Authors: Olajuwon Olawale Faleke and Philip Jeremiah Kadiri

Preforming network intrusion detection using Microsoft azure and open-source tools

25/10/2022

Table of contents

List of Figures

Introduction

This experiment will practically demonstrates how to setup and configure: a logging and data visualization stack called the Elastic Stack v5.2.0, and how to setup, configure, and parse an intrusion detection system's (Suricata) JSON log output into that stack. The experiment will also review the QakBot malware PCAP traffic, analyze the infection process using the aforementioned open source tools, suggest recommendations, and document the reflection on the procedure.

Technical Report

This technical report contains the configuration, setup, and use of the Elastic Stack and Suricata. Additionally, it aims to analyze the QakBot malware infection process and any other malicious traffic.

Setting up an Azure Virtual Machine

A free account has to be created in order to utilize Azure's services, as exhibited in *Appendix A*, Figures 63 to 74. The account created came with $200 worth of credits and free access to Azure's service for 12 months which assisted in the developer acquiring a virtual machine (vm) of choice.

The Azure validated creation process and account settings can be found in *Appendix A*, Figures 63 to 74.

When selecting the size of the VM, there are a number of limitiations as only certain regions were allowed to use specific sizes. We settled with the East US region, 64GiB Hard Disk space, 16Gib RAM memory, 2 CPUs with 2 Cores each was selected, which is believed to be sufficient for this project.

Next the SSH keys are initialised by setting permissions to read-only.

```
# Setting Permissions for SSH key to Read Only
chmod 400 SecurityAuditVM_key.pem
```

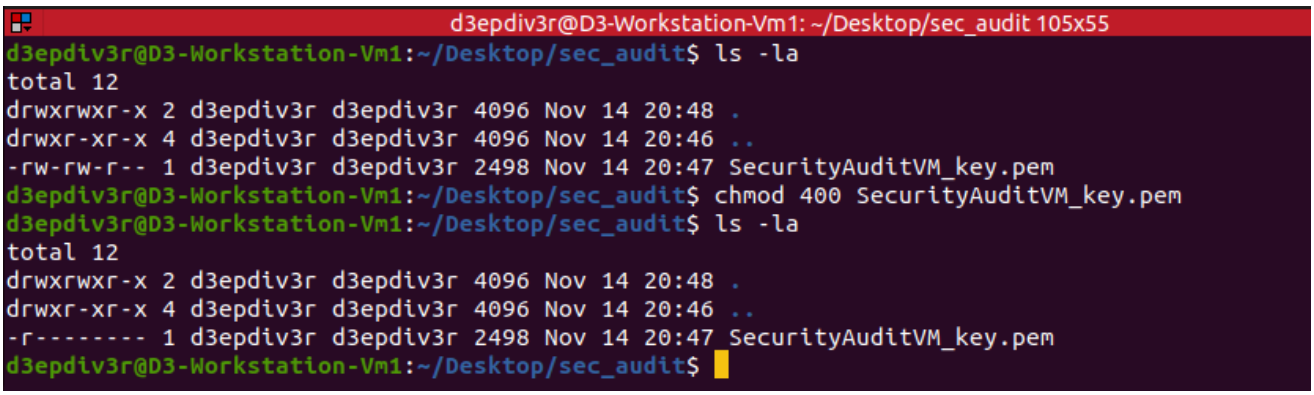


Figure 1: Setting Permissions for SSH Keys

Afterwards, the login was attempted, which was successful on the first attempt, indicating that Azure has been successfully setup.

```
# Login to Azure VM via SSH
```

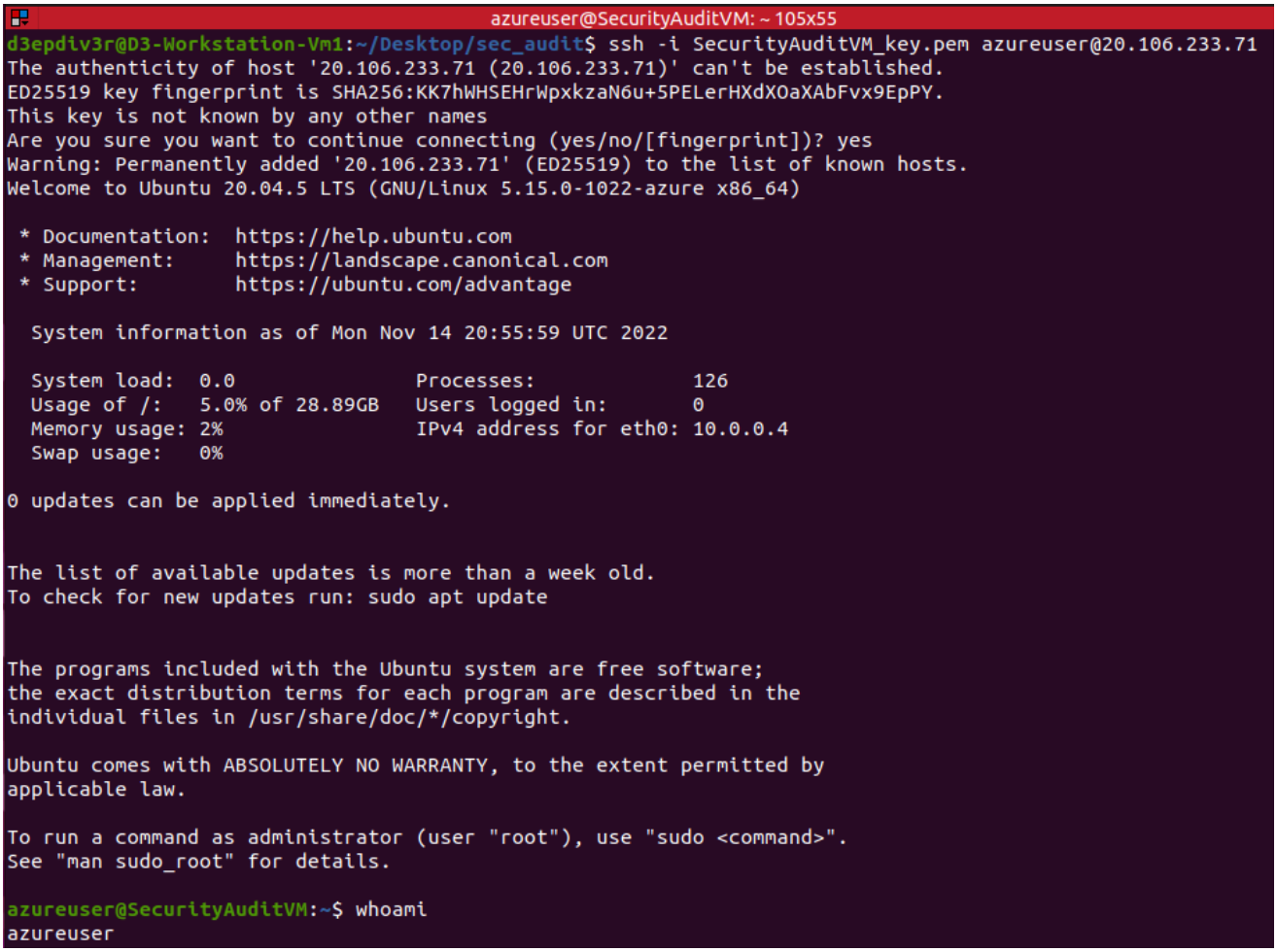


Figure 2: Successful Login to Azure Security Audit Virtual Machine

To complete the setup, the Ubuntu's package manager was updated, and installed Nginx's web server just incase we needed it. However, an update was not carried out, as the latest version of Ubuntu 20.04 was already in use. This was also done to avoid breaking any working packages.

```
# Updating package manager
sudo apt update -y

sudo apt -y install nginx

sudo apt update -y
```

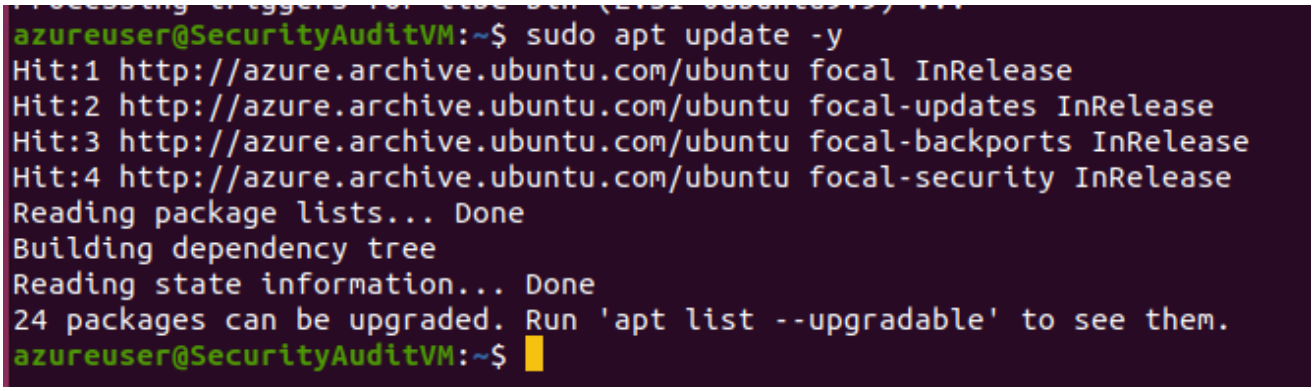


Figure 3: Successful Update

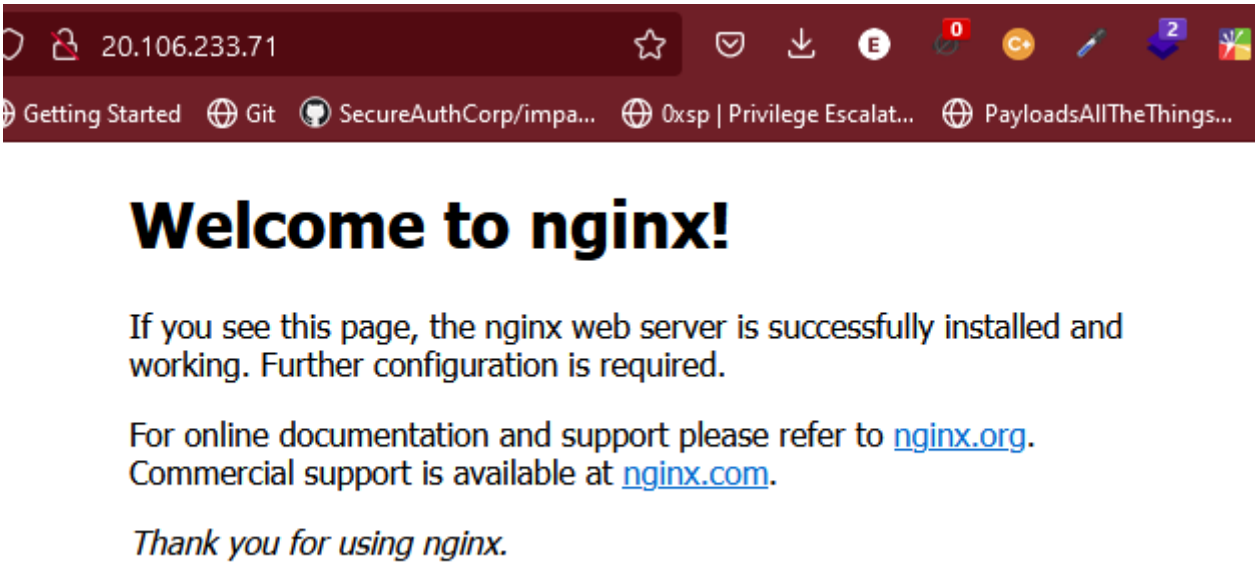


Figure 4 Shows successful installation of Nginx and access to Azure Server

Downloading PCAP Files for Network Analysis

Before proceeding, the PCAP files had to be downloaded so they would be available for the analysis. A variety of PCAP files have been downloaded from the following sources: Palo Alto Networks (Unit 42, 2020) and Cyber Defenders (CyberDefenders, 2020) to be used by Suricata and the other open source tools that need it.

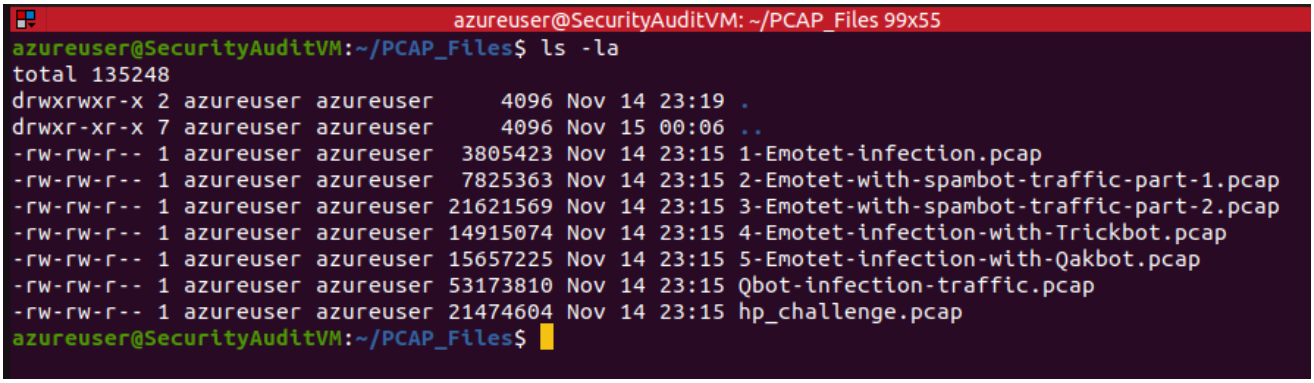


Figure 5: PCAP files

Suricata

Suricata has been installed according to the guide provided by Microsoft (Microsoft, 2022) which required that new PPA repositories be added to the package manager.

Installing Suricata

```
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt-get update
```

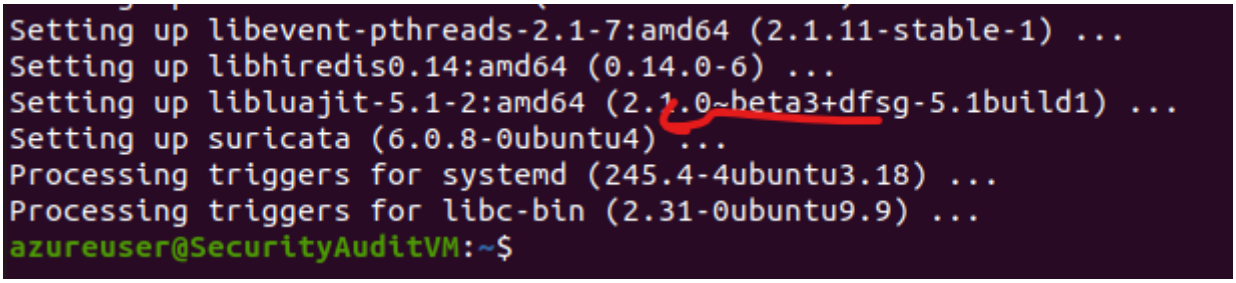


Figure 6: Successful Installation of Suricata

After setting up Suricata, the service was enabled and checked to see if it was running successfully.

```
# Enabling Suricata


# Checking Status of Suricata
sudo systemctl status suricata
```

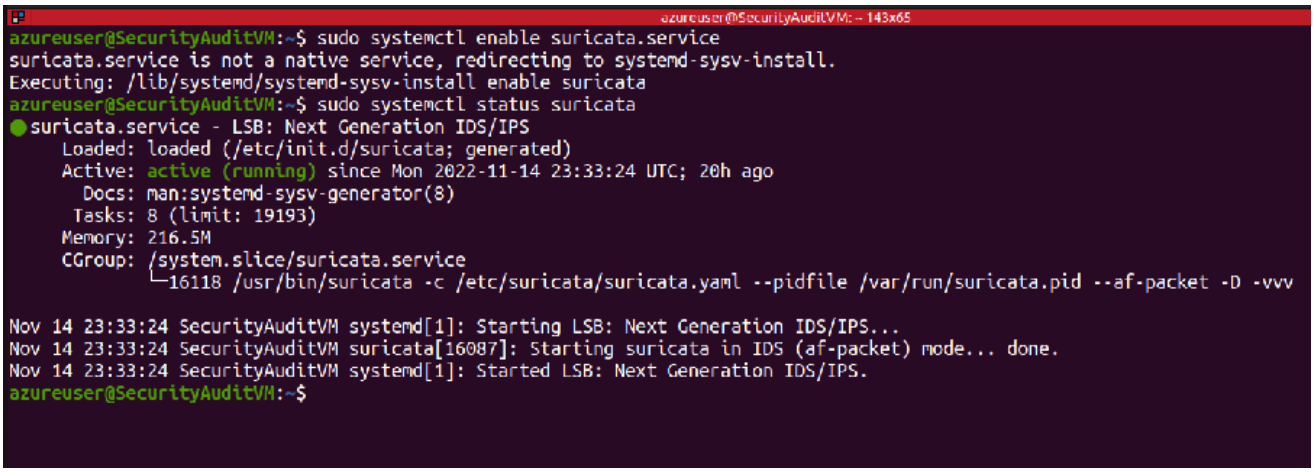


Figure 7: Enabling and verifying status of Suricata

Configuring Suricata

After validating that Suricata runs as expected, the service was stopped using `sudo systemctl stop suricata` in order to make changes to the configuration file.

Suricata's configuration file exists in `/etc/suricata/suricata.yaml`. Being a YAML file, it made modifications easy and Suricata did a great job in clearly identifying what each configuration does which also made changes easy.

