Analyzing TASI-PHIDC Network with ELK Stack

Roman Nicolai

***Abstract*—I want to research the ELK Stack and ultimately design a working ELK Stack implementation. ELK is an acronym that stands for three open-source projects: Elasticsearch, Logstash, and Kibana. These three components work in-tandem to create a powerful log ingestion and visualization tool. Elasticsearch is the JSON-based search engine, Logstash is the log ingestion pipeline, and Kibana is the visualization schema. The ELK Stack is useful for analyzing logs but perhaps most popularly, improving the cybersecurity posture of many organizations. My goal is to install an ELK Stack implementation that receives logs from different operating systems and manages to sort and report on those logs.**
Keywords—ELK Stack, Elasticsearch, Logstash, Kibana, cybersecurity, information technology, data visualization, log ingestion

I. Introduction

I have always been interested in tools like the ELK Stack but never had the knowledge or tools available to me to utilize them. However now, through my job at TASI-PHIDC, I have the opportunity to do so. I've been working at TASI for the last two years and have grown a lot as an IT/Network Engineer. Most importantly to this project, it allows me the ability to install tools like the ELK Stack. The ELK components can be installed and configured on a Linux virtual machines. At TASI, we use the VMware ESXi software to generate these virtual machines. So my research project will be done in tandem with TASI. I hope to research the ELK Stack and show a working implementation, along with solutions that the ELK Stack helps TASI achieve. Specifically, I would like to show authentication attempts to TASI's VPN, as well as other pertinent events (such as lockouts or malicious activity). This can be achieved through configuring the Logstash service to receive logs from particular IP addresses and ports, these fields being the data streams we want to receive logs from. In order to see authentication attempts, Logstash will want to receive logs from the SSLVPN firewall, as well as the Windows operating systems. Once we receive these logs, it would be ideal to parse the necessary fields so that we can easily search for them in Kibana. Additionally, I would like to report on this data on a daily basis. This would most likely require a bash script because I would prefer to avoid paying for Kibana's built-in reporting options. Lastly, I could look into additional tools such as ElastAlert which can monitor and alert on the working Elasticsearch service. Potential challenges that can arise from this project is obviously technical challenges including configuration errors, security vulnerabilities, and memory management.

A. Challenges

Aside from the obvious technical challenges that this project presents, there also is the challenge of available resources. The ELK Stack should be installed and configured in a virtual machine representing an operating system. How will this virtual machine be generated? It is possible to install the ELK components on a regular PC but that is discouraged. Also, what data streams will be utilized, and whose data will this belong to?

B. Motivations

The motivation behind this project is directly related to the previous "Challenges" section. I am motivated to go through with this project because I now have the resources and tools to build the ELK Stack through my employment with TASI-PHIDC. TASI-PHIDC can provide the virtual machine through their VMware ESXi software. They also provide the data to be ingested. Their firewall and machines together create hundreds of thousands of logs per day to analyze. I have always wanted to learn and design an ELK Stack implementation, or similar network analysis tool, but never had the opportunity to do so until now.

II. Background

A. Existing Knowledge

Before working on this project, I have some prior knowledge on the ELK Stack. I know that the ELK Stack is an open-source tool used for organizing and visualizing data from any source that can provide electronic data. It stores this data in schema-free JSON documents. It can access and manipulate this data through an HTTP web interface.

B. Related Implementations

The ELK Stack has seen use in numerous companies, including some Fortune 500 companies. Some of these companies include Walmart, Netflix, NASA, Goldman Sachs, and The United States Geological Survey [2]. Walmart has had success using the technology to detect fraud and scams aimed at customer demographics. Netflix and Goldman Sachs have used the technology to monitor enormous amounts of data created on their servers on a day-to-day basis. NASA and The United States Geological Survey use the ELK Stack for organizing and reporting on scientific data [3]. For example, the Mars rover sends data through Elasticsearch to help NASA scientists monitor its movements in real-time. The United

States Geological Survey uses the ELK Stack to track seismographic data to better anticipate and respond to natural disasters. It also integrates this with Twitter data to significantly boost its response time to these catastrophes.

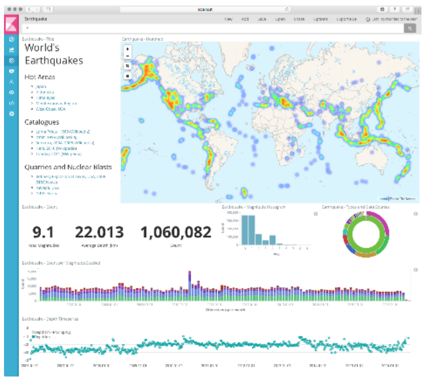


Fig. 1. Example of earthquake data analyzed with the Elastic Stack [1].

III. Problem Description

In terms of the context of this project, I want to install and configure the ELK Stack to monitor data generated by TASI-PHIDC machines. The problem that needs to be solved is in fact a series of smaller problems or steps needed to be completed. However, the overall goal is to report on TASI-PHIDC data surrounding the organization's VPN usage.

To approach this goal, it may be ideal to break it down into a series of smaller tasks. After the installation of the ELK Stack, it must be configured to accept logs from various ports in the firewall. After the data is coming in as desired, appropriate reports must be designed through Kibana and bash. Since I am trying to avoid the costs of the X-Pack features, I must create the reports on my own. A strategy is to design a bash report in tandem with crontab to automate report generation. The data I want to collect from the VPN is authentication attempts, login successes and failures, and logouts. This data is useful for monitoring what employees are using the network, and when. A step further is to alert on lockout events (i.e. when an employee gets locked out from the VPN because of too many failed login attempts). This can be accomplished through the install of ElastAlert, which is designed to interact with Elasticsearch data to manifest alerting. Perhaps the biggest "problem" would be the technical hurdles that need to be pushed through to accomplish the goal of a working ELK Stack implementation.

