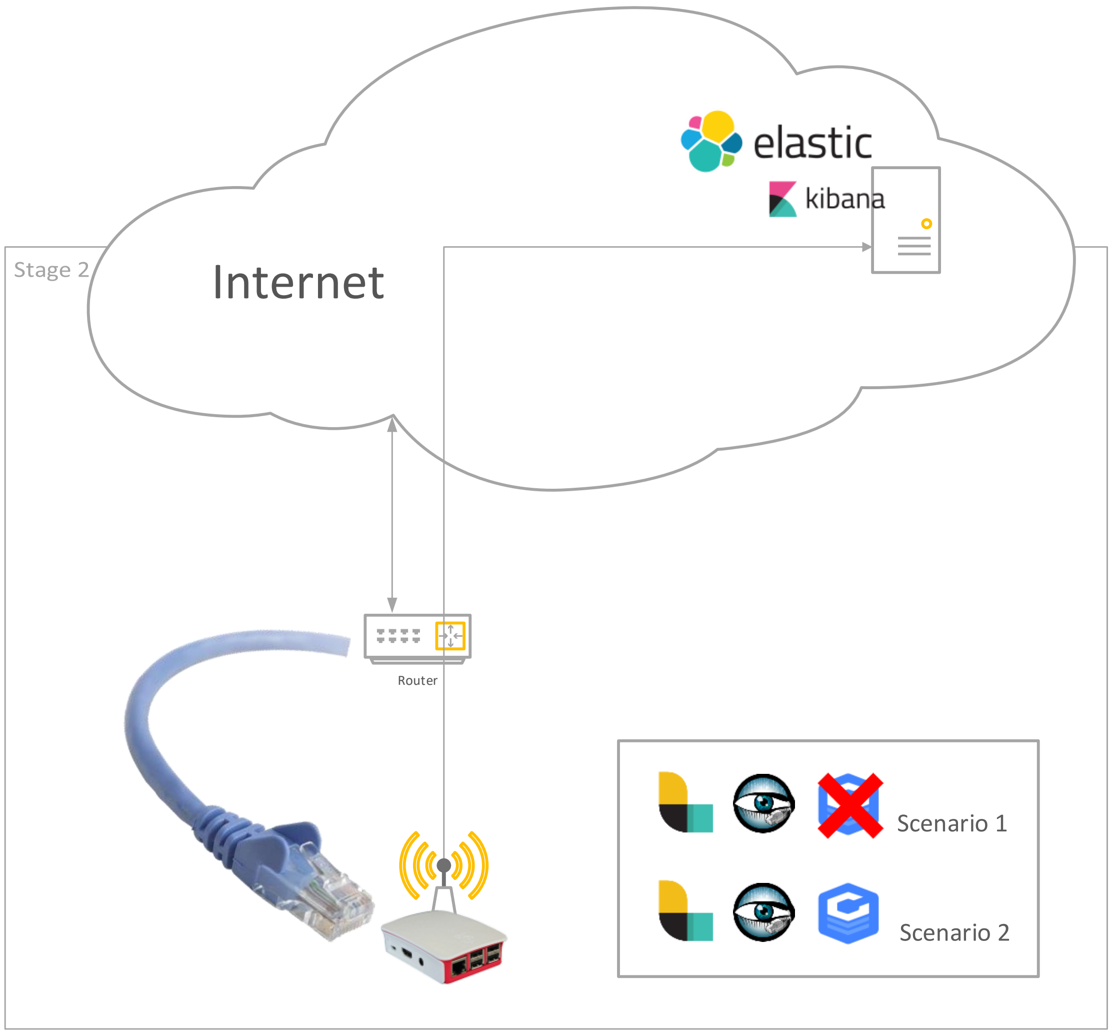


Figure 9: System performance experiment - stage 2

Finally, the third stage consists of a number of measurements split into two scenarios with the Raspberry Pi 3 up and running. In the first scenario, BRO and Logstash were enabled while Intel feature was disabled. In the second scenario, all features were enabled. During the measurements for both scenarios, 5 network devices were connected while the end users were browsing the internet intensively. The objective of this stage was to determine the results of CPU and Memory load while the whole solution was fully used by 5 network devices, (Figure 10).

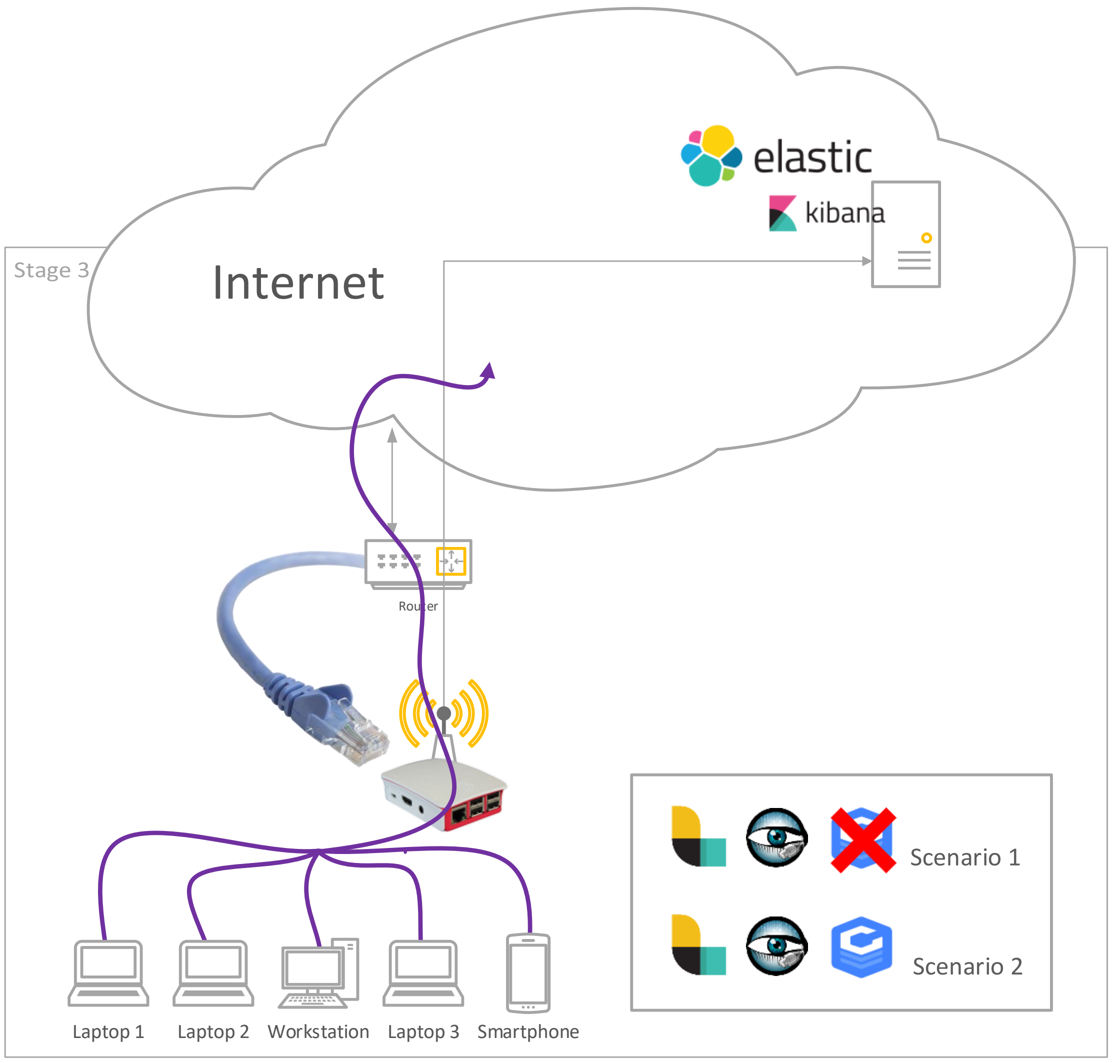
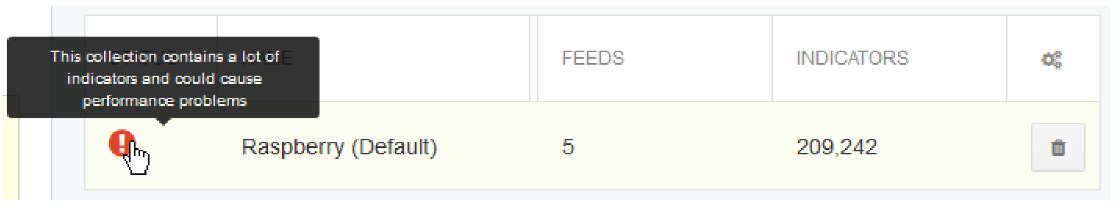


Figure 10: System performance experiment - stage 3

To be noted, that during all three stages the behavior and the functionality of the system with the Critical Stack Intel Feeds, firstly being enabled and secondly being disabled, were recorded in order to determine the impact of the indicators that were used for the proposed solution, (Figure 11).