The home subnet needed to be specified in order to increase the accuracy and performance of the alerts.

```
# Checking the IP Address of the VM for use in configuration file
azureuser@SecurityAuditVM:~$ ip -br -c a

eth0             UP             10.0.0.4/24 fe80::20d:3aff:fe53:f5b5/64
enP59470s1       UP
```

```
##
## Step 1: Inform Suricata about your network
##

vars:
  # more specific is better for alert accuracy and performance
  address-groups:
    HOME_NET: "[10.0.0.4/24]"
    #HOME_NET: "[192.168.0.0/16]"
    #HOME_NET: "[10.0.0.0/8]"
    #HOME_NET: "[172.16.0.0/12]"
    #HOME_NET: "any"

    EXTERNAL_NET: "!$HOME_NET"
    #EXTERNAL_NET: "any"
```

Figure 8: Specifying subnet in Suricata's configuration file

Next the interface had to be changed.

```
##
## Step 3: Configure common capture settings
##
## See "Advanced Capture Options" below for more options, including Netmap
## and PF_RING.
##

# Linux high speed capture support
af-packet:
  - interface: eth0
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99
    # Default AF_PACKET cluster type. AF_PACKET can load balance per flow or per hash.
```

Figure 9: Updating network interface in Suricata's config

Next, the community flow ID was enabled, which is useful for event correlation and helpful when using tools like Zeek or when trying to import logs into Suricata in JSON format.

```
# Community Flow ID
# Adds a 'community_id' field to EVE records. These are meant to give
# records a predictable flow ID that can be used to match records to
# output of other tools such as Zeek (Bro).
#
# Takes a 'seed' that needs to be same across sensors and tools
# to make the id less predictable.

# enable/disable the community id feature.
community-id: true
# Seed value for the ID output. Valid values are 0-65535.
community-id-seed: 0
```

Figure 10: Enabling community-id in Suricata's config

Afterwards, it was verified that Suricata's rules were put in place.

```
##
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /var/lib/suricata/rules

rule-files:
  - suricata.rules
##
## Auxiliary configuration files.
##
```

Figure 11: Default rule files in Suricata

Since Suricata operates using rules similar to other IDS systems, an update was run to load all rules and initialize the resources needed.

```
# Update Suricata
sudo suricata-update
```

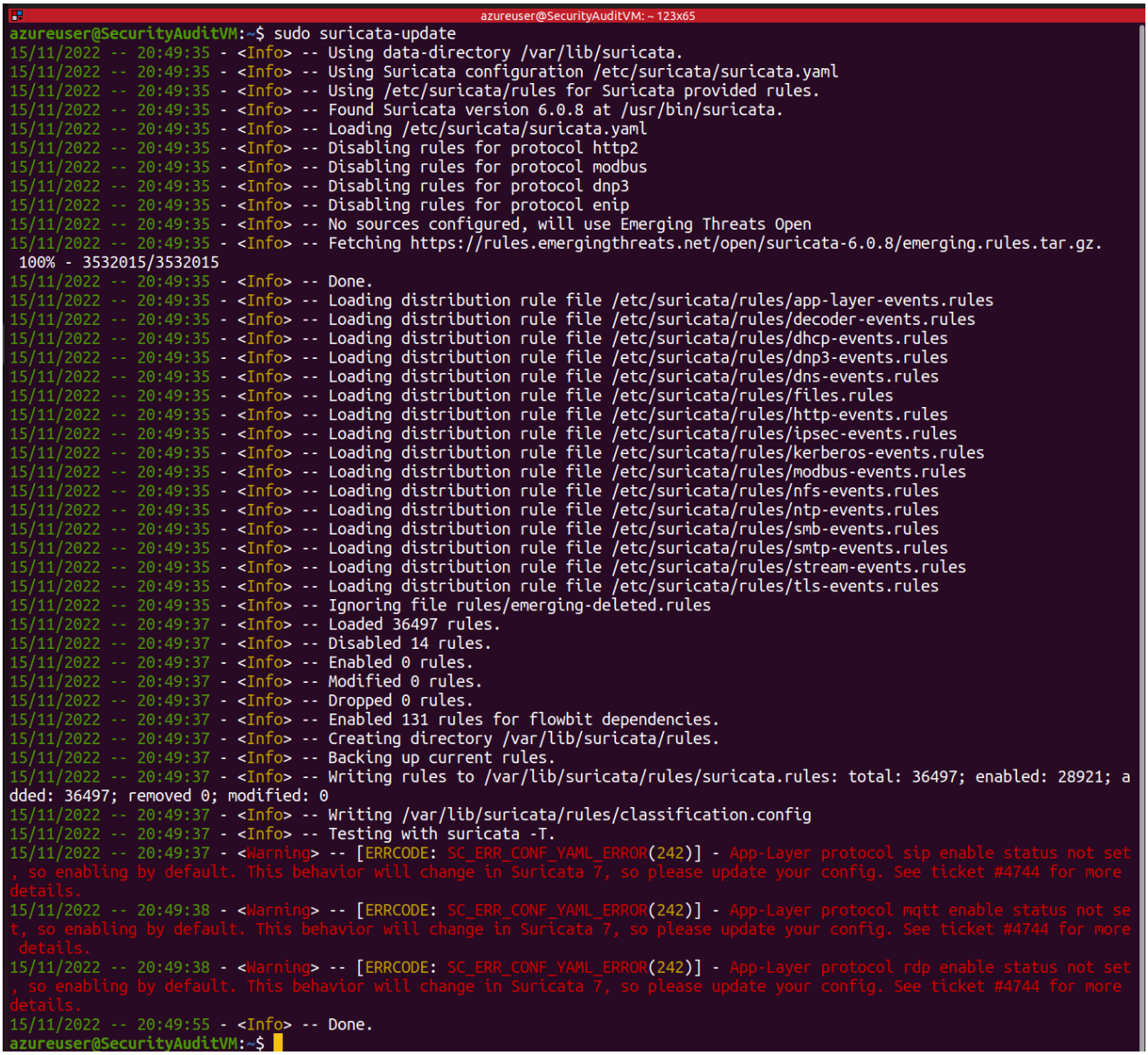


Figure 12: Updating Suricata

On running the update, it is evident that it automatically fetched the rules from Emerging Threats, but more importantly, issues were experienced (with the app-layer protocol) that seemed to be corrected in Suricata 7 but has not been changed in Suricata 6.x.x. The configurations have to be changed to fix these issues.

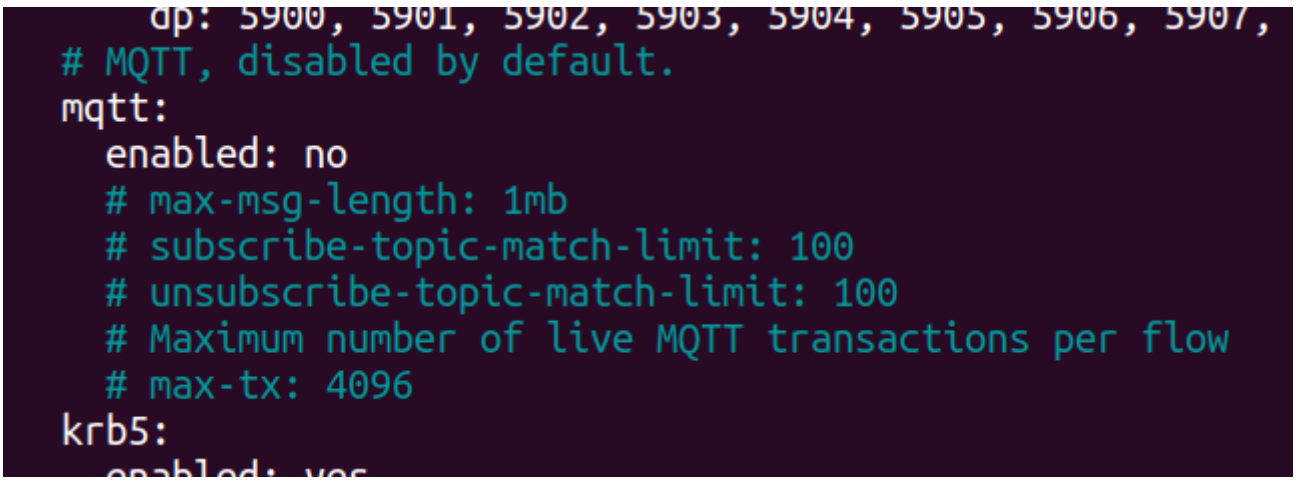


Figure 13: Disabling the MQTT app-layer protocol

```
  # memcap: 64mb
rdp:
  enabled: yes
ssh:
  enabled: yes
  #hassh: yes
```

Figure 14: Enabling the RDP app-layer protocol

```
sip:
  enabled: no
```

Figure 15: Disabling the SIP app-layer protocol

Then we updated Suricata and all worked fine.

```
azureuser@SecurityAuditVM: ~ 123x65
azureuser@SecurityAuditVM:~$ sudo suricata-update
15/11/2022 -- 21:04:12 - <Info> -- Using data-directory /var/lib/suricata.
15/11/2022 -- 21:04:12 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
15/11/2022 -- 21:04:12 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
15/11/2022 -- 21:04:12 - <Info> -- Found Suricata version 6.0.8 at /usr/bin/suricata.
15/11/2022 -- 21:04:12 - <Info> -- Loading /etc/suricata/suricata.yaml
15/11/2022 -- 21:04:12 - <Info> -- Disabling rules for protocol mqtt
15/11/2022 -- 21:04:12 - <Info> -- Disabling rules for protocol http2
15/11/2022 -- 21:04:12 - <Info> -- Disabling rules for protocol modbus
15/11/2022 -- 21:04:12 - <Info> -- Disabling rules for protocol dnp3
15/11/2022 -- 21:04:12 - <Info> -- Disabling rules for protocol enip
15/11/2022 -- 21:04:12 - <Info> -- Disabling rules for protocol sip
15/11/2022 -- 21:04:12 - <Info> -- No sources configured, will use Emerging Threats Open
15/11/2022 -- 21:04:12 - <Info> -- Last download less than 15 minutes ago. Not downloading https://rules.emergingthreats.ne
t/open/suricata-6.0.8/emerging.rules.tar.gz.
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/app-layer-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/decoder-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/dhcp-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/dnp3-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/dns-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/files.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/http-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/ipsec-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/kerberos-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/modbus-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/nfs-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/ntp-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/smb-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/smtp-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/stream-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Loading distribution rule file /etc/suricata/rules/tls-events.rules
15/11/2022 -- 21:04:13 - <Info> -- Ignoring file rules/emerging-deleted.rules
15/11/2022 -- 21:04:14 - <Info> -- Loaded 36497 rules.
15/11/2022 -- 21:04:15 - <Info> -- Disabled 14 rules.
15/11/2022 -- 21:04:15 - <Info> -- Enabled 0 rules.
15/11/2022 -- 21:04:15 - <Info> -- Modified 0 rules.
15/11/2022 -- 21:04:15 - <Info> -- Dropped 0 rules.
15/11/2022 -- 21:04:15 - <Info> -- Enabled 131 rules for flowbit dependencies.
15/11/2022 -- 21:04:15 - <Info> -- Backing up current rules.
15/11/2022 -- 21:04:17 - <Info> -- Writing rules to /var/lib/suricata/rules/suricata.rules: total: 36497; enabled: 28921; a
dded: 0; removed 0; modified: 0
15/11/2022 -- 21:04:17 - <Info> -- Writing /var/lib/suricata/rules/classification.config
15/11/2022 -- 21:04:17 - <Info> -- No changes detected, exiting.
azureuser@SecurityAuditVM:~$ sudo suricata-update -T
```

Figure 16: Successful update of Suricata without app-layer errors

Understanding what resources exist is essential, since Suricata was pulling rule lists from Emerging Threats, Here `sudo suricata-update list-sources` was run to get the information as seen below:

```
azureuser@SecurityAuditVM:~$ sudo suricata-update list-sources
[...]

  Vendor: Proofpoint

  License: MIT
[...]

  Vendor: malsilo

  License: MIT
[...]
```

It is recognisable that some of the sources are commercial and some MIT (Open source or Non-commercial) which meant that we could add additional rules can be added for free. The `malsilo/win-malware` is then added to aid Suricata conduct a more in-depth scan.

```
# Add two sources and update Suricata

```

As displayed below, everything is working correctly and the Suricata tests run successfully.

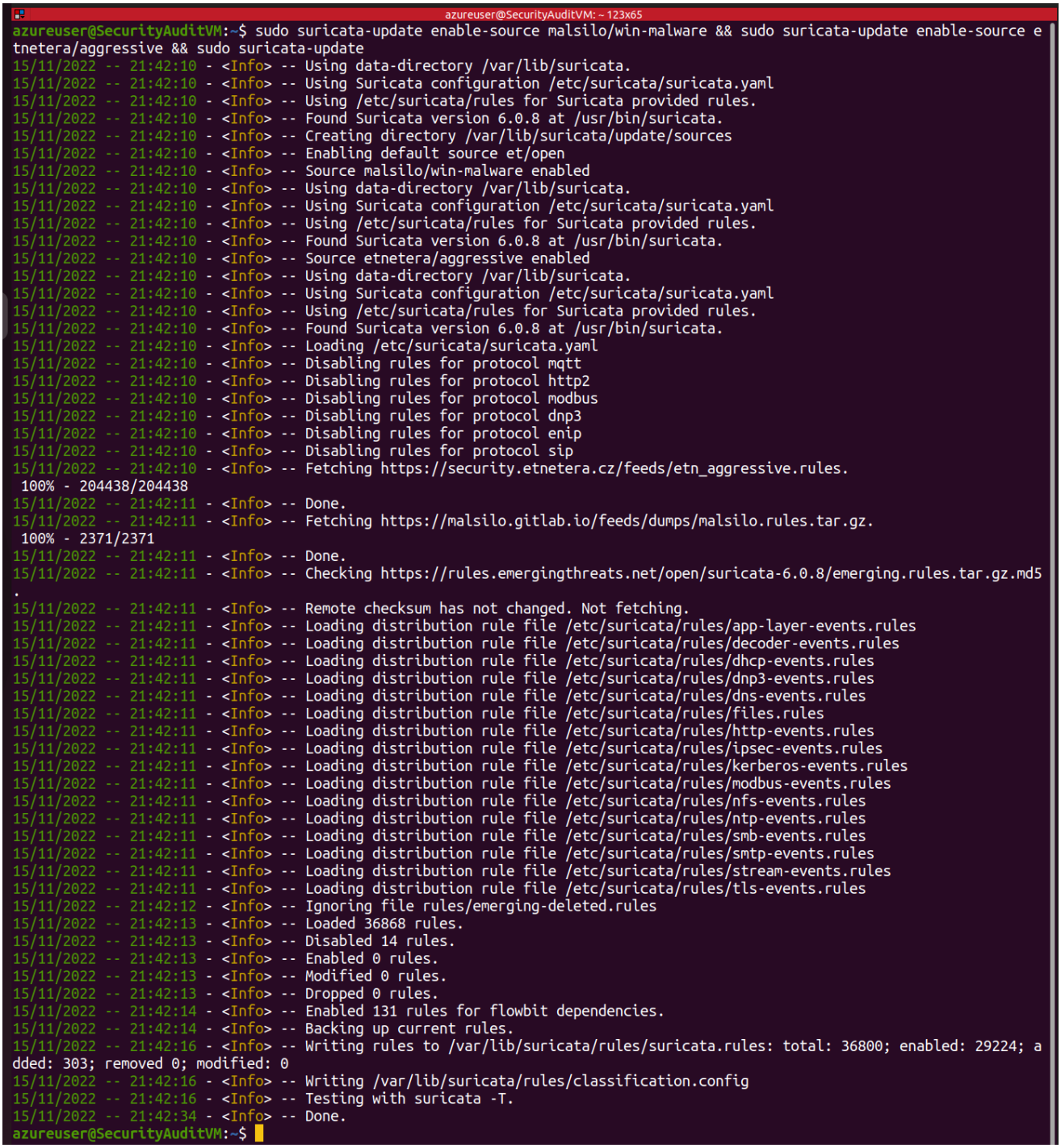


Figure 17: Adding two new source, updating, and testing Suricata configurations

The next step was to manually test Suricata's configuration to ensure everything was setup properly.

```
# Testing Suricata's Configuration File and Rules
sudo suricata -T -c /etc/suricata/suricata.yaml -v
```

```
azureuser@SecurityAuditVM:~$ sudo suricata -T -c /etc/suricata/suricata.yaml -v
16/11/2022 -- 02:06:56 - <Info> - Running suricata under test mode
16/11/2022 -- 02:06:56 - <Notice> - This is Suricata version 6.0.8 RELEASE running in SYSTEM mode
16/11/2022 -- 02:06:56 - <Info> - CPUs/cores online: 2
16/11/2022 -- 02:06:56 - <Info> - fast output device (regular) initialized: fast.log
16/11/2022 -- 02:06:56 - <Info> - eve-log output device (regular) initialized: eve.json
16/11/2022 -- 02:06:56 - <Notice> - JsonSIPLog logger not enabled: protocol sip is disabled
16/11/2022 -- 02:06:56 - <Notice> - JsonMQTTLog logger not enabled: protocol mqtt is disabled
16/11/2022 -- 02:06:56 - <Info> - stats output device (regular) initialized: stats.log
16/11/2022 -- 02:07:03 - <Info> - 1 rule files processed. 29240 rules successfully loaded, 0 rules failed
16/11/2022 -- 02:07:03 - <Info> - Threshold config parsed: 0 rule(s) found
16/11/2022 -- 02:07:03 - <Info> - 29243 signatures processed. 1483 are IP-only rules, 5171 are inspecting p
acket payload, 22386 inspect application layer, 108 are decoder event only
16/11/2022 -- 02:07:14 - <Notice> - Configuration provided was successfully loaded. Exiting.
16/11/2022 -- 02:07:14 - <Info> - cleaning up signature grouping structure... complete
azureuser@SecurityAuditVM:~$
```

Figure 18: Suricata configuration loaded successfully

The configuration was loaded successfully: one rule file (`suricata.rules`) was processed and `29,240` rules loaded successfully. Zero rules failed.