IV. Novel Ideas & Solutions

The ELK Stack is quite flexible in terms of how it can approach and solve problems. In this section I will attempt to list various solutions that the ELK Stack may help me reach.

A. *Log Sources: SSLVPN & Windows Event Logs*

It is important to target log sources that will provide the data we want. TASI-PHIDC creates various log streams, but two stand out – SSLVPN and Windows Event Logs. The SSLVPN stream contains logs such as VPN authentication attempts and failures. Windows Event Logs also contain authentication successes through DCON machines, along with lockout events corresponding to ID 4740. These are the two log streams that we would want to ingest, for the reasons stated above.

B. *User Generated Reports*

To avoid the costs that come with X-Pack (X-Pack is the paid add-on of the ELK Stack that provides features such as reporting), I must generate my own reports. A bash script is most likely the best approach. Most likely the bash script will pull some form of data from Elasticsearch or Kibana, and then email that data daily through crontab.

C. *Develop a custom Grok Filter*

The grok filter is implemented in the Logstash configuration file. As data is pipelined through Logstash, the grok filter allows the data to be parsed. The grok filter converts unstructured log data into something structured and query-able. The grok filter is highly customizable but also comes with its own syntax. Our use-case will require us to parse the SSLVPN logs rather than the Windows Event logs.

D. *Achieve memory management through ILM Polices*

If I let the logs enter the virtual machine unchecked, eventually they will flood the machine's storage until it crashes. Therefore, I must implement a memory management system or strategy. Luckily the ELK Stack has its own way to manage data. Logs are naturally stored in ELK in the form of indices. These indices can be configured by type (such as syslog/logstash or winlogbeat) and new indices can be generated after a specified period (such as day or month). The ILM policies that the user creates then deletes the index after a certain amount of time.

E. *Send data to external NAS server*

Since our data will be scheduled to be deleted eventually, a method to save this data should be designed. In this case we will send data to an external NAS (Network Attached Storage) for cold-storage. Because the NAS will have much more storage space than the virtual machine, I will not have to worry about the storage maxing out for a while. To send data to this NAS, it will have to mounted on the virtual machine.

F. *ElastAlert*

ElastAlert is a software framework designed for alerting on Elasticsearch data. In this case, I can install ElastAlert to alert on lockout events. These lockout events occur when a user fails to enter his or her VPN credentials correctly multiple times. The event is generated as a Windows log and shipped through Logstash to Elasticsearch. ElastAlert gives the capability to alert on events such as these via email or Slack.

G. *Integrate with NGINX*

I would also like to install NGINX on the server for additional security benefits. NGINX is a reverse proxy service that can intercept traffic and route it to a separate server. It can improve privacy through this. Specifically I will install NGINX for Kibana. The NGINX also allows us to give credentials to Kibana. This will force Kibana to be routed through NGINX which will require credentials for only authenticated users.

V. Experiment Setup & Configurations

A. *ELK Stack Operating System: CentOS 7 (Linux)*

I am choosing to install Elasticsearch, Logstash, and Kibana on a CentOS 7 virtual machine, which is a Linux operating system. There are other options of operating systems such as Ubuntu, Windows, Debian, and others, but I am going with Linux because I am most familiar with its terminal and commands. There are differences between the operating systems, but once the ELK Stack is installed and configured it should not matter to the scope of this project. Also, CentOS 7 gives context to other aspects of this report such as logging and general troubleshooting methods.

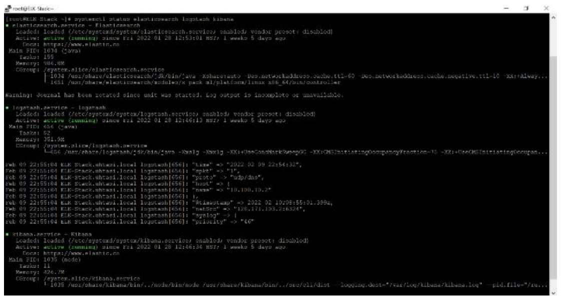
B. *VMware ESXi: Virtual Machines & Snapshotting*

To generate the CentOS 7 virtual machine, I will use VMware ESXi software. This software is a hypervisor which has the capability to deploy and serve virtual computers [4]. I am deciding to do the project in a virtual machine because it keeps the work isolated. It also allows for snapshotting. Snapshots can be taken when desired and allows the user to revert to a pervious version in the case of mistakes. This is useful because installing and configuring the ELK Stack is tedious and oftentimes reverting to a snapshot is easier than trying to untangle a mess of errors.

VI. Experiment Results

A. *ELK Stack Install*

The installation went successfully but was not without trouble. There were many errors to go through relating to configuration and permissions. However I was able to get through it.



```
root@ELK-Stack:~
[root@ELK-Stack ~]# systemctl status elasticsearch logstash kibana
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/etc/systemd/system/elasticsearch.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-01-28 12:53:01 HST; 1 weeks 5 days ago
     Docs: https://www.elastic.co
 Main PID: 1034 (java)
    Tasks: 155
   Memory: 986.0M
   CGroup: /system.slice/elasticsearch.service
           ├─1034 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.negative.ttl=10 -XX:+Alway...
           └─1651 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller

Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable.

● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-01-28 12:46:13 HST; 1 weeks 5 days ago
 Main PID: 656 (java)
    Tasks: 62
   Memory: 351.9M
   CGroup: /system.slice/logstash.service
           └─656 /usr/share/logstash/jdk/bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupan...

Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "time" => "2022-02-09 22:54:32",
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "npkt" => "1",
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "proto" => "udp/dns",
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "host" => {
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "name" => "10.100.10.2"
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: },
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "@timestamp" => 2022-02-10T08:55:01.399Z,
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "netSrc" => "128.171.153.2:6324",
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "syslog" => {
Feb 09 22:55:04 ELK-Stack.uhtasi.local logstash[656]: "priority" => "46"

● kibana.service - Kibana
   Loaded: loaded (/etc/systemd/system/kibana.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-01-28 12:46:34 HST; 1 weeks 5 days ago
     Docs: https://www.elastic.co
 Main PID: 1035 (node)
    Tasks: 11
   Memory: 426.7M
   CGroup: /system.slice/kibana.service
           └─1035 /usr/share/kibana/bin/../node/bin/node /usr/share/kibana/bin/../src/cli/dist --logging.dest="/var/log/kibana/kibana.log" --pid.file="/ru...
```

Fig. 2. The running statuses of the ELK services.