This collection contains a lot of indicators and could cause performance problems

	NAME	FEEDS	INDICATORS	
	Raspberry (Default)	5	209,242	

Figure 11: Critical Stack Intel Feeds warning

1.3.2 Network throughput utilization

Network throughput utilization experiments have been conducted in four stages with the aim to determine the BRO impact on the network throughput, as well as the impact of the solution as a whole. This is necessary to be examined, because the proposed solution is an extra node to the default network traffic route. To this end, the IPERF3 tool has been used for the measurements.

During the first and the second stages of the experiment two laptops devices were used with 72.2 Mbps Wi-Fi network adapter bandwidth each of them and connected to the Raspberry Pi 3. These laptops were placed in a three meter distance far from the Raspberry Pi 3 and in line of site. The IPERF3 tool was installed in both laptops. This tool was used to determine the maximum achievable bandwidth of the LAN created by the Raspberry Pi 3. The measurements were made firstly with BRO and Logstash disabled and secondly with BRO and Logstash enabled. To be stressed, for the 1st stage of the experiment measurements have been made for TCP protocol[7], while for the 2nd stage of the same experiment measurements have been made for UDP protocol[8]. The objective of both stages was to determine if BRO has any impact on the network throughput for TCP and UDP protocols, (Figure 12).

[7] The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating by an IP network. TCP is optimized for accurate delivery rather than timely delivery and can incur relatively long delays (on the order of seconds) while waiting for out-of-order messages or re-transmissions of lost messages.

[8] The User Datagram Protocol (UDP) is one of the core members of the Internet protocol suite. UDP uses a simple connectionless communication model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. There is no guarantee of delivery, ordering, or duplicate protection.

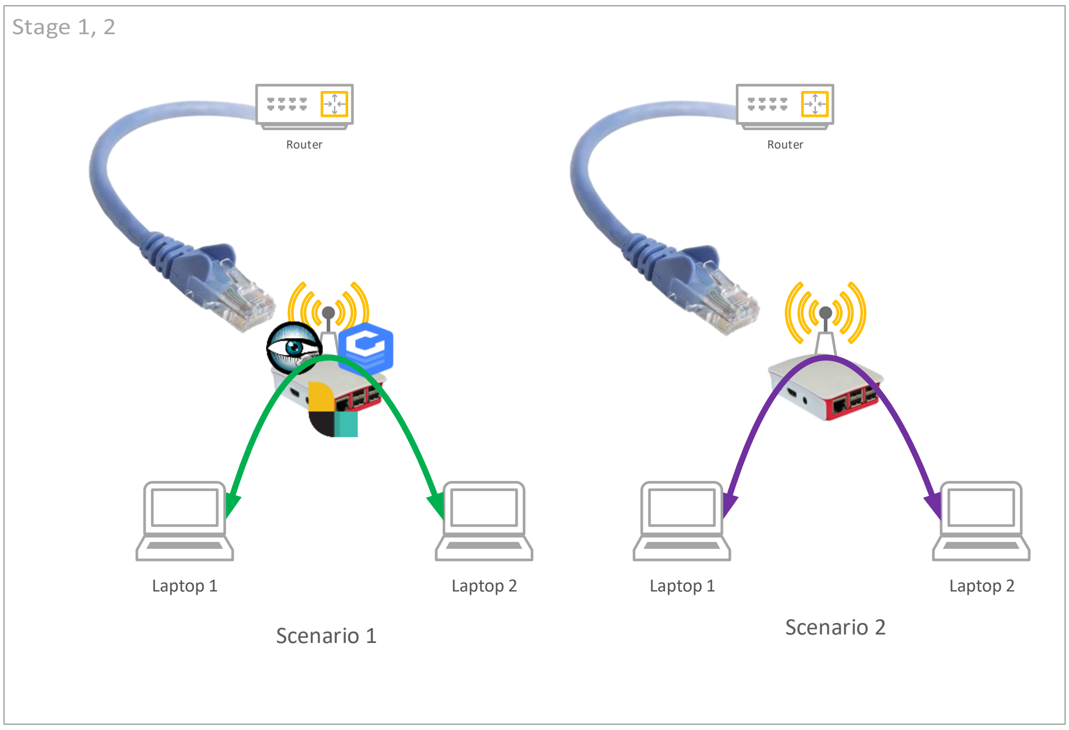


Figure 12: Network throughput utilization experiment - stage 1, 2

During the third and fourth stages of the experiment one laptop device has been used with 72.2 Mbps Wi-Fi network adapter bandwidth. The laptop has been placed in a three meter distance far from the Raspberry Pi 3 and the home router/access point, and in line of site. The IPERF3 tool has been installed on the laptop device and on the cloud server (Elastic stack). The measurements have been made firstly utilizing the default gateway that is the home router/access point, and secondly, utilizing the Raspberry Pi 3, with all features enabled. To be stressed, for the 3rd stage of the experiment measurements have been made for TCP protocol while for the 4th stage of the same experiment measurements have been made for UDP protocol. The objective of both stages was to determine the network throughput impact of the whole solution for TCP and UDP protocols, (Figure 13).

In order to ensure that measurements will be realistic, the necessary speed tests took place that showed that the ISP bandwidth used can be supported by the proposed architecture.

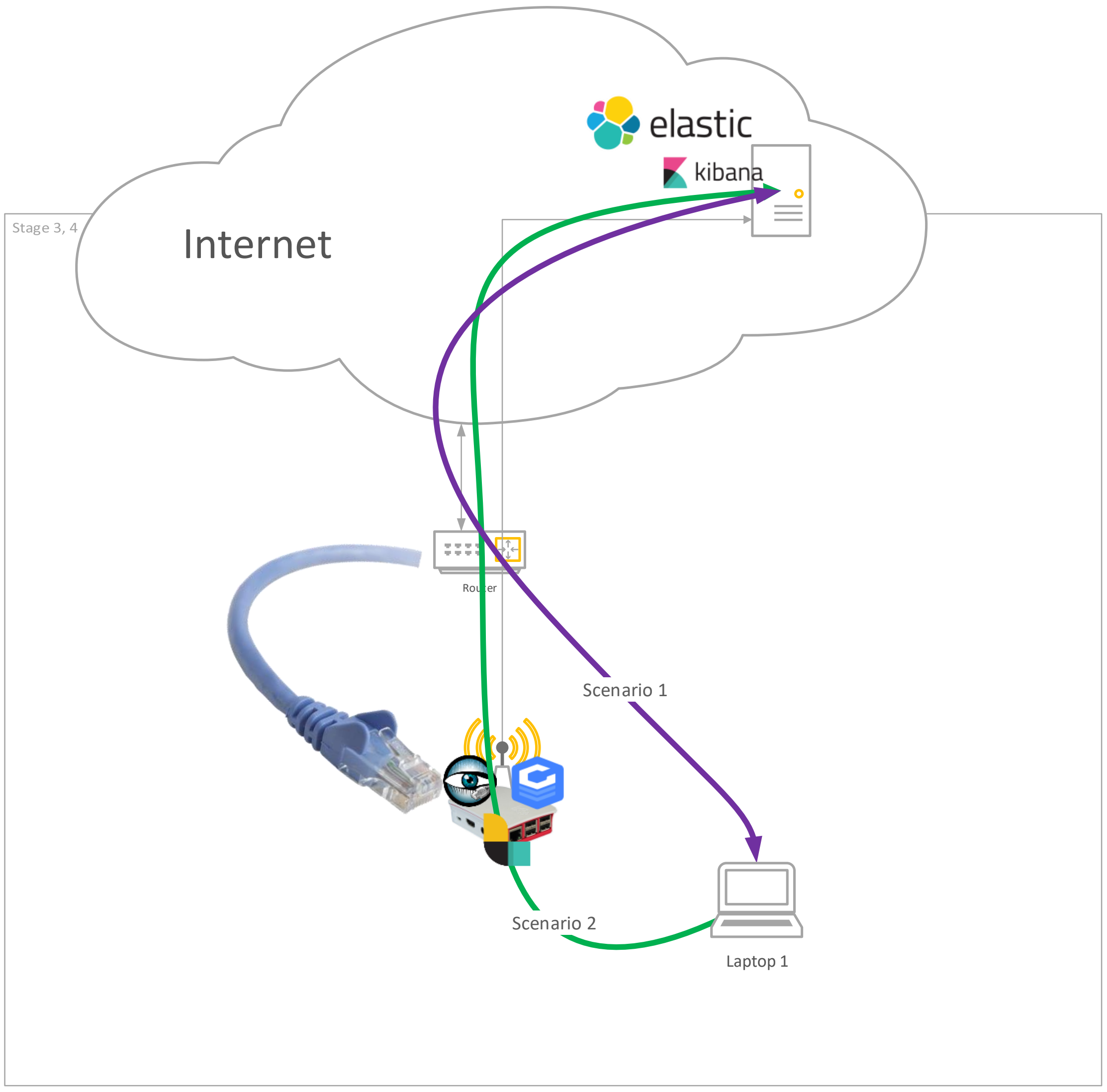


Figure 13: Network throughput utilization experiment - stage 3, 4

Chapter 5

1 Results

Presentation of results takes place into two parts. The first part provides measurements regarding the system performance experiments, and the second part, provides results referring to network throughput utilization experiments.