Afterwards, Suricata is run to ensure everything works as expected.

```
azureuser@SecurityAuditVM:~$ sudo systemctl start suricata.service
azureuser@SecurityAuditVM:~$ sudo systemctl status suricata.service
● suricata.service - LSB: Next Generation IDS/IPS
     Loaded: loaded (/etc/init.d/suricata; generated)
     Active: active (running) since Wed 2022-11-16 02:28:29 UTC; 13s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 31093 ExecStart=/etc/init.d/suricata start (code=exited, status=0/SUCCESS)
      Tasks: 1 (limit: 19193)
     Memory: 424.7M
     CGroup: /system.slice/suricata.service
             └─31099 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid --af-pa

Nov 16 02:28:29 SecurityAuditVM systemd[1]: Starting LSB: Next Generation IDS/IPS...
Nov 16 02:28:29 SecurityAuditVM suricata[31093]: Starting suricata in IDS (af-packet) mode... done.
Nov 16 02:28:29 SecurityAuditVM systemd[1]: Started LSB: Next Generation IDS/IPS.
azureuser@SecurityAuditVM:~$
```

Figure 19: Running status for Suricata

Testing Suricata

When testing Suricata, the logs were are checked to see if they have been created correctly for verification purposes, as logs can be written to whichever directory the user specifies.

```
azureuser@SecurityAuditVM:~$ ls -al /var/log/suricata/
total 1228
drwxr-xr-x  5 root root     4096 Nov 16 00:47 .
drwxrwxr-x 12 root syslog   4096 Nov 16 00:47 ..
drwxr-xr-x  2 root root     4096 Sep 28 05:05 certs
drwxr-xr-x  2 root root     4096 Sep 28 05:05 core
-rw-r--r--  1 root root   954990 Nov 16 02:32 eve.json
-rw-r--r--  1 root root     1961 Nov 16 02:29 fast.log
drwxr-xr-x  2 root root     4096 Sep 28 05:05 files
-rw-r--r--  1 root root   208742 Nov 16 02:32 stats.log
-rw-r--r--  1 root root     1542 Nov 16 02:28 suricata-start.log
-rw-r--r--  1 root root    57203 Nov 16 02:28 suricata.log
azureuser@SecurityAuditVM:~$
```

Figure 20: Suricata log files

As seen in *Figure 20* above, `eve.json` (contains the intrusion logs stored in `JSON` format) and `fast.log` (contains the same log stored in standard format) the two essential files to the scope of this project are created.

Next, Suricata was tested to see if it was monitoring the network by making a request to `curl http://testmynids.org/uid/index.html`. A website and framework for testing NIDS detection (3CORESec, 2021).

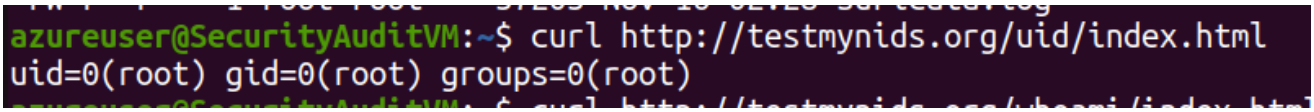


Figure 21: Testing NID detection

Viewing `sudo cat /var/log/suricata/fast.log` proves that Suricata produces attack alerts coming from `testmynids.com` in *Figure 22* and *Figure 23*

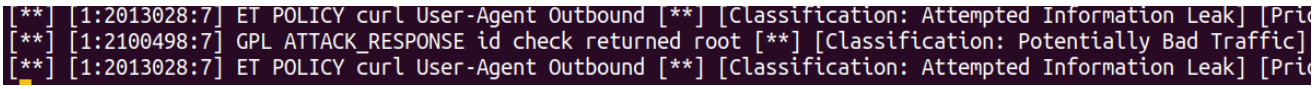


Figure 22: Suricata detects information leak attempt on the network 1

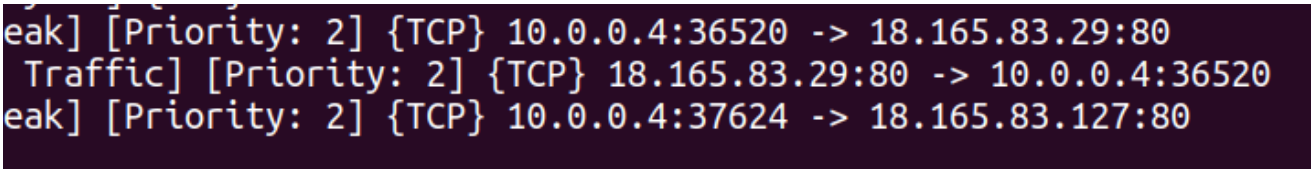


Figure 23: Suricata logs outbound destination IP

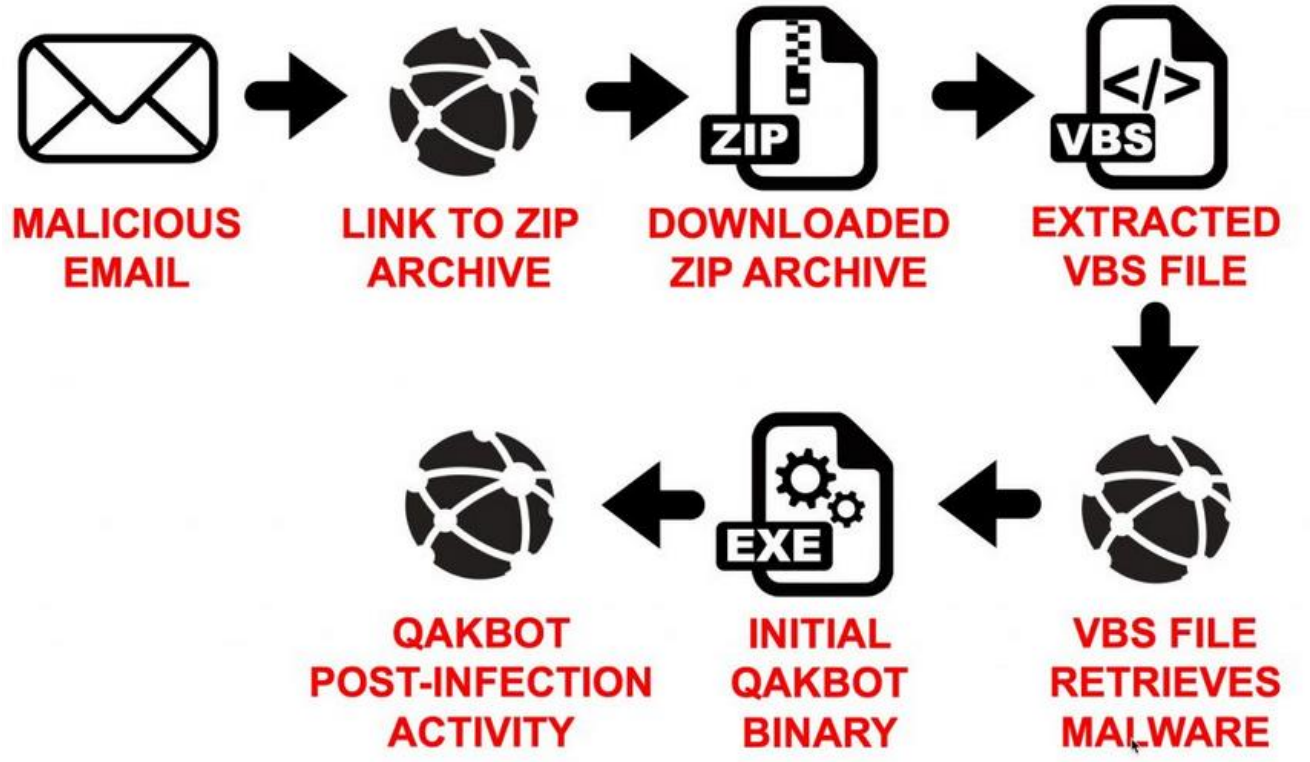
This shows that Suricata is set up correctly and running in the background!

Checking QakBot Traffic with Suricata

Seeing as the amount of PCAP files we downloaded earlier (as seen in *Figure 5*) analysing of all may be beyond the scope, it is wiser to focus specifically on the `Qbot-infection-traffic.pcap` file for analysis.

Figure 24

QakBot Infection Stages Overview



Directly copied from Duncan, B. (2020).

Knowing QakBot's infection process is helpful because it assist in detecting intrusion on the network.

Next, *Figure 25* below displays an attempt to parse the PCAP file into Suricata.

```
# Parsing PCAP to Suricata

/var/log/suricata
# OR
```

```
azureuser@SecurityAuditVM:~/PCAP_Files$ ls
1-Emotet-infection.pcap                    3-Emotet-with-spambot-traffic-part-2.pcap  5-Emotet-infection-with-Qakbot.pcap
2-Emotet-with-spambot-traffic-part-1.pcap  4-Emotet-infection-with-Trickbot.pcap      Qbot-infection-traffic.pcap
azureuser@SecurityAuditVM:~/PCAP_Files$ sudo suricata -c /etc/suricata/suricata.yaml -r Qbot-infection-traffic.pcap
16/11/2022 -- 04:26:24 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER mode
16/11/2022 -- 04:26:24 - <Notice> - JsonSIPLog logger not enabled: protocol sip is disabled
16/11/2022 -- 04:26:24 - <Notice> - JsonMQTTLog logger not enabled: protocol mqtt is disabled
16/11/2022 -- 04:26:42 - <Notice> - all 3 packet processing threads, 4 management threads initialized, engine started.
16/11/2022 -- 04:26:43 - <Notice> - Signal Received.  Stopping engine.
16/11/2022 -- 04:26:43 - <Notice> - Pcap-file module read 1 files, 74404 packets, 51983322 bytes
azureuser@SecurityAuditVM:~/PCAP_Files$
```

Figure 25: Parsing QakBot traffic into Suricata

It is evident in Figure above that Suricata ingested the PCAP file successfully, as `74,404` packets were read.

Suricata's digested output is stored in `eve.json` (JSON Format) and `fast.log` (standard format). The output in the `eve.json` file was chosen to be utilised, meaning `jq` had to be installed to make the JSON easier to read.

```
# Install jq
```

Since we know the infection process of **QakBot** is made aware, developers understood what to look for. The `eve.json` file was analysed and GET requests were found to download malicious `9312.zip` files from `bhatner.com` what appears to be a WordPress CMS (indicated by `wp-` `L30O6EHZ.zip` being sent from `"daikei-` `<toiawase@monotaro.com>` . These events happen over and over again and this sort of behaviour is common with QakBot malware. This shows that the PCAP file was being ingested properly.

```
# Searching for patterns that match ".zip" in Suricata's log file (eve.json)
jq -C "." /var/log/suricata/eve.json | grep ".zip"
```

Figure 26: QakBot malicious `.zip` file downloads (one)

```
      "L3006EHZ.zip"
      "L3006EHZ.zip"
    "filename": "L3006EHZ.zip",
    "url": "/wp-content/uploads/2020/01/ahead/9312.zip",
    "http_content_type": "application/zip",
    "filename": "9312.zip",
    "url": "/wp-content/uploads/2020/01/ahead/9312.zip",
    "http_content_type": "application/zip",
      "L3006EHZ.zip"
      "L3006EHZ.zip"
    "filename": "L3006EHZ.zip",
    "url": "/wp-content/uploads/2020/01/ahead/9312.zip",
    "http_content_type": "application/zip",
    "filename": "9312.zip",
    "url": "/wp-content/uploads/2020/01/ahead/9312.zip",
    "http_content_type": "application/zip",
      "L3006EHZ.zip"
      "L3006EHZ.zip"
    "filename": "L3006EHZ.zip",
    "url": "/wp-content/uploads/2020/01/ahead/9312.zip",
    "http_content_type": "application/zip",
    "filename": "9312.zip",
    "url": "/wp-content/uploads/2020/01/ahead/9312.zip",
    "http_content_type": "application/zip",
    "url": "/wp-content/uploads/2020/01/ahead/9312.zip",
    "http_content_type": "application/zip",
    "filename": "9312.zip",
    "url": "/wp-content/uploads/2020/01/ahead/9312.zip",
    "http_content_type": "application/zip",
azureuser@SecurityAuditVM:~/PCAP_Files$
```


Figure 27: QakBot malicious `.zip` file downloads (two)

Setting up the Elastic Stack

As displayed *Figure 26* and *Figure 27* the logs produced by Suricata are not very presentable and they are a little difficult to read and understand. "By connecting Suricata with the Elastic Stack (a SaaS which combine Elasticsearch, Logstash and Kibana), we can create a Kibana dashboard which allows us to search, graph, analyze, and derive insights from our logs" Microsoft. (2022).

Note that the Microsoft documentation provided is out-of-date as it does not state that the versions of Elastic Stack (5.2.*) written about, can only work on Ubuntu v16.04: 32bit specifically (and other specific Linux versions) as seen in *Figure 28* below.

Azure only offers Ubuntu v18.04 and greater for free. In order to use v16.04 on Azure, the user would have to pay! As a result, from this point on, another server was provisioned with Ubuntu v16.04 to carry out the task. The previous steps were repeated on that server.

Last updated: 2022-11-01

	RHEL/CentOS 6‡	RHEL/CentOS 7	RHEL 8	RHEL 9	CentOS 8	Oracle Linux 6*‡	Oracle Linux 7*	Oracle Linux 8	Ubuntu 14.04	Ubuntu 16.04	Ubuntu 18.04	Ubuntu 20.04	Ubuntu 22.04
Elasticsearch 5.0.x	✔	✔	✘	✘	✘	✔	✔	✘	✔	✔	✘	✘	✘
Elasticsearch 5.1.x	✔	✔	✘	✘	✘	✔	✔	✘	✔	✔	✘	✘	✘
Elasticsearch 5.2.x	✔	✔	✘	✘	✘	✔	✔	✘	✔	✔	✘	✘	✘
Elasticsearch 5.3.x	✔	✔	✘	✘	✘	✔	✔	✘	✔	✔	✘	✘	✘
Elasticsearch 5.4.x	✔	✔	✘	✘	✘	✔	✔	✘	✔	✔	✘	✘	✘
Elasticsearch 5.5.x	✔	✔	✘	✘	✘	✔	✔	✘	✔	✔	✘	✘	✘

Figure 28: Elastic Stack support matrix

The next step here was to install all the required dependencies to use the Elastic Stack.

Elasticsearch

Elasticsearch 5.0 and above requires Java 8 to run, since Java did not exist on the system, it had to be installed.

Installing Elasticsearch

```
azureuser@SecurityAuditVM:~/PCAP_Files$ java -version

Command 'java' not found, but can be installed with:

sudo apt install default-jre              # version 2:1.11-72, or
sudo apt install openjdk-11-jre-headless  # version 11.0.17+8-1ubuntu2~20.04
sudo apt install openjdk-13-jre-headless  # version 13.0.7+5-0ubuntu1~20.04
sudo apt install openjdk-16-jre-headless  # version 16.0.1+9-1~20.04
sudo apt install openjdk-17-jre-headless  # version 17.0.5+8-2ubuntu1~20.04
sudo apt install openjdk-8-jre-headless   # version 8u352-ga-1~20.04

azureuser@SecurityAuditVM:~/PCAP_Files$ uname -a
```

Figure 29: Java not found on system

```
# We updated the system first
sudo apt update

sudo apt install apt-transport-https
# Then we Installed Java-8

# After downloading Java 8, we added the path to our environment
sudo nano /etc/environment

source /etc/environment
java -version
```

```
azureuser@SecurityAudittVM:~$ java -version
openjdk version "1.8.0_352"
OpenJDK Runtime Environment (build 1.8.0_352-8u352-ga-1~20.04-b08)
OpenJDK 64-Bit Server VM (build 25.352-b08, mixed mode)
azureuser@SecurityAudittVM:~$ echo $JAVA_HOME
/usr/lib/jvm/java-8-openjdk-amd64
azureuser@SecurityAudittVM:~$
```

Figure 30: Java version and home path

The next stage of the experiment was to install Elasticsearch.

```
# Update system
sudo apt-get update

curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.2.0.deb

sudo dpkg -i elasticsearch-5.2.0.deb
# Update system
```

```
azureuser@SecurityAudittVM:~$ curl -L -O https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-5.2.0.deb
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 31.8M  100 31.8M    0     0  12.5M      0  0:00:02  0:00:02 --:--:-- 12.5M
azureuser@SecurityAudittVM:~$ sudo dpkg -i elasticsearch-5.2.0.deb
Selecting previously unselected package elasticsearch.
(Reading database ... 73794 files and directories currently installed.)
Preparing to unpack elasticsearch-5.2.0.deb ...
Creating elasticsearch group... OK
Creating elasticsearch user... OK
Unpacking elasticsearch (5.2.0) ...
Setting up elasticsearch (5.2.0) ...
Not setting net/ipv4/conf/all/promote_secondaries (explicit setting exists).
Not setting net/ipv4/conf/default/promote_secondaries (explicit setting exists).
Processing triggers for systemd (245.4-4ubuntu3.18) ...
azureuser@SecurityAudittVM:~$ sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://azure.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://azure.archive.ubuntu.com/ubuntu focal-security InRelease
Hit:5 http://ppa.launchpad.net/oisf/suricata-stable/ubuntu focal InRelease
Reading package lists... Done
azureuser@SecurityAudittVM:~$
```