B. *Grok Filter*

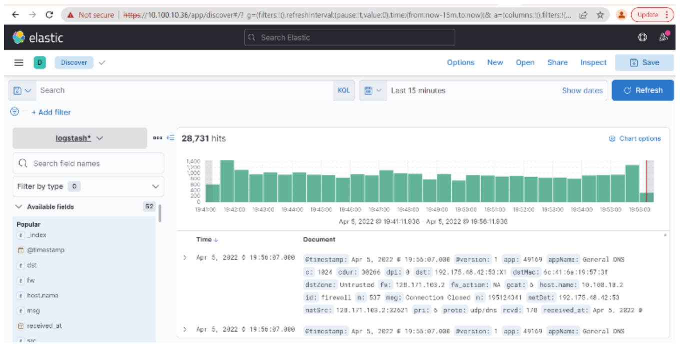
I wrote a grok filter located in our /etc/logstash/conf.d/logstash-syslog.conf file. However before discussing the grok filter, it may be better to introduce the input section of the Logstash configuration file. The input receives data from the sources on the specified ports. This is the code block:

```
input {
    tcp {
      port => 5055
      type => syslog
    }
    udp {
      port => 5055
      type => syslog
    }
    beats {
      port => 5044
      type => winlogbeat
    }
  }
```

Now I will discuss the grok filter itself. It is designed to parse syslog and winlogbeat fields. It was initially tagging a 'kv' field so I found syntax to remove it. It also assigns a date in the format desired to the syslog logs. Lastly it mutates data that is not ecs (Elastic Common Schema) which is data that is deemed to be common fields or data by the Elastic community. This is the grok filter:

```
  filter {
    if [type]=="syslog" {
          grok {
                  match => {
 "message" =>
"^<%{NUMBER:[syslog][priority]}>%{SPACE}++%{GREEDY
DATA:[syslog][message]}$"
                  }
    add_field => [ "received_at", "%{@timestamp}"
]
    add_field => [ "received_from", "%{host}"]
    remove_field => "message"
          }
          kv {
    source => "[syslog][message]"
    remove_field => "[syslog][message]"
          }
          date {
    match => [ "[time]", "yyyy-MM-dd HH:mm:ss" ]
    target => "@timestamp"
          }
    }

  if ![ecs] {
  mutate {
  rename => ["host", "[host][name]" ]
  update => { "[ecs][version]" => "1.5.0" }
  add_tag => [ "ecs_converted" ]
                  }
    }
  }
```

After the grok filter was designed, I started up the Logstash service along with the Kibana service. I then began receiving logs that were viewable and sortable in Kibana.



```
Apr 5, 2022 @ 19:56:07.000  @timestamp: Apr 5, 2022 @ 19:56:07.000 @version: 1 app: 49169 appName: General DNS
                            c: 1024 cdur: 30266 dpi: 0 dst: 192.175.48.42:53:X1 dstMac: 6c:41:6a:19:57:3f
                            dstZone: Untrusted fw: 128.171.103.2 fw_action: NA gcat: 6 host.name: 10.100.10.2
                            id: firewall m: 537 msg: Connection Closed n: 195124341 natDst: 192.175.48.42:53
                            natSrc: 128.171.103.2:32621 pri: 6 proto: udp/dns rcvd: 178 received_at: Apr 5, 2022 @
Apr 5, 2022 @ 19:56:07.000  @timestamp: Apr 5, 2022 @ 19:56:07.000 @version: 1 app: 49169 appName: General DNS
```

Fig. 3. Logs in Kibana’s Discover feature.

C. *ILM Policies & Index Templates*

I created the daily logstash and winlogbeat indices and patterns through two different methods. I created the initial logstash index through Dev Tools in Kibana, in which I also assigned it an alias. On the other hand, I create the winlogbeat daily indices through the output block of the logstash-syslog.conf file. The reason behind this is that the winlogbeat logs have a usable timestamp, while the syslog logs seem to not have one. There may be a workaround, but as of now I am going with this solution.

I then created an ILM policy that specifies a deletion phase after 25 hours. 25 hours gives enough time for the daily reports to have the necessary data as well as giving time for the scripts to run and pull data from Kibana. Lastly I created index patterns for logstash-* and winlogbeat-* which automatically assigns the generated indices to an ILM policy.

D. *Reporting*

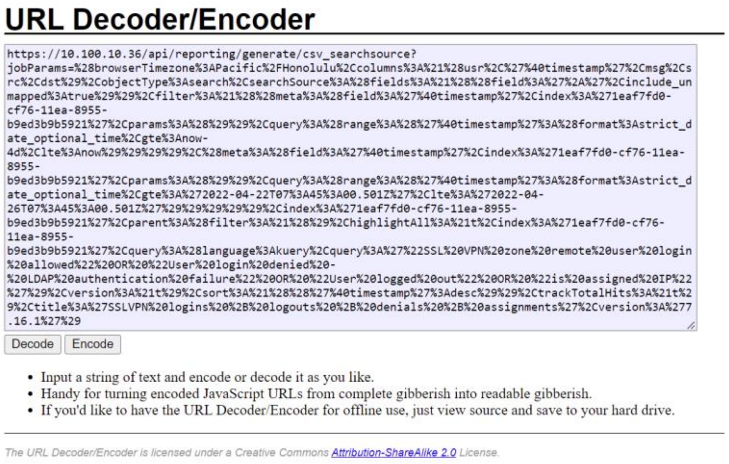
I managed to achieve reporting through a homemade bash script. I also utilized the POST URL generated through Kibana. The POST URL was difficult to deal with initially because I had to decode it and add date math and then reencode it.

```
(range:('@timestamp':(format:strict_date_optional_
time,gte:'2020-11- 08T02:34:13.350Z',lte:'2020-11-
09T02:34:13.350Z'))))
```

