Open source intrusion detection systems evaluation for small and medium-sized enterprise environments

Markku Hänninen

Master’s Thesis
December 2019
School of Technology
Master’s degree programme in information technology
Cyber Security

jamk.fi

Description

Author(s) Hänninen, Markku Mikael	Type of publication Master’s thesis	Date 12/2019
		Language of publication: English
	Number of pages 41	Permission for web: x

Title of publication

Open source IDS evaluation for small and medium-sized enterprise environments

Degree programme

Master’s degree programme in Information Technology

Supervisor(s)

Nevala Jarmo, Saharinen Karo

Assigned by

Laurio Markus, Paytrail Oyj

Abstract

Paytrail offers internet payment services to organizations and due to these services, Paytrail has a very public and visible presence on the Internet. Their information systems contain sensitive information about customers and general public using the services.

One of the biggest threats to any company today is intrusion to their information systems and networks. While preventive methods can raise the cost of intrusion, there are no 100% secure systems and the next best thing is to notice the intrusion early on.

The selected open source intrusion detection systems for networks were compared to provide awareness of their capabilities concerning detection, outputs, administration and software maintenance and development.

The research method used a was collective case study. In collective case study a common set of research goals are used to study individual cases. The cases themselves were quantitative and qualitative evaluations of the capabilities mentioned above. These cases and their research results then provide the basis for comparing the different intrusion detection systems by the factors that had been set by Paytrail.

The results of the research are used by Paytrail to gain awareness what the strengths and weaknesses of open source NIDS products are. The information gained can be then used to provide options for future network security development at Paytrail

Keywords/tags (subjects)

IDS, intrusion detection systems, network security, Snort, Suricata, Zeek

Miscellaneous

jamk.fi

Kuvailulehti

Tekijä(t) Hänninen, Markku	Julkaisun laji Opinnäytetyö, ylempi AMK	Päivämäärä Joulukuu 2019
	Sivumäärä 41	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Avoimen lähdekoodin tunkeutujan havaitsemisjärjestelmien arviointi pienien ja keskisuurien yrityksien ympäristöille

Tutkinto-ohjelma

Cyber security, Insinööri (YAMK)

Työn ohjaaja(t)

Jarmo Nevala, Karo Saharinen

Toimeksiantaja(t)

Paytrail Oyj, Markus Laurio

Tiivistelmä

Paytrail tarjoaa verkkomaksutapapalveluita organisaatioille, ja johtuen näistä palveluista, Paytraililla on erittäin julkinen ja näkyvä paikka Internetissä. Paytrailin tietojärjestelmät sisältävät arkoja tietoja asiakkaista ja yleisöstä joka maksupalveluita käyttää.

Yksi suurimmista uhista mille tahansa yritykselle nykyään on tunkeutuminen heidän tietojärjestelmiin. Vaikka ennalta ehkäisevät menetelmät voivat nostaa tunkeutumisen hintaa, ei ole olemassa 100-prosenttisesti turvallisia järjestelmiä. Tärkeintä onkin huomata tunkeutuminen mahdollisimman nopeasti.

Valittuja avoimen lähdekoodin tietoverkkojen tunkeutumisen havaitsemisjärjestelmiä vertailtiin, jotta voitiin lisätä tietoisuutta niiden kyvykkyydestä havaitsemisen, ulostulojen, hallinnoinnin, ohjelmistokehityksen ja ylläpidon suhteen.

Tutkimusmetodi, jota käytettiin, oli kollektiivinen tapaustutkimus. Kollektiivisessa tapaustutkimuksessa yhteisiä tutkimustavoitteita käytetään monen yksittäistapauksen tutkimiseen. Tutkimustavoitteet olivat kvantitatiivisia sekä kvalitatiivisia arvioita yllä mainituista kyvykkyyksistä. Nämä tutkimustapaukset ja niiden tulokset toimivat pohjana eri tunkeutujan havaitsemisjärjestelmien vertailuille niiden osatekijöiden perusteilla, jotka Paytrail oli määritellyt.

Paytrail käyttää tutkimustuloksia saavuttaakseen paremman tietoisuuden eri avoimen lähdekoodin tietoverkon tunkeutujan havaitsemisjärjestelmien vahvuuksista ja heikkouksista. Saavutettua tietoa voidaan käyttää jatkossa tarjoamaan vaihtoehtoja Paytrailin tulevalle verkkoturvallisuuden kehittämiselle.

Avainsanat (asiasanat)

IDS, instrusion detection systems, network security, Snort, Suricata, Zeek

Muut tiedot

Contents

Figures

Tables

Acronyms

APT	Advanced Persistent Threat
BWA	Broken Web Applications
DDOS	Distributed Denial-Of-Service
DOS	Denial-Of-Service
Gbps	Gigabits per second
HIDS	Host-based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
IDS	Intruder Detection System
IGMP	Internet Group Management Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
IPv6	Internet Protocol Version 6
IT	Information Technology
LAMP	Linux, Apache, MySQL, PHP/PERL/Python
LAN	Local Area Network
NFS	Network File System
NIDS	Network Intruder Detection System
OWASP	Open Web Application Security Project
SMTP	Simple Mail Transfer Protocol
SYN	Synchronize
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

1 Introduction

In the thesis, three intrusion detection systems (IDS) are compared in terms of their capabilities and usability from the point of view of small and medium-sized enterprises. The motivation for the thesis in the area of IDS was provided by the author’s role as a security officer at Paytrail Oyj. As a security officer, it is necessary to train and gather more information about security related tools and solutions and keep ahead of the growing threat environment. This thesis provided an opportunity to delve into the world of intrusion detection systems and better understand the systems themselves and the differences there are between the readily available open source solutions.

As the information technology (IT) threat environment is continually evolving, the tools and processes used to counter these threats have to develop at a similar pace or one gets left behind (Macdonnell 2014). As the security tools themselves develop, it is difficult to choose proper tools for any given task without firsthand experience or a comprehensive review.

IDS in a corporate local area network (LAN) provides a line of defense against a legion of threats that can infect hosts and provide malicious actor access to network. When a malicious actor has already access to the LAN, the best possible remedy is detection as fast as possible; hence, possible repercussion can be limited as much as possible. In addition, IDS can provide help with malware and faulty configurations causing traffic to LAN and taxing company resources. IDS is today considered a normal part of the security layers in security conscious organizations. Figure 1 shows how common different security technologies are; IDS systems are used in 50.4% of companies that responded to the survey.

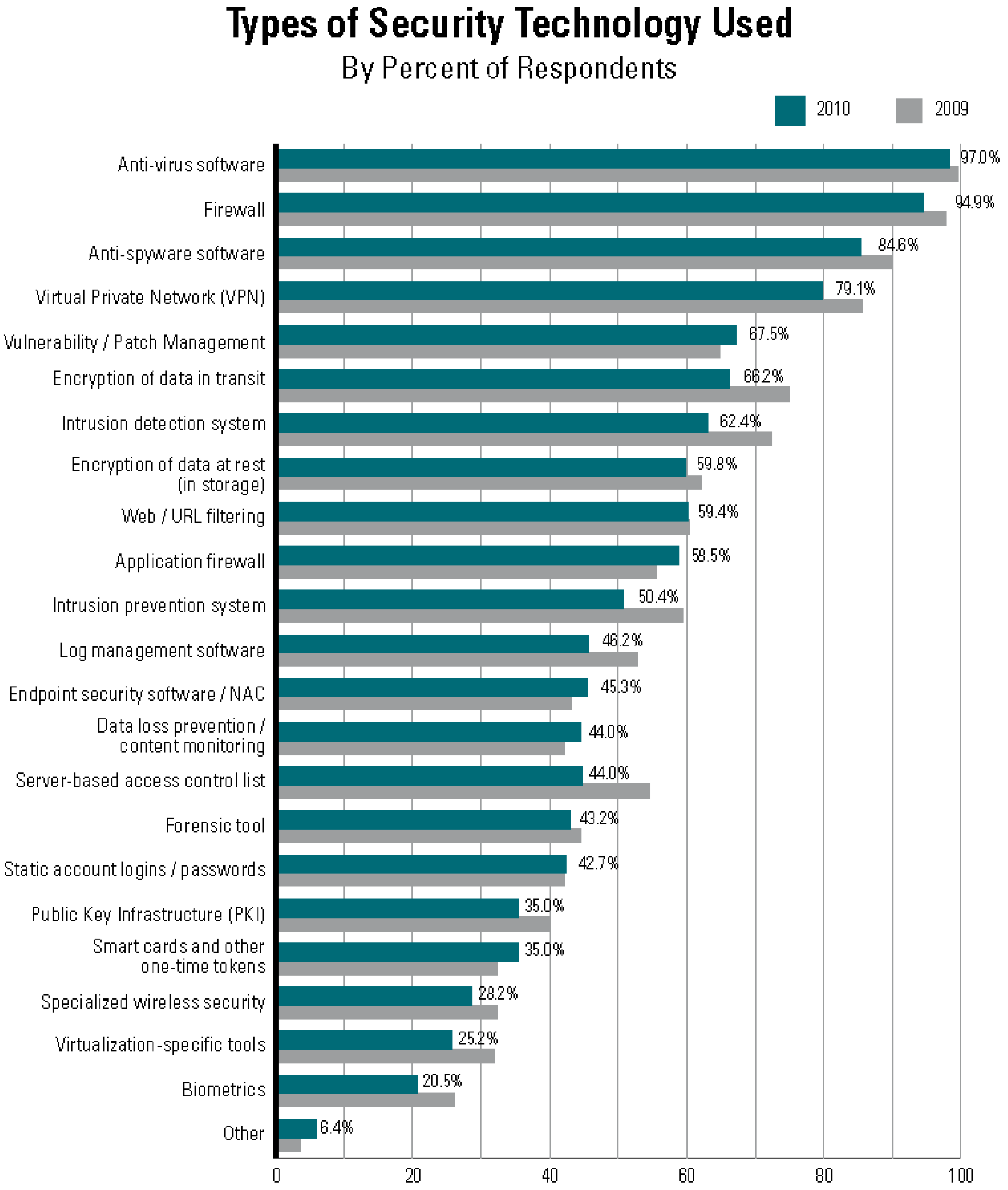


Figure 1. Percentage of security technologies used in CSI survey (Computer Security Institute 2011, 33)

Paytrail is a payment institution providing services to e-commerce merchants and consumers, enabling payments through the internet with a variety of payment methods. The company was founded in 2007 as Suomen Verkkomaksut, and Paytrail brand was established in 2013. Currently Paytrail provides internet payments for over 10 000 webstores and services. During its history from 2007 to 2019, Paytrail has provided internet payments worth over EUR 8 billion to its customers. Paytrail is situated in Jyväskylä, Finland and is fully owned by Nets A/S from Denmark.

As Paytrail(in future also Company) continually develops its information security, it is necessary to evaluate the tools and compare them, to provide necessary knowledge to make informed decisions. While IT operations in the assigner company had already experience of IDS tools, there has been no evaluation or comparison done between different products concerning how they would serve the needs of the company. The Company uses a lot of open source technology, has not adopted open source tool under its own administration for this purpose, instead using commercial services on this front. Open source IDS tool was seen as excellent addition to the existing tools and services in use.

In the assignment, the assigner is interested in three different factors. The first factor is what kind of security performance can be expected without big work effort, how does the solution work out of the box? The second factor is the trustworthiness of the open source project. What does the project development look like, is the future of the project credible? The third factor is the integration. To work well in an existing system, the solutions must be easily integrated to existing logging solutions. How good are the integration options in the solution?

The thesis provides a comparison between the IDS solutions and an understanding of what are the strengths and weaknesses of the different solutions based on these factors. By giving a different emphasis in areas of scoring, the reader of the thesis can customize the results to fit the needs of their company, organization or as an individual. To the assigner, the thesis provides way to compare the open source solutions and make informative decision about the future developments on the open source IDS front with some knowledge.

2 Theoretical background

2.1 Related research

Intrusion detection systems are a target of constant research and internet media articles. In non-academic articles that compare the products, one can find comparisons including Snort, Suricata and Zeek; however, they are short reviews based on features as well as brief lists of strengths and weaknesses. The comparisons in such articles are very superficial and can be used to list potential tools for use; however, not to get any actual knowledge about the capabilities of the products. A good example is provided by Bricata (2018) in their white paper Suricata vs. Snort vs. Bro IDS. The paper describes the products on a general level, mentioning their strengths but it does not provide enough substance to make informed decisions. More in-depth articles on the internet media usually concern the use of one product and certain use cases, how to execute the use cases or how one solution tackles some issue.

The effectiveness of IDS in general is a topic that is touched on in research papers. Hartstein (2008) views available information on the effectiveness of Snort against malicious programs. In the Harstein study, the results of a Snort research study were noted, where 1,259 programs that created network traffic for Snort to detect, Snort created alerts from 68% of them that would get the attention of administrator. The author also notes that while malicious programs might escape detection on the host by being packaged differently, other behavior stays the same, thus being detectable by IDS by its network traffic.

In academic research, there are papers available comparing Snort, Suricata and Zeek; however, they focus on specific areas. For example, Pihelgas (2012) research about network performance of these solutions focuses on that area and does not provide any additional information about the feasibility of these products for any given user. In the area of network performance, Suricata and Zeek both were able to process 1Gbps network link traffic without dropped packets. Snort dropped 41% packets; yet, the desired configuration was not achieved in the research, so the result on Snort is inconclusive. In Pihelgas (2012), in terms of memory usage, Suricata used most memory, with Snort and Zeek using markedly less. In terms of CPU, Suricata seemed the most efficient using CPU resources, with Snort and Zeek being significantly less efficient.

Rødfoss (2011) compares the systems by testing few different attack methods on them and what kinds of alerts are provided; however, the research is very limited by scope and does not provide the overall view but a very limited focus on specific issues. In the research that was carried out, Snort and Suricata had a different level of rulesets in use; thus, Snort alerted much more traffic than Suricata. Zeek uses built-in rules so the result was more comparable, Zeek losing to Snort in terms of alerts. The number of alerts in the research was staggering at hundreds of thousands. While the research carried out a raw comparison between the products, it was not very informative in practical terms.

Ivvala (2017) takes in-depth look at Snort and its capabilities on IPS front against DOS and DDOS attacks. The study is very IPS focused and DOS specific review of Snort's capabilities. The research showed that Snort is capable IPS against DOS attacks with good reliability up to a point, after which packet loss starts and the IPS loses the legitimate traffic.

In Open Source IDS High Performance Shootout, Khalil (2015) looks at existing literature and research about the performance of Snort, Suricata and Zeek. The research discovered that Zeek and Suricata are capable of handling 10Gbps traffic and beyond. Snort was recommended to environments or between links with less than 1Gbps speeds due to single thread design.

On the literature front, Caswell, Beale and Baker (2007) take delve deep into Snort and its capabilities; however, the book itself does not provide any insight in comparison with other solutions. The succeeds well in explaining how IDS works and providing practical information about Snort at the same time.

In conclusion, while the works for the focused area of comparison or in-depth dive to one solution could be found, there was no research about the solutions on overall level, comparing the solutions in a manner, that would provide enough information to make an informed choice about which of the products would best serve enterprise IT environment. Some of the existing studies can be used to provide additional information on their specific areas about the solutions.

2.2 Intrusion detection systems (IDS)

In computing, intrusion means unauthorized access or use of a computer system. Schell and Martin (2006, 180) define the act of intrusion as "to compromize computer system by breaking the security or causing it to enter into an insecure state”. To monitor and alert system administrators of such unauthorized used, a tool is needed. Rehman (2003, 5-6) describes IDSs as systems with methods and techniques to detect such an unauthorized activity based on rules and signatures. These intrusion detection systems provide system administrators a viable tool that can be used to automatically monitor systems and provide system administrators with alerts. Using these systems, administrators can discover an unauthorized use of their systems and trace suspicious usage.

Intrusion detection systems have a decade- long history alongside the development of computer systems. Khan Pathan (2014, chapter 2.1.1) tracked the start of IDS history to the 1980s when the first research into such systems was published. The systems’ development just as a subject of academic research continued, until in the early 1990s when first commercial IDS products were launched.

Intrusion detection systems can be categorized using different ways based on their detection methods, the situation in system architecture and post-detection actions. According to Khan Pathan (2014, chapter 2.1.2) there are signature and anomaly-based detection methods, host-based, network-based or hybrid systems using sensor location and intrusion detection systems and intrusion prevention systems based on post-detection actions.