Figure 31: Elasticsearch installation

Next, the daemon was restarted, enabled and started Elasticsearch. This ran successfully, denoted by the active status as seen in *Figure 32* below.

```
# Reload systemctl daemon
sudo systemctl daemon-reload

sudo /etc/init.d/elasticsearch start
# Enable the Elasticsearch service

# Start Elasticsearch

# Check Elasticsearch running status
```

```
azureuser@SecurityAudittVM:~$ sudo systemctl status elasticsearch.service
● elasticsearch.service - Elasticsearch
     Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2022-11-17 18:55:00 UTC; 2s ago
       Docs: http://www.elastic.co
    Process: 8894 ExecStartPre=/usr/share/elasticsearch/bin/elasticsearch-systemd-pre-exec (code=exi
   Main PID: 8895 (java)
      Tasks: 16 (limit: 19193)
     Memory: 2.1G
     CGroup: /system.slice/elasticsearch.service
             └─8895 /bin/java -Xms2g -Xmx2g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFracti

Nov 17 18:55:00 SecurityAudittVM systemd[1]: Starting Elasticsearch...
Nov 17 18:55:00 SecurityAudittVM systemd[1]: Started Elasticsearch.
```

Figure 32: Elasticsearch running status

Configuring Elasticsearch

Some of the configurations within Elasticsearch's config file needed to be modified.

```
# Stop Elasticsearch before configuration
sudo systemctl stop elasticsearch.service
# Editing Elasticsearch configurations

# Afterward we started the service
sudo systemctl start elasticsearch
```

`network.host` was uncommented and the IP to `0.0.0.0` was changed. The `http.port: 9200` from within the config file was also uncommented.

```
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- Network -----------------------------------
#
# Set the bind address to a specific IP (IPv4 or IPv6):
#
network.host: 0.0.0.0
#
# Set a custom port for HTTP:
#
http.port: 9200
#
# For more information, consult the network module documentation.
#
# --------------------------------- Discovery ----------------------------------
```

Figure 33: Modifying Elasticsearch configuration

Testing Elasticsearch

After the setup, a test on Elasticsearch was conducted.

```
// JSON response from Elasticsearch API endpoint
azureuser@SecurityAudittVM:~/PCAP_Files$ curl -X GET "localhost:9200"
{

  "cluster_name" : "elasticsearch",
```

```
  "cluster_uuid" : "zIJMKN3RkWMmgsfiAlMvg",

    "number" : "5.2.0",

    "build_date" : "2017-02-24T17:26:45.8352",

    "lucene_version" : "6.4.0"

  "tagline" : "You Know, for Search"
}
```

The JSON response shows that Elasticsearch has been configured correctly!

Logstash

In this section of the report, Logstash is installed. This is "a tool that collects data from different sources. The data it collects is parsed by Kibana and stored in Elasticsearch" (Mamidwar, S, 2022).

Installing Logstash

```
# Downloading logstash

sudo dpkg -i logstash-5.2.0.deb
# Starting Logstash

# Enabling Logstash

# Checking Logstash status
sudo systemctl status logstash
```

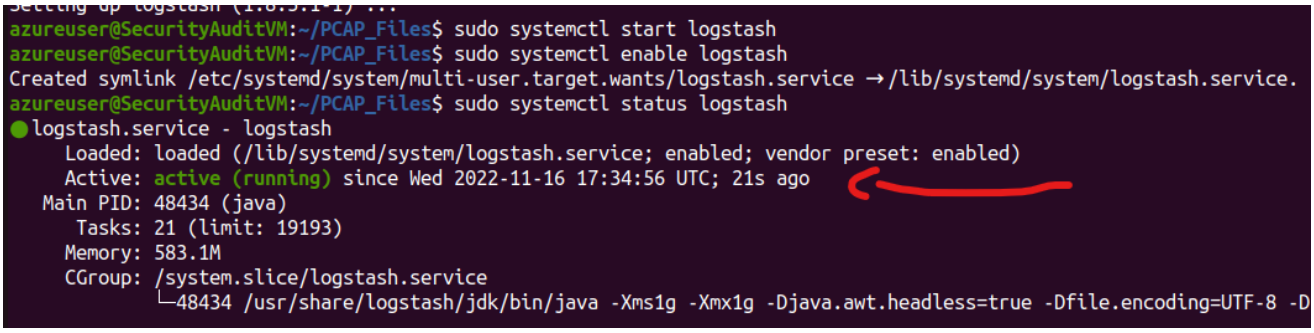


Figure 34: Successful installation of Logstash

Configuring Logstash

Logstash was stopped before proceeding to configure it to read from the output of the `eve.json` file created by Suricata.

```
# Stop Logstash

# We changed permissions of eve.json and logstash file so it can digest data
# both were initially 777 then switched permissions back to 776

sudo chmod -R 776 /usr/share/logstash
# Creating configuration file

# Adding configurations
```

We added the data below into `/etc/logstash/conf.d/logstash.conf` .

```
input {
file {

    codec =>  "json"

}

}

filter {

    date {

    }

    code => "

        event.set('[fileinfo][type]', event.get('[fileinfo]
[magic]').to_s.split(',')[0])

    "


    ruby{

        if event.get('[event_type]') == 'alert'

        if (sp.length == 2) and /\A\d+\z/.match(sp[1])

        end

        "


}

if [src_ip]  {
```

```
    geoip {

    target => "geoip"

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

    }

    convert => [ "[geoip][coordinates]", "float" ]

    if ![geoip.ip] {

        geoip {

        target => "geoip"

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

        }

        convert => [ "[geoip][coordinates]", "float" ]

    }

}
}

output {
elasticsearch {

}
}
```

Next the following changes are made, as seen in (*Figures: 35, 36, 37, 38*) below to the `/etc/logstash/logstash.yml` file.

```
#
# Settings can be specified either in hierarchical f
#
   pipeline:
     batch:
       size: 1000
       delay: 20
#
# Or as flat keys:
#
#   pipeline.batch.size: 125
#   pipeline.batch.delay: 5
#
```

Figure 35: Logstash pipeline configuration 1

```
# ------------ Data path ------------------
#
# Which directory should be used by logstash
# for any persistent needs. Defaults to LOGS
#
path.data: /var/lib/logstash
#
# ------------ Pipeline Settings -----------
```

Figure 36: Logstash pipeline configuration 2

```
# ------------ Pipeline Configuration Settings ----------
#
# Where to fetch the pipeline configuration for the main p
#
path.config: /etc/logstash/conf.d
#
# Pipeline configuration string for the main pipeline
#
# config.string:
#
```

Figure 37: Logstash pipeline configuration 3

```
# ------------ Metrics Settings --------------
#
# Bind address for the metrics REST endpoint
#
http.host: "0.0.0.0"
#
# Bind port for the metrics REST endpoint, this option also accept a range
# (9600-9700) and logstash will pick up the first available ports.
#
http.port: 9600
#
# ------------ Debugging Settings --------------
#
# Options for log.level:
#   * fatal
#   * error
#   * warn
#   * info (default)
#   * debug
#   * trace
#
# log.level: info
path.logs: /var/log/logstash
#
# ------------ Other Settings --------------
```

Figure 38: Logstash pipeline configuration 4

Testing Logstash

Next, Logstash was manually started as `systemctl` was causing some issues.

```
# Starting Logstash directly
```

```
Could not find log4j2 configuration at path /usr/share/logstash/config/log4j2.properties. Using default config which logs to console
15:27:28.447 [[main]-pipeline-manager] INFO  logstash.outputs.elasticsearch - Elasticsearch pool URLs updated {:changes=>{:removed=>[], :added=>[http://localhost:9200/]}}
15:27:28.452 [[main]-pipeline-manager] INFO  logstash.outputs.elasticsearch - Running health check to see if an Elasticsearch connection is working {:healthcheck_url=>http://localhost:9200/, :path=>"/"}
15:27:28.590 [[main]-pipeline-manager] WARN  logstash.outputs.elasticsearch - Restored connection to ES instance {:url=>#<URI::HTTP:0x895d5 URL:http://localhost:9200/>}
15:27:28.591 [[main]-pipeline-manager] INFO  logstash.outputs.elasticsearch - Using mapping template from {:path=>nil}
15:27:28.890 [[main]-pipeline-manager] INFO  logstash.outputs.elasticsearch - Attempting to install template {:manage_template=>{"template"=>"logstash-*", "version"=>50001, "settings"=>{"index.refresh_interval"=>"5s"}, "mappin
gs"=>{"_default_"=>{"_all"=>{"enabled"=>true, "norms"=>false}, "dynamic_templates"=>[{"message_field"=>{"path_match"=>"message", "match_mapping_type"=>"string", "mapping"=>{"type"=>"text", "norms"=>false}}}, {"string_fields"=>
{"match"=>"*", "match_mapping_type"=>"string", "mapping"=>{"type"=>"text", "norms"=>false, "fields"=>{"keyword"=>{"type"=>"keyword"}}}}}], "properties"=>{"@timestamp"=>{"type"=>"date", "include_in_all"=>false}, "@version"=>{"t
ype"=>"keyword", "include_in_all"=>false}, "geoip"=>{"dynamic"=>true, "properties"=>{"ip"=>{"type"=>"ip"}, "location"=>{"type"=>"geo_point"}, "latitude"=>{"type"=>"half_float"}, "longitude"=>{"type"=>"half_float"}}}}}}}}
15:27:28.898 [[main]-pipeline-manager] INFO  logstash.outputs.elasticsearch - New Elasticsearch output {:class=>"LogStash::Outputs::ElasticSearch", :hosts=>[#<URI::Generic:0x1710021 URL://localhost:9200>]}
15:27:28.906 [[main]-pipeline-manager] INFO  logstash.filters.geoip - Using geoip database {:path=>"/usr/share/logstash/vendor/bundle/jruby/1.9/gems/logstash-filter-geoip-4.0.4-java/vendor/GeoLite2-City.mmdb"}
15:27:28.938 [[main]-pipeline-manager] INFO  logstash.filters.geoip - Using geoip database {:path=>"/usr/share/logstash/vendor/bundle/jruby/1.9/gems/logstash-filter-geoip-4.0.4-java/vendor/GeoLite2-City.mmdb"}
15:27:28.942 [[main]-pipeline-manager] INFO  logstash.pipeline - Starting pipeline {"id"=>"main", "pipeline.workers"=>2, "pipeline.batch.size"=>125, "pipeline.batch.delay"=>5, "pipeline.max_inflight"=>250}
15:27:28.949 [[main]-pipeline-manager] INFO  logstash.pipeline - Pipeline main started
15:27:29.000 [Api Webserver] INFO  logstash.agent - Successfully started Logstash API endpoint {:port=>9601}
```

Figure 39: Logstash running successfully

Kibana

Afterwards, Kibana was installed. "It is a graphical user interface for parsing and interpreting collected log files" (Mamidwar, S, 2022).

Installing Kibana

```
# Download Kibana

# Verify and extract kibana

tar -xzf kibana-5.2.0-linux-x86.tar.gz
cd kibana-5.2.0-linux-x86/
```

```
# Start Kibana
./bin/kibana
```

```
  log   [23:08:05.770] [info][status][plugin:kibana@5.2.0] Status changed from uninitialized to green - Ready
  log   [23:08:05.831] [info][status][plugin:elasticsearch@5.2.0] Status changed from uninitialized to yellow - Waiting for El
asticsearch
  log   [23:08:05.869] [info][status][plugin:console@5.2.0] Status changed from uninitialized to green - Ready
  log   [23:08:05.891] [info][status][plugin:elasticsearch@5.2.0] Status changed from yellow to green - Kibana index ready
  log   [23:08:06.047] [info][status][plugin:timelion@5.2.0] Status changed from uninitialized to green - Ready
  log   [23:08:06.052] [info][listening] Server running at http://localhost:5601
  log   [23:08:06.053] [info][status][ui settings] Status changed from uninitialized to green - Ready
```

Figure 40: Kibana running successfully

There was no need to make any change in the Kibana configuration file.

After setting up the configurations above, the Azure Portal needed to be logged into, then navigate to the **Virtual Machine**, **Settings**, then selected **Networking**, within this section an **inbound port rule** was created to allow public access to the server port at `5601` for the purpose of this project.

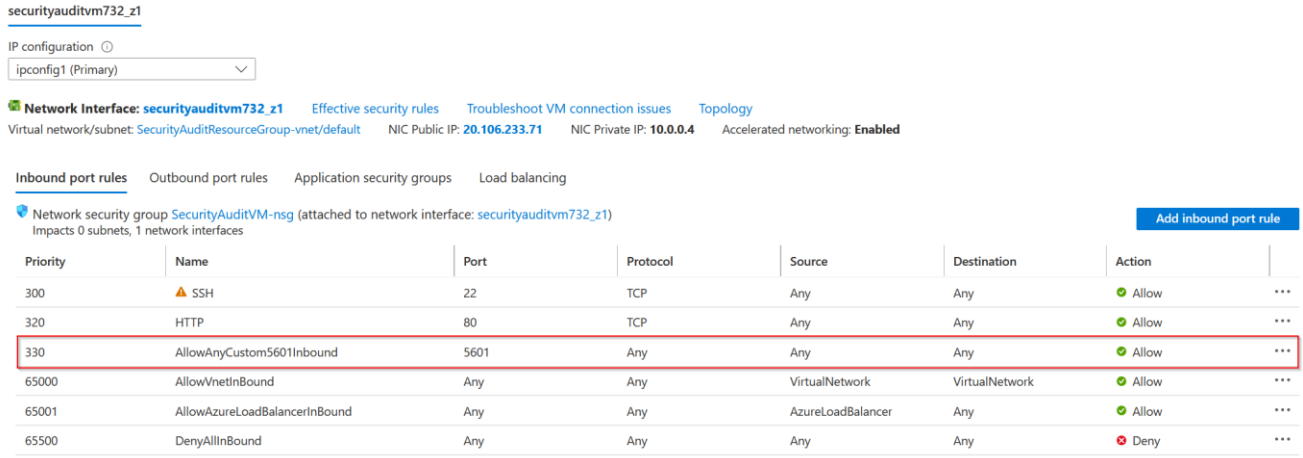


Figure 41: Setting up inbound port rules for Kibana on Azure

After the changes, Kibana's service were restarted and it ran successfully.

Testing Kibana

To access Kibana, a web browser was opened and browsed to our Azure's public IP address and port

```
# Kibana interface
http://20.106.233.71:5601
```

As shown in *Figure 42* below, Kibana has been installed, configured, and loaded correctly!

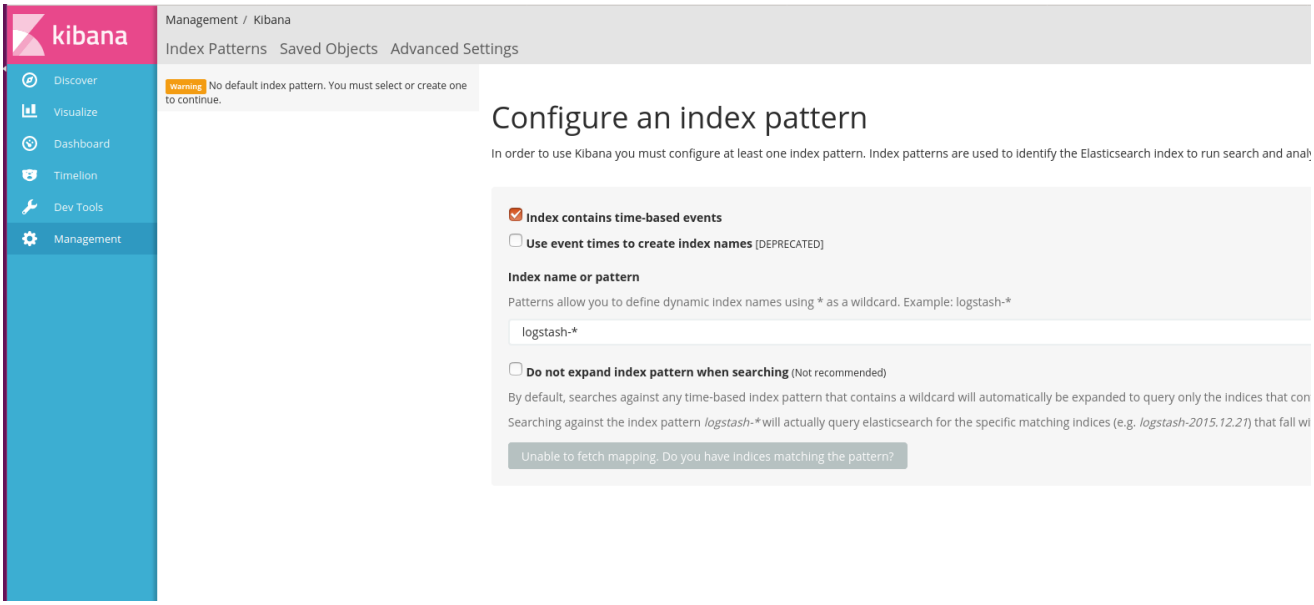


Figure 42: Kibana management screen

Creating the Kibana Dashboard

Developers needed to create a default index pattern and download the dashboard, visualization, and saved search file.

At this point in the experiment all services had to be restarted, so that they can be synchronized to properly load all index patterns. As seen in *Figure 42* above, no index pattern for `logstash-*` was found which prevented the experiment from moving forward.