We want the gte and lte time to be fluid and not fixed on a static range. To do this we add “date math” in place of the real times. This is what the new range looks like with “date math”:

```
(range:('@timestamp':(format:strict_date_optional_
time,gte:now-1d,lte:now))))
```

This allows the POST URL to constantly readjust its scope in terms of the date. Without this adjustment, the POST URL will only look at the fixed range, which is not ideal.



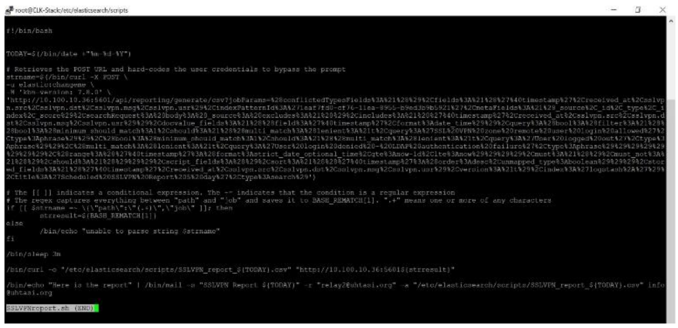
```
https://10.100.10.36/api/reporting/generate/csv_searchsource?
jobParams=%28browserTimezone%3APacific%2FHonolulu%2Ccolumns%3A%21%28usr%2C%27%40timestamp%27%2Cmsg%2Cs
rc%2Cdst%29%2CobjectType%3Asearch%2CsearchSource%3A%28fields%3A%21%28%28field%3A%27%2A%27%2Cinclude_un
mapped%3Atrue%29%29%2Cfilter%3A%21%28%28meta%3A%28field%3A%27%40timestamp%27%2Cindex%3A%271eaf7fd0-
cf76-11ea-8955-
b9ed3b9b5921%27%2Cparams%3A%28%29%29%2Cquery%3A%28range%3A%28%27%40timestamp%27%3A%28format%3Astrict_d
ate_optional_time%2Cgte%3Anow-
4d%2Clte%3Anow%29%29%29%29%2C%28meta%3A%28field%3A%27%40timestamp%27%2Cindex%3A%271eaf7fd0-cf76-11ea-
8955-
b9ed3b9b5921%27%2Cparams%3A%28%29%29%2Cquery%3A%28range%3A%28%27%40timestamp%27%3A%28format%3Astrict_d
ate_optional_time%2Cgte%3A%272022-04-22T07%3A45%3A00.501Z%27%2Clte%3A%272022-04-
26T07%3A45%3A00.501Z%27%29%29%29%29%29%2Cindex%3A%271eaf7fd0-cf76-11ea-8955-
b9ed3b9b5921%27%2Cparent%3A%28filter%3A%21%28%29%2ChighlightAll%3A%21t%2Cindex%3A%271eaf7fd0-cf76-
11ea-8955-
b9ed3b9b5921%27%2Cquery%3A%28language%3Akuery%2Cquery%3A%27%22SSL%20VPN%20zone%20remote%20user%20login
%20allowed%22%20OR%20%22User%20login%20denied%20-
%20LDAP%20authentication%20failure%22%20OR%20%22User%20logged%20out%22%20OR%20%22is%20assigned%20IP%22
%27%29%2Cversion%3A%21t%29%2Csort%3A%21%28%28%27%40timestamp%27%3Adesc%29%29%2CtrackTotalHits%3A%21t%2
9%2Ctitle%3A%27SSLVPN%20logins%20%2B%20logouts%20%2B%20denials%20%2B%20assignments%27%2Cversion%3A%277
.16.1%27%29
```

- Input a string of text and encode or decode it as you like.
- Handy for turning encoded JavaScript URLs from complete gibberish into readable gibberish.
- If you'd like to have the URL Decoder/Encoder for offline use, just view source and save to your hard drive.

The URL Decoder/Encoder is licensed under a Creative Commons Attribution-ShareAlike 2.0 License.

Fig. 4. The URL Decoder/Encoder used to insert the date math.