Signature and anomaly detection methods differ in the way the system evaluates the traffic. Signature-based systems detect intrusions by storing the signatures of attacks and the behavior of known intrusion methods and comparing these signatures to actions, commands and network traffic that these known intrusion methods use. When a match is found, the event is reported. An example of signature detection would be to monitor network traffic to system service port sending data packets that are trying to exploit a known bug.

In anomaly-based detection, the system is monitoring the actions, commands and network traffic and is aware of normal acceptable behavior. When the behavior is sufficiently different from a normal baseline, the event is reported. An example of anomaly detection would be a backend server trying to take an outbound SSH connection to the internet host when it is not allowed per company policy.

Host-based IDS (HIDS) are installed on every host that requires intrusion detection and they report all events that take place on the host they are installed on, for example file changes or suspicious commands. Network-based IDSs (NIDS) monitor network traffic and report all events regarding the network traffic they monitor, for example suspicious connections or data packets containing known attack patterns. Hybrid IDS is a system that uses a combination of both host-based and network-based IDS methods and techniques to discover intrusions.

Post-detection methods divide IDS into two different categories. The logical evolution of an intrusion detection system is an intrusion prevention systems (IPS). NIST-800-94 (2-1) describes IPS as “software that has all the capabilities of an intrusion detection system and can also attempt to stop possible incidents”. By using the capabilities of IDS to discover possible intrusions, the IPS systems take active action to prevent the intrusion. An example of IPS event would be malicious traffic to a network service port; after detection, the IPS will alert the administration of such an event, but also block the network traffic from the source IP to the network service port to prevent any intrusion. This also means that the IPS needs to be in position to perform these preventive actions, for example as passthrough entity in network, or as HIDS installed on the host which is the target of the intrusion. Figure 2 shows the difference between IDS and IPS in network security.

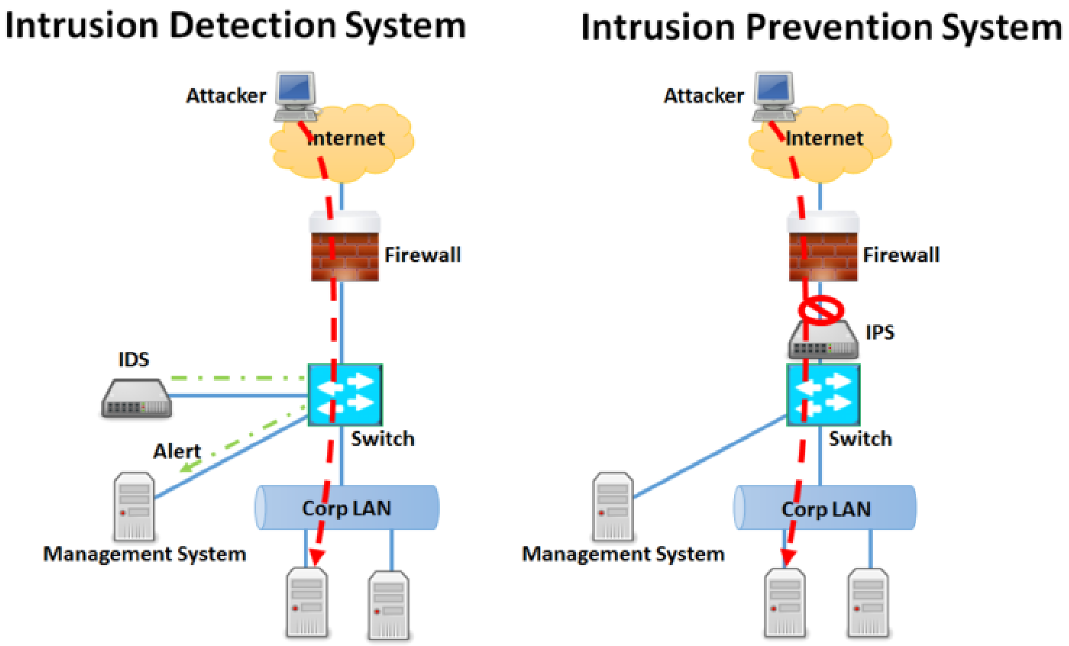


Figure 2. Difference between IPS and IDS in network security (*Difference between IPS and IDS in network security*, 2017)

This thesis concentrates on IDS instead of IPS, and NIDS specifically, instead of HIDS or hybrid systems. As can be surmised from the previous chapters, this selection is made based on location and post detection action basis. The assigner of this work, Paytrail Oyj, designated the need based on their needs in network security and thus the scope of this work.

NIDS as a solution has certain strengths and weaknesses that are based on its operation model. When looking at the strengths Khan Pathan (2014, chapter 2.1.2.2.1.2) mentions real time attack discovery and invisibility to the attacker. Having real time information enables the administration to react immediately when intrusions are detected and thus minimize the possible damage. Invisibility to the attacker creates issues for the intruder, as the intruder cannot be sure if their actions are monitored or not and thus cannot know if a reaction to the intrusion has started.

NIST-800-94 (4-9 – 4-10) adds few more strengths to the list, an in-depth analysis of a wide variety of common protocols and detection of policy violations and unauthorized services. Most NIDS can analyze all common application protocols such as HTTP, NFS, and SMTP, transport layer protocols such as TCP and UDP and network layer protocols such as IPv6 or IGMP. As these protocols are the ones that enable the attacks as well, most are covered by the monitoring. NIDS can also detect policy violations and unauthorized services. As security issues can be caused by users and misconfigured services as well, it is important that such incidents are discovered as soon as possible. NIDSs are generally capable of discovering such issues from network as well if they are configured to do so.

As weaknesses, according to Khan Pathan (2014, chapter 2.1.2.2.1.2) there are the inability to monitor encrypted traffic and restricted application on fragmented networks. As traffic can be encrypted, the NIDS cannot analyze such traffic in detail and it must rely only on basic routing information and handshakes. This causes NIDS to miss malicious traffic that is sent within encrypted transmissions as payload. On the other hand, if a network is very fragmented, it causes issues to NIDS as it must be able to listen to on all the traffic that needs to be analyzed. In very fragmented networks, this would require a huge number of NIDS installations that might not be feasible due to workload or the complexity of the whole.

NIST-800-94 (4-9 – 4-10) mentions a number of false positives and issues with complex operating environments. False positives cause issues with alerts and administration. If there are plenty of false positive alerts that require personnel to check them through; unnecessary work is created as well as distrust of the technology. If the number of false positives is truly great, the real positives can be buried under a mountain of false positives. The modern network environments can be very complex for the NIDS due to the sheer number of devices, operating systems, protocols and communication methods. It is almost impossible to be up to date with all changes and new content that might be transmitted in the network; hence, there is always a possibility of IDS not understanding the transmissions and thus being unable to analyze the content. This might lead to false positives, missing real positives and simply not understanding the traffic at all, thus ignoring it.

2.3 Risk, threat and vulnerability

In information technology, risk, threat and vulnerability are terms interwoven together and sometimes not used properly. According to Kim & Solomon (2014), risk is the probability that something bad is going to happen. A threat is any action that can damage or compromise an asset. A vulnerability is a weakness in the design or software code itself.

In this thesis, the scope of threats and vulnerabilities is limited to issues that are observable from network activity. As described in chapter 2.2 concerning IDS, these activities are recognized either from signature or as statistical anomaly, through rule relating to it. In the following pages, network threats and vulnerabilities how NIDS might mitigate them are discussed more in detail.

2.4 Network threats

The scope of the thesis contains network threats that are already active in a corporate LAN. The evaluation is performed based on the ability to catch malicious traffic in network and on use cases where a breach has already occurred to some system, and that system generates a malicious network activity.

In the thesis the threats that IDS can detect are divided into two different categories: Advanced Persistent Threat (APT) and malware. APT activities are such, that they are controlled fully or to a high degree by a human whereas malware activities are performed mostly by a program without active human guidance. In both cases, the attention is only on malicious activities. Based on these threats, use cases are created that are then evaluated in the thesis. Both, APT and malware, are threats that are relatively common, while not the most common threats. Figure 3 illustrates how common different security related events are. Malware and unauthorized use of computers, networks and servers by outsiders has occurred in 20% and 10% of the corporations respectively, within a 12-month observation period.

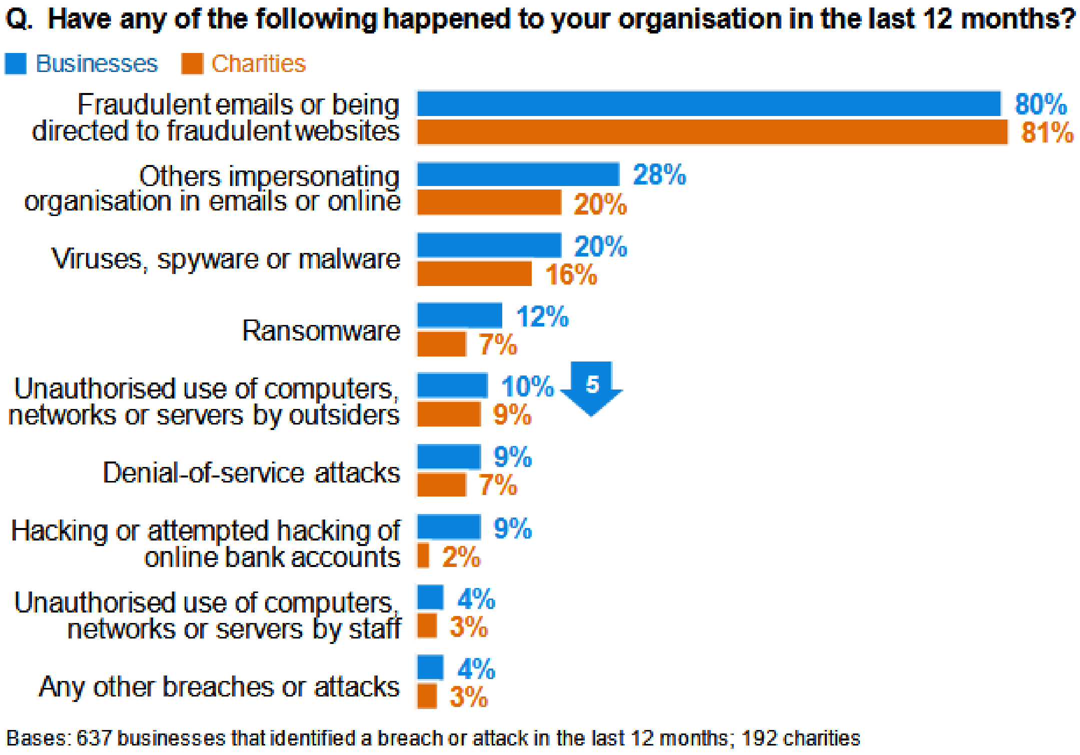


Figure 3. Types of breaches suffered in 12 month period (Department for Digital, Culture, Media, and Sport 2019, 43)

2.4.1 Advanced Persistent Threat

In case of APT activities, actions taken by a malicious actor in the network are the topic. In these cases, a malicious actor is in complete or partial control of some device attached to a network, which enables communication through the network. In simplest cases, the activity is simple network tool use from the original host to another, such as SSH connection. In more advanced cases, the malicious user has access to advanced tools, such as port scanners for information gathering or exploit tools like Sqlmap.

To create test and evaluate cases against human controlled activities, the threat model used to create the evaluation cases is Advanced Persistent Threat. Cyber security company Kaspersky (2019) defines Advanced Persistent Threat as using continuous, clandestine and sophisticated hacking techniques to gain access to a system and remain inside for a prolonged period of time, with potentially destructive consequences. APT provides a threat use case where all the steps during the network intrusion demonstrate different challenges when it comes to noticing and alerting the administrators.

Kaspersky (2019) defines APT attack in following five stages:

Stage one – Gain access

Malicious actor gains entry to a network through some means, e.g. infected file, or vulnerability. This provides the attacker access to the network that the infected system is connected to. In stage one, NIDS can only notice these events if the Internet traffic is monitored and access is gained to publicly offered resource.

Stage two – Establish a foothold

Malicious actors implant malware to provide means to move around the systems undetected. It will also enable a better control through more sophisticated backdoors around the systems. NIDS can detect control connection in stage two. The malicious actor needs a control connection to send commands to a compromised system. The best way to detect intrusions at this point is a host-based intrusion detection system.

Stage three – Deepen access

Malicious actors aim to gain administrator rights to gain more control and greater access. This will enable the attacker to gain full access to a system and a better capability to attack other systems. As in stage two, NIDS can only detect control connection at this point. HIDS are designed to create a layer of security against this stage as well.

Stage Four – Move laterally

Having administrator rights, malicious actors attempt to gain access to other systems and networks. An attacker tries to gain access to systems that it is interested in. Different systems might provide ways to previously inaccessible segmented networks. This is where NIDS in a local area network are most useful. As attacker tries to gain access to other systems, the network traffic is then analyzed by the NIDS and attacks are detected if they are performed.

Stage Five – Look, learn and remain

Malicious actors aim to gain a deeper understanding of the system and means to harvest information they want at will. NIDS can still detect attacks at this phase based on the network traffic that the attacker causes, be it attacks against system or exfiltration of data. Figure 4 shows the attack from stages one through five.

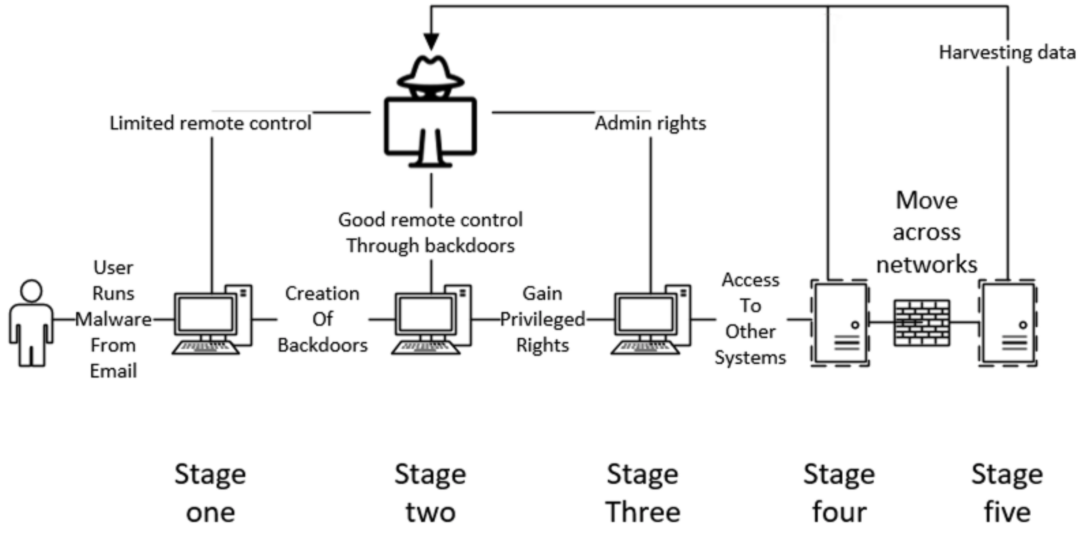


Figure 4. Stages of APT attack

As the thesis is about NIDS, only network activities are relevant from the threat perspective. For that reason, stages two and three have much less meaning for the thesis in the use cases than stages one, four, and five.

2.4.2 Malware

Malware works without or with minimum amount of human guidance by the attacker. From figure 3 what was introduced earlier illustrates that malware is still a common threat in corporate environments and more common than APT attacks. All malware can perform a range of malicious actions based on their programming from creating backdoors to encrypting file systems for ransom.

Cisco (2018) offers definitions for different categories of malware using the technical method of operation: worms, viruses, Trojans and bots are explained as follows.

Worms are malware that aim to replicate functional copies of themselves. Worms are standalone software that spread by exploiting vulnerabilities or by social engineering. Worms are usually visible to NIDS as the common way to replicate is through networks to other hosts through vulnerability that they exploit.

Viruses are malware that spread by inserting a copy of itself to another computer program. Viruses are usually attached to executable files and are usually only active when the executable program is run with the virus as part of it. Viruses in general are usually not detectable by NIDS; however, they might include some functionality that causes network traffic and thus be visible.

Trojan is malware that tries to trick users to load or execute it and causing the Trojan to attack. Unlike a worm or virus, a Trojan does not reproduce by itself. Trojans by themselves are not visible to NIDS as the infection occurs at host level; nevertheless, Trojans can install a backdoor and download additional malware through a network connection causing visible traffic in network.