1.1 System Performance

The results of all different stages are presented in similar tables, where the CPU and Memory utilization is depicted in the form of percentage and in the form of the base unit of the reserved resource. Moreover, a graph that summarizes all information about results.

The tables show the CPU and Memory (RAM and SWAP area) utilization of the BRO and Logstash, which are the more intensive software packages that have been used, and of the entire system.

1.1.1 System performance: 1st stage

The first stage consists of a number of experiments in which the system was up and running in an idle state, with BRO and Logstash disabled, (Table 1).

Table 1: Pi running, all features disabled - stage 1

System performance idle			
Running tasks	Resources	Percentage	Base unit
BRO	CPU (4 cores)	N/A	N/A
	RAM (923MB)	N/A	N/A

	SWAP (99MB)	N/A	N/A
	Total MEM (1022MB)	N/A	N/A
Logstash	CPU (4 cores)	N/A	N/A
	RAM (923MB)	N/A	N/A
	SWAP (99MB)	N/A	N/A
	Total MEM (1022MB)	N/A	N/A
SYSTEM	CPU (4 cores)	00.16%	7.68 MHz
	RAM(923MB)	17.33%	160 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM(1022MB)	15.65%	160.00 MB
Overall	CPU (4 cores)	00.16%	07.68 MHz
	IDLE CPU (4 cores)	99.84%	4792.32 MHz
	RAM(923MB)	17.33%	160.00 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM(1022MB)	15.65%	160.00 MB
	FREE RAM (923MB)	82.67%	763.00 MB
	FREE SWAP (99MB)	100.00%	99.00 MB
	FREE Total MEM (1022MB)	84.35%	862.00 MB

According to Table 1, it is observed that the system in idle state utilizes 00.16% of CPU and 17.33% (160.00 MB) of Memory.

1.1.2 System performance: 2nd stage

The second stage consists of a number of experiments in which the system was up and running in an idle state, with BRO and Logstash enabled, without any network device being connected. At this stage, experiments have been made firstly with the Critical Stack Intel Feeds feature (hence Intel feature) disabled (Table 2), and secondly with this feature enabled, (Table 3).

Table 2: Pi running, BRO and Logstash enabled, Intel feature disabled – stage 2

Critical Stack Intel Feeds feature disabled			
Running tasks	Resources	Percentage	Base unit
BRO	CPU (4 cores)	11.10%	532.80 MHz
	RAM (923MB)	18.80%	173.90 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	17.02%	173.90 MB
Logstash	CPU (4 cores)	01.80%	86.40 MHz
	RAM (923MB)	27.50%	253.50 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	24.80%	253.50 MB
SYSTEM	CPU (4 cores)	00.16%	07.68 MHz
	RAM (923MB)	18.50%	170.60 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	16.70%	170.60 MB
Overall	CPU (4 cores)	13.06%	626.88 MHz
	IDLE CPU (4 cores)	86.94%	4173.12 MHz
	RAM (923MB)	64.79%	598.00 MB

	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	58.51%	598.00 MB
	FREE RAM (923MB)	35.21%	325.00 MB
	FREE SWAP (99MB)	100.00%	99.00 MB
	FREE Total MEM (1022MB)	41.49%	424.00 MB

Table 3: Pi running, BRO and Logstash enabled, Intel feature enabled - stage 2

Critical Stack Intel Feeds feature enabled			
Running tasks	Resources	Percentage	Base unit
BRO	CPU (4 cores)	11.10%	532.80 MHz
	RAM (923MB)	48.46%	447.30 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	43.77%	447.30 MB
Logstash	CPU (4 cores)	01.80%	86.40 MHz
	RAM (923MB)	28.46%	262.70 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	25.70%	262.70 MB
SYSTEM	CPU (4 cores)	00.17%	08.16 MHz
	RAM (923MB)	19.18%	177.00 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	17.32%	177.00 MB
Overall	CPU (4 cores)	13.07%	627.36 MHz
	IDLE CPU (4 cores)	86.93%	4172.64 MHz

	RAM (923MB)	96.10%	887.00 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	86.80%	887.00 MB
	FREE RAM (923MB)	03.90%	36.00 MB
	FREE SWAP (99MB)	100.00%	99.00 MB
	FREE Total MEM (1022MB)	13.20%	135.00 MB

According to Table 2, it is observed that the entire system in idle state with Intel feature disabled utilizes 13.06% of CPU and 58.51% of Memory, which corresponds to 598.00 MB. In particular, BRO utilizes 11.10% of CPU and 17.02% of Memory, which corresponds to 173.90 MB; Logstash utilizes 01.80 % of CPU and 24.08% of Memory, which corresponds to 253.50 MB.

On the other hand, according to Table 3, it is observed that the entire system in idle state with Intel feature enabled utilizes 13.07% of CPU and 86.80% of Memory, which corresponds to 887.00 MB. In particular, BRO utilizes 11.10% of CPU and 43.77% of Memory, which corresponds to 477.30 MB; Logstash utilizes 01.80 % of CPU and 25.07% of Memory, which corresponds to 262.70 MB.

1.1.3 System performance: 3rd stage

The third stage consists of a number of experiments in which the system was up and running with BRO and Logstash enabled. The proposed solution was running with 5 network devices connected while end users were browsing the internet intensively. At this stage, experiments have been made firstly with the Intel feature disabled (Table 4) and secondly with the Intel feature enabled (

Table 5).

Table 4: Pi running, BRO and Logstash enabled, Intel feature disabled - stage 3

Critical Stack Intel Feeds feature disabled			
Running tasks	Resources	Percentage	Base unit
BRO	CPU (4 cores)	11.60%	556.80 MHz
	RAM (923MB)	20.36%	187.90 MB

	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	18.39%	187.90 MB
Logstash	CPU (4 cores)	01.85%	88.80 MHz
	RAM (923MB)	27.60%	255.00 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	24.95%	255.00 MB
SYSTEM	CPU (4 cores)	~00.17%	08.16 MHz
	RAM (923MB)	18.54%	171.10 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	16.74%	171.10 MB
Overall	CPU (4 cores)	13.62%	653.76 MHz
	IDLE CPU (4 cores)	86.38%	4146.24 MHz
	RAM (923MB)	66.52%	614.00 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	66.01%	614.00 MB
	FREE RAM (923MB)	33.48%	309.00 MB
	FREE SWAP (99MB)	100.00%	99.00 MB
	FREE Total MEM (1022MB)	39.92%	408.00 MB

Table 5: Pi running, BRO and Logstash enabled, Intel feature enabled - stage 3

Critical Stack Intel Feeds feature enabled			
Running tasks	Resources	Percentage	Base unit
BRO	CPU (4 cores)	~11.60%	556.80 MHz