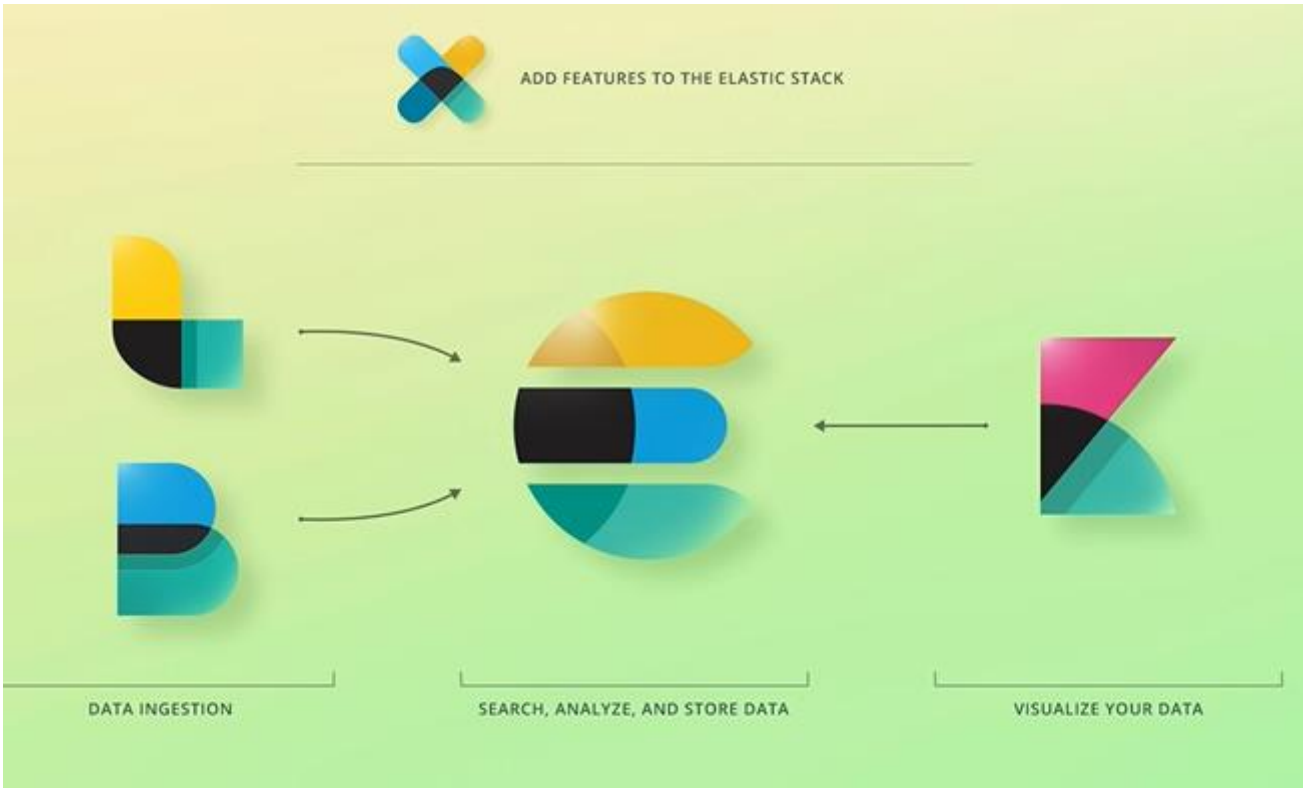


Figure 43: ELK/B Stack functionality

Next, after all services were running in the background, Suricata's PCAP ingestion process had to be re-ran, so that as events generated by Suricata reflect in the `eve.json` file,

Logstash will pick it up and parse it to Elasticsearch, so that it can be displayed by Kibana.

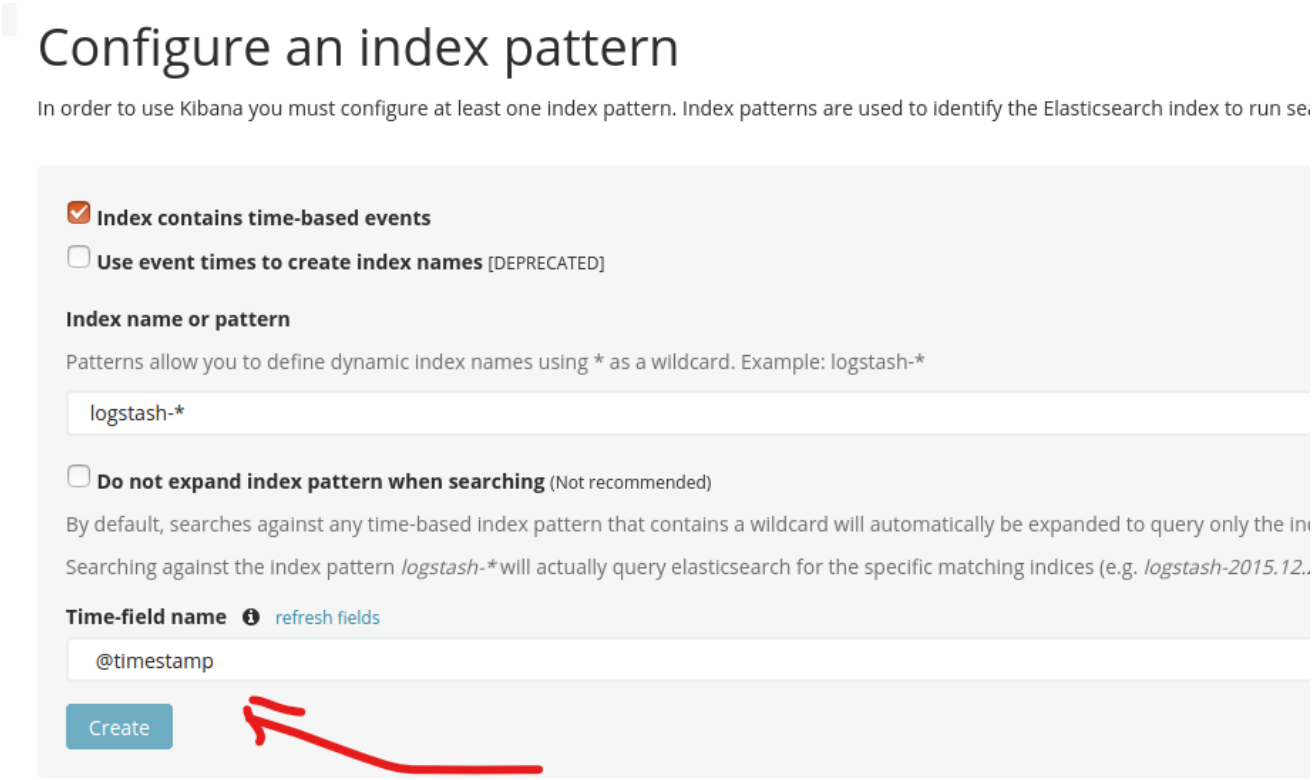


Figure 44: Access to create index-pattern after Suricata PCAP ingestion

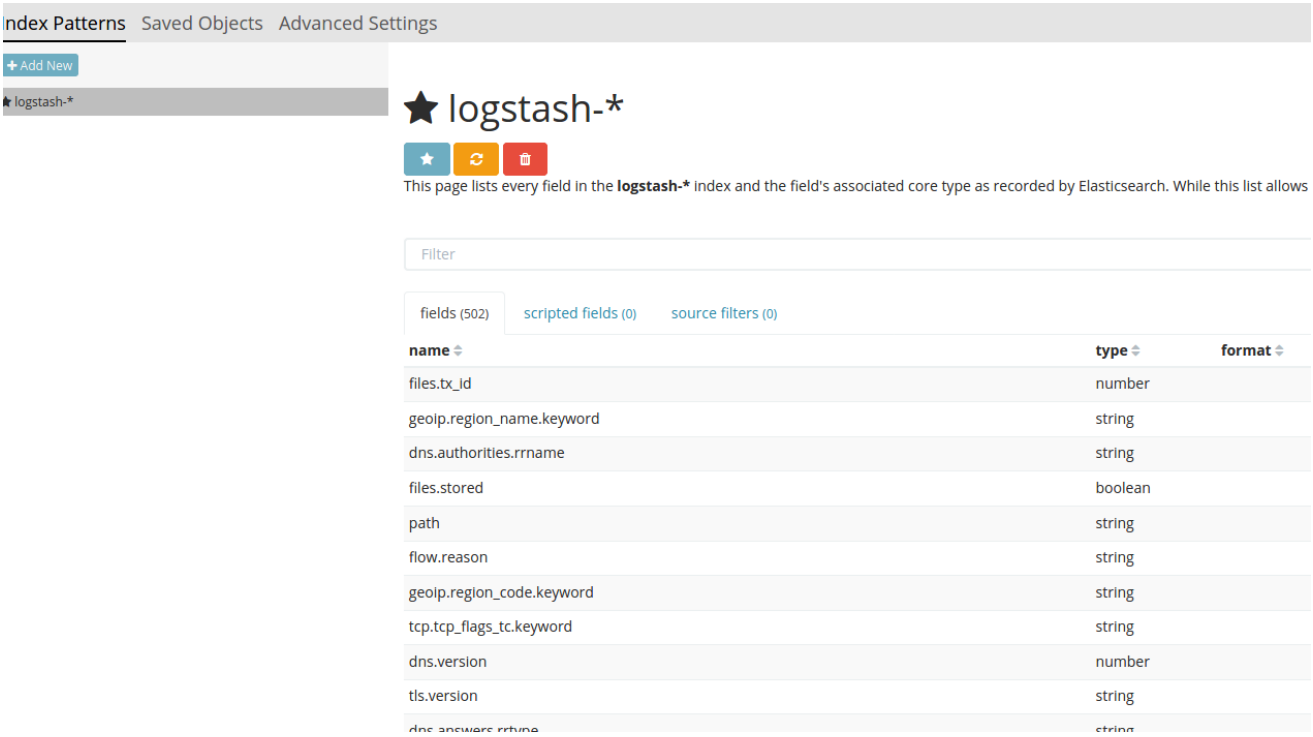


Figure 45: Created `logstash-*` pattern

Next, the dashboards were imported.

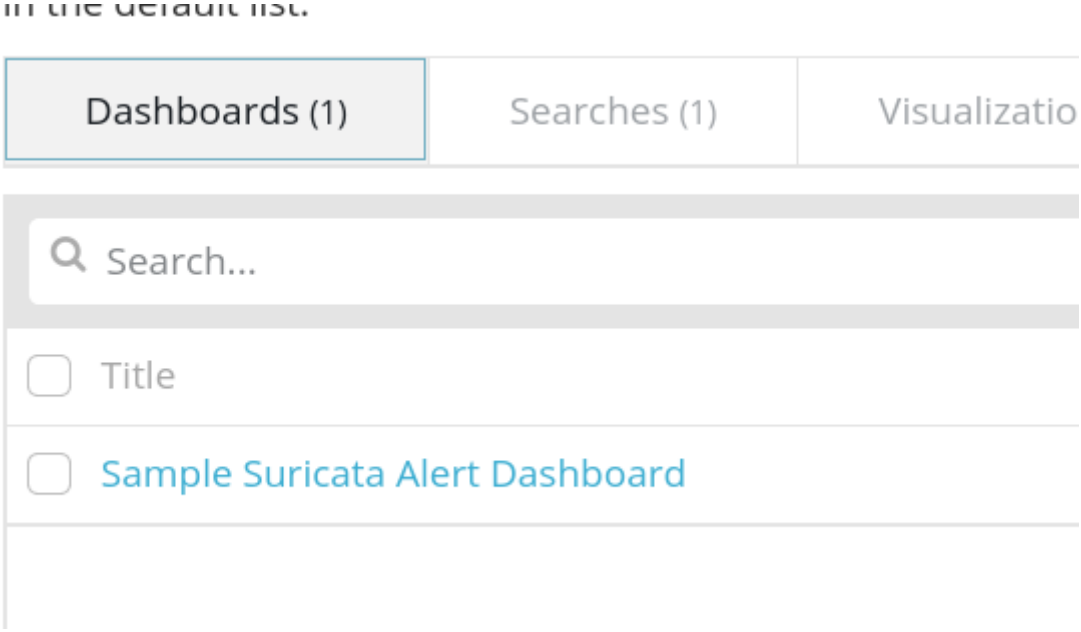


Figure 46: Sample Suricata Alert Dashbort import

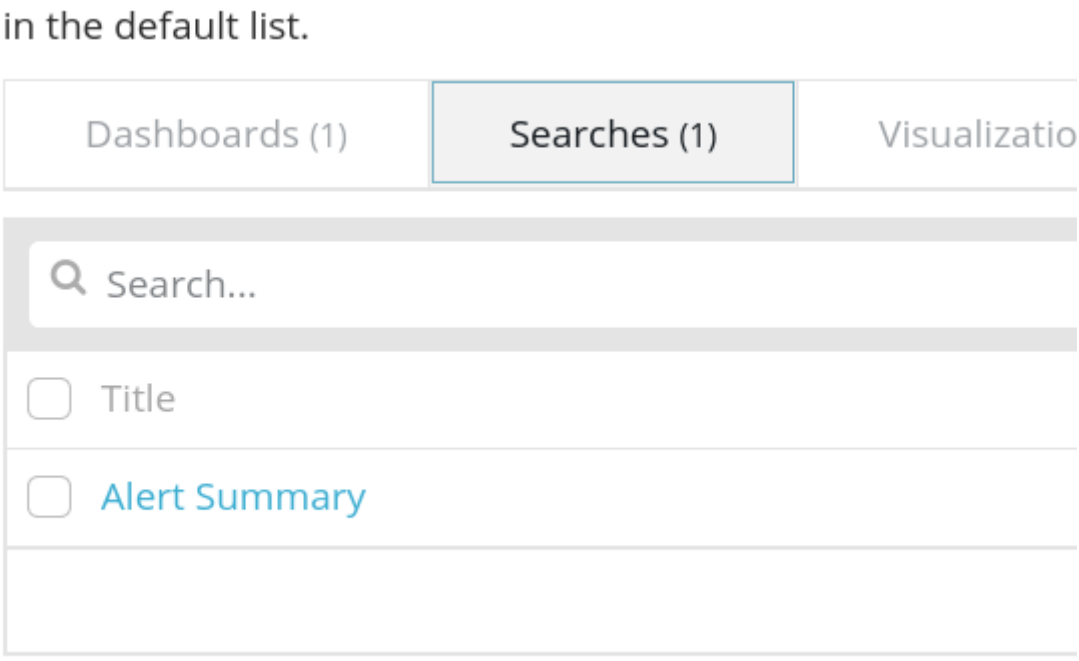


Figure 47: Alert summary search import

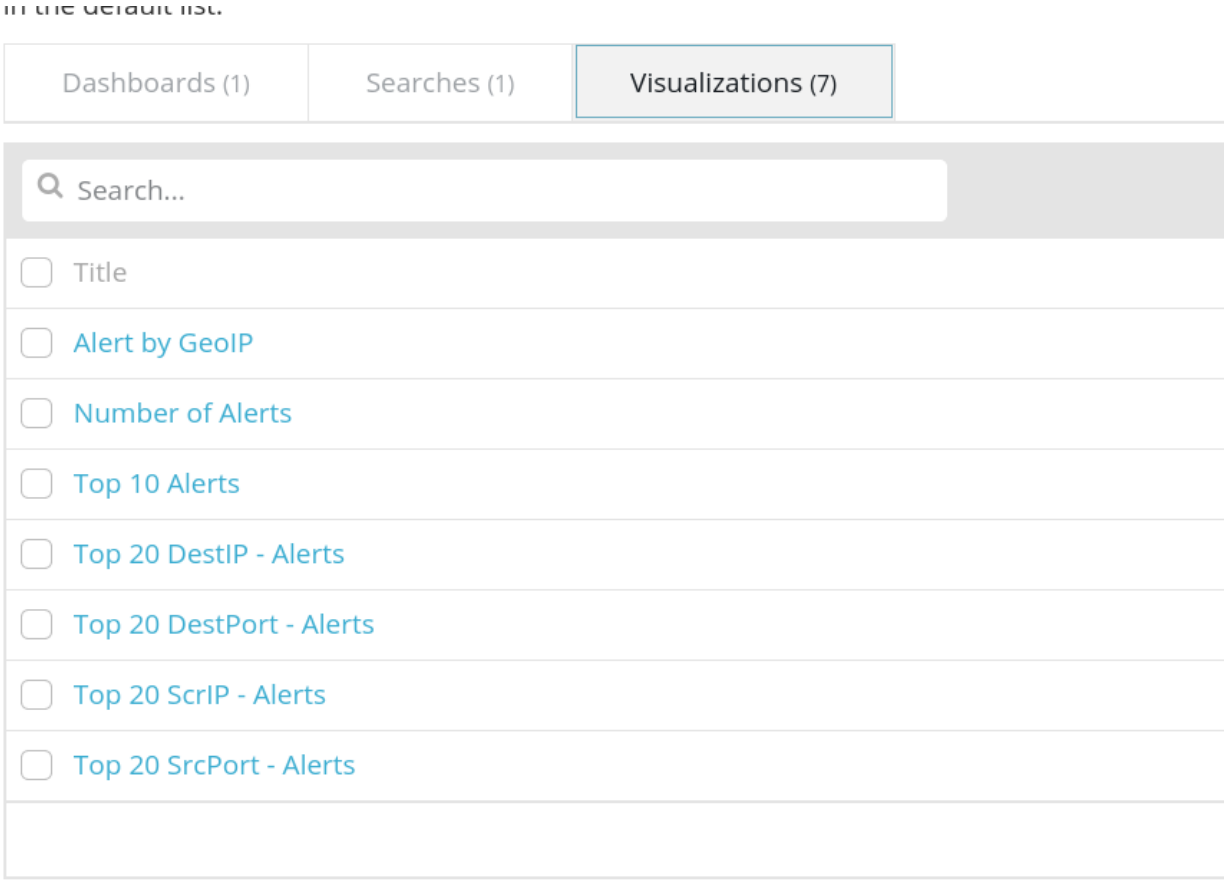


Figure 48: Visualization imports

Visualizing Alerts in Kibana

As shown in *Figure 49* below, the output of the **Discover** tab in Kibana shows a spike-bar in the years and dates for when the **QakBot** malware hit the network. It also, shows that it ingests the logs from Suricata `eve.json` properly.