Bots are malware that infect the host system and then connect back to command and control system for further instructions. Bots have worm like capabilities; however, they are much more versatile and can create botnets, network of bots that command and control can then use to attack their targets. Bots often work through a network by having a command channel and when they are used for distributed attacks. Bots often cause network traffic visible to NIDS.

2.5 Network attack vectors

To keep the scope of thesis reasonable, the testing of attack vectors is limited to common attacks through network, which can be performed either as APT attack or malware. In their book Network Security Bible, Cole, Krutz & Conley (2005, chapter 17) give definitions to five common attacks by intruders:

Denial-of-server (DoS) attacks try to incapacitate the system by consuming the system resources to a point where the system is unable to function. There are many ways to achieve this result, for example TCP SYN flood where so many requests are created towards the system that it is unable to respond to all of them. Another way could be a buffer overflow attack, where the system is provided for example with too big a data packet, which can then cause the system to crash or become unstable. (ibid, chapter 17).

In the thesis, DOS attacks are not given great attention due to the nature of the attack vector. When a DOS attack takes place, the target usually becomes aware of the attack without any special monitoring, as the services become non-functional, service logs usually provide the origin of these attacks as well. In the use cases, just basic flooding attacks are taken against a firewall to see that the if the NIDS tested will notice the attack and report the origin of the attack in the network.

With spoofing, an attacker tries to confuse the target system with spoofed messages, causing it to send transmission to attacker instead of the correct system. Spoofing enables attacks such as Man-In-The-Middle as common spoofing attack is trying to get the victim host to route network traffic through the attacking host. (ibid, chapter 17). The thesis will test if the NIDS can notice spoofing attempts and report them correctly.

Port scanning can be used to gather information about the structure of networks and hosts. Port scanners can also scan hosts for vulnerabilities and software versions to aid in further intrusions. Port scanning is an important part in the thesis due to its usability in steps one and four in an ATP attack. (ibid, chapter 17). Several different port scanning attacks are performed with differing parameters and port scanning software to see how well each NIDS notices and reports such scans.

By guessing passwords, using existing accounts and passwords is a common and effective attack vector. In addition of social engineering and guessing, password attacks can be performed in a random or systematic manner. Brute-force means guessing passwords in a random manner until a working password is found. In a dictionary attack, commonly used password lists are used to gain access. (ibid, chapter 17). In the thesis, password guessing is tried against web applications and operating systems to see if NIDS can notice and report these attacks.

Software exploitation provides many different attack vectors towards systems to gain information and privileged access. Different methods of exploiting software are tested, for example SQL injections, parameter fuzzing and scanning for known vulnerabilities. Vulnerable software is very useful in all steps of ATP attacks and is a pre-requisite for almost all automated attacks not based on end user actions. NIDS monitoring is tested against software exploitation of an application server, an operating system and a network service.

2.6 Open source projects

Snort, Suricata and Zeek are open source projects. The thesis evaluates what the development of said solutions looks in terms of future by looking at near history of the projects. With commercial projects it is easier to observe the reputability of the company and how common any given solution is. If it seems commercially viable, its development will most likely continue. In open source world, even usable solutions can end up withering away due to lacking active contributors for the project.

The Open Source Guide(*Starting an Open Source Project)* defines project as open source when “anybody can view, use, modify, and distribute your project for any purpose. These permissions are enforced through an open source license.”. As such, there are different reasons for sustainability for open source project and commercial project. To evaluate sustainability of these projects, factors that make sustainable and successful open source projects are observed.

In their article Making Lightning Strike Twice, Weinstock and Hissam (2005, 147-148) go through the requirements for a successful open source software project. They argue for a potential developer pool and dedicated developer community as a meaningful factor for successful open source software project as seen in figure 5 showing contributor history of relatively successful openSUSE project.

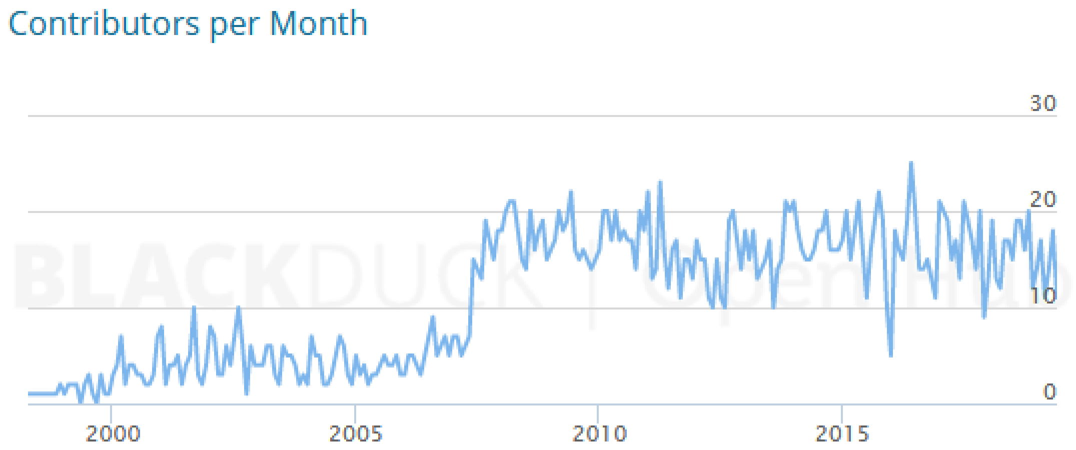


Figure 5 Contributors per month in openSUSE open source project. (*openSUSE Linux Report*, 2019)

With such a community, software should be updated frequently and the releases will be of good quality. Time is also seen as an essential factor as time horizon for mature project might be too short and contributors disappear. OSS Watch (2014) continues the same line in their article How to evaluate sustainability of an open source project. It lists code activity, release history, user community, longevity and ecosystem as meaningful factors for success. As most of these issues are recorded in the open source projects, the situation can be evaluated where different projects currently are.

3 Research

3.1 Case study

The Research method in use is a collective case study, usually qualitative research method; however, in this case quantitative data is included as well. The method was selected to provide information about features, quality and usability of different open source products in same category. The research itself is conducted on virtual machines with a simple network structure usually found in small and medium enterprises.

Case study provided a way to compare the solutions by using selected narrow cases to see the differences and similarities of the solutions. The research goals required the use of both qualitative and quantitative methods to provide the necessary information about the solutions in order to carry out overall evaluation. Tight (2017) explains that case studies provide a method to study narrowly scoped areas; however, still providing results that are useful in a wider sense.

In the thesis, the solutions are evaluated with criteria based on the factors set by the assigner. The criteria led to creation of four sub areas of the research, where test cases were created for comparing the solutions in them and using both qualitative and quantitative methods to set them in order of preference based on the results. These results were then used to compare the solutions on an overall level. While the overall result can be used in a quantitative manner, it is the qualitative evaluation of the factors and their emphasis to any subjective reader that determines the real end result.

3.2 Research objectives

When the research was organized, the assigner provided us the three factors they use to evaluate the usefulness of IDS solution in their use. The factors are as follows:

1. Security performance out of the box, without a big work effort
2. Credibility of the open source project in terms of its future
3. Integration options of the solution

Based on the three factors mentioned, following research goals were devised:

1. How well do the IDSs detect the variety of network attacks using built-in scripts
2. How much effort does configuration of the IDSs require?
3. How well are the IDSs maintained and developed?
4. What kind of alert outputs are supported?

Goals 1 and 2 provide us with information to first factor set by the assigner. Goal 3 provides information to second factor. Goal 4 answers third factor. The results of this research should provide insight how these different products work out of the box and what are their strengths and weaknesses are concerning the factors set.

3.3 Targets of research

The targets of research are well established open source IDS tools that are available as separate tools and as part of commercial products. The tools were set by the assigner for this research to provide information about them for making informative decisions about their use in corporate network setting. Snort, Suricata and Zeek are the generic open source NIDS that are available in the open source market at this time.

3.3.1 Suricata

Suricata is open source NIDS owned and supported by the Open Information Security Foundation (Open Information Security Foundation). Suricata's history starts from grant funding provided to Open Information Security Foundation in 2009. First release of Suricata was in 2010 and it has roots in Snort architecture. (White, Fitzsimmons & Matthews. 2013, chapter 1.2). Suricata is the most recent entry of the three at the IDS stage. It has the advantage of starting from scratch with lessons learned from the previous solutions and trying to improve upon them. Suricata's design has roots in Snort development and Suricata can use rules and signatures designed for Snort with little effort. Suricata is primarily a signature-based detection tool.

3.3.2 Snort

Snort is an open source NIDS that was originally created by Martin Roesch and released in 1998. Later in 2001, Roesch created a company, Sourcefire Inc., to develop and offer commercial products based on Snort. Sourcefire Inc. was acquired by Check Point Software Technologies in 2005 and later, Cisco Systems in 2013, and Cisco Systems is currently managing the development of Snort. (*What is Snort?*, 2017). Snort has a long development history and is used in commercial products for

IDS purposes. While a long history can be a burden, it is also a source of strength through wider community and recognition. In any list of open source IDSs, Snort is certainly found in the listing. It is usually sitting on top of the list as most recognizable. Snort is primarily a signature-based detection tool.

3.3.3 Zeek (formerly known as Bro)

Zeek is an open source NIDS project initially started by Vern Paxson in 1995 when he was a researcher at the Lawrence Berkeley National Laboratory. The first operational deployment was carried out in 1996 and a research paper about Zeek (then known as Bro) was published in 1998. In 2003 National Science Foundation began supporting Zeek development and a team of researchers and students has been developing the system since. The development of Bro has been different from other open source IDSs, as the development has mostly been based on academic research projects. (Zeek documentation 2019, chapter History). From the three, Zeek is clearly less known, most likely related to its development history as research project. Regardless, there is a commercial solution offered that is based on Zeek and it can be expected that Zeek will be better known in the future. Zeek is primarily an anomaly-based detection tool.

3.4 Test environment

The test environment was created with virtual servers running on Oracle VirtualBox virtualization software. VirtualBox software enabled running all the hosts in testing on the same machine and lessened the complexity of setting up the testing. Oracle VirtualBox is an open-source project and free to use for personal and educational use.

The test environment has three target servers, two clients, all in the same network with NIDS solutions listening traffic in promiscuous mode. Promiscuous mode provides all VirtualBox related network traffic to virtual servers as if they were part of a hub instead of a switch, thus enabling the use of NIDSs in such a manner as they were in real networks with port mirroring.

With all NIDSs in promiscuous mode, it is possible to provide an identical test setting in all performed tests. The tests are run once, and results documented from all NIDSs, thus making tests and results comparable without the possibility of a test error between solutions due to the environment in the testing situation. Figure 6 shows logical graph of the testing environment in VirtualBox.

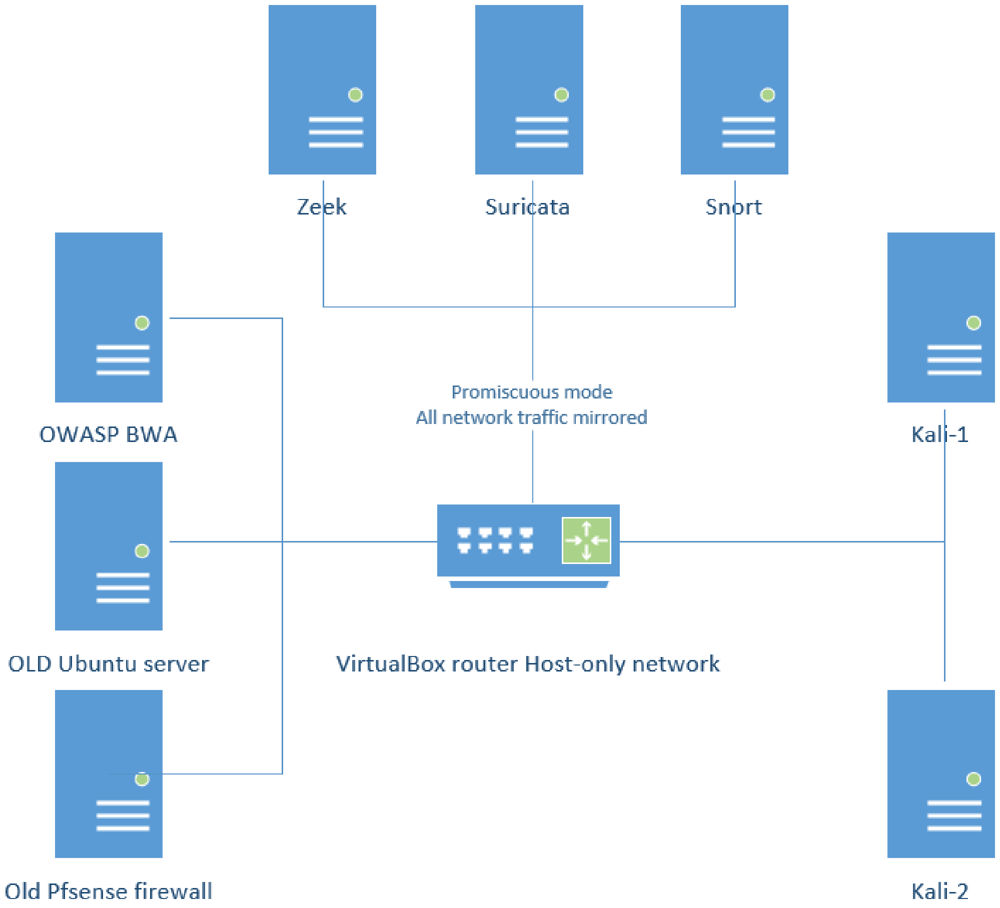


Figure 6. Test environment

3.4.1 Network and the hosts

All the hosts in the VirtualBox environment were set up with IP address in host-only network 192.168.56.0/24. Table 1. Shows the test network environment used in the testing.

Table 1. Test network environment

Name	Version	IP
Network	VirtualBox 6.0.12	192.168.56.0/24
OWASPBWA	1.2	192.168.56.103

Old Ubuntu server	9.10 Karmic Koala 64-bit	192.168.56.150
PfSense	2.3.5 64-bit	192.168.56.140
Kali-1	2019_4	192.168.56.115
Kali-2	2019_4	192.168.56.116

OWASP BWA is a tool created by the Open Web Application Security Project (OWASP) which provides tools to improve software security. Broken Web Application (BWA) project provides tools to learn about software vulnerabilities and learn by doing in different web application environments. As vulnerable web application Mutillidae II tool is used, which has been created by OWASP foundation to provide training environment for cyber security concerning web applications. (Owasp, 2015).

An old Ubuntu server is used to provide the testing environment with a vulnerable Linux installation for testing attacks against networked Linux OS and network services that run on it. By using the old Linux version, exploits attacking against known vulnerabilities are easily tested. Following Linux services were installed: OpenSSH, Mail server, LAMP server, PostgreSQL database, Print server, Samba file server and Tomcat Java server.

PfSense is an open-source firewall used as the network device in the tests. Firewall is mainly used for testing DOS attacks, but also attacks against vulnerabilities. PfSense is based on FreeBSD and provides a community edition which is distributed under Apache 2.0 license. PfSense is offered in a cloud and appliance versions in a commercial setting as well.

Kali-1 and Kali-2 are both Kali Linux installations and are used to create attacks against targets. Kali Linux is a Linux distribution developed by Offensive security containing a set of tools that can be used, among other things, for network attacks.

3.4.2 Snort configuration

Snort was installed and configured to use the community rules and subscriber rules from www.snort.org. Subscriber rules are rules maintained by the http://www.talosintelligence.com and 30 day old rules are given free of charge to registered users at www.snort.org

Snort version: 2.9.15 (Build 7)

Rules fetched: 18.10.2019

3.4.3 Suricata configuration

Suricata-update tool provides rulesets for Suricata. All non-commercial rulesets were enabled with tool and used during the tests.

Suricata version: 4.1.5

Rules fetched: 20.10.2019

Suricata JA3 fingerprint rules require that JA3 is enabled in Suricata configuration. JA3 fingerprint is method of fingerprinting SSL handshakes. The following line was modified in */etc/suricata/suricata.yaml* as yes:

ja3-fingerprints: no

3.4.4 Zeek configuration

Zeek has rule-scripts built-in that just need to be enabled in the configuration. All available detection scripts were enabled in the configuration.

Zeek version: 3.0.0

Rules built-in. Version fetched 23.10.2019.