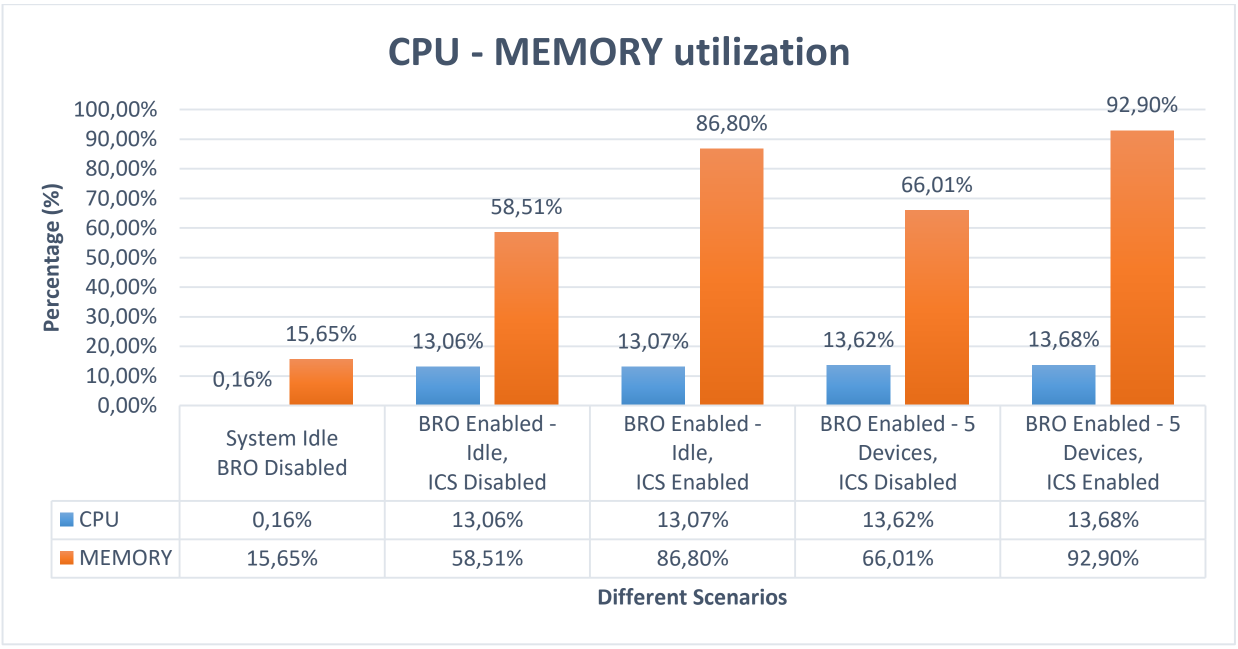
	RAM (923MB)	49.70%	458.80 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	44.89%	458.80 MB
Logstash	CPU (4 cores)	~1.90%	91.20 MHz
	RAM (923MB)	28.50%	263.40 MB
	SWAP (99MB)	00.00%	00.00 MB
	Total MEM (1022MB)	25.77%	263.40 MB
SYSTEM	CPU (4 cores)	~0.18%	08.64 MHz
	RAM (923MB)	21.60%	201.80 MB
	*SWAP (99MB)	*26.77%	*26.50MB
	Total MEM (1022MB)	22.34%	228.30 MB
Overall	CPU (4 cores)	13.68%	656.64 MHz
	IDLE CPU (4 cores)	86.32%	4143.36 MHz
	RAM (923MB)	100.00%	923.00 MB
	SWAP (99MB)	26.77%	26.50 MB
	Total MEM (1022MB)	92.90%	949.50 MB
	FREE RAM (923MB)	00.00%	00.00 MB
	FREE SWAP (99MB)	73.24%	72.50 MB
	FREE Total MEM (1022MB)	07.10%	72.50 MB

*SWAP area utilization has been included in the SYSTEM Memory load, because it could not be measured which exactly process utilizes it.

According to Table 4, it is observed that the entire system with Intel feature disabled utilizes 13.62% of CPU and 66.01% of Memory, which corresponds to 614.00 MB. In particular, BRO utilizes 11.60% of CPU and 18.39% of Memory, which corresponds to 187.90 MB; Logstash utilizes 01.85 % of CPU and 24.95% of Memory, which corresponds to 255.00 MB.

On the other hand, according to

Table 5, it is observed that the entire system in idle state with Intel feature enabled utilizes 13.68% of CPU and 92.90% of Memory, which corresponds to 949.50 MB. In particular, BRO utilizes 11.60% of CPU and 44.89% of Memory, which corresponds to 458.80 MB; Logstash utilizes 01.90 % of CPU and 25.77% of Memory, which corresponds to 263.40 MB.



	System Idle BRO Disabled	BRO Enabled - Idle, ICS Disabled	BRO Enabled - Idle, ICS Enabled	BRO Enabled - 5 Devices, ICS Disabled	BRO Enabled - 5 Devices, ICS Enabled
🟦 CPU	0,16%	13,06%	13,07%	13,62%	13,68%
🟧 MEMORY	15,65%	58,51%	86,80%	66,01%	92,90%

Figure 14: CPU–Memory utilization - All stages included

The above graph (Figure 14) shows the average percentage of overall CPU and Memory utilization of Raspberry Pi 3 for five different scenarios. The Y-axis shows the average percentage and the X-axis displays the results of the five different scenarios of the conducted tests. In the first scenario, while the Raspberry was powered on and in idle state the overall CPU load was 0.16% and the overall Memory load was 15.65%. In the second scenario, where the BRO was enabled and the Intel feature disabled, and without utilizing the solution, the overall CPU load was 13.06% and the overall Memory load was 58.51%. In the third scenario, where all features were enabled but the solution was remaining idle, the overall CPU load was 13.07% and the overall Memory was 86.80%. In the fourth scenario, the BRO was enabled with the Intel feature disabled, and 5 network devices were connected with the end users browsing the internet intensively. In that scenario, the overall CPU load was 13.62% and Memory load was 66.01%. Finally, in the fifth scenario where all features were enabled and the solution was utilized by 5 network devices connected with the end users browsing the internet intensively, the overall CPU load was 13.68% and the overall Memory load was 92.90%.

1.1.4 Discussion of Results

From the above data, I conclude that Raspberry Pi 3 is able to operate for all examined scenarios. It must be mentioned that the second scenario of the third stage reveals that when BRO and Logstash were enabled, including the Intel feature, the Raspberry Pi's 3 Memory was running at its limits. As a result, Memory is a critical factor for establishing BRO and Logstash with the Intel feature enabled. However, the CPU load was always in low levels. This confirms the BRO developers' rule of thumb that 4 Cores should be enough to handle the network traffic for SOHO (50Mbps-1Gbps bandwidth).

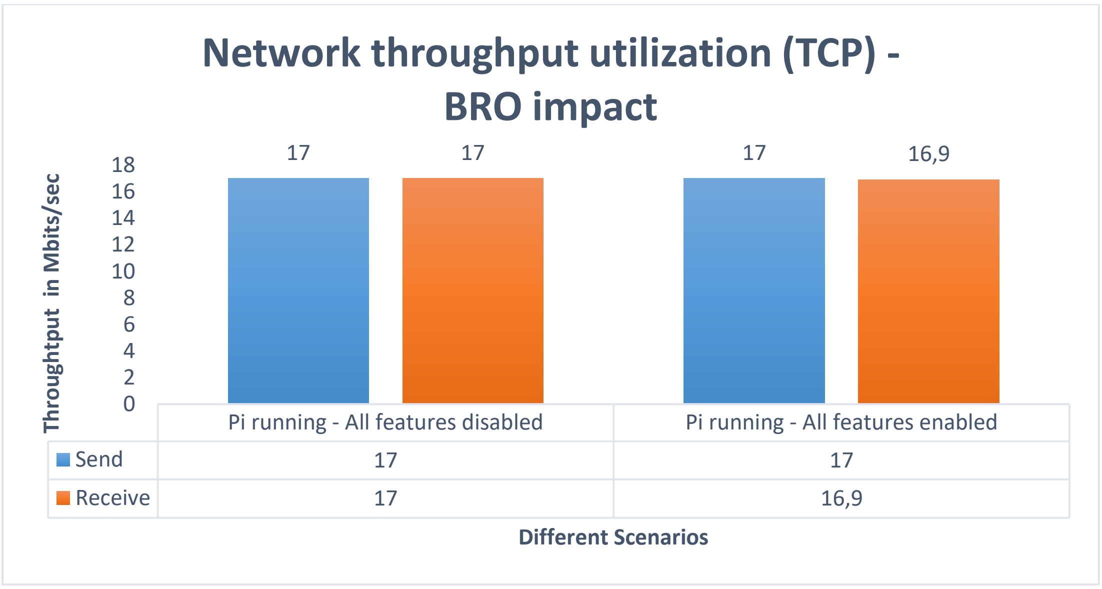
More specifically, regarding Memory utilization, it was observed that Raspberry Pi's 3 Memory for running BRO with Intel feature enabled and 5 devices connected, it was ranged approximately at 50% at all times while the CPU load ranged around 12%. This shows that Raspberry Pi's 3 resources are able to support the operation of BRO. Moreover, the Logstash software that is responsible for parsing and forwarding the logs to the cloud server was utilizing almost 25% of Memory. This demonstrates that Logstash, compared to BRO, is utilizing disproportionate amount of memory for the purpose it performs. As a result, for a realistic scenario, the Logstash software needs to be either optimized or developed or replaced in order to ensure that Raspberry Pi's 3 Memory will not run at its limits at all times.

1.2 Network Throughput utilization

The results of the network throughput utilization experiments are presented in graphs for both TCP and UDP protocols. The throughput utilization is depicted in the form of base unit, Mbps. Additionally, for UDP protocol the "lost datagrams" and the "delay jitter"[9] have been measured in the form of percentage (%) and milliseconds (ms) respectively.