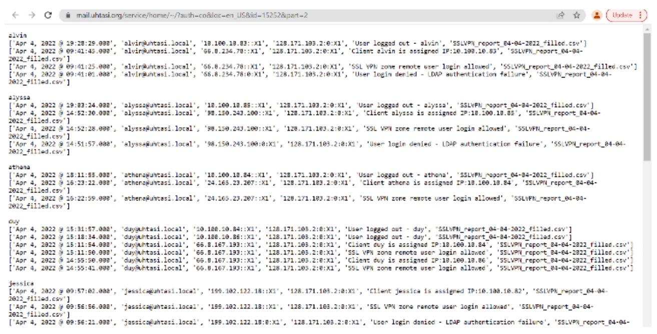
The script itself was slightly complex because of the POST URL and formatting. Small quirks such as specifying the entire command path and sleeping between curling and mailing were required.



```
#!/bin/bash

TODAY=$(/bin/date +"%m-%d-%Y")

# Retrieves the POST URL and hard-codes the user credentials to bypass the prompt
strname=$(/bin/curl -X POST \
-u elastic:changeme \
-H 'kbn-version: 7.8.0' \
'http://10.100.10.36:5601/api/reporting/generate/csv?jobParams=%28conflictedTypesFields%3A%21%28%29%2Cfields%3A%21%28%27%40timestamp%27%2Creceived_at%2Csslvp
n.src%2Csslvpn.dst%2Csslvpn.msg%2Csslvpn.usr%29%2CindexPatternId%3A%271eaf7fd0-cf76-11ea-8955-b9ed3b9b5921%27%2CmetaFields%3A%21%28_source%2C_id%2C_type%2C_i
ndex%2C_score%29%2CsearchRequest%3A%28body%3A%28_source%3A%28excludes%3A%21%28%29%2Cincludes%3A%21%28%27%40timestamp%27%2Creceived_at%2Csslvpn.src%2Csslvpn.d
st%2Csslvpn.msg%2Csslvpn.usr%29%29%2Cdocvalue_fields%3A%21%28%28field%3A%27%40timestamp%27%2Cformat%3Adate_time%29%29%2Cquery%3A%28bool%3A%28filter%3A%21%28%
28bool%3A%28minimum_should_match%3A1%2Cshould%3A%21%28%28multi_match%3A%28lenient%3A%21t%2Cquery%3A%27SSL%20VPN%20zone%20remote%20user%20login%20allowed%27%2
Ctype%3Aphrase%29%29%2C%28bool%3A%28minimum_should_match%3A1%2Cshould%3A%21%28%28multi_match%3A%28lenient%3A%21t%2Cquery%3A%27User%20logged%20out%27%2Ctype%3
Aphrase%29%29%2C%28multi_match%3A%28lenient%3A%21t%2Cquery%3A%27User%20login%20denied%20-%20LDAP%20authentication%20failure%27%2Ctype%3Aphrase%29%29%29%29%29
%29%29%29%2C%28range%3A%28%27%40timestamp%27%3A%28format%3Astrict_date_optional_time%2Cgte%3Anow-1d%2Clte%3Anow%29%29%29%29%2Cmust%3A%21%28%29%2Cmust_not%3A%
21%28%29%2Cshould%3A%21%28%29%29%29%2Cscript_fields%3A%28%29%2Csort%3A%21%28%28%27%40timestamp%27%3A%28order%3Adesc%2Cunmapped_type%3Aboolean%29%29%29%2Cstor
ed_fields%3A%21%28%27%40timestamp%27%2Creceived_at%2Csslvpn.src%2Csslvpn.dst%2Csslvpn.msg%2Csslvpn.usr%29%2Cversion%3A%21t%29%2Cindex%3A%27logstash%2A%27%29%
2Ctitle%3A%27Scheduled%20SSLVPN%20Report%205%20day%27%2Ctype%3Asearch%29')

# The [[ ]] indicates a conditional expression. The =~ indicates that the condition is a regular expression
# The regex captures everything between "path" and "job" and saves it to BASH_REMATCH[1]. ".+" means one or more of any characters
if [[ $strname =~ \{\"path\":\"(.+)\",\"job\" ]]; then
        strresult=${BASH_REMATCH[1]}
else
        /bin/echo "unable to parse string $strname"
fi

/bin/sleep 3m

/bin/curl -o "/etc/elasticsearch/scripts/SSLVPN_report_${TODAY}.csv" "http://10.100.10.36:5601${strresult}"

/bin/echo "Here is the report" | /bin/mail -s "SSLVPN Report ${TODAY}" -r "relay2@uhtasi.org" -a "/etc/elasticsearch/scripts/SSLVPN_report_${TODAY}.csv" info
@uhtasi.org

SSLVPNreport.sh (END)
```

Fig. 5. The finished reporting script.

Overall it manages to successfully email the report in text form. It is technically a .csv file but appears to look like just a text file.