Zeek has automated notice (alarm) suppression, which means that it does not report same issue twice, unless specified time interval has gone by. To prevent this, following option was set in *base/frameworks/notice/main.zeek* file:
option default suppression interval = 0secs;

3.5 Testing areas and scoring

3.5.1 Network attack monitoring

In the thesis, network attack vectors are defined as attack vectors that attack through network connections. Furthermore, the scope of this thesis is limited to attack vectors that use technical means; therefore, social engineering / dumpster diving and other non-technical information gathering methods are outside of the scope of this thesis. Excluded are also all technical attack vectors that are invisible to network monitoring, as they have no relevancy to NIDS performance analysis.

Evaluating network attack monitoring provides information how each product performs in its main function. In this question it was interesting to see how well different solutions observe and report different attacks. Additionally, it was to be seen if there are meaningful differences between the solutions in reporting attacks. All solutions had all their built-in scripts and signature libraries enabled if they were available at no cost.

The only log that is followed is the alert log or notice log that the software uses to flag priority events. Normal traffic logs are not inspected. With Suricata and Snort, the alert log Priority: 2 and Priority: 1 -messages are examined. With Zeek, all entries logged in notice.log are examined. The reason for this is that all IDS log massive amounts of informational data, and the amount of information makes such logging levels useless in terms of human observation. It is possible to spot the attack from such logs; however, going through such amounts of logs is not feasible in a real life environment. Thus, only those entries are examined, that the IDSs themselves consider more important. With Zeek, it was necessary to exclude certificate errors from the notice.log as well, because Zeek informed about all non-valid certificates in the log.

To answer to the research goal 1, the most common network attack vectors were studied and the test cases were built for them. These test cases are performed in a network, where each solution is monitoring the traffic while attacks are performed and then results are scored. The test documentation includes information about which common attack vector it relates to.

3.5.2 Configuration

On the configuration and updates side the focus is on how easily the changes in configuration are performed. With configuration changes the focus will be on simple manually created alert rules for network traffic that are useful in any environment. These tests answer to research the goal 2. In the test time between starting to create rule is measured in the system and the time when the rule is successfully triggered in the solution by the traffic defined in the test. The test is performed without previous knowledge of the syntax used in creating the rules.

3.5.3 Software maintenance and development

A comparison is made between the IDSs to see how well they are maintained and if the products are developed in an active manner. The results of these tests provide information concerning the research goal 3. The time between the major version releases and development of minor releases between major versions is observed. Contributor numbers in the last 12-month period are checked from the development projects. These factors are seen most relevant for comparisons between solutions considering they all have a long development history.

Tool https://www.openhub.net/ is used to check the code contributions.

3.5.4 Alert outputs

As the relevant integration in IDS use is logging and alerts, a comparison of the number and quality of outputs that the solutions support is performed. Each of the solutions' documentation is checked to verify how many different outputs the solution can provide and if there are any differences between the solutions, such as the extent of the output options. These tests were performed to provide information relevant to research goal 4.

3.6 Scoring

In the network attack tests one score point is given if the attack is reported. One score point is given if the report of the attack matches the attack itself and is not misleading. Thus, any single test can provide a maximum of two points to a solution and a minimum of zero. The scores from all tests are then tallied, and different solutions can be compared by their total score and their score with each test case. The total scores are then compared, and the best solution gives three points, the second best two points and the last solution one point from this area of tests.

In the configuration tests, the solutions were timed in each test. Each test was scored with three points to the best solution, two points for the second best and one point for the third best. The total scores were then compared and the area was scored by giving three points for the best solution, two points for the second and one point for the third.

In software maintenance and development, the project was evaluated based on the major releases, development in the last few years and the number of contributors they had. The evaluation then scored the projects by giving three points for the best solutions, two to the second best and one for the last. The area score was gained directly from the result of the evaluation.

Alert outputs were evaluated by looking at the number and quality of outputs that each solution provided. Based on the evaluation, the best solution scored three points, second one was given two points and the third best, one point. The area score is based on the result of the evaluation scores.

If solutions tied in scoring results of an area, both solutions were given the points based on the position they were tied in. The position immediately behind the tie was considered filled at that point. If solutions were tied at first place, both were given three points, and the last solution one as it is deemed placed third. If the solutions were tied at the second place, both were awarded two points, and none one point.

4 Results and analysis

4.1 Network attack monitoring

Network attack monitoring is divided into five areas of different attacks that were tested based on the common attack vectors listed in chapter 2.5. The basis for the test measurement is described in chapter 3.5.1. Each solution was given one point if it reported an attack related to the issue tested. Another point was given if the reported attack was correctly identified in the alert.

Denial of service attacks included a network attack with TCP, an attack against web server and an attack against the operating system service. The attacks were performed by using Metasploit tool. DOS attacks can be used to cause a service outage to a specific server or service. Depending on the sophistication of the attack, DOS attacks can be a nuisance or part of more elaborate attack to replace some infrastructure service such as DNS with the attacker's own version, providing malicious data. Specific tests and results in Appendix 1. Table 2 shows the results of the DOS tests.

Table 2. Results of Denial of Service attacks

Denial Of Service	Snort	Suricata	Zeek
TCP DOS	0	0	0
Web server DOS	0	0	0
OS service DOS	0	2	0

Only one attack against operating system service Samba was detected during all the tests by Suricata with correct information about the attack. The result was not expected, especially TCP DOS is something that is easily observable from the firewall statistics and the traffic is not meaningful in any manner. This might be something that can be tweaked in the sensitivity settings of the solutions; however, out of the box, the results were not good.

TCP Hijacking attack was performed by spoofing ARP messages to the client and gateway using ARPSpoof tool. ARPspoofing can be used to provide faulty routing data to network hosts causing them to send network traffic to a malicious host. This false routing can be used to disrupt the network or as the first step for a more complex man-in-the-middle attack. The specific test is found in Appendix 2.

As we can see from table 3, none of the solutions reported this attack. This was not an expected result. While the ARP messages themselves are perfectly appropriate, one would expect the solutions to compare the ARP messages monitored to those moving in the same network by other hosts and deem the attack as anomalous traffic. This might be something that can be tweaked in the settings; however, out of the box, none alerted.

Table 3. Results of TCP Hijacking attacks

TCP Hijacking	Snort	Suricata	Zeek
ARPspoof	0	0	0

Port scanning attacks included two port scans and one fingerprinting action. The tools used in attacks were NMAP and Dmitry. Port scanning and fingerprinting can be used for reconnaissance to learn more about the network and detect vulnerable versions of network services and hosts. The Details of the attack can be found in Appendix 3. Table 4 shows the results of port scanning tests.

Table 4. Results of Port Scanning attacks

Port Scanning	Snort	Suricata	Zeek
NMAP port scan	0	1	2
Dmitry port scan	0	0	2
Fingerprinting	0	2	0

In port scanning better results were observed with Suricata and Zeek reporting attacks and the attacks were identified correctly. Port scanning and fingerprinting inevitably rely on some set limit of connections to ports to make the determination of suspicious action. It is likely that the limits on the solutions out of the box are too high for the test, as they do not alert about attacks even though the tools connect many different ports in short order.

Two password guessing attacks were made with Hydra, one against the web server application and one against the operating system (OS) service. Easy passwords provide low lying fruit for any attacker. One bad password can provide the first step to any system, even if the related account is an unprivileged one. More detailed information about the test can be found in Appendix 4.

In this case, Suricata alerted about strange traffic against the Web Server; however, it was unable to identify it being a password guessing attack. In the attack against the OS service, Zeek was able to identify and report the attack. Table 5 shows the results of the tests. These were not expected results, since password attacks are well known and should be identifiable from the traffic, especially HTTP as the attempt is made in unencrypted form. The reason for this can be that there is some high limit of password tries in the IDS for this and the limit was not reached.

Table 5. Results of Password Guessing attacks

Password Guessing	Snort	Suricata	Zeek
Hydra - Web Server	0	1	0

Hydra - OS Service	0	0	2

Six different attacks were performed with software exploitation: vulnerability scan, SQL injection, parameter fuzzing, forced directory browsing and two vulnerability exploits. Vulnerable or badly configured software often offers various ways for a malicious attacker to exploit the systems and either gain access or deepen it without a huge effort. Appendix 5 provides detailed information about the tests. Table 6 shows the results of the software exploitation attacks.

Table 6. Results of Software Exploitation attacks

Software Exploitation	Snort	Suricata	Zeek
NMAP - Vulnerability scan	2	2	1
SQLMAP SQL injection	2	2	0
Burp Suite Parameter fuzzing	0	2	0
Nikto Forced directory browsing	0	2	0
Linux Samba vulnerability exploit	0	0	0
Linux Apache/Bash vulnerability exploit	2	2	0

Software exploitation shows great differences between the solutions with Suricata coming on top. Snort performs at adequate level; however, Zeek is clearly struggling. Suricata's and Snort's reliance on signature detection is their advantage. These attacks contain a payload that is easy to configure as signature. Zeek, on the other hand, works in a different manner and does not have extensive signature libraries.

4.2 Configuration

The comparison tests concerning configuration illustrated how easy simple network traffic rules are to configure to different solutions. The tests were performed without previous knowledge of the syntax for the rules, and the effort was timed. Three points were given to the best solution, two to the second best and one point to the last one. The rules used in testing are such that they are used in any given installation of IDS in the real world to set up rules for acceptable and non-acceptable network traffic. Table 7 shows the results of the configuration tests and Appendix 6 contains detailed information about the tests.

Table 7. Results of configuration tests

Configuration	Snort	Suricata	Zeek
Custom rule for network traffic	3	3	1
Network traffix rule exception	3	3	3

In this case, Snort and Suricata used a very similar syntax and configuration for the custom rules and were given the same time, as the experience from the first test benefitted the second. All the solutions had simple systems for creating an alert from specified network traffic. Zeek suffered from a greater attention to more complex rules that the solution offers, which caused a longer time to find the syntax for the simpler signature rules that could be used to create an alert notice from specified network traffic.

4.3 Software maintenance and development

Version update dates were checked for major versions and number and size of updates between last two major version to see what the development state of the different solutions is. The tests also included a check of the number of contributors in the last twelve months. The results are displayed in table 8. These checks provided information about basic activity in the open-source project to evaluate what their future looks at this moment. The best was given a score of three points, the second two and the third one point. Detailed information about the versions can be found in Appendix 7.

Suricata and Zeek showed strong active development in recent years and had constant updates every year with a good number of contributors. Snort had few releases and a very inactive year in 2018 with only one release and a relatively low number of contributors in the last twelve months. Table 8 shows the results of the evaluation. Snort 3.0 has been in Beta since 09/2018 and this inactivity in releases and updates might be related to it; however, as this was not public knowledge, The scoring was carried out accordingly

Table 8. Results of updates comparison

Updates	Snort	Suricata	Zeek
Version updates	1	3	3

4.4 Alert outputs

All the solutions were compared to see what alert output methods were supported out of the box. The outputs were considered by their variety and importance. Outputs are an important factor when considering how the solutions are integrated to existing systems. Varied and extensive outputs offer an easy way to integrate the solutions to any given need whereas narrow options likely lead to an increased work effort. Information about the comparison can be found in Appendix 8.

As table 9 shows, Suricata and Snort tied with their outputs, with Zeek getting the last place. By sheer numbers, Snort had the most options, however, it lacked JSON output, which is often used in logging solutions and provides human readable text log. Suricata had a good variety of outputs and an excellent JSON output parser. Zeek had only few outputs and lacked some common ones.

Table 9. Result of Alert output comparison

Alert outputs	Snort	Suricata	Zeek
Comparison of alert outputs	3	3	1

5 Discussion

5.1 Solution comparison

By scoring each of the testing areas, giving best solution three points, second two and the third one point, results shown in table 10 were gained.

Table 10. Results of testing areas

Testing area	Snort	Suricata	Zeek
Network Attack Monitoring	1	3	2
Configuration	3	3	1
Updates	1	3	3
Alert outputs	3	3	1
Total	8	12	7

By looking at the results, it can be observed that Suricata was the first or shared the first place in each area of tests. Snort places as second with one point over Zeek. The result was somewhat expected, Suricata is the newest product of the three; however, it is already mature with nine years of releases. Zeek performed better than expected, being on same level with Snort, even though Zeek is based on a research project. Part of Snort's struggle might be related to its upcoming version 3.0 that has been in beta since 2018. The results can be easily tweaked to different situations by either going through the tests and evaluating their importance to any given use case or giving multipliers to testing areas based on the needs. It is also possible that the signatures currently offered by Snort out of the box are not as varied as Suricata's and thus the marked differences in the results of the network attack portion of the tests.

5.2 Research

The First goal of the research was to test how well different IDS solutions detected a variety of network attacks with freely available scripts and rules. The attacks were devised based on common attack vectors used by malware and APT. The tests themselves provided clear quantifiable results. The only question about the test is how well the selected tests correspond to real world events. While the results in chapter 4.1 show the difference between the solutions when testing these specific issues, different results might be gained by testing something else. The tests were based on common attack vectors and the results did provide a clear division between the solutions. The testing itself was carried out without great issues. The only trouble during the testing was Zeek as it logged all the certificate errors, thus making the logs less readable. This issue was corrected by parsing all but certificate errors in the log.

The second goal was related to the first goal, as they both were questions that answered the same factor set by the assigner company. The tests performed towards the second goal were very simple; however, most likely done in all real-world implementations of an IDS solution. The results in chapter 4.2 showed the effort required to carry out such changes in practice and thus provided good information towards the goal set. During the testing it was apparent that Snort and Suricata have rules based on a very similar syntax, thus creating rules for either is relatively simple if one is familiar with the syntax of the other. Luckily, this enabled setting the same time for both solutions, as there was little difference in creating them.

Gathering information towards the third goal was different from first two goals. Evaluating the test results with active development and maintenance of solutions is not as simple as comparing time and log results. Research to literature provided some good solutions to performing these tests and in the end, the results in chapter 4.3 provided at least a glimpse of the current situation, even if forecasting future developments is of course impossible. The development of open source projects can be cyclic and thus this result as a comparison of the solutions is not the most reliable but does give a basis to believe that all three projects are under active development.

Looking at research goal 4, alert outputs, the test comprised a comparison of natively supported output modules of each solution. While it is easy to compare the amount of supported outputs, evaluating which format is better or more meaningful than another is a highly subjective matter. The results in chapter 4.4 showed that Zeek was easy to set aside as the “loser” of this comparison due to very limited options, whereas Suricata and Snort had both strengths and weaknesses in the results that could push the result to any direction. The result between Suricata and Snort might change depending on who is performing the evaluation and the requirements of that particular use case.

When the overall results in chapter 5.1 are observed and the research goals stated in this thesis and the overall picture they paint are concerned, the answer seems clear: to a small and medium enterprise, Suricata seems the best option. Suricata can detect attacks, it is relatively easy to create alerts concerning corporate network, the solution is under active development and one can be sure that one gets the alerts to the logging system of one’s choice with relative ease.

This is not the whole story though. When the research was carried out, novelties of the different solutions became apparent as well. Zeek, while third in this comparison, for example had detailed logging in terms of protocols and packets and would prove useful if such information is needed for forensics when investigating network issues or a breach. Snort is considered an industry standard, and information as well as help and literature about Snort is available in greater degree than there is about Suricata or Zeek. Snort is also part of many commercial IDS devices and is in part developed by Cisco.

While this thesis would show that if an enterprise needs to improve its network security, Suricata would be an efficient and low effort solution to provide that security; however, the other solutions have their places as well. As IDSs do not actively manipulate the network, there is no issue of having many IDSs in use and aggregating their alerts. There is a free open source Linux distribution called Security Onion that offers all these tools and more in a single package for use in any network environment. While the administration of such combined tool is most likely more demanding and requires a greater effort, the security level it provides should be higher as well, as different solutions have different strengths and weaknesses.

From the point of view of the research goals, the configuration portion could have provided more useful information by going more in depth to rules and signatures as well as comparing the creation of complex rules and signatures. This area was limited to simple useful rules due to great growth in effort that learning the creation of complex rules and signatures would have required.

The accuracy of research should be good as this thesis enabled the comparison of the IDS solutions by seeing their results based on the same unique time the attack was made. The rules and signatures used were limited to those easily available to make the research as useful as possible to all sorts of organizations. There might be a small variance in results when using commercial rules and signatures, in this case the effect should be limited as only Suricata had signatures that were left out based on their commercial nature. Snort used a non-commercial ruleset; however, it is the same ruleset the commercial one, it is just released 30 days later for free to registered users.