The network throughput utilization experiments were conducted with the IPERF3 tool in four stages. These experiments have been conducted with the aim to determine the BRO impact and the impact of the solution as a whole on the network throughput for TCP and UDP protocols.

1.2.1 BRO impact (TCP): 1st stage



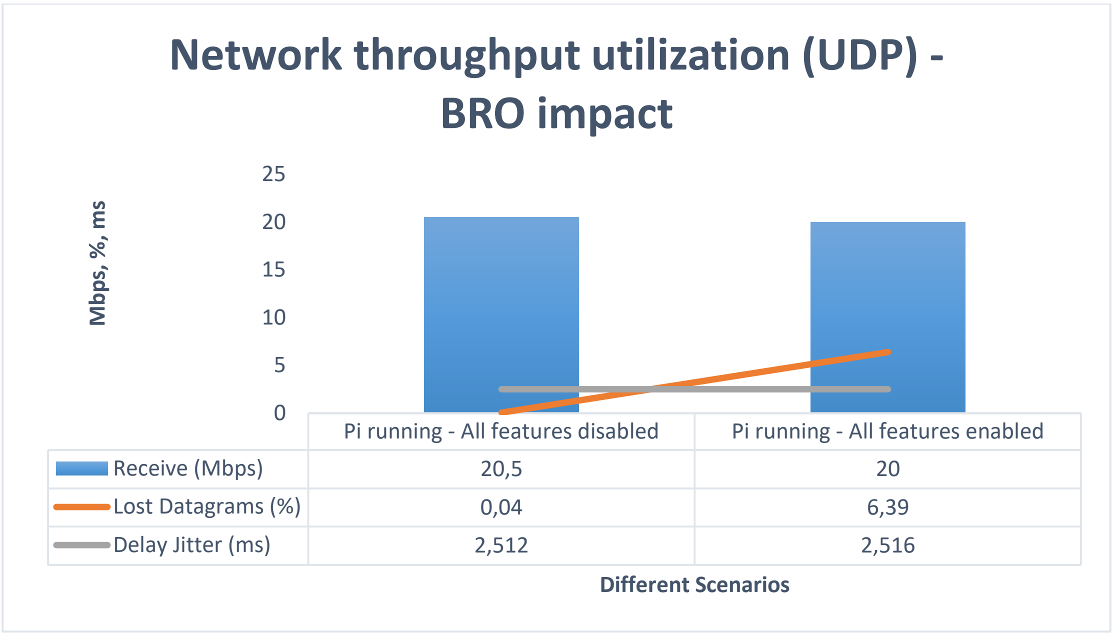
	Pi running - All features disabled	Pi running - All features enabled
🟦 Send	17	17
🟧 Receive	17	16,9

Figure 15: Network throughput utilization (TCP) - BRO impact

The above graph (Figure 15) shows the average network throughput utilization for TCP protocol in Mbps between two laptops, with the Raspberry Pi 3 running firstly with all the features disabled, and secondly with all features enabled. The Y-axis shows the throughput measured in Mbps and the X-axis displays the two different scenarios of the conducted experiments. In the first scenario, Raspberry Pi 3 was running with all features disabled and the average throughput was 17.0 Mbps for sending and receiving data. In the second scenario, Raspberry was running with all features enabled and the average throughput for sending and receiving data was 17.0 Mbps and 16,9 Mbps respectively.

[9] In computer networking, packet delay variation (PDV), also known as delay jitter, is the difference in end-to-end one-way delay between selected packets in a flow with any lost packets being ignored.

1.2.2 BRO impact (UDP): 2nd stage

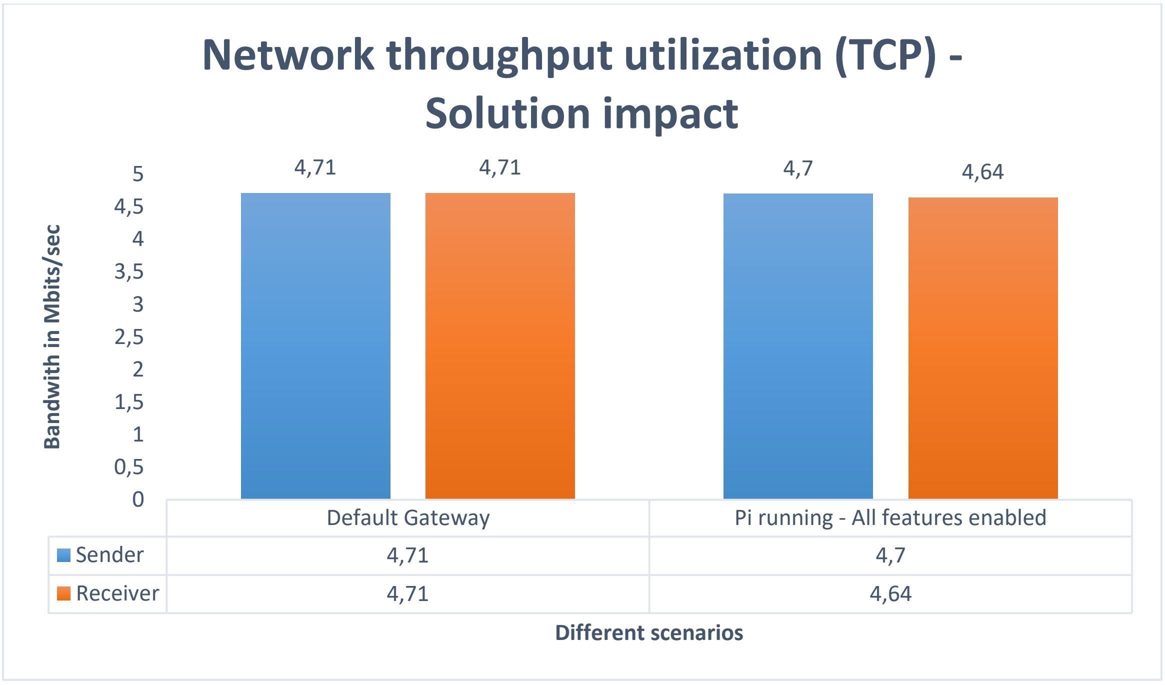


	Pi running - All features disabled	Pi running - All features enabled
🟦 Receive (Mbps)	20,5	20
🟧 Lost Datagrams (%)	0,04	6,39
⬜ Delay Jitter (ms)	2,512	2,516

Figure 16: Network throughput utilization (UDP) - BRO impact

The above graph (Figure 16) shows the average network throughput utilization for UDP protocol in Mbps, the lost datagrams in the form of percentage (%) and the delay jitter in milliseconds (ms) between two laptops, with the Raspberry Pi 3 running firstly with all the features disabled, and secondly with all features enabled. The Y-axis shows the throughput measured in Mbps, the lost datagrams measured in the form of percentage (%) and the delay jitter in ms. The X-axis displays the two different scenarios of the conducted experiments. In the first scenario, Raspberry was running with all features disabled and the average throughput was 20.5 Mbps for receiving data. The lost datagrams were approximately 4% and the delay jitter was 2.512 ms. In the second scenario, Raspberry was running with all features enabled and the average throughput for receiving data was 20.0 Mbps, the lost datagrams were approximately 6.39% and the delay jitter was 2.516 ms.

1.2.3 Impact of the solution (TCP): 3rd stage

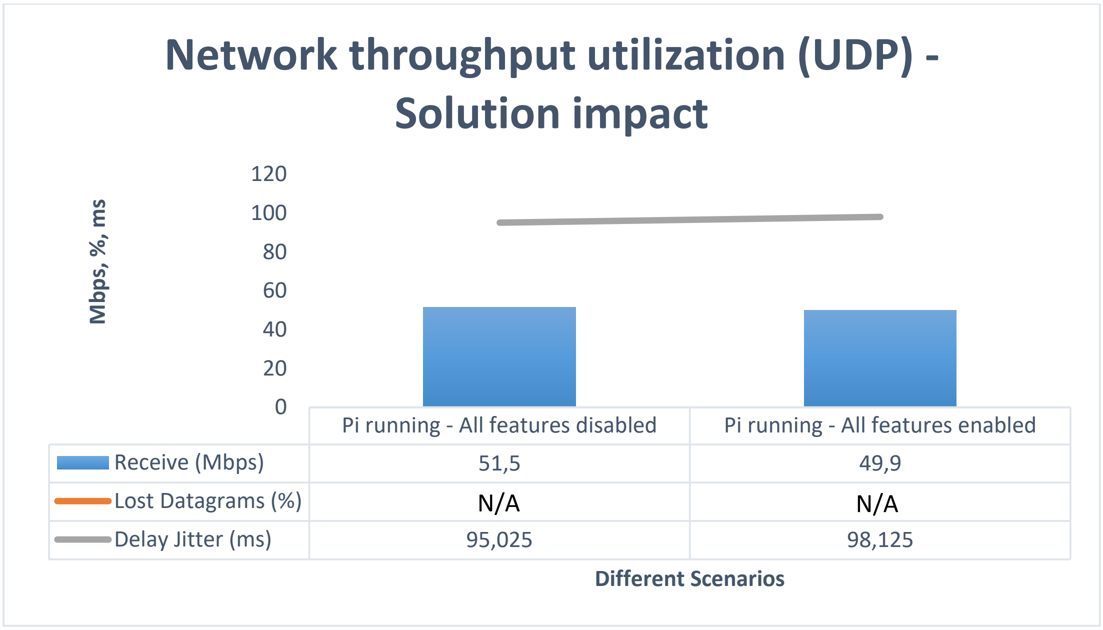


	Default Gateway	Pi running - All features enabled
🟦 Sender	4,71	4,7
🟧 Receiver	4,71	4,64