Figure 49: Bar chart showing the year and date of QakBot malware attack

Figure 50: Alert by Geo-IP

Top 10 Alerts

alert.signature.keyword: Descending	Count
ET JA3 Hash - [Abuse.ch] Possible Quakbot	1,548
SURICATA Applayer Detect protocol only one direction	1,188
ET POLICY OpenSSL Demo CA - Internet Widgits Pty (O)	168
ET MALWARE Win32/Emotet CnC Activity (POST) M9	142
ET USER_AGENTS Microsoft Device Metadata Retrieval Client User-Agent	140
ET MALWARE Win32/Emotet CnC Activity (POST) M8	126
ET JA3 Hash - [Abuse.ch] Possible Gozi	84
SURICATA SMTP invalid reply	78
SURICATA Applayer Wrong direction first Data	50
ET POLICY PE EXE or DLL Windows file download HTTP	42

Figure 51: Top 10 Alerts

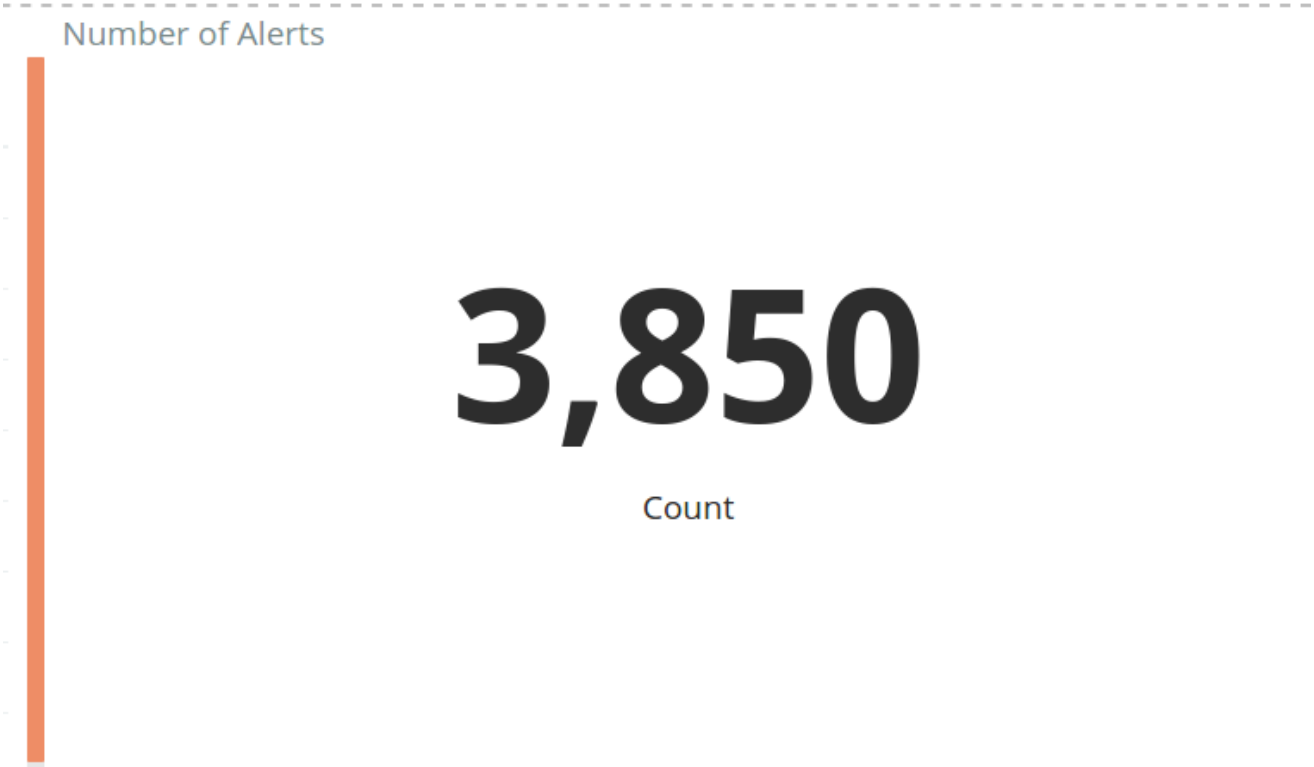


Figure 52: Number of Alerts

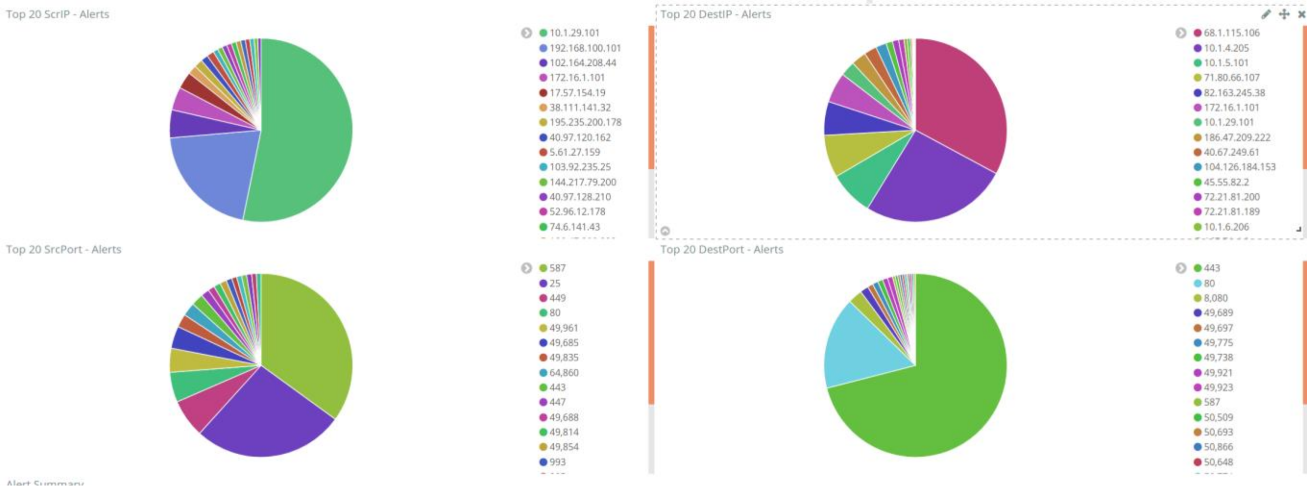
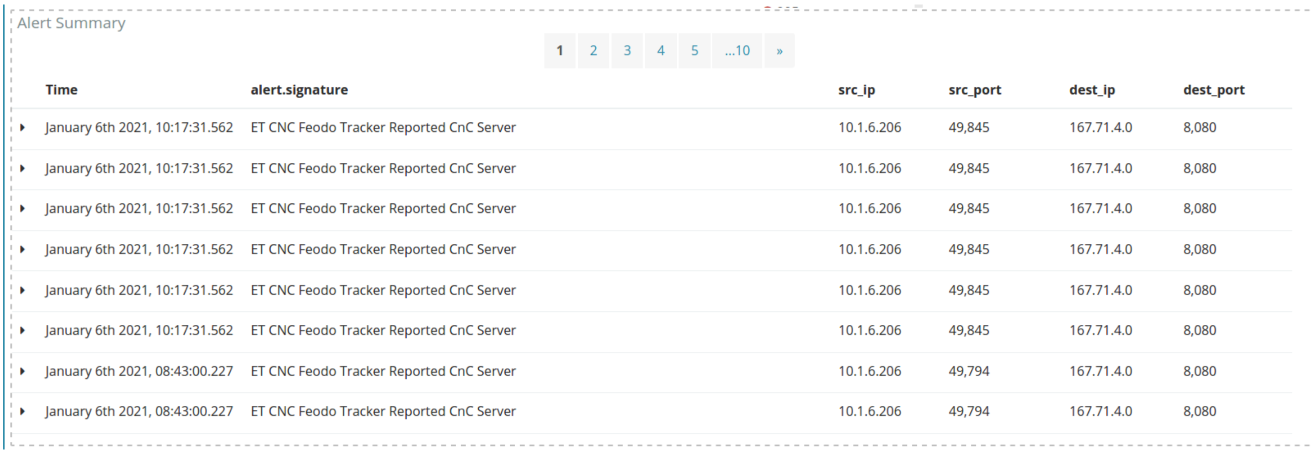


Figure 53: Alerts by `src-dest-IP` and `src-dest-port`


Alert Summary

Time	alert.signature	src_ip	src_port	dest_ip	dest_port
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 08:43:00.227	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,794	167.71.4.0	8,080
January 6th 2021, 08:43:00.227	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,794	167.71.4.0	8,080

Figure 54: Alert Summary

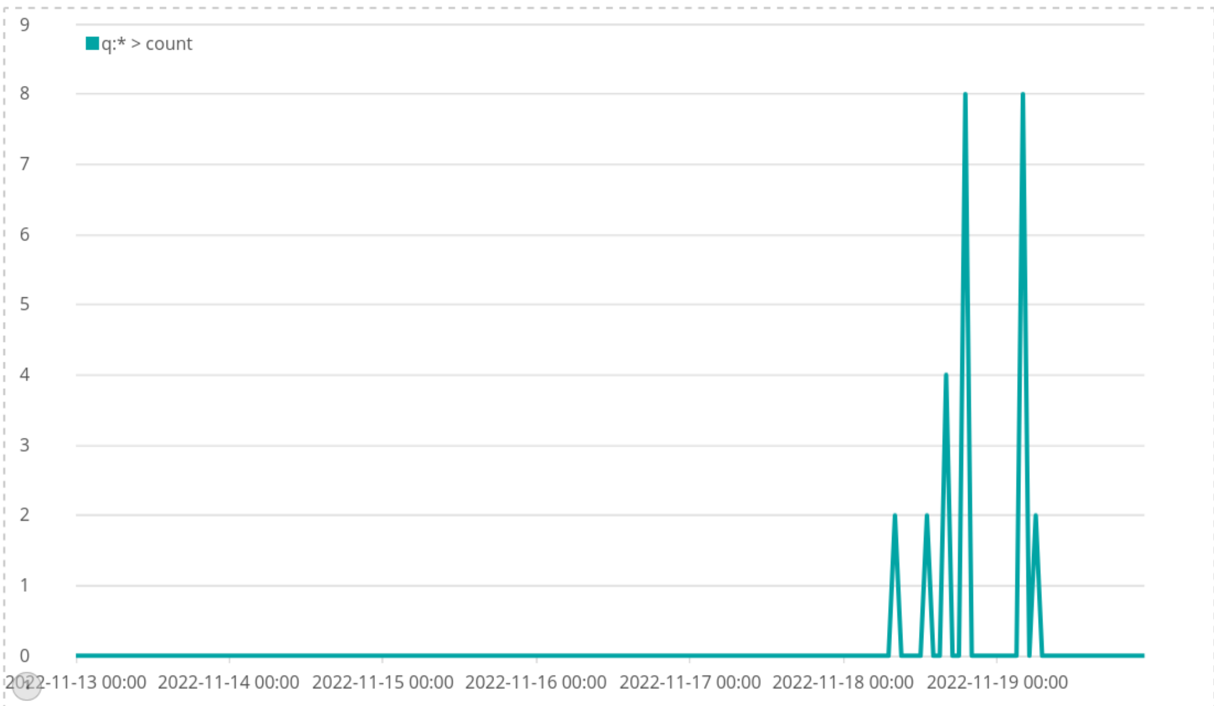


Figure 55: Timeline graph

Analyzing Alerts in Kibana

As seen in the ***Visualizing Alerts in Kibana*** section the benefits of the Elastic Stack is beyond question, as the logs generated by Suricata have been displayed in a visually and graphically grouped manner that makes analyzing, asssimilation, and understanding easy. Suricata does a great job in detecting intrusions (for an open source software), but by parsing the log output to the Elastic Stack more sense can be made out of the data. As seen above, Kibana displays: the **number of Alerts** (3,850), the **attack by geolocation**, the **date of the attack** (2019-12-31/2020-12-31), the **top ten alerts** (in which it immediately picks out **QakBot** and its infection process in a glance), and many more information!

Here, the aim is to briefly analyze the data generated by Kibana.

In addtion to the discovered `9312.zip` file by Suricata (see *Figure 27*) which contains the malicious VBS file that retrieves the QakBot malware (see *Figure 24*), as shown in *Figure 56* and *Figure 57*, it is clear that Kibana displays alerts of a potential second stage download (presumably the malware retrieved by the malicious VBS file) in a more human readable and noticeable way.

January 6th 2021, 08:42:26.512	ET INFO EXE - Served Attached HTTP	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:26.512	ET POLICY PE EXE or DLL Windows file download HTTP	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:26.512	ET INFO EXE - Served Attached HTTP	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:26.512	ET INFO Executable Retrieved With Minimal HTTP Headers - Potential Second Stage Download	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:26.512	ET INFO Executable Retrieved With Minimal HTTP Headers - Potential Second Stage Download	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:26.512	ET POLICY PE EXE or DLL Windows file download HTTP	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:26.512	ET INFO Executable Retrieved With Minimal HTTP Headers - Potential Second Stage Download	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:26.512	ET INFO Executable Retrieved With Minimal HTTP Headers - Potential Second Stage Download	103.92.235.25	80	10.1.6.206	49,775

Figure 56: Potential Second Stage Download and DLL Windows file download HTTP

t	http.hostname	seo.udaipurkart.com
t	http.http_content_type	application/octet-stream
t	http.http_method	GET
#	http.length	45,044
t	http.protocol	HTTP/1.1
#	http.status	200
t	http.url	/rx-5700-6hnr7/Sgms/
t	metadata.flowbits	min.gethttp, ET.http.binary
t	path	/var/log/suricata/eve.json
#	pcap_cnt	131,231
t	proto	TCP
t	src_ip	103.92.235.25
#	src_port	80

Figure 57: HTTP Headers for Downloaded File

The attack chain occurs continually in a loop.

January 6th 2021, 08:42:27.328	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:27.328	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:27.328	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:27.328	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:27.328	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	103.92.235.25	80	10.1.6.206	49,775
January 6th 2021, 08:42:27.328	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)	103.92.235.25	80	10.1.6.206	49,775

Figure 58: Alerts informing usage of a Malware Anti-Degubbung EXE

#	alert.severity	3
t	alert.signature	ET INFO EXE IsDebuggerPresent (Used in Malware Anti-Debugging)
#	alert.signature_id	2,015,744
t	app_proto	http
t	community_id	1:to7YeekESWCT2IQYVyRc/E+Qdu8=
t	dest_ip	10.1.6.206
#	dest_port	49,775
t	event_type	alert
?	files	{ "filename": "nDUrg8uFD5hl.dll", "size": 76792, "stored": false, "state": "UNKNOWN", "tx_id": 0, "gaps": false, "sid": [] }

Figure 59: Malicious DLL FIle and Potential **QakBot** malware

In *Figure 58* and *Figure 59*, there are continous requests to download the malicious DLL file which is believed to be the QakBot malware.