In some rules and signatures there are sensitivity settings as well. By tweaking such settings, different kind of alert results might have been reached, especially in testing that causes a certain amount of traffic, like port scanning. Nevertheless as it is very difficult to compare such tweaking between solutions, it was left out. However, results might be slightly better in real world, where such tweaking is probable at least on some level.

5.3 Field of study

While the author was unable to find similar researches comparing these products in the same manner, there was plenty of research available about these products, comparisons between them that are closely related and provide different kind of information about the products, for example the research about Snort, Suricata and Zeek network performance by Pihelgas (2012).

Using this thesis as base, other studies can be used to gather comparison information about different areas of IDSs not included in this thesis and then provide even more information to make informed decisions about which IDS is a good fit for any given environment. This creates possibilities for future research, as literature review of different comparisons could provide a very good overall view of these products. This kind of research could provide better knowledge about the solutions than any single research comparing some given features between the solutions.

6 Conclusions

Without having user experience about different IDS solutions, it is very difficult to make an informed decision to choose one for any given use. This thesis looked at three different open source solutions, Snort, Suricata and Zeek, to see how they compare with each other in terms of providing security to a small and medium-sized enterprise network environments. Snort, Suricata and Zeek are established open source IDS tools suitable for generic use, thus a reasonable target for a research that excludes commercial products. Other open source products are either host-based or limited somehow, for example OpenWIPS-NG, which is targeted at wireless networks. There are also many solutions that can use data provided by an IDS; however, they are in the domain of network security monitoring.

The research questions were set in a manner to provide information about how the systems would perform when set up to run with a moderate amount of effort and with out of the box features. They were then compared to see how they performed against common attack situations in a network, how easy it was to create small IDS rules in a simple network environment, how actively they were developed and finally, how varied their output capabilities were, so the alerts are easy get out of them to other systems.

On the terms set above, the thesis provided clear results in chapter 4.5. If one has use for an open source IDS solution, Suricata is a good choice to start with. Both Snort and Zeek are good solutions themselves and provide similar features and their own strengths; however, out of the box they do not provide as good a solution as Suricata

Snort gave the impression of a well working and widely known solution, there was easily the greatest amount of information available about it in books, forums and articles. It left the feeling that one cannot go wrong with Snort, yet it also felt somewhat dated and was not updated as often as the competitors. Snort is also the only solution not offering any multi-threaded functionality, thus limiting the capabilities of a single Snort instance in terms of network traffic volume. This feature is promised in the upcoming version 3.0; however, it has been in beta phase for over a year.

Zeek was a very different product from either Snort or Suricata. It has evolved as an academic research project and it showed. The information about it was most scarcely available and it had no support outside of few online forums. Zeek also uses primarily anomaly detection whereas Snort and Suricata are signature-based. Zeek created the most interest in author’s IT expert side, and with infinite resources, the author would certainly put time and effort to study all aspects of Zeek.

Suricata felt the freshest solution of the three. Snort and Suricata are very similar products from user perspective, Suricata just felt like it was a newer better version of Snort with additional features. Information about Suricata was easily available, yet not as widely as it was about Snort.

The results followed the impressions that the solutions created during this thesis. If one solution had to be recommended for small and medium-sized enterprises to use, Suricata would be that recommendation by the thesis and the author.

The thesis could have been broadened by adding different areas to testing such as performance or delving deeper in areas such as rule and signature creation and alert outputs. The scope of the thesis was difficult to set and a great deal of balancing between the size of the thesis and practical need for information had to be done. In the end I am satisfied with the thesis and the knowledge it provides about the solutions.

Related research can also be used to complement this thesis. For example, performance knowledge from Pihelgas (2012) and Khalil (2015) can help to decide what kind of networks or points the IDS is used to monitor, or what product to select in any given case based on network performance. Caswell, Beale and Baker (2007) provide information for the organizations taking the logical step from IDS to IPS as the requirements on security front develop. Rødfoss's (2011) study is very similar to parts of this thesis in looking at network threats. The way Rødfoss's study was conducted was such that the results are not comparable with the results of this thesis. The problem is that Rødfoss's study did not use all available rulesets in their tests, thus leading to results that do not provide a fair comparison between solutions.

Harstein (2008) viewed the results of Snort against malicious programs with network traffic, showing 68% result for alerts that an administrator would take seriously. In this thesis, with much smaller test set for network attack threats, Suricata reached 60%, Snort 20% and Zeek 26% detection rate. Compared to the Harstein (2008), this test was arguably more difficult to the IDS, as the traffic produced was in great part based on human created traffic instead of being automatically created by a known malicious program. Network traffic of malicious program is often more easily detected via signatures and rules, as it is more regular. The low detection rate during the research was a surprise, especially in terms of Suricata vs. Snort, as both are based on a similar signature detection method. Part of the reason is likely related to more varied signature sources that Suricata provides out of the box and to the more modern detection engine.

One area of interest came up during the research relating to comparison of the products. As there is no way to have 100% security, security breaches in networks take place despite best efforts. When a such breach occurs, network forensics is a very important tool to investigate the breach and to get information about the attacker. All the IDSs provide extensive logging features to provide such forensic information. These capabilities to provide forensic information would be an interesting area of research when comparing different solutions and would provide more information to decision makers when selecting solutions for their network security.

7 Recommendation

Based on the whole thesis, all tests and the experience gained during it, the recommendation as single open source IDS solution for a corporation would be Suricata. It provided the best results out of the box and has the advantage of being built upon the shoulders of Snort. It is multi-threaded and can be used in networks where Snort and even Zeek might need additional installations to keep up. Suricata's detection engine also contains features that Snort does not support. The only weakness that Suricata really has is that it is based primarily on signatures. In an optimal case, there would be both mature anomaly detection and signature detection in the same tool. While Zeek did not do as well as Suricata in tests, Zeek has better capabilities to detect attacks with no known signature yet, as it is primarily an anomaly-based IDS.

If there are more resources available, a combination of Zeek and Suricata would provide more in-depth security than Suricata alone. To be complete, IDS solutions can be supplemented by a network security monitoring system that provides administrators clearer view to security events in their network. Therefore, the best result, as always in security is not achieved by use of one tool but by use of many tools together to provide a comprehensive view to the security environment in question.

REFERENCES

Bricata, 2018. *Suricata vs. Snort vs. Bro IDS.* Accessed on 29. October 2019 https://storage.pardot.com/350921/13942/Bricata_BroVsSnortSuricata_Whitepaper_2018.pdf

Caswell, U. Beale, J. Baker, A. 2007. *Snort Intrusion Detection and Prevention Toolkit.* Burlington, MA: Syngress Publishing Inc.

Cole, E., Krutz, R., Conley, J. 2005. *Network Security Bible*. John Wiley & Sons. E-book from Books24x7. Accessed on September 3, 2019 *http://common.books24x7.com.ezproxy.jamk.fi:2048/toc.aspx?bookid=12199*

Computer Security Institute. 2011. *2010/2011 Computer Crime and Security Survey*. Accessed on November 17, 2019. Retrieved from https://cours.etsmtl.ca/gti619/documents/divers/CSIsurvey2010.pdf

Department for Digital, Culture, Media & Sport. 2019. *Cyber Security Breaches Survey 2019*. Accessed on November 17, 2019. Retrieved from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/813599/Cyber_Security_Breaches_Survey_2019_-_Main_Report.pdf

Difference between IPS and IDS in network security. Ipwithease website, 2017. Accessed November 17, 2019. Retrieved from https://ipwithease.com/difference-between-ips-and-ids-in-network-security/

Hartstein, B. 2008. *Intrusion Detection Likelihood: A Risk-Based Approach*. SANS Institute. Accessed November 18, 2019. Retrieved from https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-likelihood-risk-based-approach-32938

Ivvala, A. 2017. *Assessment of Snort Intrusion Prevention Systems in Virtual Environment Against DoS and DDos Attacks*. Master's thesis, Institute of Technology. Blekinge Institute of Technology, Faculty of Computing. Accessed on 31. October 2019. Retrieved from http://www.diva-portal.org/smash/get/diva2:1085340/FULLTEXT02

Kaspersky, 2019. *What Is an Advanced Persistent Threat (APT)?.* Accessed October 4, 2019. Retrieved from https://usa.kaspersky.com/resource-center/definitions/advanced-persistent-threats

Khalil, G. 2015. *Open Source IDS High Performance Shootout*. SANS Institute. Accessed November 17, 2019. Retrieved from https://www.sans.org/reading-room/whitepapers/intrusion/open-source-ids-high-performance-shootout-35772

Khan Pathan, A. 2014. *The State of the Art in Intrusion Prevention and Detection. Auerbach Publications.* E-book from Books24x7. Accessed on October 4, 2019. Retrieved from http://common.books24x7.com.ezproxy.jamk.fi:2048/toc.aspx?bookid=61769, Books24x7.

Kim, D, & Solomon, M. 2014. *Fundamentals of Information Systems Security*, Second Edition. Jones and Bartlett Learning. E-book from books24x7. Accessed on January 29, 2019. Retrieved from http://common.books24x7.com.ezproxy.jamk.fi:2048/toc.aspx?bookid=69815, Books24x7

Klein, D., 2019. *Understanding the types of cyber threats on the rise in 2019*. Accessed on 29. January 2019. Retrieved from https://www.guardicore.com/2019/1/types-of-cyber-threats

MacDonnell, U. 2014. *Cyber Threat!: How to Manage the Growing Risk of Cyber Attacks*. New Jersey: John Wiley / Sons, incorporated.

NIST-800-94. *Guide to Intrusion Detection and Prevention Systems (IDPS)***.** Gaithersburg: National Institute of Standards and Technology. February 2007. Accessed on November 5, 2019. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf

Open Information Security Foundation. *Suricata homepage*. Accessed October 4, 2019. Retrieved from https://suricata-ids.org/

openSUSE Linux Report. 2019. Report page on Black Duck Inc. corporation's openhub website. Accessed November 18, 2019. https://www.openhub.net/p/opensuse-linux

OSS Watch. 2014. *How to evaluate the sustainability of an open source project*. Accessed November 17, 2019. https://opensource.com/life/14/1/evaluate-sustainability-open-source-project

OWASP. 2015. *OWASP Mutillidae 2 Project*. Accessed November 28, 2019. https://www.owasp.org/index.php/OWASP_Mutillidae_2_Project

Pihelgas, M. 2012. *A Comparative Analysis of Open-Source Intrusion Detection Systems*. Master's Thesis, University of Technology. Tallinn University of Technology, Department of Computer science. Accessed on 31.2019. Retrieved from http://mauno.pihelgas.eu/files/Mauno_Pihelgas-A_Comparative_Analysis_of_OpenSource_Intrusion_Detection_Systems.pdf

Rehman, R. 2003. *Intrusion Detection Systems with Snort*. New Jersey: Prentice Hall PTR.

Rødfoss, J. 2011. *Comparison of Open Source Network Intrusion Detection Systems*. Thesis, University College. Oslo University College, Network and System Administration. Accessed on 31. October 2019. Retrieved from https://www.duo.uio.no/bitstream/handle/10852/8951/Rodfoss.pdf

Schell, B. Martin, C. 2006. *Webster's New World Hacker Dictionary*. Indianapolis, IN: Wiley Publishing Inc.

Starting an Open Source Project. Page on GitHub's open source guide website. 2019. Accessed November 17, 2019. Retrieved from https://opensource.guide/starting-a-project/

Tight, M, 2017. *Understanding Case Study Research: Small-scale Research with Meaning*. Croydon: CPI Group (UK) Ltd.

What is Snort? Page on InfosecAddicts website, 2017. Accessed October 4, 2019. Retrieved from https://infosecaddicts.com/snort/

Cisco, 2018. *What Is the Difference: Viruses, Worms, Trojans, and Bots?* Accessed on October 4, 2019. Retrieved from https://tools.cisco.com/security/center/resources/virus_differences

White, J., Fitzsimmons, T. & Matthews, J. 2013. *Quantitative analysis of Intrusion Detection Systems: Snort and Suricata*. Accessed October 4, 2019. Retrieved from https://web2.clarkson.edu/class/cs644/ids/Paper/build/PDF/SnortVsSuricata.pdf, Clarkson University.

Weinstock, C & Hissam, S. 2005. *Making Lightning Strike Twice*. In publication Perspectives on Free and Open Source Software. Cambridge, MA: The MIT Press.

Zeek documentation, 2019. Accessed October 4, 2019. Retrieved from https://docs.zeek.org/en/stable/intro/

APPENDICES

Appendix 1

Network attack monitoring - DOS

TCP DOS

Threats: APT(Stage one, four), Worm, Virus, Bot.

TCP SYN flood attack against PfSense firewall with Metasploit module auxiliary/dos/tcp/synflood.rb

Attack command:

use auxiliary/dos/tcp/synflood.rb

set RHOSTS 192.168.56.140

exploit

Expected:

Alert relating to TCP, DOS, Flooding should be reported towards 192.168.56.140

Snort:

No results

Suricata:

No results

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	0
Zeek	0

Web server DOS

Threats: APT(stage four), Worm, Bot.

Apache Range Header DoS with Metasploit module auxiliary/dos/http/apache_range_dos.rb

Attack command:

use auxiliary/dos/http/apache_range_dos.rb

set RHOSTS 192.168.56.150

exploit

Expected:

Alert relating to Apache, Header, DOS, Flooding should be reported towards 192.168.56.150

Snort:

No results

Suricata:

No results

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	0
Zeek	0

OS service DOS

Threats: Worm, Virus, Bot.

SMBLoris NBSS Denial of Service attack against Ubuntu server Samba service with Metasploit module auxiliary/dos/smb/smb_loris.rb

Attack command:

use auxiliary/dos/smb/smb_loris.rb

set RHOSTS 192.168.56.150

exploit

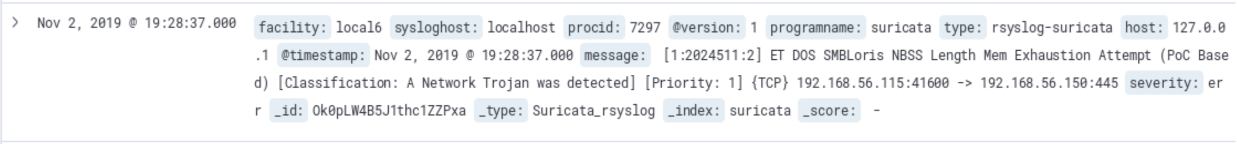
Expected:

Alert relating to Samba, DOS, Flooding should be reported towards 192.168.56.150

Snort:

No results

Suricata:



```
> Nov 2, 2019 @ 19:28:37.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                               .1 @timestamp: Nov 2, 2019 @ 19:28:37.000 message:  [1:2024511:2] ET DOS SMBLoris NBSS Length Mem Exhaustion Attempt (PoC Base
                               d) [Classification: A Network Trojan was detected] [Priority: 1] {TCP} 192.168.56.115:41600 -> 192.168.56.150:445 severity: er
                               r _id: Ok0pLW4B5J1thc1ZZPxa _type: Suricata_rsyslog _index: suricata _score:  -
```

1 hit

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	2
Zeek	0

Appendix 2

Network attack monitoring - TCP Hijacking

ARPspoof

Threats: APT(Stage four, five)

Arpspoof attack against kali-2 machine in 192.168.56.116

Attack command:

arpspoof -t 192.168.56.116 192.168.56.1

arpspoof -t 192.168.56.1 192.168.56.115

Expected:

Attack notification about ARP, spoofing or network related attack.

Snort:

No results

Suricata:

No results

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	0
Zeek	0

Appendix 3

Network attack monitoring - Port Scanning

NMAP portscan

Threats: APT(Stage four, five)

Network portscan with nmap scans the network for 100 most common ports.