mail.uhtasi.org/service/home/~/?auth=co&loc=en_US&id=15252&part=2
Update

```
alvin
['Apr 4, 2022 @ 19:28:29.000', 'alvin@uhtasi.local', '10.100.10.83::X1', '128.171.103.2:0:X1', 'User logged out - alvin', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 09:41:45.000', 'alvin@uhtasi.local', '66.8.234.78::X1', '128.171.103.2:0:X1', 'Client alvin is assigned IP:10.100.10.83', 'SSLVPN_report_04-04-
2022_filled.csv']
['Apr 4, 2022 @ 09:41:25.000', 'alvin@uhtasi.local', '66.8.234.78::X1', '128.171.103.2:0:X1', 'SSL VPN zone remote user login allowed', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 09:41:01.000', 'alvin@uhtasi.local', '66.8.234.78:0:X1', '128.171.103.2:0:X1', 'User login denied - LDAP authentication failure', 'SSLVPN_report_04-04-
2022_filled.csv']

alyssa
['Apr 4, 2022 @ 19:03:24.000', 'alyssa@uhtasi.local', '10.100.10.85::X1', '128.171.103.2:0:X1', 'User logged out - alyssa', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 14:52:30.000', 'alyssa@uhtasi.local', '98.150.243.100::X1', '128.171.103.2:0:X1', 'Client alyssa is assigned IP:10.100.10.85', 'SSLVPN_report_04-04-
2022_filled.csv']
['Apr 4, 2022 @ 14:52:28.000', 'alyssa@uhtasi.local', '98.150.243.100::X1', '128.171.103.2:0:X1', 'SSL VPN zone remote user login allowed', 'SSLVPN_report_04-04-
2022_filled.csv']
['Apr 4, 2022 @ 14:51:57.000', 'alyssa@uhtasi.local', '98.150.243.100:0:X1', '128.171.103.2:0:X1', 'User login denied - LDAP authentication failure', 'SSLVPN_report_04-04-
2022_filled.csv']

athena
['Apr 4, 2022 @ 18:11:55.000', 'athena@uhtasi.local', '10.100.10.84::X1', '128.171.103.2:0:X1', 'User logged out - athena', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 16:23:22.000', 'athena@uhtasi.local', '24.165.23.207::X1', '128.171.103.2:0:X1', 'Client athena is assigned IP:10.100.10.84', 'SSLVPN_report_04-04-
2022_filled.csv']
['Apr 4, 2022 @ 16:22:59.000', 'athena@uhtasi.local', '24.165.23.207::X1', '128.171.103.2:0:X1', 'SSL VPN zone remote user login allowed', 'SSLVPN_report_04-04-
2022_filled.csv']

duy
['Apr 4, 2022 @ 15:31:57.000', 'duy@uhtasi.local', '10.100.10.84::X1', '128.171.103.2:0:X1', 'User logged out - duy', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 15:18:34.000', 'duy@uhtasi.local', '10.100.10.86::X1', '128.171.103.2:0:X1', 'User logged out - duy', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 15:11:54.000', 'duy@uhtasi.local', '66.8.167.193::X1', '128.171.103.2:0:X1', 'Client duy is assigned IP:10.100.10.84', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 15:11:50.000', 'duy@uhtasi.local', '66.8.167.193::X1', '128.171.103.2:0:X1', 'SSL VPN zone remote user login allowed', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 14:55:50.000', 'duy@uhtasi.local', '66.8.167.193::X1', '128.171.103.2:0:X1', 'Client duy is assigned IP:10.100.10.86', 'SSLVPN_report_04-04-2022_filled.csv']
['Apr 4, 2022 @ 14:55:41.000', 'duy@uhtasi.local', '66.8.167.193::X1', '128.171.103.2:0:X1', 'SSL VPN zone remote user login allowed', 'SSLVPN_report_04-04-2022_filled.csv']

jessica
['Apr 4, 2022 @ 09:57:02.000', 'jessica@uhtasi.local', '199.102.122.18::X1', '128.171.103.2:0:X1', 'Client jessica is assigned IP:10.100.10.82', 'SSLVPN_report_04-04-
2022_filled.csv']
['Apr 4, 2022 @ 09:56:56.000', 'jessica@uhtasi.local', '199.102.122.18::X1', '128.171.103.2:0:X1', 'SSL VPN zone remote user login allowed', 'SSLVPN_report_04-04-
2022_filled.csv']
['Apr 4, 2022 @ 09:56:21.000', 'jessica@uhtasi.local', '199.102.122.18:0:X1', '128.171.103.2:0:X1', 'User login denied - LDAP authentication failure', 'SSLVPN_report_04-04-
```

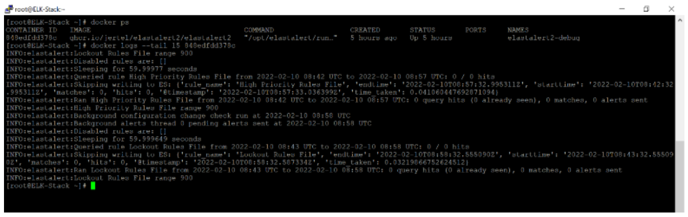
Fig. 6. The report once generated from the POST URL and emailed.

E. *Integrate with NGINX*

Before installing nginx, Kibana and Elasticsearch must be configured to be on localhost. After this is done, nginx may be installed. After nginx is installed, the credentials may be specified in the /etc/nginx/htpasswd.users file. Additionally, a configuration file for nginx must be added. Once this is set-up, only authorized users will be able to access Kibana's data.

F. *ElastAlert*

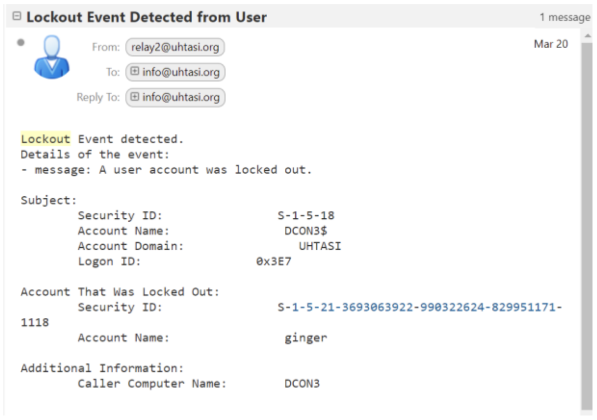
I installed Docker on the server to run ElastAlert2 through a modified docker run command. The docker run command specified volumes that were to be used such as the configuration file and SMTP authentication file. It also specified the rules directory containing the various rule YAML files. The docker run command was later given the debug flag to run in debug mode because of the ability to see more granular logs for troubleshooting reasons.



```
[root@ELK-Stack ~]# docker ps
CONTAINER ID   IMAGE                                  COMMAND                   CREATED       STATUS       PORTS     NAMES
848edfdd378c   ghcr.io/jertel/elastalert2/elastalert2   "/opt/elastalert/run…"   5 hours ago   Up 5 hours             elastalert2-debug
[root@ELK-Stack ~]# docker logs --tail 15 848edfdd378c
INFO:elastalert:Lockout Rules File range 900
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 59.99977 seconds
INFO:elastalert:Queried rule High Priority Rules File from 2022-02-10 08:42 UTC to 2022-02-10 08:57 UTC: 0 / 0 hits
INFO:elastalert:Skipping writing to ES: {'rule_name': 'High Priority Rules File', 'endtime': '2022-02-10T08:57:32.995311Z', 'starttime': '2022-02-10T08:42:32
.995311Z', 'matches': 0, 'hits': 0, '@timestamp': '2022-02-10T08:57:33.036399Z', 'time_taken': 0.041060447692871094}
INFO:elastalert:Ran High Priority Rules File from 2022-02-10 08:42 UTC to 2022-02-10 08:57 UTC: 0 query hits (0 already seen), 0 matches, 0 alerts sent
INFO:elastalert:High Priority Rules File range 900
INFO:elastalert:Background configuration change check run at 2022-02-10 08:58 UTC
INFO:elastalert:Background alerts thread 0 pending alerts sent at 2022-02-10 08:58 UTC
INFO:elastalert:Disabled rules are: []
INFO:elastalert:Sleeping for 59.999649 seconds
INFO:elastalert:Queried rule Lockout Rules File from 2022-02-10 08:43 UTC to 2022-02-10 08:58 UTC: 0 / 0 hits
INFO:elastalert:Skipping writing to ES: {'rule_name': 'Lockout Rules File', 'endtime': '2022-02-10T08:58:32.555090Z', 'starttime': '2022-02-10T08:43:32.55509
0Z', 'matches': 0, 'hits': 0, '@timestamp': '2022-02-10T08:58:32.587334Z', 'time_taken': 0.032198667526245117}
INFO:elastalert:Ran Lockout Rules File from 2022-02-10 08:43 UTC to 2022-02-10 08:58 UTC: 0 query hits (0 already seen), 0 matches, 0 alerts sent
INFO:elastalert:Lockout Rules File range 900
[root@ELK-Stack ~]#
```