Figure 17: Network throughput utilization (TCP) - Solution impact

The above graph (Figure 17) shows the average network throughput utilization in Mbps for TCP protocol between one laptop and the cloud server (Elastic stack). The first scenario takes place utilizing only the default gateway that is the home router/access point, without the laptop being connected with the Raspberry Pi 3. The second scenario is realized with the laptop utilizing the Raspberry Pi 3 with all the features enabled. The Y-axis shows the throughput measured in Mbps and the X-axis displays the two different scenarios of the conducted experiments. In the first scenario, the average throughput for sending and receiving data was 4.71 Mbps. In the second scenario, with the Raspberry Pi 3 running with all features enabled, the average throughput for sending and receiving data was 4.70 Mbps and 4.64 Mbps respectively.

1.2.4 Impact of the solution (UDP): 4th stage



	Pi running - All features disabled	Pi running - All features enabled
Receive (Mbps)	51,5	49,9
Lost Datagrams (%)	N/A	N/A
Delay Jitter (ms)	95,025	98,125

Figure 18: Network throughput utilization (UDP) - Solution impact

The above graph (Figure 18) shows the average network throughput utilization in Mbps for UDP protocol between one laptop and the cloud server (Elastic stack). The first scenario takes place utilizing only the default gateway that is the home router/access point, without the laptop being connected with the Raspberry Pi 3. The second scenario is realized with the laptop utilizing the Raspberry Pi 3 with all the features enabled. The Y-axis shows the throughput measured in Mbps, the lost datagrams measured in the form of percentage (%) and the delay jitter in ms, while the X-axis displays the two different scenarios of the conducted experiments. In the first scenario, Raspberry was running with all features disabled and the average throughput was 51.5 Mbps for receiving data. The lost datagrams were not able to be measured[10] while the delay jitter was 95.012 ms. In the second scenario, Raspberry was running with all features enabled and the average throughput for receiving data was 49.9 Mbits/sec, the lost datagrams were not able to be measured[11] while the delay jitter was 98.125 ms.

[10] Too many packages were lost during the experiments, thus making this making this measurement unreliable.

[11] Too many packages were lost during the experiments, thus making this making this measurement unreliable.

1.2.5 Discussion of Results

Based on the above data, I conclude that BRO and the proposed solution as a whole do not cause a substantial overhead to the network. More specifically, the results reflect that the upstream average network throughput remains almost intact, even though the drop rate of UDP packages is increasing from 0,04% to 6,39% while BRO and Logstash are running. The downstream average network throughput has been slightly decreased for both TCP and UDP protocols, and for BRO and the proposed architecture. A possible explanation for this decrease can be justified either by the addition of the extra node to the default routing path or by the Wi-Fi network adapter card capabilities or by the combination of both.

Chapter 6

1 Conclusion

This thesis attempts to make a contribution to the security area in the information technology field. To this end this research provides a novel architecture for a complete model built on commodity hardware and open source software for providing security services in SOHO environments. Special emphasis has been given on NIDS technology, a defensive security service in the field of IT capable to protect computer systems and computer networks from a variety of network threats. Although numerous defensive technologies have been developed, less attention has been given to SOHO environments. For this reason, this thesis focuses on providing security solution for these environments.

In practical terms, for implementing this architecture, open source software has been installed on the Raspberry Pi 3. This software includes BRO, a powerful network analysis framework capable to provide NIDS services and Logstash, a log pipeline tool for parsing logs, executing different transformations and forwarding the data to various targets. In addition, a SIM tool has been deployed on a cloud server, with the use of the Elastic stack, mainly for logging analysis,.

This research has shown that the Raspberry Pi 3 Model B can be used as a centralized network NIDS for SOHO environments. Particularly, it has been observed that the routing of network traffic through Raspberry Pi 3 and the inspection and analysis of the network traffic from BRO do not cause substantial overhead to the network throughput utilization. In addition, it was observed that as far as the performance of Raspberry Pi 3, and specifically the CPU load, Raspberry Pi 3 has more than enough CPU clock speed to run all the necessary services. However, the conducted experiments have shown that Logstash, compared to BRO, utilizes almost the 50% of BRO’s Memory utilization, which is a disproportionate amount of Raspberry Pi’s 3 Memory for the purpose it performs. As a result, Logstash software needs to either be optimized or developed or replaced in order to ensure that Raspberry Pi’s 3 Memory will not run at its limits at all times.

Thus, this thesis proposes an effective and efficient solution based on the capabilities provided by the Raspberry Pi 3 Model B, BRO, Logstash and Elastic stack aiming to empower small office and home office (SOHO) users to deal with security threats. In order this solution be more effective, the end users must be knowledgeable enough to identify the graphs output and respond accordingly by taking the necessary actions. Seen in in this context, it can be said that the proposed solution represents a very useful defensive system for SOHO environments against network attacks and a number of other suspicious activities.

2 Further research

The proposed architecture can be used as the basis for providing security services to a wider range of network environments beyond the SOHO ones. This can be achieved by replacing the open source software used in this thesis with technologies that cover other security aspects in the information technology field.

Additionally, taking into account the experiment findings, the following tasks can be undertaken to achieve an improved framework: either optimizing or replacing Logstash, or optimizing and developing BRO with a special focus on particular network attacks.

Appendix A

Raspberry Pi 3 Model B installation & configuration

A.0.1 BRO

Installation

```
sudo apt-get install cmake make gcc g++ flex bison libpcap-dev
libssl-dev python-dev swig zlib1g-dev

git clone --recursive git://git.bro.org/bro

./configure

make

make install
```

Configuration

```
sudo nano /usr/local/bro/etc/node.cfg

      [bro]

      type=standalone

      host=localhost

      interface=wlan0


sudo nano /usr/local/bro/share/bro/site/load.bro:

      .........

      @load /opt/critical-stack/frameworks/intel
      @load policy/frameworks/intel/seen

      @load policy/frameworks/intel/do_notice

      .........

sudo /usr/local/bro/bin/broctl install
```

A.0.2 Critical Stack Intel Feeds

Installation

```
sudo wget https://intel.criticalstack.com/client/critical-stack-
intel-arm.deb --no-check-certificate

sudo dpkg -i critical-stack-intel-arm.deb

sudo -u critical-stack critical-stack-intel api $apikey
```

Configuration

```
sudo chown critical-stack:critical-stack /etc/sudoers.d

sudo -u critical-stack critical-stack-intel config --set
bro.include.path=/usr/local/bro/share/bro/site/local.bro

sudo -u critical-stack critical-stack-intel config --set
bro.broctl.path=/usr/local/bro/bin/broctl

sudo -u critical-stack critical-stack-intel config --set
bro.path=/usr/local/bro/bin/bro

sudo chmod -R 777 /usr/local/bro/share/bro/site/local.bro

sudo chown critical-stack:critical-stack /etc/sudoers.d/99-critical-
stack

sudo -u critical-stack critical-stack-intel pull

sudo chown root:root /etc/sudoers.d/99-critical-stack

sudo chown root:root /etc/sudoers.d
```

A.0.3 Logstash

Installation

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo
apt-key add -

sudo apt-get install apt-transport-https
```

```
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main"
| sudo tee -a /etc/apt/sources.list.d/elastic-5.x.list

sudo apt-get update && sudo apt-get install Logstash
```

Configuration

```
sudo nano/etc/logstash/conf.d/bro-logstash.conf

      input { file {

            type => "BRO_intellog"

            path => "/usr/local/bro/logs/current/intel.log"

            }

      file {

          type => "BRO_softlog"

          path => "/usr/local/bro/logs/current/software.log"

            }

      file {

          type => "BRO_noticelog"

          path => "/usr/local/bro/logs/current/notice.log"

            }

      file {

          type => "BRO_dhcplog"

          path => "/usr/local/bro/logs/current/dhcp.log"