Attack command:

nmap --top-ports 100 --script-args http.useragent="" -n 192.168.56.0/24

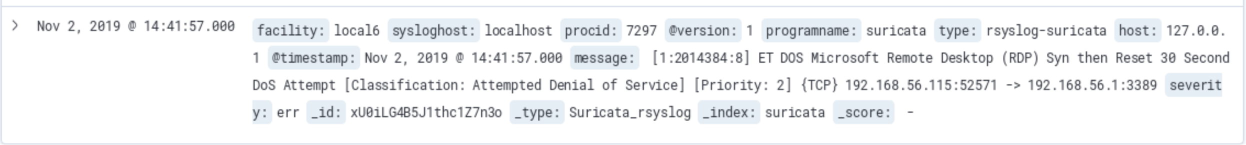
Expected:

Portscan should be reported

Snort:

No results

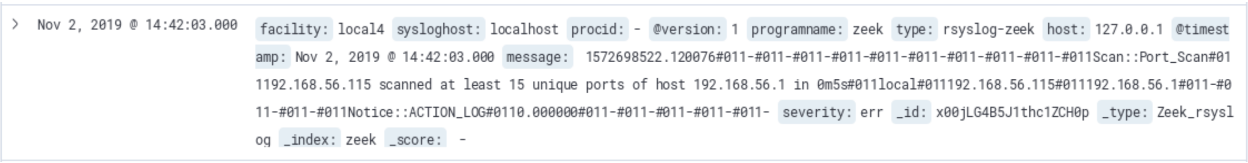
Suricata:



```
>  Nov 2, 2019 @ 14:41:57.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0.
                                1 @timestamp: Nov 2, 2019 @ 14:41:57.000 message:  [1:2014384:8] ET DOS Microsoft Remote Desktop (RDP) Syn then Reset 30 Second
                                DoS Attempt [Classification: Attempted Denial of Service] [Priority: 2] {TCP} 192.168.56.115:52571 -> 192.168.56.1:3389 severit
                                y: err _id: xU0iLG4B5J1thc1Z7n3o _type: Suricata_rsyslog _index: suricata _score:  -
```

1 hit with wrong attack reported

Zeek:



```
>  Nov 2, 2019 @ 14:42:03.000   facility: local4 sysloghost: localhost procid: - @version: 1 programname: zeek type: rsyslog-zeek host: 127.0.0.1 @timest
                                amp: Nov 2, 2019 @ 14:42:03.000 message:  1572698522.120076#011-#011-#011-#011-#011-#011-#011-#011-#011-#011Scan::Port_Scan#01
                                1192.168.56.115 scanned at least 15 unique ports of host 192.168.56.1 in 0m5s#011local#011192.168.56.115#011192.168.56.1#011-#0
                                11-#011-#011Notice::ACTION_LOG#0110.000000#011-#011-#011-#011-#011-  severity: err _id: x00jLG4B5J1thc1ZCH0p _type: Zeek_rsysl
                                og _index: zeek _score:  -
```

7 hits

Scoring:

	Points
Snort	0
Suricata	1
Zeek	2

Dmitry

Threats: APT(stage four, five)

Network portscan with dmitry scans IP 192.168.56.150 TCP ports.

Attack command:

dmitry -p 192.168.56.150

Expected:

Portscan should be reported towards 192.168.56.150

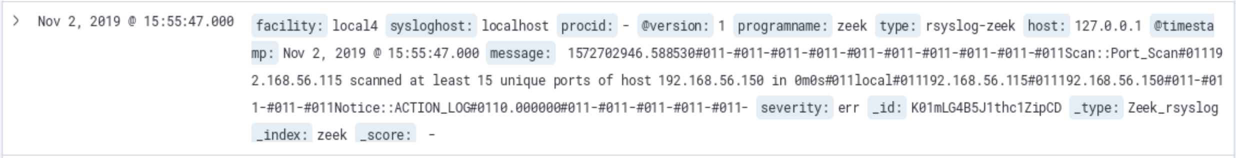
Snort:

No results

Suricata:

No results

Zeek:



```
>  Nov 2, 2019 @ 15:55:47.000   facility: local4 sysloghost: localhost procid: - @version: 1 programname: zeek type: rsyslog-zeek host: 127.0.0.1 @timesta
                               mp: Nov 2, 2019 @ 15:55:47.000 message:  1572702946.588530#011-#011-#011-#011-#011-#011-#011-#011-#011-#011Scan::Port_Scan#01119
                               2.168.56.115 scanned at least 15 unique ports of host 192.168.56.150 in 0m0s#011local#011192.168.56.115#011192.168.56.150#011-#01
                               1-#011-#011Notice::ACTION_LOG#0110.000000#011-#011-#011-#011-#011-  severity: err _id: K01mLG4B5J1thc1ZipCD _type: Zeek_rsyslog
                               _index: zeek _score:  -
```

1 hit

Scoring:

	Points
Snort	0
Suricata	0
Zeek	2

Fingerprinting

Threats: APT(stage three, four, five)

Service fingerprint attack against Ubuntu server with nmap

Attack command:

nmap --script-args http.useragent="" -sV 192.168.56.150

Expected:

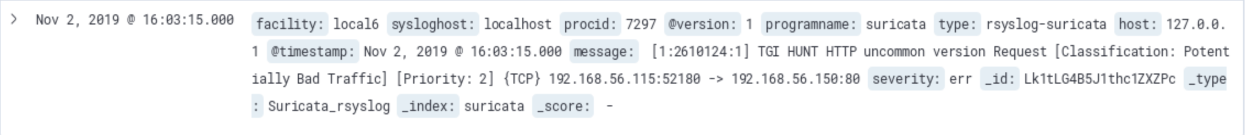
Portscan or fingerprint related attack should be reported towards 192.168.56.150

Result:

Snort:

No results

Suricata:



```
> Nov 2, 2019 @ 16:03:15.000  facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0.
                              1 @timestamp: Nov 2, 2019 @ 16:03:15.000 message:  [1:2610124:1] TGI HUNT HTTP uncommon version Request [Classification: Potent
                              ially Bad Traffic] [Priority: 2] {TCP} 192.168.56.115:52180 -> 192.168.56.150:80 severity: err _id: Lk1tLG4B5J1thc1ZXZPc _type
                              : Suricata_rsyslog _index: suricata _score:  -
```

1 hit

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	2
Zeek	0

Appendix 4

Network attack monitoring - Password guessing

Hydra – Web Server

Threats: APT(one, three, four, five), Bot, Worm

Web application password brute force attack against OWASP Broken Web applications Mutillidae using Hydra

Attack command:

hydra -l muhvi -P /usr/share/wordlists/fasttrack.txt 192.168.56.103 http-post-form "/mutillidae/login.php:username=^USER^&password=^PASS^&Login=Login:Password incorrect

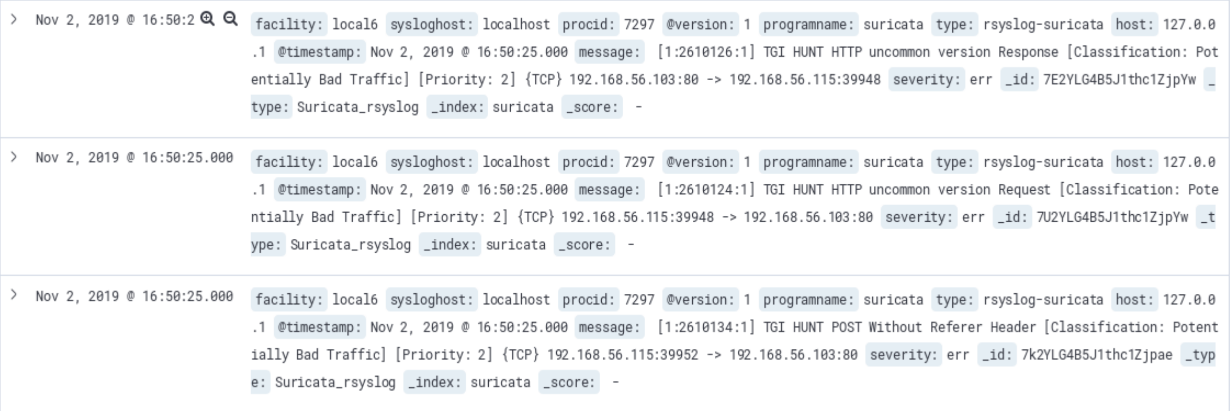
Expected:

Brute force related attack should be reported towards 192.168.56.103

Snort:

No results

Suricata:



```
> Nov 2, 2019 @ 16:50:2   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                          .1 @timestamp: Nov 2, 2019 @ 16:50:25.000 message:  [1:2610126:1] TGI HUNT HTTP uncommon version Response [Classification: Pot
                          entially Bad Traffic] [Priority: 2] {TCP} 192.168.56.103:80 -> 192.168.56.115:39948 severity: err _id: 7E2YLG4B5J1thc1ZjpYw _
                          type: Suricata_rsyslog _index: suricata _score:  -

> Nov 2, 2019 @ 16:50:25.000  facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                          .1 @timestamp: Nov 2, 2019 @ 16:50:25.000 message:  [1:2610124:1] TGI HUNT HTTP uncommon version Request [Classification: Pote
                          ntially Bad Traffic] [Priority: 2] {TCP} 192.168.56.115:39948 -> 192.168.56.103:80 severity: err _id: 7U2YLG4B5J1thc1ZjpYw _t
                          ype: Suricata_rsyslog _index: suricata _score:  -

> Nov 2, 2019 @ 16:50:25.000  facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                          .1 @timestamp: Nov 2, 2019 @ 16:50:25.000 message:  [1:2610134:1] TGI HUNT POST Without Referer Header [Classification: Potent
                          ially Bad Traffic] [Priority: 2] {TCP} 192.168.56.115:39952 -> 192.168.56.103:80 severity: err _id: 7k2YLG4B5J1thc1Zjpae _typ
                          e: Suricata_rsyslog _index: suricata _score:  -
```

3 hits

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	1
Zeek	0

Hydra – OS service

Threats: APT(stage three, four, five)

SSH password brute force attack against Ubuntu server using Hydra

Command used:

hydra -l root -P /usr/share/wordlists/fasttrack.txt 192.168.56.150 -t 4 ssh

Expected:

Brute force related attack should be reported towards 192.168.56.150

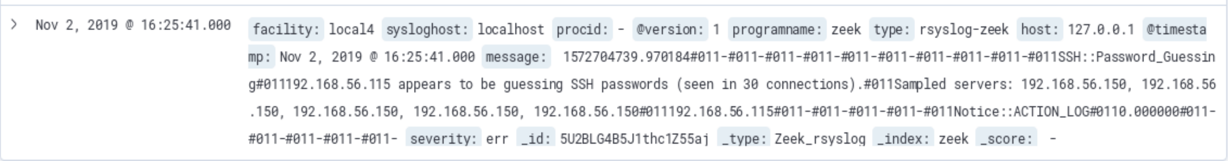
Snort:

No results

Suricata:

No results

Zeek:



```
> Nov 2, 2019 @ 16:25:41.000   facility: local4 sysloghost: localhost procid: - @version: 1 programname: zeek type: rsyslog-zeek host: 127.0.0.1 @timesta
                               mp: Nov 2, 2019 @ 16:25:41.000 message:  1572704739.970184#011-#011-#011-#011-#011-#011-#011-#011-#011-#011SSH::Password_Guessin
                               g#011192.168.56.115 appears to be guessing SSH passwords (seen in 30 connections).#011Sampled servers: 192.168.56.150, 192.168.56
                               .150, 192.168.56.150, 192.168.56.150, 192.168.56.150#011192.168.56.115#011-#011-#011-#011-#011Notice::ACTION_LOG#0110.000000#011-
                               #011-#011-#011-#011- severity: err _id: 5U2BLG4B5J1thc1Z55aj _type: Zeek_rsyslog _index: zeek _score:  -
```

1hit

Scoring:

	Points
Snort	0
Suricata	0
Zeek	2

Appendix 5

Network attack monitoring - Software exploitation

NMAP - Vulnerability scan

Threats: APT(Stage one, four, five)

Vulnerability scan is done by using NMAP -vuln script that scans server for many known vulnerabilities.

Attack command:

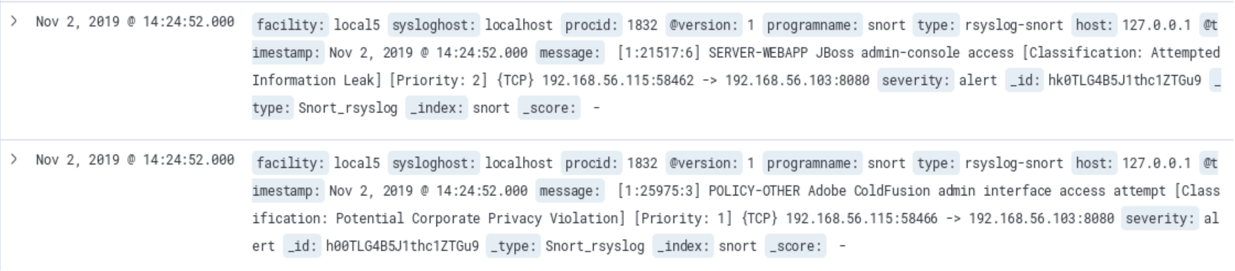
nmap -Pn --script vuln --script-args http.useragent=”” 192.168.56.103

Expected:

Several different attacks should be reported

Result:

Snort

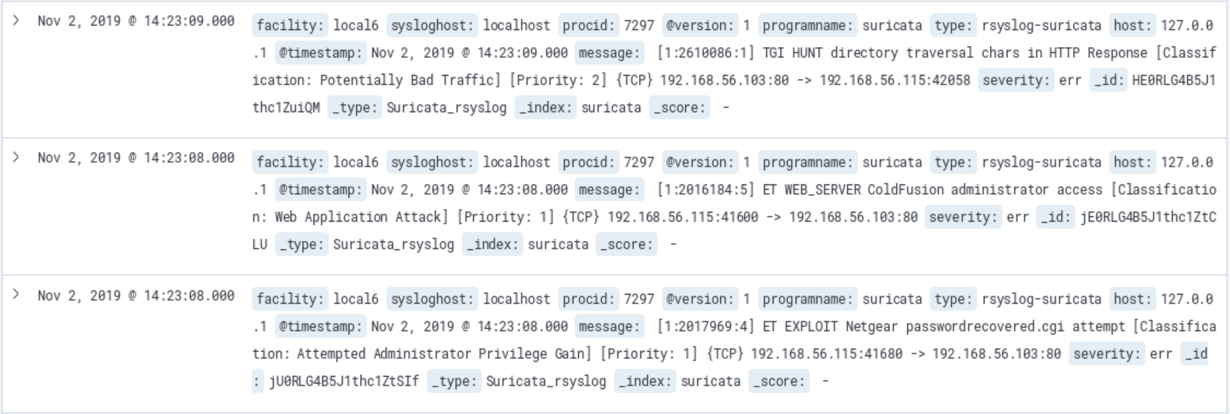


```
> Nov 2, 2019 @ 14:24:52.000  facility: local5 sysloghost: localhost procid: 1832 @version: 1 programname: snort type: rsyslog-snort host: 127.0.0.1 @t
                              imestamp: Nov 2, 2019 @ 14:24:52.000 message:  [1:21517:6] SERVER-WEBAPP JBoss admin-console access [Classification: Attempted
                              Information Leak] [Priority: 2] {TCP} 192.168.56.115:58462 -> 192.168.56.103:8080 severity: alert _id: hk0TLG4B5J1thc1ZTGu9 _
                              type: Snort_rsyslog _index: snort _score:  -

> Nov 2, 2019 @ 14:24:52.000  facility: local5 sysloghost: localhost procid: 1832 @version: 1 programname: snort type: rsyslog-snort host: 127.0.0.1 @t
                              imestamp: Nov 2, 2019 @ 14:24:52.000 message:  [1:25975:3] POLICY-OTHER Adobe ColdFusion admin interface access attempt [Class
                              ification: Potential Corporate Privacy Violation] [Priority: 1] {TCP} 192.168.56.115:58466 -> 192.168.56.103:8080 severity: al
                              ert _id: h00TLG4B5J1thc1ZTGu9 _type: Snort_rsyslog _index: snort _score:  -
```