Time	alert.signature	src_ip	src_port	dest_ip	dest_port
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 10:17:31.562	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,845	167.71.4.0	8,080
January 6th 2021, 08:43:00.227	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,794	167.71.4.0	8,080
January 6th 2021, 08:43:00.227	ET CNC Feodo Tracker Reported CnC Server	10.1.6.206	49,794	167.71.4.0	8,080

Figure 60: Potential Execution of CnC Server (QakBot DLL)

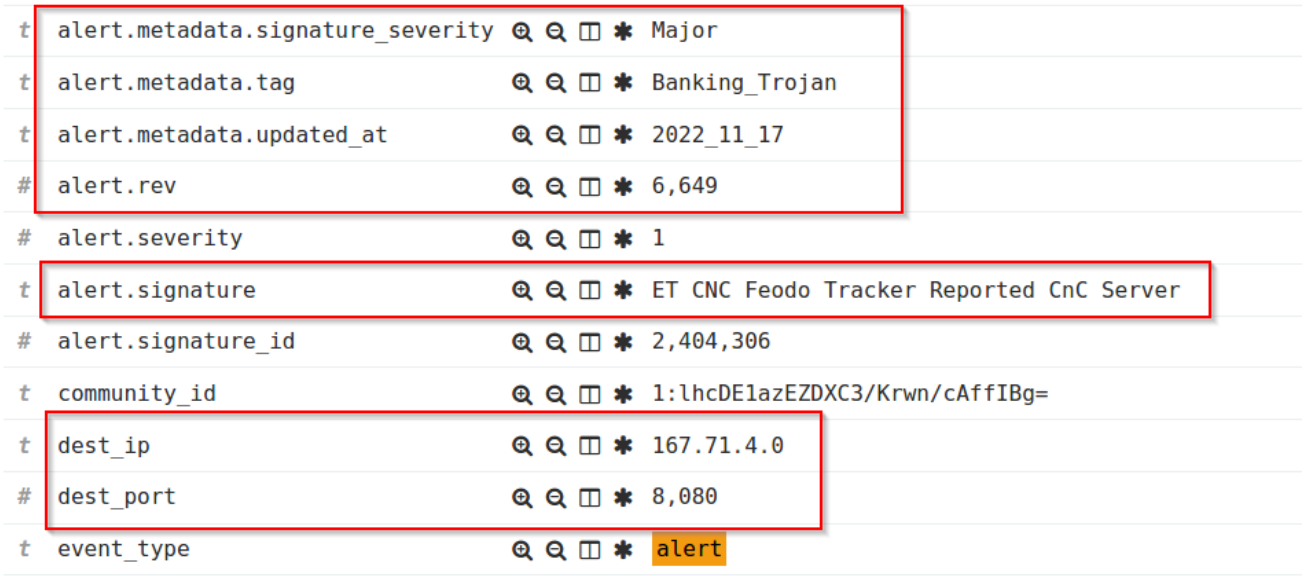


Figure 61: QakBot (Banking Trojan) Signature in Kibana

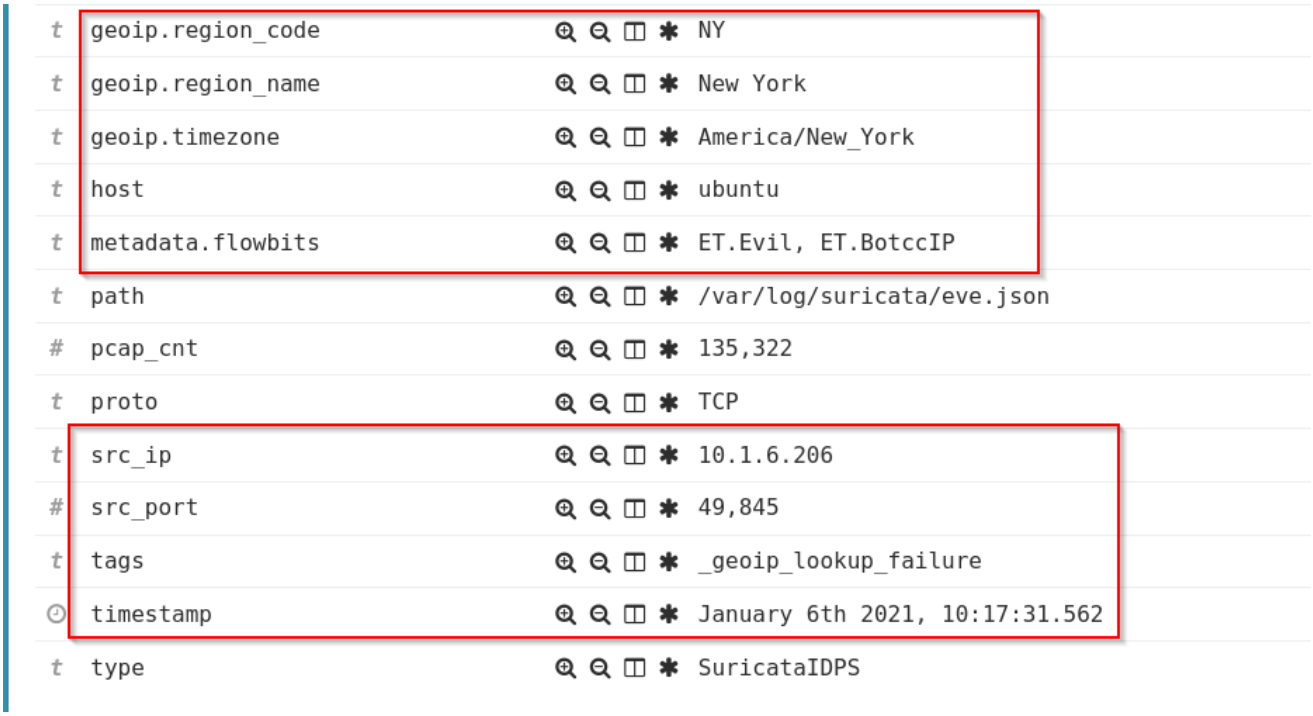


Figure 62: Geolocation Data of Host the Executed Malware

Obviously, *Figures 60, 61, and 62* shows that the QakBot malware had begun its post infection activities at this stage, which shows that it had successfully exploited an Ubuntu node on a network in New York, USA!

This is how multiple open-source tools can be combined to successfully detect and analyze intrusions on a network.

Recommendations

International Organization for Standardization. (2013). Section 12.2.1 states that we need to: "scan electronic mail attachments and downloads for malware before use; this scan should be carried out at different places, e.g. at electronic mail servers, desk top computers and when entering the network of the organization;"

Therefore, this attack would have been prevented if the user(s) followed the ISO controls set against malware.

Personal Reflection

This experiment taught the developers a lot about security monitoring and intrusion detection. The process of using an IDS to ingest and log data (to detect attacks on the network) was learned, and how that data can be graphically represented by combining multiple open source tools (Azure, Logstash, Elasticsearch, and Kibana).

During this experiment, issues were encountered, some of which are:

- The Microsoft tutorial did not show how to properly setup and configure Suricata, and the Elastic Stack. This information had to be learnt from a different resource. Although challenging, this gave the developers more understanding of the inner workings of Suricata.
- As seen in **Figure 28**, only Ubuntu v16.04 perfectly works with Elasticsearch v5.2.*. A failure always occurs when the Elasticsearch service is ran on versions > v16.04. A way had to be figured out to get the machine, as every Ubuntu v16.04 on Asure requires payment to use (only versions higher than that: v18.04, v20.04, and v22.04 are free). The Microsoft documentation does not state this.
- Running different releases of Kibana and Elasticsearch (e.g. Kibana 5.x and Elasticsearch 6.x) is not supported, nor is running a version of Kibana that is newer than the version of Elasticsearch (e.g. Kibana 5.1 and Elasticsearch 5.0).

Regardless of all the errors, the general experience and practical knowledge gained, of setting up, configuring, and deploying an IDS and using a data visualization stack to view logs produced by the IDS was very intriguing, helpful as much as necessary for network monitoring.

Conclusion

The benefits of the Elastic Stack in network monitoring is beyond question, as it makes the job easier. The experiment shows basic configuration of the stack and Suricata. The utilization of the open source tools helped to practically analyze malicious network traffic containing events that correlate to the operations of the QakBot malware. Not only did the stack and IDS help analyze, they also identified and successfully detected the malware execution process and attack.
The Elastic Stack and Suricata picked up on these malicious activities and graphically displayed the alerts in a grouped manner making it easy to understand.

In all, it was a great experience. Learning how to setup and configure an IDS, data visualization software, read attack logs, and more.

References

Duncan, B. (2020). *Wireshark Tutorial: Examining Qakbot Infections*. Unit 42. https://unit42.paloaltonetworks.com/tutorial-qakbot-infection/

Duncan, B. (2021). *Wireshark Tutorial: Examining Emotet Infection Traffic*. Unit 42. https://unit42.paloaltonetworks.com/wireshark-tutorial-emotet-infection/

The Honeynet Project. (2020). *EscapeRoom*. CyberDefenders. https://cyberdefenders.org/

Microsoft. (2022). *Perform network intrusion detection with Network Watcher and open source tools*. https://learn.microsoft.com/en-us/azure/network-watcher/network-watcher-intrusion-detection-open-source-tools

3CORESec. (2021). *3CORESec/testmynids.org: A website and framework for testing NIDS detection*. GitHub. https://github.com/3CORESec/testmynids.org

Mamidwar, S. (2022). *How to Install Elastic Stack 8 on Ubuntu 20.04 LTS* FOSSTechNix. https://www.fosstechnix.com/how-to-install-elastic-stack-8-on-ubuntu-20-04/

Elastic. (2022). *Support Matrix*. https://www.elastic.co/support/matrix

International Organization for Standardization. (2013). *Information technology Security techniques Code of practice for information security controls* (ISO Standard No. 27002:2013(E)). https://www.iso.org/obp/ui#iso:std:iso-iec:27002:dis:ed-3:v1:en

Appendices

Appendix A

Setting up an Azure Virtual Machine

This appendix documents the necessary steps that were taken to successfully create an Azure virtual machine.

Registration of Free Microsoft Azure Account

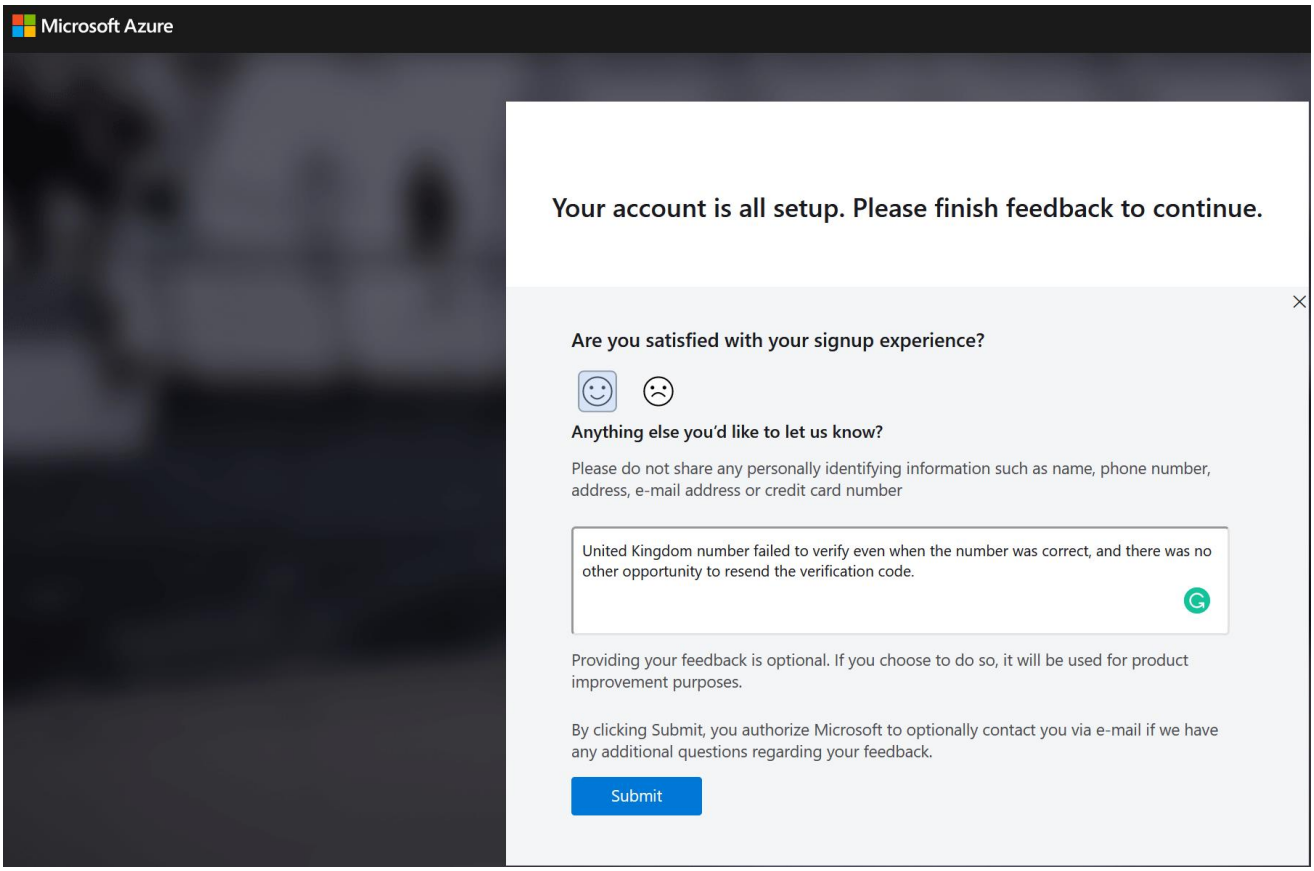


Figure 63: Complete Registration of Free Microsoft Azure Account.