Fig. 7. ElastAlert Docker container logs.

The rule that I have been running is the Lockout Rule File which searches Elasticsearch data containing the query "A user account was locked out.". The rule file also specifies the email's contents. Most importantly, it specifies which fields will be included in the email. Unfortunately, I could only include the message field in my email alerts. I debugged this through my docker logs command and could see that only the message field was available in the overarching _source field. This led me to the next process of attempting to add existing fields to this _source field. However I have yet to find success with this, so currently I am stuck with only the message field in my alerts.



Lockout Event Detected from User — 1 message

From: relay2@uhtasi.org — Mar 20
To: info@uhtasi.org
Reply To: info@uhtasi.org

```
Lockout Event detected.
Details of the event:
- message: A user account was locked out.

Subject:
        Security ID:                  S-1-5-18
        Account Name:                  DCON3$
        Account Domain:                  UHTASI
        Logon ID:                  0x3E7

Account That Was Locked Out:
        Security ID:                  S-1-5-21-3693063922-990322624-829951171-
1118
        Account Name:                  ginger

Additional Information:
        Caller Computer Name:          DCON3
```

Fig. 8. Example of an alert generated through our Lockout Rule File.

G. *Isolate Winlogbeat Index for Testing*

During the process of working on ElastAlert, I noticed I did not have access to some of the fields. I wanted to play with the Elasticsearch mapping to fix this, but did not want to interfere with my existing winlogbeat indices. To remedy this I sent logs from an unused Windows server owned by TASI and isolated these logs from the overall winlogbeat stream. To do this I adjusted the output of my logstash configuration file as so:

```
if [type]=="winlogbeat" {
          if [agent][hostname]=="Uku"{
          elasticsearch {
          hosts => ["localhost:9200"]
        index => "winlogbeattest-uku-
%{+yyyy.MM.dd}"

        }

        } else {

        elasticsearch {

        hosts => ["localhost:9200"]

        index => "winlogbeat-%{+yyyy.MM.dd}"

        }

        }

stdout { codec => rubydebug }

}
```

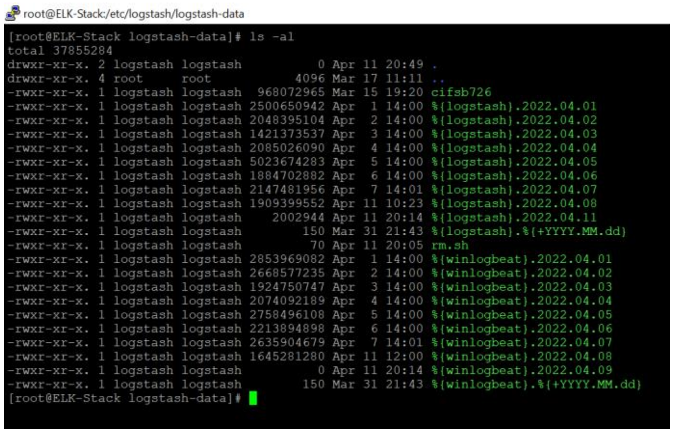
This code block sends logs from the "Uku" host to its own index separate from the normal Winlogbeat index. By creating an index template for the winlogbeattest-* pattern, I am able to alter its settings and mapping without interfering with the normal Winlogbeat index. Note that the stdout code portion refers to the logs that are outputted to the screen (standard out).

H. *Send data to external NAS (Network Attached Storage)*

To send data to the NAS, I have to set a mount on the VM. You can accomplish this through the actual mount command or you can set it permenantly through the /etc/fstab file. I decided to set it through the /etc/fstab file and added this line:

```
//10.100.10.52/Backups/ELK_Backups
/etc/logstash/logstash-data cifs
vers=1.0,uid=logstash,gid=logstash,username=******
**,password=******** 0 0
```