22 hits

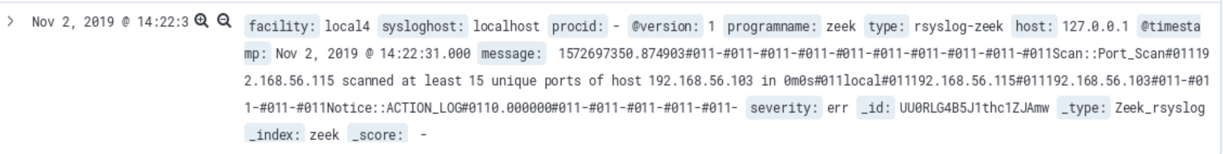
Suricata



```
> Nov 2, 2019 @ 14:23:09.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                               .1 @timestamp: Nov 2, 2019 @ 14:23:09.000 message:  [1:2610086:1] TGI HUNT directory traversal chars in HTTP Response [Classif
                               ication: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.56.103:80 -> 192.168.56.115:42058 severity: err _id: HE0RLG4B5J1
                               thc1ZuiQM _type: Suricata_rsyslog _index: suricata _score:  -
> Nov 2, 2019 @ 14:23:08.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                               .1 @timestamp: Nov 2, 2019 @ 14:23:08.000 message:  [1:2016184:5] ET WEB_SERVER ColdFusion administrator access [Classificatio
                               n: Web Application Attack] [Priority: 1] {TCP} 192.168.56.115:41600 -> 192.168.56.103:80 severity: err _id: jE0RLG4B5J1thc1ZtC
                               LU _type: Suricata_rsyslog _index: suricata _score:  -
> Nov 2, 2019 @ 14:23:08.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                               .1 @timestamp: Nov 2, 2019 @ 14:23:08.000 message:  [1:2017969:4] ET EXPLOIT Netgear passwordrecovered.cgi attempt [Classifica
                               tion: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.56.115:41680 -> 192.168.56.103:80 severity: err _id
                               : jU0RLG4B5J1thc1ZtSIf _type: Suricata_rsyslog _index: suricata _score:  -
```

30 hits

Zeek



```
> Nov 2, 2019 @ 14:22:3   facility: local4 sysloghost: localhost procid: - @version: 1 programname: zeek type: rsyslog-zeek host: 127.0.0.1 @timesta
                          mp: Nov 2, 2019 @ 14:22:31.000 message:  1572697350.874903#011-#011-#011-#011-#011-#011-#011-#011-#011-#011Scan::Port_Scan#01119
                          2.168.56.115 scanned at least 15 unique ports of host 192.168.56.103 in 0m0s#011local#011192.168.56.115#011192.168.56.103#011-#01
                          1-#011-#011Notice::ACTION_LOG#0110.000000#011-#011-#011-#011-#011- severity: err _id: UU0RLG4B5J1thc1ZJAmw _type: Zeek_rsyslog
                          _index: zeek _score:  -
```

1 hit

Scoring:

	Points
Snort	2
Suricata	2
Zeek	1

SQLMAP SQL injection

Threats: APT(stage one, three, four, five)

SQL injection attack against OWASP Broken Web Applications Mutillidae using SQLMAP

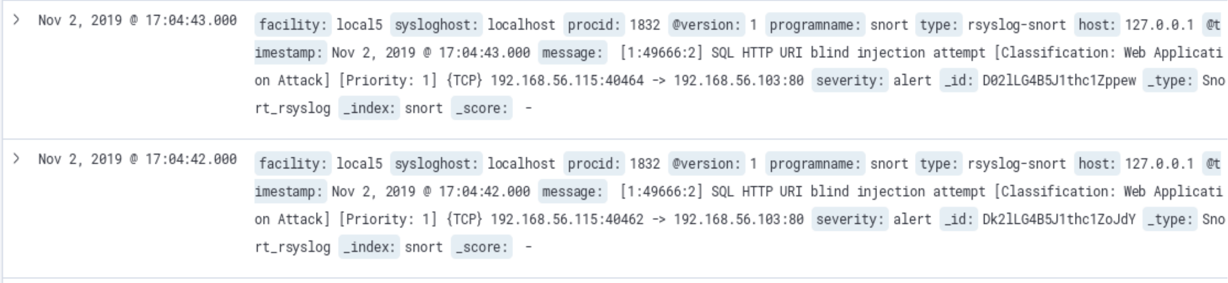
Attack command:

sqlmap -u "http://192.168.56.103/mutillidae/index.php?page=user-info.php" --data="username=muhvi"

Expected:

SQL injection related attack should be reported towards 192.168.56.103

Snort:

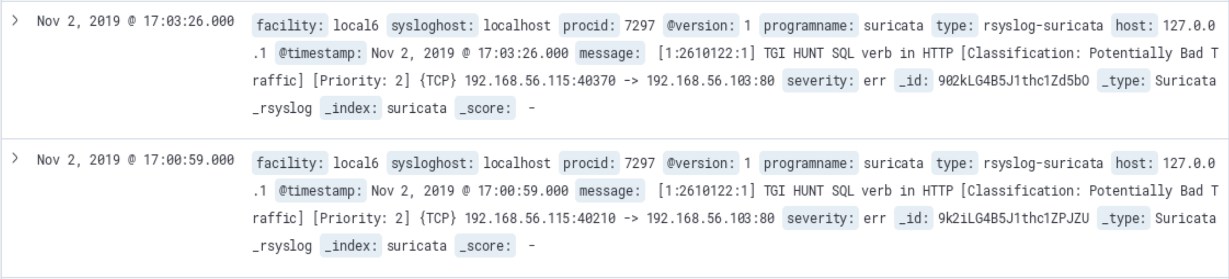


```
>  Nov 2, 2019 @ 17:04:43.000   facility: local5 sysloghost: localhost procid: 1832 @version: 1 programname: snort type: rsyslog-snort host: 127.0.0.1 @t
                                imestamp: Nov 2, 2019 @ 17:04:43.000 message:  [1:49666:2] SQL HTTP URI blind injection attempt [Classification: Web Applicati
                                on Attack] [Priority: 1] {TCP} 192.168.56.115:40464 -> 192.168.56.103:80 severity: alert _id: D02lLG4B5J1thc1Zppew _type: Sno
                                rt_rsyslog _index: snort _score:  -

>  Nov 2, 2019 @ 17:04:42.000   facility: local5 sysloghost: localhost procid: 1832 @version: 1 programname: snort type: rsyslog-snort host: 127.0.0.1 @t
                                imestamp: Nov 2, 2019 @ 17:04:42.000 message:  [1:49666:2] SQL HTTP URI blind injection attempt [Classification: Web Applicati
                                on Attack] [Priority: 1] {TCP} 192.168.56.115:40462 -> 192.168.56.103:80 severity: alert _id: Dk2lLG4B5J1thc1ZoJdY _type: Sno
                                rt_rsyslog _index: snort _score:  -
```

23 hits

Suricata:



```
>  Nov 2, 2019 @ 17:03:26.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                                .1 @timestamp: Nov 2, 2019 @ 17:03:26.000 message:  [1:2610122:1] TGI HUNT SQL verb in HTTP [Classification: Potentially Bad T
                                raffic] [Priority: 2] {TCP} 192.168.56.115:40370 -> 192.168.56.103:80 severity: err _id: 902kLG4B5J1thc1Zd5bO _type: Suricata
                                _rsyslog _index: suricata _score:  -

>  Nov 2, 2019 @ 17:00:59.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                                .1 @timestamp: Nov 2, 2019 @ 17:00:59.000 message:  [1:2610122:1] TGI HUNT SQL verb in HTTP [Classification: Potentially Bad T
                                raffic] [Priority: 2] {TCP} 192.168.56.115:40210 -> 192.168.56.103:80 severity: err _id: 9k2iLG4B5J1thc1ZPJZU _type: Suricata
                                _rsyslog _index: suricata _score:  -
```

7 hits

Zeek:

No results

Scoring:

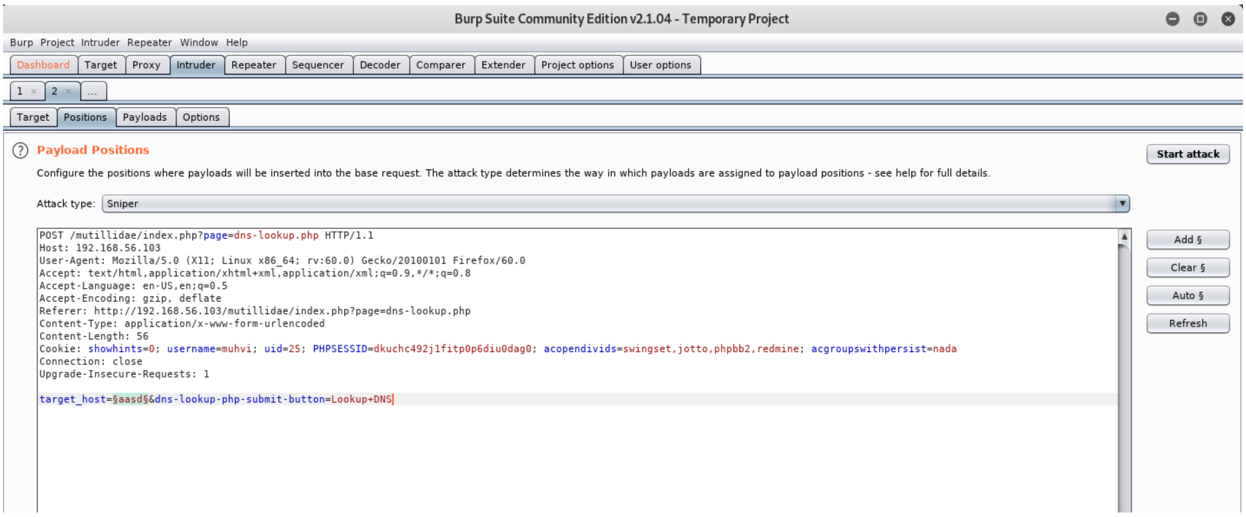
	Points
Snort	2
Suricata	2
Zeek	0

Burp Suite Parameter fuzzing

Threats: APT(stage one, four, five)

Parameter fuzzing attack against OWASP Broken Web Applications Mutillidae using Burp Suite

Attack:


Burp Suite Community Edition v2.1.04 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
1 × 2 × ...
Target Positions Payloads Options
Start attack
Add §
Clear §
Auto §
Refresh

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
Host: 192.168.56.103
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.103/mutillidae/index.php?page=dns-lookup.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Cookie: showhints=0; username=muhvi; uid=25; PHPSESSID=dkuchc492j1fitp0p6diu0dag0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1

target_host=§aasd§&dns-lookup-php-submit-button=Lookup+DNS
```

Wordlist used:

/usr/share/wfuzz/wordlist/Injections/

bad_chars.txt

XML.txt

XSS.txt

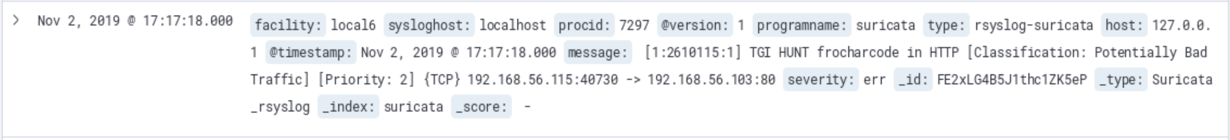
Expected:

Alert about injection, characters or http related attack should be reported towards 192.168.56.103

Snort:

No results

Suricata:



```
>  Nov 2, 2019 @ 17:17:18.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0.
                              1 @timestamp: Nov 2, 2019 @ 17:17:18.000 message:  [1:2610115:1] TGI HUNT frocharcode in HTTP [Classification: Potentially Bad
                              Traffic] [Priority: 2] {TCP} 192.168.56.115:40730 -> 192.168.56.103:80 severity: err _id: FE2xLG4B5J1thc1ZK5eP _type: Suricata
                              _rsyslog _index: suricata _score:  -
```

1 hit

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	2
Zeek	0

Nikto Forced directory browsing

Threats: APT(stage one, four, five), Worm, Bot

Forced directory browsing – directory enumeration against Ubuntu server Nikto

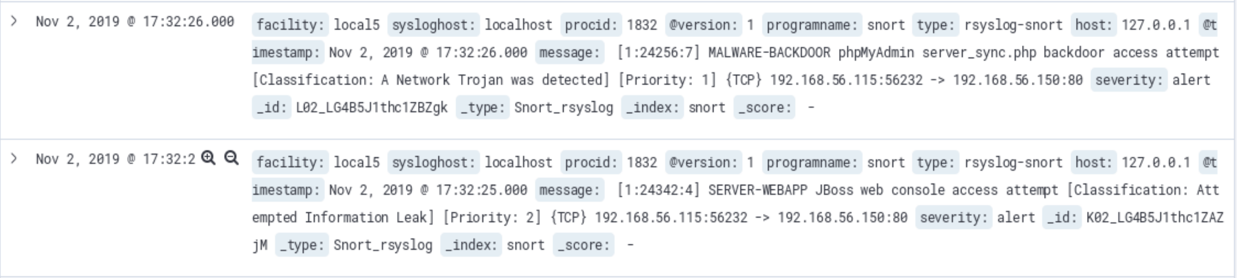
Attack command:

nikto -h 192.168.56.150

Expected:

Alert about scanning, vulnerabilities, information gathering or http should be reported towards 192.168.56.150

Snort:

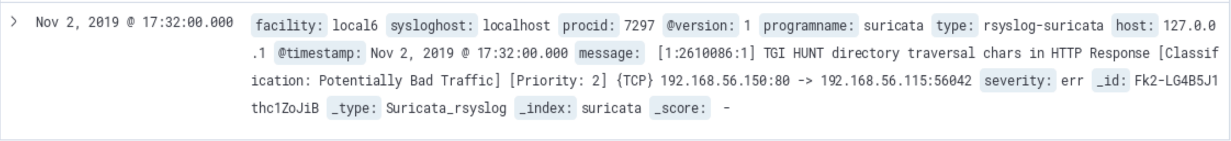


```
> Nov 2, 2019 @ 17:32:26.000   facility: local5 sysloghost: localhost procid: 1832 @version: 1 programname: snort type: rsyslog-snort host: 127.0.0.1 @t
                               imestamp: Nov 2, 2019 @ 17:32:26.000 message:  [1:24256:7] MALWARE-BACKDOOR phpMyAdmin server_sync.php backdoor access attempt
                               [Classification: A Network Trojan was detected] [Priority: 1] {TCP} 192.168.56.115:56232 -> 192.168.56.150:80 severity: alert
                               _id: L02_LG4B5J1thc1ZBZgk _type: Snort_rsyslog _index: snort _score:  -

> Nov 2, 2019 @ 17:32:2        facility: local5 sysloghost: localhost procid: 1832 @version: 1 programname: snort type: rsyslog-snort host: 127.0.0.1 @t
                               imestamp: Nov 2, 2019 @ 17:32:25.000 message:  [1:24342:4] SERVER-WEBAPP JBoss web console access attempt [Classification: Att
                               empted Information Leak] [Priority: 2] {TCP} 192.168.56.115:56232 -> 192.168.56.150:80 severity: alert _id: K02_LG4B5J1thc1ZAZ
                               jM _type: Snort_rsyslog _index: snort _score:  -
```

135 hits

Suricata:



```
> Nov 2, 2019 @ 17:32:00.000   facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                               .1 @timestamp: Nov 2, 2019 @ 17:32:00.000 message:  [1:2610086:1] TGI HUNT directory traversal chars in HTTP Response [Classif
                               ication: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.56.150:80 -> 192.168.56.115:56042 severity: err _id: Fk2-LG4B5J1
                               thc1ZoJiB _type: Suricata_rsyslog _index: suricata _score:  -
```

133 hits

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	2
Zeek	0

Linux Samba vulnerability exploit

Threats: APT(stage three, four, five), Worm

Linux Samba vulnerability attack with Metasploit module exploits/linux/samba/setinfopolicy_heap.rb

Attack command:

use exploits/linux/samba/setinfopolicy_heap.rb

set RHOSTS 192.168.56.150

set SMBUser <username>

set SMBPass <password>

exploit

Expected:

Alert relating to samba or connections related issues to its ports 139 or 445 should be reported towards 192.168.56.150

Snort:

No results

Suricata:

No results

Zeek:

No results

Scoring:

	Points
Snort	0
Suricata	0
Zeek	0

Linux Apache/Bash vulnerability exploit

Threats: APT(two, three, four, five) Worm, Bot

Linux bash vulnerability attack through Apache CGI with Metasploit module exploits/multi/http/apache_mod_cgi_bash_env_exec.rb

Attack command:

use exploits/multi/http/apache_mod_cgi_bash_env_exec.rb

set RHOSTS 192.168.56.103

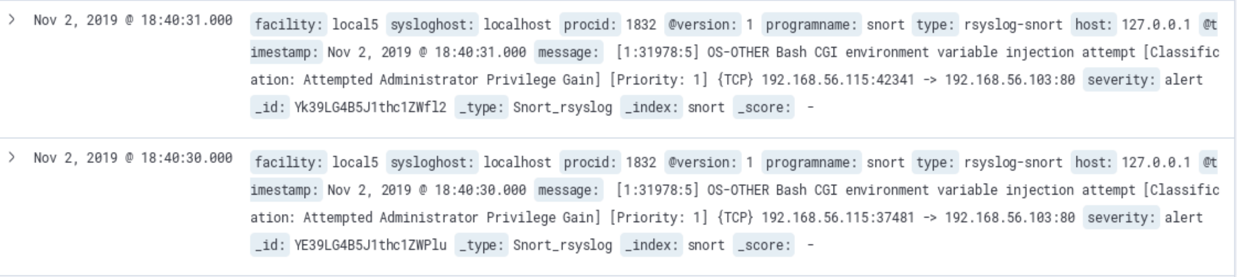
set TARGETURI /cgi-bin/<somecgi.cgi>

exploit

Expected:

Alert relating to Apache, CVE-2014-6271, CVE-2014-6278, bash or CGI should be reported towards 192.168.56.150

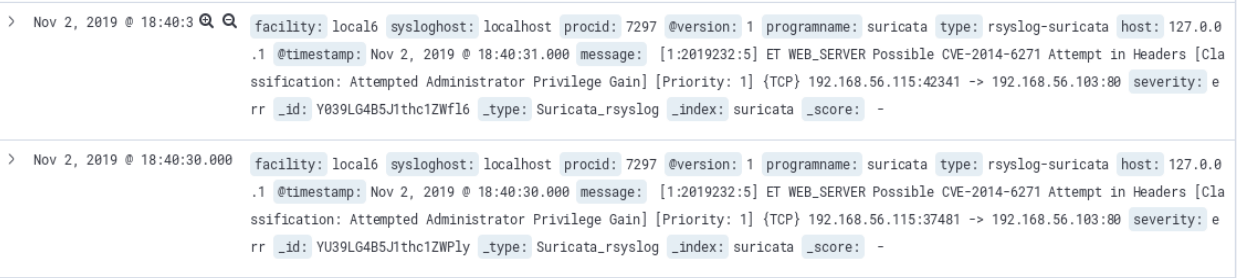
Snort:



```
> Nov 2, 2019 @ 18:40:31.000  facility: local5 sysloghost: localhost procid: 1832 @version: 1 programname: snort type: rsyslog-snort host: 127.0.0.1 @t
                              imestamp: Nov 2, 2019 @ 18:40:31.000 message:  [1:31978:5] OS-OTHER Bash CGI environment variable injection attempt [Classific
                              ation: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.56.115:42341 -> 192.168.56.103:80 severity: alert
                              _id: Yk39LG4B5J1thc1ZWfl2 _type: Snort_rsyslog _index: snort _score:  -
> Nov 2, 2019 @ 18:40:30.000  facility: local5 sysloghost: localhost procid: 1832 @version: 1 programname: snort type: rsyslog-snort host: 127.0.0.1 @t
                              imestamp: Nov 2, 2019 @ 18:40:30.000 message:  [1:31978:5] OS-OTHER Bash CGI environment variable injection attempt [Classific
                              ation: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.56.115:37481 -> 192.168.56.103:80 severity: alert
                              _id: YE39LG4B5J1thc1ZWPlu _type: Snort_rsyslog _index: snort _score:  -
```

2 hits

Suricata:



```
> Nov 2, 2019 @ 18:40:3       facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                              .1 @timestamp: Nov 2, 2019 @ 18:40:31.000 message:  [1:2019232:5] ET WEB_SERVER Possible CVE-2014-6271 Attempt in Headers [Cla
                              ssification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.56.115:42341 -> 192.168.56.103:80 severity: e
                              rr _id: Y039LG4B5J1thc1ZWfl6 _type: Suricata_rsyslog _index: suricata _score:  -
> Nov 2, 2019 @ 18:40:30.000  facility: local6 sysloghost: localhost procid: 7297 @version: 1 programname: suricata type: rsyslog-suricata host: 127.0.0
                              .1 @timestamp: Nov 2, 2019 @ 18:40:30.000 message:  [1:2019232:5] ET WEB_SERVER Possible CVE-2014-6271 Attempt in Headers [Cla
                              ssification: Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.56.115:37481 -> 192.168.56.103:80 severity: e
                              rr _id: YU39LG4B5J1thc1ZWPly _type: Suricata_rsyslog _index: suricata _score:  -
```

2 hits

Zeek:

No results

Scoring:

	Points
Snort	2
Suricata	2
Zeek	0

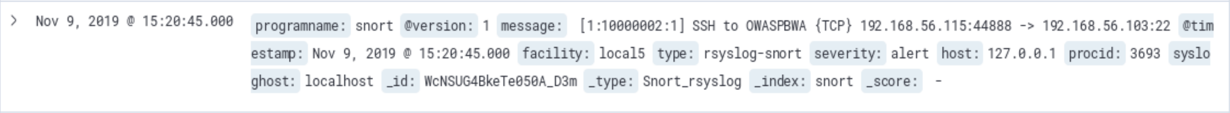
Appendix 6

Configuration

Custom rule for network traffic

SSH connection from Any host in the network to OWASP BWA creates an alert or a notice.

Snort:



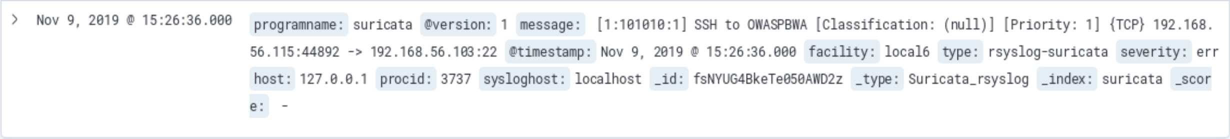
```
> Nov 9, 2019 @ 15:20:45.000  programname: snort @version: 1 message:  [1:10000002:1] SSH to OWASPBWA {TCP} 192.168.56.115:44888 -> 192.168.56.103:22 @tim
                              estamp: Nov 9, 2019 @ 15:20:45.000 facility: local5 type: rsyslog-snort severity: alert host: 127.0.0.1 procid: 3693 syslo
                              ghost: localhost _id: WcNSUG4BkeTe050A_D3m _type: Snort_rsyslog _index: snort _score:  -
```

Rule:

alert tcp any any -> 192.168.56.103 22 (msg:”SSH to OWASPBWA”; sid:1000002; rev:001;)

Time to create: 16min

Suricata:



```
> Nov 9, 2019 @ 15:26:36.000  programname: suricata @version: 1 message:  [1:101010:1] SSH to OWASPBWA [Classification: (null)] [Priority: 1] {TCP} 192.168.
                              56.115:44892 -> 192.168.56.103:22 @timestamp: Nov 9, 2019 @ 15:26:36.000 facility: local6 type: rsyslog-suricata severity: err
                              host: 127.0.0.1 procid: 3737 sysloghost: localhost _id: fsNYUG4BkeTe050AWD2z _type: Suricata_rsyslog _index: suricata _scor
                              e:  -
```

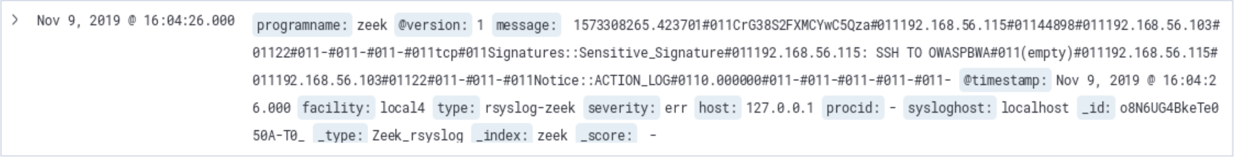
Rule:

alert tcp any any -> 192.168.56.103 22 (msg:”SSH TO OWASPBWA”; priority:1; sid:101010; rev:1;)

Time to create: 16min

As the rule syntax for Suricata is so similar to Snort, I deem the time same for Suricata as for Snort. After noticing the similar rule syntax, the rule for Suricata took only few minutes to make.

Zeek:



```
> Nov 9, 2019 @ 16:04:26.000   programname: zeek @version: 1 message:  1573308265.423701#011CrG38S2FXMCYwC5Qza#011192.168.56.115#01144898#011192.168.56.103#
                               01122#011-#011-#011-#011tcp#011Signatures::Sensitive_Signature#011192.168.56.115: SSH TO OWASPBWA#011(empty)#011192.168.56.115#
                               011192.168.56.103#01122#011-#011-#011Notice::ACTION_LOG#0110.000000#011-#011-#011-#011-#011- @timestamp: Nov 9, 2019 @ 16:04:2
                               6.000 facility: local4 type: rsyslog-zeek severity: err host: 127.0.0.1 procid: - sysloghost: localhost _id: o8N6UG4BkeTe0
                               50A-T0_ _type: Zeek_rsyslog _index: zeek _score:  -
```

Rule:

signature ssh {

ip-proto == tcp

dst-ip == 192.168.56.103

dst-port == 22

event “SSH TO OWASPBWA”

}

Time to create: 30min

Scoring:

	Points
Snort	3
Suricata	3
Zeek	1

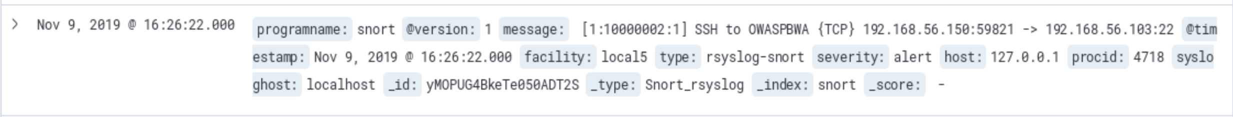
Network traffic rule exception

SSH connection from Kali-1 and Kali-2 to OWASP BWA 192.168.56.103 will not create an alert or a notice

Snort:

From kali-1 and kali-2 : No results

From Ubuntuserver:



```
> Nov 9, 2019 @ 16:26:22.000  programname: snort @version: 1 message:  [1:10000002:1] SSH to OWASPBWA {TCP} 192.168.56.150:59821 -> 192.168.56.103:22 @tim
                              estamp: Nov 9, 2019 @ 16:26:22.000 facility: local5 type: rsyslog-snort severity: alert host: 127.0.0.1 procid: 4718 syslo
                              ghost: localhost _id: yMOPUG4BkeTe050ADT2S _type: Snort_rsyslog _index: snort _score:  -
```

Rule:

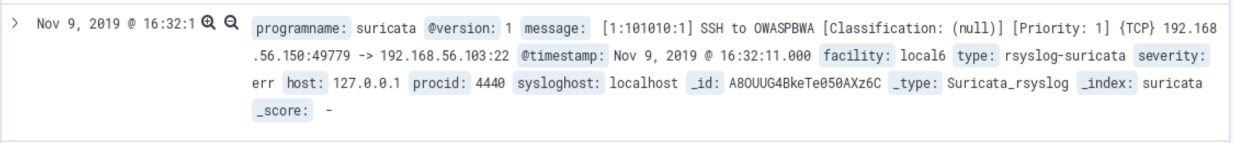
alert tcp ![192.168.56.115,192.168.56.116] any -> 192.168.56.103 22 (msg:"SSH TO OWASPBWA"; sid:10000002; rev:001;)

Time to create: 5min

Suricata:

From kali-1 and kali-2: No results

From Ubuntuserver:



```
> Nov 9, 2019 @ 16:32:1  programname: suricata @version: 1 message:  [1:101010:1] SSH to OWASPBWA [Classification: (null)] [Priority: 1] {TCP} 192.168
                         .56.150:49779 -> 192.168.56.103:22 @timestamp: Nov 9, 2019 @ 16:32:11.000 facility: local6 type: rsyslog-suricata severity:
                         err host: 127.0.0.1 procid: 4440 sysloghost: localhost _id: A8OUUG4BkeTe050AXz6C _type: Suricata_rsyslog _index: suricata
                         _score:  -
```

Rule:

alert tcp any any -> 192.168.56.103 22 (msg:"SSH TO OWASPBWA"; priority:1; sid:101010; rev:1;)

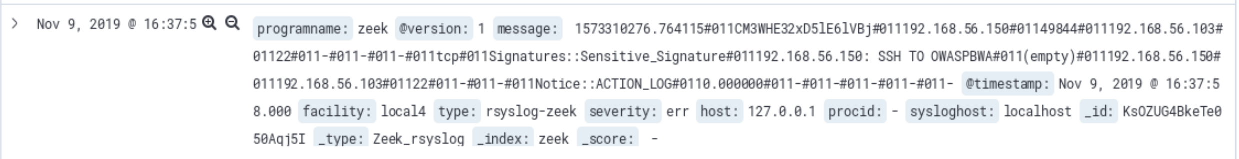
Time to create: 5min

The rule syntax with Suricata is so similar with Snort so same time is recorded with Suricata as with Snort

Zeek:

From kali-1 and kali-2: No results

From Ubuntuserver:



```
> Nov 9, 2019 @ 16:37:5   programname: zeek @version: 1 message:  1573310276.764115#011CM3WHE32xD5lE6lVBj#011192.168.56.150#01149844#011192.168.56.103#
                          01122#011-#011-#011-#011tcp#011Signatures::Sensitive_Signature#011192.168.56.150: SSH TO OWASPBWA#011(empty)#011192.168.56.150#
                          011192.168.56.103#01122#011-#011-#011Notice::ACTION_LOG#0110.000000#011-#011-#011-#011-#011- @timestamp: Nov 9, 2019 @ 16:37:5
                          8.000 facility: local4 type: rsyslog-zeek severity: err host: 127.0.0.1 procid: - sysloghost: localhost _id: KsOZUG4BkeTe0
                          50Aqj5I _type: Zeek_rsyslog _index: zeek _score:  -
```

Rule:

signature ssh {

ip-proto == tcp

dst-ip == 192.168.56.103

dst-port == 22

event "SSH TO OWASPBWA"

}

Time to create: 5min

Scoring

	Points
Snort	3
Suricata	3
Zeek	3

Appendix 7

Software maintenance and development

Version updates

We look at version histories to see how active the development has been between last major versions and after latest major version.

Snort:

Major releases:

2.9.0 in 2011

Snort has been developing on 2.9.x versions since 2011 and in last few years the updates have been farther between and have included mostly bug fixes. 2019 had three smaller releases, but 2018 only one release and 2017 three releases. Snort 3 has been in beta since 09/2018 but stable 3.0 release date is not yet available. While development seems active, it does not seem as active as with Suricata and Zeek.

Contributors in last 12 month period: 13

Suricata:

Major releases:

5.0 in 2019

4.0 in 2017

3.0 in 2016

Very active development with many version releases every year between 5.0 and 4.0 providing features and bug fixes. 5.0 released very recently in October 2019 so no history data available of post 5.0 development.

Contributors in last 12 month period: 36

Zeek:

Major releases:

3.0 in 2019

2.0 in 2012

1.0 in 2005

Very active development with many version releases every year between 2.0 and 3.0 providing features and bug fixes. Active development after release of 3.0.

Contributors in last 12 month period: 71

Scoring:

	Points
Snort	1
Suricata	3
Zeek	3

Appendix 8

Alert outputs

Comparison of alert outputs

Alert outputs of the IDS were collected from documentation and table of supported outputs was created:

Output	Snort	Suricata	Zeek
syslog	X	X	
Ascii	X	X	X
JSON		X	X
CSV	X		
TCPdump	X		
Unified2	X	X	
Unixsock	X		
SQLite			X

Syslog is well known and widely used standard for logging.

Ascii refers to normal text file logs

JSON, JavaScript Object Notation, is widely used file format that is easily readable

CSV, comma separated values, simple delimited text file format.

TCPdump, common file format for analyzing network traffic, used by many network sniffer tools.

Unified2, output data format used in IDS tools to provide additional data along normal event as unified output

Unixsock, output is provided via UNIX domain socket

SQLite, file-based SQL database system

Evaluation of outputs

Snort provides the greatest amount of logging options, but it is lacking JSON output, which is very common way to parse logs and supported by wide variety of logging solutions. Suricata has smaller number of logging outputs than Snort, but provides very extensive JSON logging out of the box. Zeek has very limited logging options out of the box, while it supports JSON, the JSON output does not have as extensive features as Suricata's JSON output. Zeek also lacks native support for syslog, which is industry standard in log messaging.

Scoring:

	Points
Snort	3
Suricata	3
Zeek	1