            }

      file {

          type => "BRO_known_deviceslog"

          path => "/usr/local/bro/logs/current/known_devices.log"

            }

      file {

          type => "BRO_connlog"

          path => "/usr/local/bro/logs/current/conn.log"
            }
      file {
          type => "BRO_sshlog"
          path => "/usr/local/bro/logs/current/ssh.log"
            }
      }
      filter { if [message] =~ /^#/ {
            drop { }
            } else {
      # BRO_intellog ######################
            if [type] == "BRO_intellog" {
              grok {
                match => [ "message",
      "(?<ts>(.*?))\t(?<uid>(.*?))\t(?<id.orig_h>(.*?))\t(?<id.orig_p
      >(.*?))\t(?<id.resp_h>(.*?))\t(?<id.resp_p>(.*?))\t(?<seen.indi
      cator>(.*?))\t(?<seen.indicator_type>(.*?))\t(?<seen.where>(.*?
      ))\t(?<seen.node>(.*?))\t(?<matched>(.*?))\t(?<sources>(.*?))\t
      (?<fuid>(.*?))\t(?<file_mime_type>(.*?))\t(?<file_desc>(.*))"
      ]
              }
            }


      # BRO_softlog ######################
            if [type] == "BRO_softlog" {
              grok {
                match => [ "message",
      "(?<ts>(.*?))\t(?<hosts>(.*?))\t(?<host_p>(.*?))\t(?<software_t
      ype>(.*?))\t(?<name>(.*?))\t(?<version.major>(.*?))\t(?<version
      .minor>(.*?))\t(?<version.minor2>(.*?))\t(?<version.minor3>(.*?
      ))\t(?<version.addl>(.*?))\t(?<unparsed_version>(.*?))\t(?<url>
      (.*))"
      ]

              }

            }


      # BRO_noticelog ######################

            if [type] == "BRO_noticelog" {

              grok {

                match => [ "message",

      "(?<ts>(.*?))\t(?<uid>(.*?))\t(?<id.orig_h>(.*?))\t(?<id.orig_p
      >(.*?))\t(?<id.resp_h>(.*?))\t(?<id.resp_p>(.*?))\t(?<fuid>(.*?
      ))\t(?<file_mime_type>(.*?))\t(?<file_desc>(.*?))\t(?<proto>(.*
      ?))\t(?<note>(.*?))\t(?<msg>(.*?))\t(?<sub>(.*?))\t(?<src>(.*?)
      )\t(?<dst>(.*?))\t(?<p>(.*?))\t(?<n>(.*?))\t(?<peer_descr>(.*?)
      )\t(?<actions>(.*?))\t(?<suppress_for>(.*?))\t(?<dropped>(.*?))
      \t(?<remote_location.country_code>(.*?))\t(?<remote_location.re
      gion>(.*?))\t(?<remote_location.city>(.*?))\t(?<remote_location
      .latitude>(.*?))\t(?<remote_location.longitude>(.*))"

      ]

              }

            }

      # BRO_dhcplog ######################

            if [type] == "BRO_dhcplog" {

              grok {

                match => [ "message",

      "(?<ts>(.*?))\t(?<uid>(.*?))\t(?<id.orig_h>(.*?))\t(?<id.orig_p
      >(.*?))\t(?<id.resp_h>(.*?))\t(?<id.resp_p>(.*?))\t(?<mac>(.*?)
      )\t(?<assigned_ip>(.*?))\t(?<lease_time>(.*?))\t(?<trans_id>(.*
      ))"

      ]

              }

            }
```

```
# BRO_known_deviceslog #####################

      if [type] == "BRO_known_deviceslog" {

        grok {

          match => [ "message",

"(?<ts>(.*?))\t(?<mac>(.*?))\t(?<dhcp_host_name>(.*))" ]

        }

      }

# BRO_connlog #####################

      if [type] == "BRO_connlog" {

        grok {

          match => [ "message",

"(?<ts>(.*?))\t(?<uid>(.*?))\t(?<id.orig_h>(.*?))\t(?<id.orig_p
>(.*?))\t(?<id.resp_h>(.*?))\t(?<id.resp_p>(.*?))\t(?<proto>(.*
?))\t(?<service>(.*?))\t(?<duration>(.*?))\t(?<orig_bytes>(.*?)
)\t(?<resp_bytes>(.*?))\t(?<conn_state>(.*?))\t(?<local_orig>(.
*?))\t(?<local_resp>(.*?))\t(?<missed_bytes>(.*?))\t(?<history>
(.*?))\t(?<orig_pkts>(.*?))\t(?<orig_ip_bytes>(.*?))\t(?<resp_p
kts>(.*?))\t(?<resp_ip_bytes>(.*?))\t(?<tunnel_parents>(.*?))\t
(?<vlan>(.*?))\t(?<inner_vlan>(.*?))\t(?<orig_l2_addr>(.*?))\t(
?<resp_l2_addr>(.*))"

]

        }

      }

# BRO_sshlog #####################

      if [type] == "BRO_sshlog" {

        grok {

         match => [ "message",

"(?<ts>(.*?))\t(?<uid>(.*?))\t(?<id.orig_h>(.*?))\t(?<id.orig_p
>(.*?))\t(?<id.resp_h>(.*?))\t(?<id.resp_p>(.*?))\t(?<version>(
.*?))\t(?<auth_success>(.*?))\t(?<auth_attempts>(.*?))\t(?<dire
ction>(.*?))\t(?<client>(.*?))\t(?<server>(.*?))\t(?<cipher_alg
>(.*?))\t(?<mac_alg>(.*?))\t(?<compression_alg>(.*?))\t(?<kex_a
lg>(.*?))\t(?<host_key_alg>(.*?))\t(?<host_key>(.*?))\t(?<remot
e_location.country_code>(.*?))\t(?<remote_location.region>(.*?)
)\t(?<remote_location.city>(.*?))\t(?<remote_location.latitude>
(.*?))\t(?<remote_location.longitude>(.*))"

]

        }

      }

      }

}

filter {

  geoip {

    source => "id.resp_h"

    target => "geoip"

    database => "/etc/logstash/GeoLiteCity.dat"

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][longitude]}" ]

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][latitude]}" ]

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][city\_name]}" ]

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][continent\_code]}"

]

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][country\_code2]}"

]

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][country\_code3]}"

]

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][country\_name]}" ]
```

```
    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][dma\_code]}" ]

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][postal\_code]}" ]

    add_field        =>        [        "[geoip][coordinates]",
"%{[geoip][region\_name]}" ]

  }

}

filter {

  if [type] == "BRO_intellog" {

      translate {

      field => "sources"

      destination => "sources_full"

      dictionary_path => "/usr/share/logstash/bin/trans.yml"

    }

  }

}

filter {

  if [id.resp_h] {

    if ![id.resp_h-resolved] {

      mutate {

        add_field => [ "id.resp_h-resolved", "%{id.resp_h}" ]

      }

      dns {

        reverse => [ "id.resp_h-resolved" ]

        action => "replace"

      }

    }

  }
```

```
}

output { elasticsearch {

        hosts => [ "83.212.112.5:9200" ]

            #user => elastic password => changeme

    }

}
```

A.0.4 hostapd

Installation

```
sudo apt-get install hostapd
```

Configuration

```
sudo systemctl stop hostapd

sudo nano /etc/hostapd/hostapd.conf

      # This is the name of the WiFi interface we configured above

      interface=wlan0

      # Use the nl80211 driver with the brcmfmac driver

      driver=nl80211

      # This is the name of the network

      ssid=Pi3-AP

      # Use the 2.4GHz band

      hw_mode=g

      # Use channel 6

      channel=6

      # Enable 802.11n

      ieee80211n=1

      # Enable WMM
```

```
    wmm_enabled=1

    # Enable 40MHz channels with 20ns guard interval

    ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]

    # Accept all MAC addresses

    macaddr_acl=0

    # Use WPA authentication

    auth_algs=1

    # Require clients to know the network name

    ignore_broadcast_ssid=0

    # Use WPA2

    wpa=2

    # Use a pre-shared key

    wpa_key_mgmt=WPA-PSK

    # The network passphrase

    wpa_passphrase=PASSWORD

    # Use AES, instead of TKIP

    rsn_pairwise=CCMP


sudo nano /etc/default/hostapd

DAEMON_CONF="/etc/hostapd/hostapd.conf"

sudo nano /etc/sysctl.conf

    net.ipv4.ip_forward=1


sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"

iptables-restore < /etc/iptables.ipv4.nat


sudo nano /etc/network/interfaces
```

```
      source-directory /etc/network/interfaces.d


      auto lo

      iface lo inet loopback


      iface eth0 inet dhcp


      allow-hotplug wlan0

      iface wlan0 inet static

      address 172.24.1.1

      netmask 255.255.255.0

      broadcast 172.24.1.255

      network 172.24.1.0

      #    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf


      allow-hotplug wlan1

      iface wlan1 inet manual

            wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

```
sudo service hostapd start
```