Creating the Virtual Machine

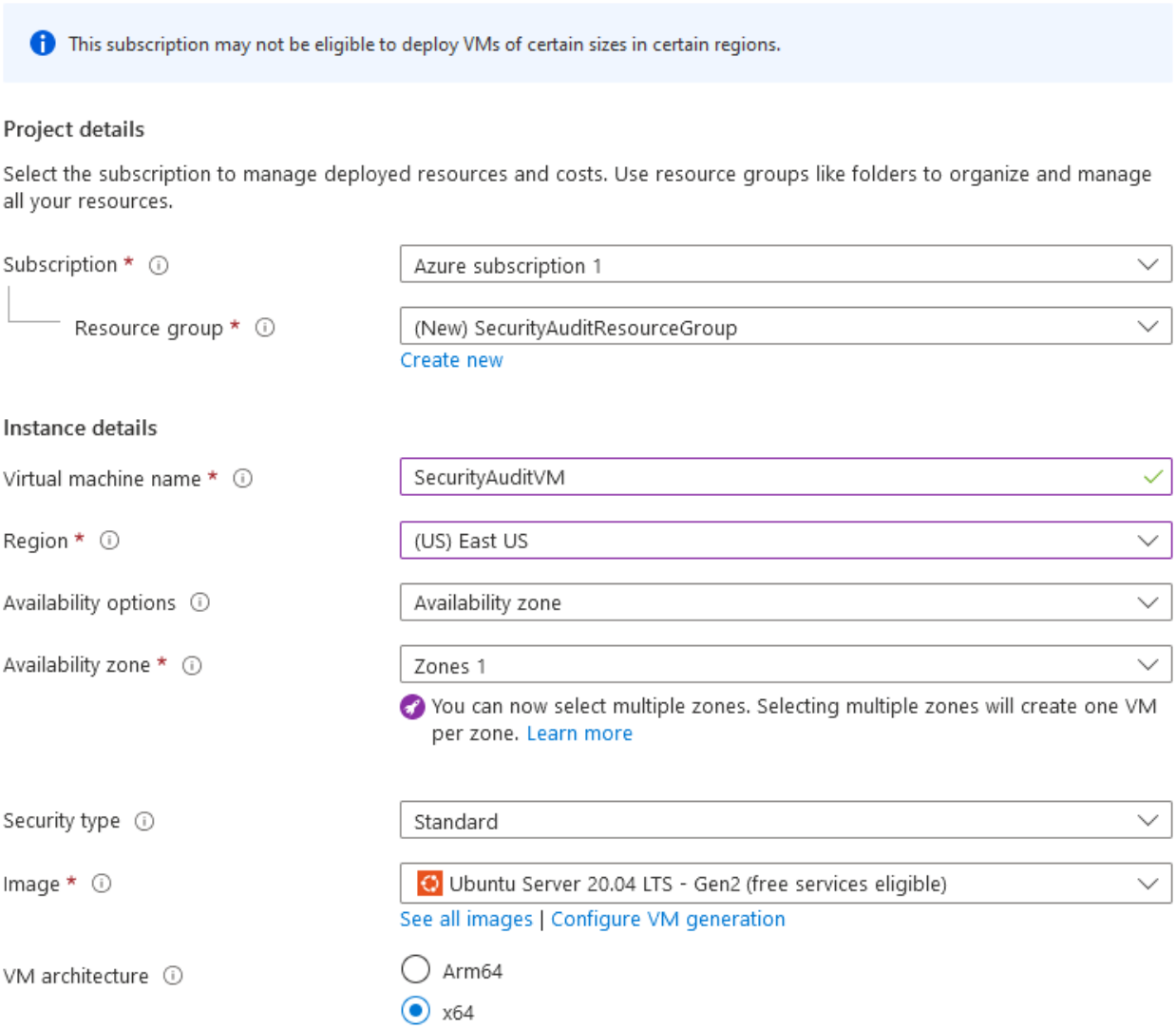


Figure 64: Azure Machine - Project Details

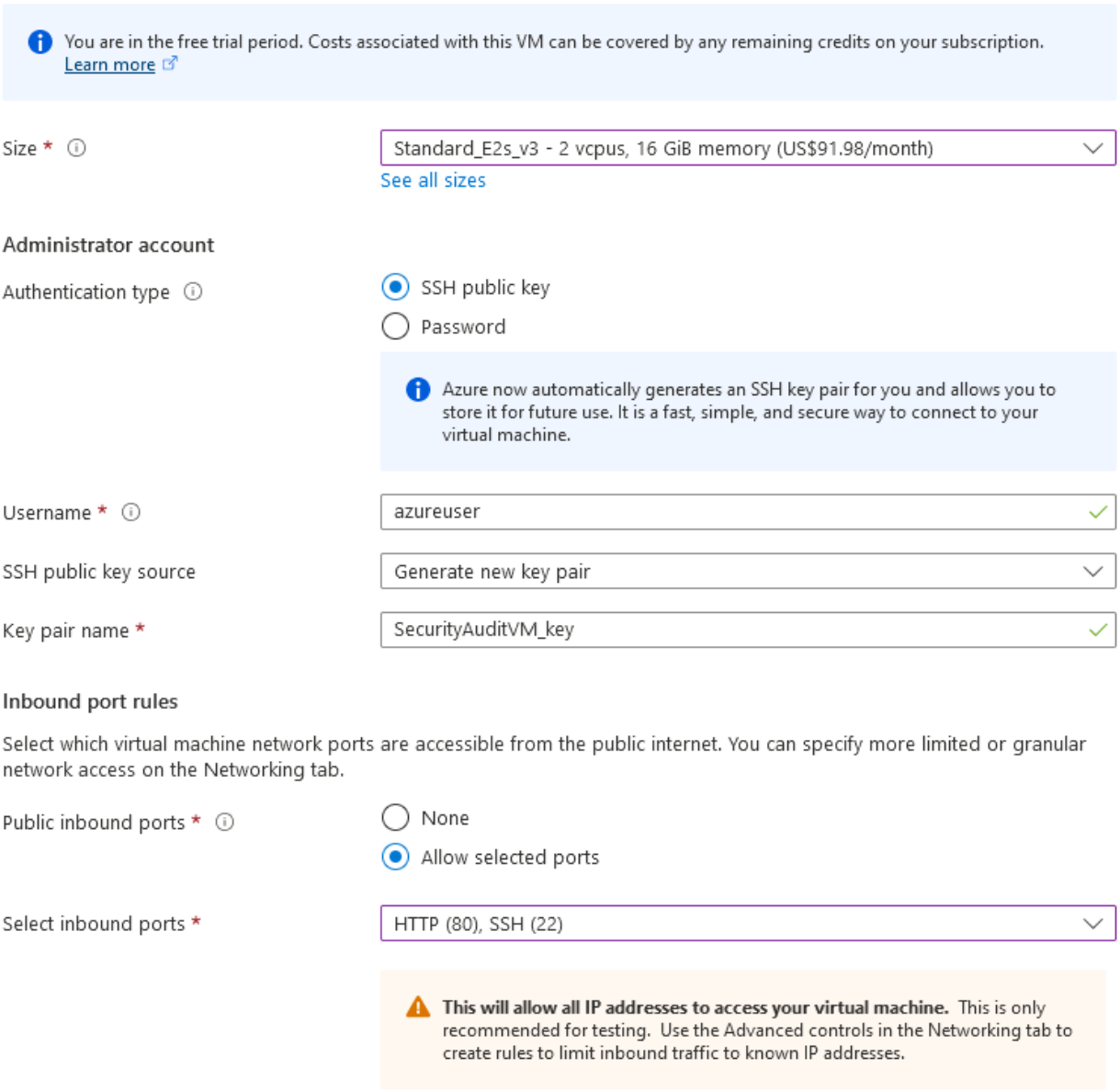


Figure 65: Azure VM Account settings and virtual machine memory size

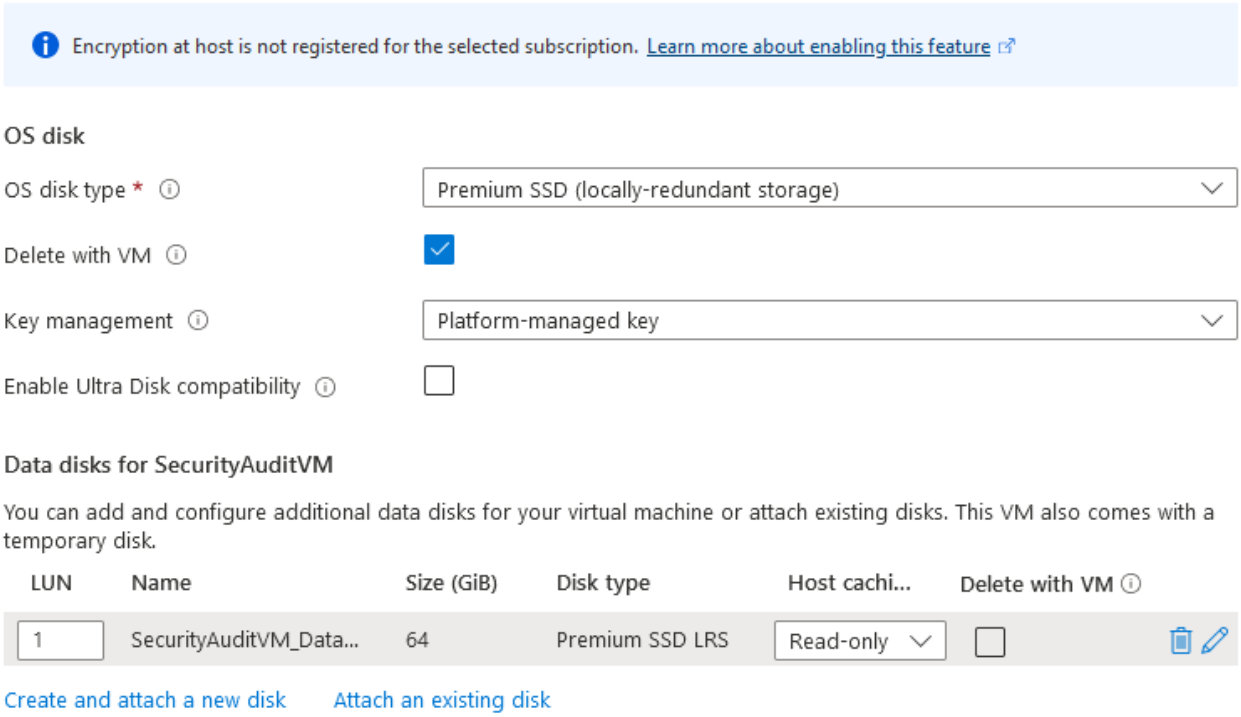


Figure 66: Virtual Machine Disk Size Allocation

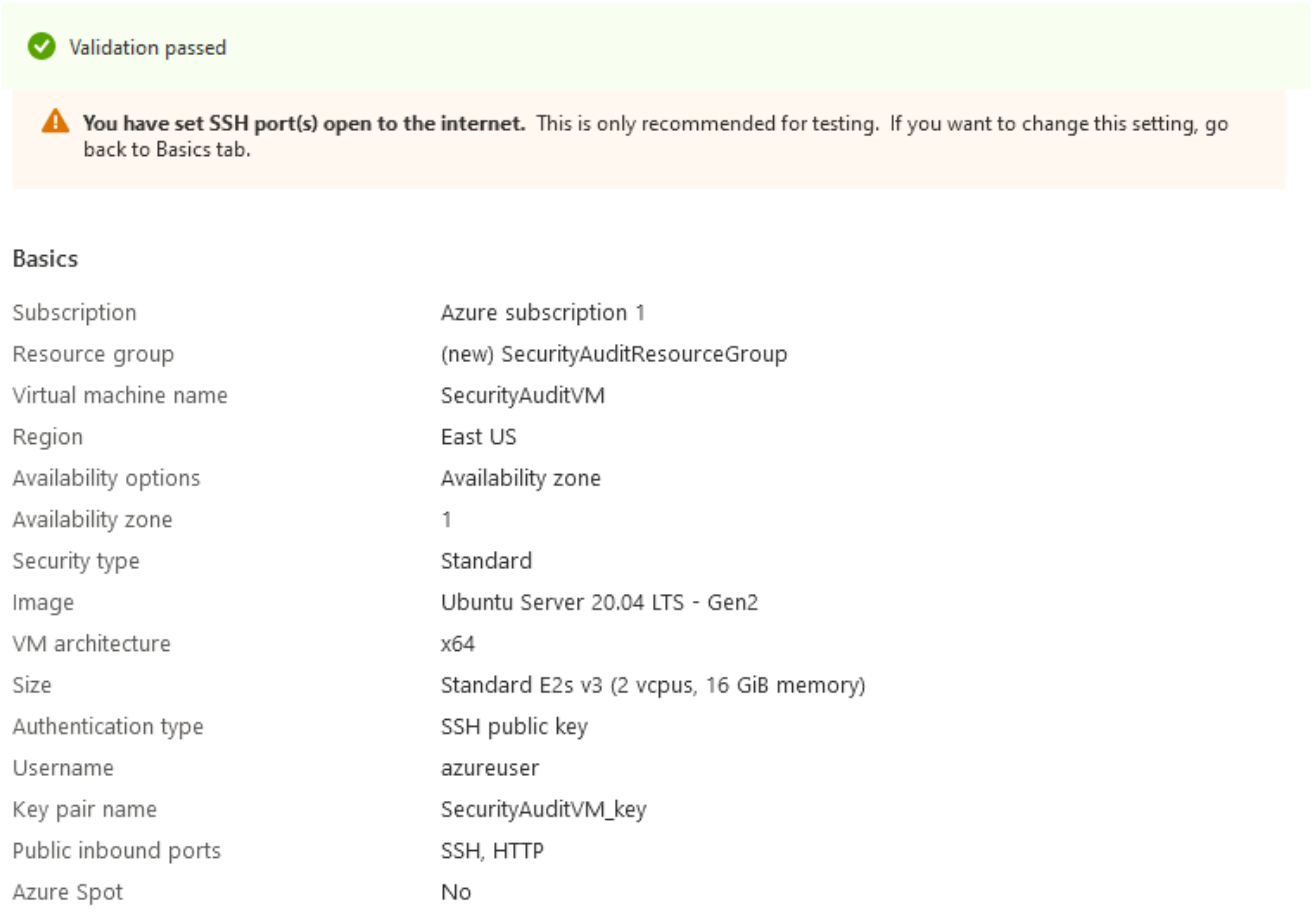


Figure 67: Basic Settings

Disks

OS disk type	Premium SSD LRS
Use managed disks	Yes
Delete OS disk with VM	Enabled
Data disks	1
Delete data disk with VM	0 disks enabled
Ephemeral OS disk	No

Networking

Virtual network	(new) SecurityAuditResourceGroup-vnet
Subnet	(new) default (10.0.0.0/24)
Public IP	(new) SecurityAuditVM-ip
Accelerated networking	On
Place this virtual machine behind an existing load balancing solution?	No
Delete public IP and NIC when VM is deleted	Disabled

Figure 68: Disks and Networking Settings

Management

Microsoft Defender for Cloud	Basic (free)
System assigned managed identity	Off
Login with Azure AD	Off
Auto-shutdown	Off
Backup	Disabled
Enable hotpatch	Off
Patch orchestration options	Image Default

Monitoring

Alerts	Off
Boot diagnostics	On
Enable OS guest diagnostics	Off

Advanced

Extensions	None
VM applications	None
Cloud init	No
User data	No
Disk controller type	SCSI
Proximity placement group	None
Capacity reservation group	None

Figure 69: Management, Monitoring, and Advanced Settings

Figure 70: Generating the SSH Key Pairs

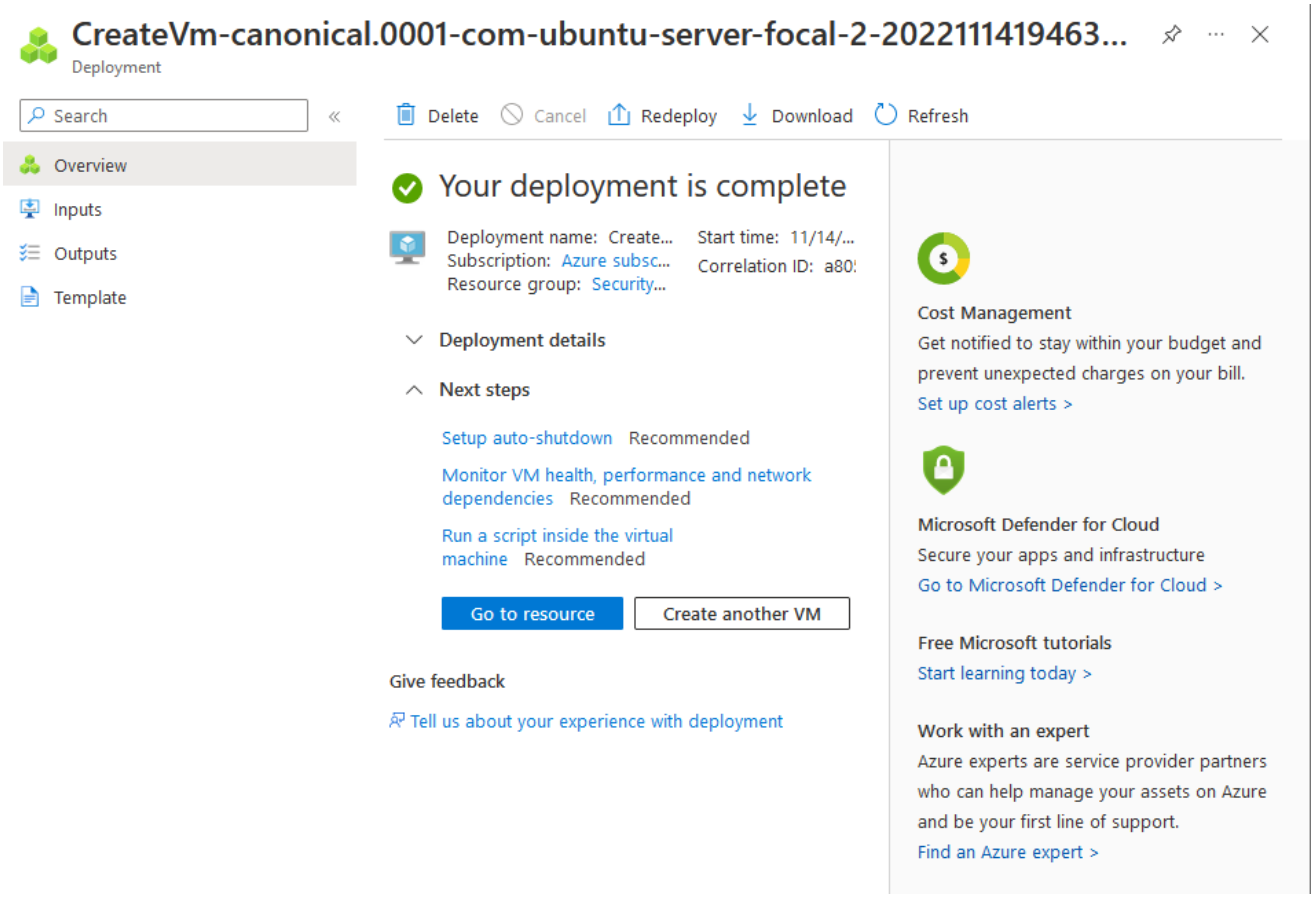


Figure 71: Complete Deployement of Azure VM

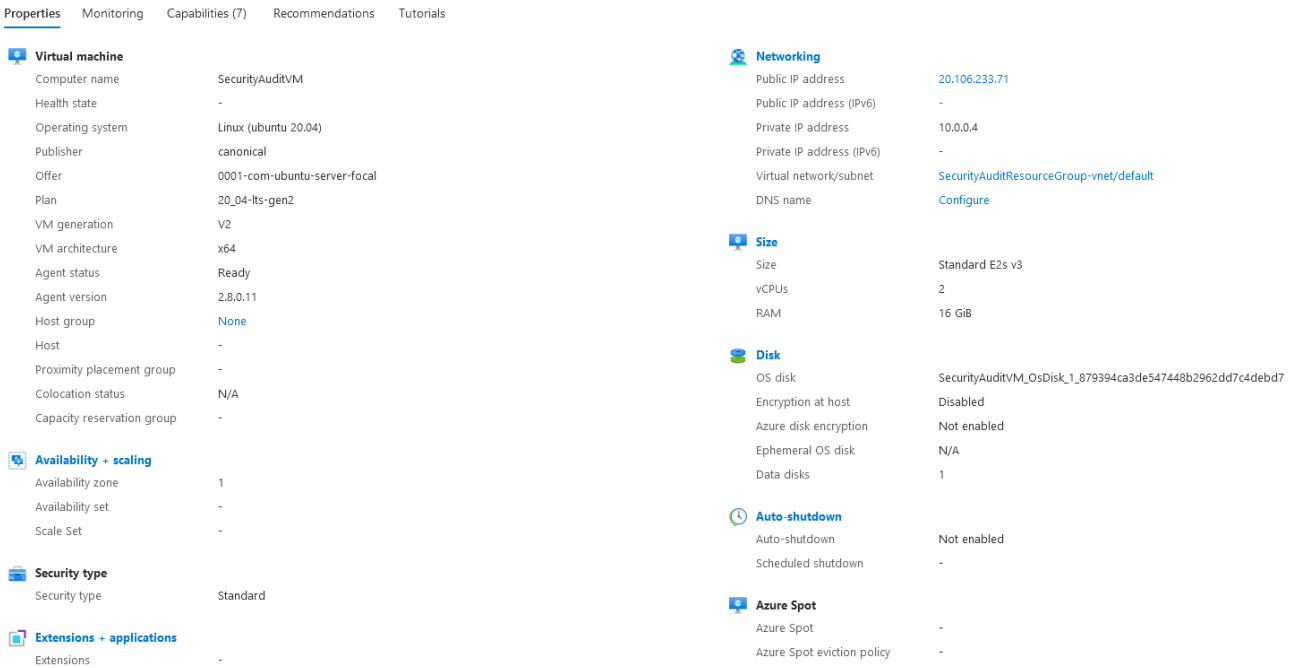


Figure 72: Virtual Machine Properties

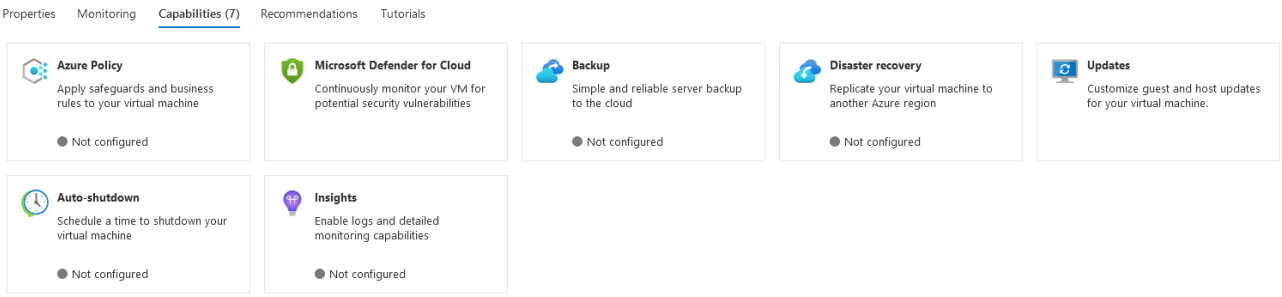


Figure 73: Virtual Machine Capabilities

Figure 74: VM Essentials