A.0.5 dnsmasq

Installation

```
sudo apt-get install dnsmasq
```

Configuration

```
sudo systemctl stop dnsmasq

sudo nano /etc/dhcpcd.conf

      interface wlan0

      static ip_address=172.24.1.1/24

      denyinterfaces eth0

      denyinterfaces wlan0


sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig

sudo nano /etc/dnsmasq.conf

      interface=wlan0      # Use interface wlan0

      listen-address=172.24.1.1  #  Explicitly  specify  the  address  to
      listen on

      bind-interfaces       # Bind to the interface to make sure we
      aren't sending things elsewhere

      server=8.8.8.8       # Forward DNS requests to Google DNS

      domain-needed        # Don't forward short names

      bogus-priv           # Never forward addresses in the non

      routed address spaces.

      dhcp-range=172.24.1.2,172.24.1.20,24h  #  Assign  IP  addresses
      between 172.24.1.50 and 172.24.1.150 with a 12 hour lease time
```

Appendix B

Ubuntu server installation & configuration

B.0.1 Elasticsearch

Installation

```
wget-qO- https://artifacts.elastic.co/GPG-KEY-elasticsearch |sudo
apt-key add -
```

```
sudo apt-get install apt-transport-https
```

```
echo"deb https://artifacts.elastic.co/packages/5.x/apt stable main"|
sudo tee -a /etc/apt/sources.list.d/elastic-5.x.list
```

```
sudo apt-get update && sudo apt-get install elasticsearch
```

Configuration

```
sudo nano /etc/elasticsearch/elasticsearch.yml
```

```
      .........
      http.port: 9200
      .........
```

B.0.2 Kibana

Installation

```
wget-qO- https://artifacts.elastic.co/GPG-KEY-elasticsearch |sudo
apt-key add -
```

```
sudo apt-get install apt-transport-https
```

```
echo"deb https://artifacts.elastic.co/packages/5.x/apt stable
main"|sudo tee -a /etc/apt/sources.list.d/elastic-5.x.list
```

```
sudo apt-get update && sudo apt-get install kibana
```

Configuration

```
sudo nano /etc/kibana/kibana.yml
```

```
      .........

      server.port: 5600

      server.host: "IP"

      elasticsearch.url: http://IP:9200

      .........
```

Bibliography

Ahmad Sharifi, A., Akram Noorollahi, B. & Farnoosh, F., 2014. Intrusion Detection and Prevention Systems (IDPS) and Security Issues. *IJCSNS International Journal of Computer Science and Network Security,* 14(11), pp. 80-84.

Anderson, J. P., 1980. *COMPUTER SECURITY THREAT MONITORING AND SURVEILLANCE.* [Online]
Available at: https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/ande80.pdf
[Accessed 2 2 2018].

Committee on National Security Systems, 2015. *Committee on National Security Systems (CNSS) Glossary.* [Online]
Available at: https://www.cnss.gov/CNSS/openDoc.cfm?hjVEBIRUL3SSxMy2KWO76g==
[Accessed 17 1 2018].

Critical Stack Team, n.d. *find [Evil] in the noise.* [Online]
Available at: https://intel.criticalstack.com/
[Accessed 14 1 2018].

Elastic Team, n.d. *Elastic Stack and Product Documentation.* [Online]
Available at: https://www.elastic.co/guide/index.html
[Accessed 12 1 2018].

Garcia-Theodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G. & Vazquez, E., 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security,* 28(1-2), pp. 18-28.

Imes, C., n.d. *IptablesHowTo.* [Online]
Available at: https://help.ubuntu.com/community/IptablesHowTo
[Accessed 14 1 2018].

iPerf3 Team, n.d. *iPerf - The ultimate speed test tool for TCP, UDP and SCTP.* [Online]

Available at: https://iperf.fr/iperf-doc.php
[Accessed 14 1 2018].

Isikdag, U., 2015. Internet of Things: Single-Board Computers. In: U. Isikdag, ed. *Enhanced Building Information Models Using IoT Services and Integration Patterns.* New York: Springer, pp. 43-53.

Johnston, S. J., Apetroaie-Christea, M., Scott, M. & Cox, S. J., 2016. *Applicability of commodity, low cost, single board computers for Internet of Things devices.* Reston, VA, USA, Institute of Electrical and Electronics Engineers.

Karthikeyan, R. K. & Indra, A., 2010. Instrusion Detection Tools and Techniques - A Survey. *International Journal of Computer Theory and Engineering,* December, 2(6), pp. 901-906.

Kelley, S., n.d. *Dnsmasq.* [Online]
Available at: http://www.thekelleys.org.uk/dnsmasq/doc.html
[Accessed 14 1 2018].

Kent, K. & Souppaya, M., 2006. *Guide to Computer Security Log Management SP 800-92.* [Online]
Available at: https://csrc.nist.gov/publications/detail/sp/800-92/final
[Accessed 17 1 2018].

Kirkland, D., n.d. *ubuntu manuals.* [Online]
Available at: http://manpages.ubuntu.com/manpages/trusty/man1/mpstat.1.html
[Accessed 12 1 2018].

Kirkland, D., n.d. *ubuntu manuals.* [Online]
Available at: http://manpages.ubuntu.com/manpages/trusty/man1/free.1.html
[Accessed 12 1 2018].

Kirkland, D., n.d. *ubuntu manuls.* [Online]
Available at:
https://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiw9qzs-4zZAhUEfywKHdzmDZkQFggpMAA&url=http%3A%2F%2Fmanpages.ubuntu.com%2Fmanpages%2Fxenial%2Fman1%2Ftop.1.html&usg=AOvVaw2mB2T7Zd7-EGR0UJ-VsDrs
[Accessed 12 1 2018].

Kruegel, C., Valeur, F. & Vigna, G., 2005. *Intrusion Detection and Correlation: Challenges and Solutions.* United States of America: Springer.

Liao, et al., 2013. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications,* January, 36(1), pp. 16-24.

Malinen, J., n.d. *hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator.* [Online]
Available at: https://w1.fi/hostapd/
[Accessed 14 1 2018].

Okeanos GRNET SA, n.d. *Okeanos.* [Online] Available at: https://okeanos.grnet.gr/about/what/ [Accessed 12 1 2018].

Raspberry Pi Foundation, n.d. *RASPBERRY PI DOCUMENTATION.* [Online] Available at: https://www.raspberrypi.org/documentation/ [Accessed 12 1 2018].

Raspberry Pi Foundation, n.d. *Raspbian.* [Online] Available at: https://www.raspberrypi.org/downloads/raspbian/ [Accessed 14 1 2018].

Scarfone, K. & Mell, P., 2007. *Guide to Intrusion Detection and Prevention Systems (IDPS) SP 800-94.* [Online] Available at: https://csrc.nist.gov/publications/detail/sp/800-94/final [Accessed 12 January 2018].

Stallings, W., 2005. *Cryptography and Network Security Principles and Practices.* Fourth ed. United States of America: Prentice Hall.

Stavroulakis, P. & Stamp, M., 2010. *Handbook of Information and Communication Security.* 1st ed. Berlin: Springer.

The Bro Project, n.d. *Documentation and Training.* [Online] Available at: https://www.bro.org/documentation/index.html [Accessed 12 1 2018].

Ubuntu Team, n.d. *Our philosophy.* [Online] Available at: https://www.ubuntu.com/about/about-ubuntu/our-philosophy [Accessed 14 1 2018].

Ubuntu Team, n.d. *Scale out with Ubuntu Server.* [Online] Available at: https://www.ubuntu.com/server [Accessed 14 1 2018].

Wang, W. & Battiti, R., 2006. *Identifying Intrusions in Computer Networks with Principal Component Analysis.* Vienna, Austria, IEEE Computer Society Washington, DC, USA 2006, pp. 270-279.

Wang, W., Guan, X., Zhang, X. & Yang, L., 2006. Profiling program behavior for anomaly intrusion detection based on the transition and frequency property of computer audit data. *Computers & Security,* 25(7), pp. 539-550.

Wang, W. & Zhanfg, W., 2015. Guest editorial: web applications and techniques. *World Wide Web,* 18(5), pp. 1391-1392.

Κάτσικας, Σ. Κ., Γκρίτζαλης, Δ. & Γκρίτζαλης, Σ., 2004. *ΑΣΦΑΛΕΙΑ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ.* 1η ed. ΑΘΗΝΑ: Εκδόσεις νέων τεχνολογιών.