The line gives logstash permissions to the directory and specifies that it is a CIFS storage. It also provides the credentials to the NAS directory. Note that I replaced the actual credentials with asterisks. The files in the mounted directory on the virtual machine will automatically exist in the other folder (//10.100.10.52/Backups/ELK_Backups in this case). Also the storage is all on the 10.100.10.52 side, so you do not have to worry about managing the storage on the virtual machine in regards to the folder. Here is the shared directory with the indices:



```
root@ELK-Stack:/etc/logstash/logstash-data
[root@ELK-Stack logstash-data]# ls -al
total 37855284
drwxr-xr-x. 2 logstash logstash          0 Apr 11 20:49 .
drwxr-xr-x. 4 root     root           4096 Mar 17 11:11 ..
-rwxr-xr-x. 1 logstash logstash  968072965 Mar 15 19:20 cifsb726
-rwxr-xr-x. 1 logstash logstash 2500650942 Apr  1 14:00 %{logstash}.2022.04.01
-rwxr-xr-x. 1 logstash logstash 2048395104 Apr  2 14:00 %{logstash}.2022.04.02
-rwxr-xr-x. 1 logstash logstash 1421373537 Apr  3 14:00 %{logstash}.2022.04.03
-rwxr-xr-x. 1 logstash logstash 2085026090 Apr  4 14:00 %{logstash}.2022.04.04
-rwxr-xr-x. 1 logstash logstash 5023674283 Apr  5 14:00 %{logstash}.2022.04.05
-rwxr-xr-x. 1 logstash logstash 1884702882 Apr  6 14:00 %{logstash}.2022.04.06
-rwxr-xr-x. 1 logstash logstash 2147481956 Apr  7 14:01 %{logstash}.2022.04.07
-rwxr-xr-x. 1 logstash logstash 1909399552 Apr 11 10:23 %{logstash}.2022.04.08
-rwxr-xr-x. 1 logstash logstash    2002944 Apr 11 20:14 %{logstash}.2022.04.11
-rwxr-xr-x. 1 logstash logstash        150 Mar 31 21:43 %{logstash}.%{+YYYY.MM.dd}
-rwxr-xr-x. 1 logstash logstash         70 Apr 11 20:05 rm.sh
-rwxr-xr-x. 1 logstash logstash 2853969082 Apr  1 14:00 %{winlogbeat}.2022.04.01
-rwxr-xr-x. 1 logstash logstash 2668577235 Apr  2 14:00 %{winlogbeat}.2022.04.02
-rwxr-xr-x. 1 logstash logstash 1924750747 Apr  3 14:00 %{winlogbeat}.2022.04.03
-rwxr-xr-x. 1 logstash logstash 2074092189 Apr  4 14:00 %{winlogbeat}.2022.04.04
-rwxr-xr-x. 1 logstash logstash 2758496108 Apr  5 14:00 %{winlogbeat}.2022.04.05
-rwxr-xr-x. 1 logstash logstash 2213894898 Apr  6 14:00 %{winlogbeat}.2022.04.06
-rwxr-xr-x. 1 logstash logstash 2635904679 Apr  7 14:01 %{winlogbeat}.2022.04.07
-rwxr-xr-x. 1 logstash logstash 1645281280 Apr 11 12:00 %{winlogbeat}.2022.04.08
-rwxr-xr-x. 1 logstash logstash          0 Apr 11 20:14 %{winlogbeat}.2022.04.09
-rwxr-xr-x. 1 logstash logstash        150 Mar 31 21:43 %{winlogbeat}.%{+YYYY.MM.dd}
[root@ELK-Stack logstash-data]#
```

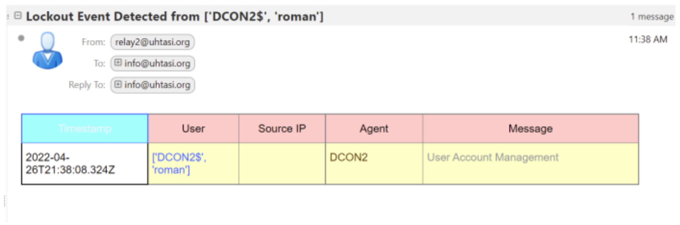
Fig. 9. The mounted /etc/logstash/logstash-data directory.

I. *Format ElastAlert alerts in HTML*

After the ElastAlert alerts were running as intended, I wanted to go a step further and format them in HTML. Previously, I could not do this because I only had access to limited fields in _source. I managed to fix this which enabled me more freedom in designing alerts, specifically HTML. I wanted to format the data in a table format. I used an online HTML table generator and played around with the formatting and color schemes. I then added the actual HTML code in the alert_text field in the ElastAlert rule file. This is possible if you set the email_format field to html. Note that in order to be interpreted by Python, the single brackets must be turned into double brackets [5]. This is the code:

```
ALERT_TEXT: |-

  <STYLE TYPE="TEXT/CSS">

  .TG  {{BORDER-COLLAPSE:COLLAPSE;BORDER-SPACING:0;}}

  .TG TD{{BORDER-COLOR:BLACK;BORDER-STYLE:SOLID;BORDER-WIDTH:1PX;FONT-FAMILY:ARIAL, SANS-SERIF;FONT-SIZE:14PX;

  OVERFLOW:HIDDEN;PADDING:10PX 5PX;WORD-BREAK:NORMAL;}}

  .TG TH{{BORDER-COLOR:BLACK;BORDER-STYLE:SOLID;BORDER-WIDTH:1PX;FONT-FAMILY:ARIAL, SANS-SERIF;FONT-SIZE:14PX;

  FONT-WEIGHT:NORMAL;OVERFLOW:HIDDEN;PADDING:10PX 5PX;WORD-BREAK:NORMAL;}}

  .TG .TG-78UE{{BACKGROUND-COLOR:#96FFFB;BORDER-COLOR:#3166FF;COLOR:#ECF4FF;TEXT-ALIGN:CENTER;VERTICAL-ALIGN:TOP}}

  .TG .TG-0S4R{{BACKGROUND-COLOR:#FFFFC7;BORDER-COLOR:INHERIT;COLOR:#643403;TEXT-ALIGN:LEFT;VERTICAL-ALIGN:TOP}}

  .TG .TG-23C7{{BACKGROUND-COLOR:#FFFFC7;BORDER-COLOR:INHERIT;COLOR:#3166FF;TEXT-ALIGN:LEFT;VERTICAL-ALIGN:TOP}}

  .TG .TG-BOLJ{{BACKGROUND-COLOR:#FFCCC9;BORDER-COLOR:INHERIT;TEXT-ALIGN:CENTER;VERTICAL-ALIGN:TOP}}

  .TG .TG-0LAX{{TEXT-ALIGN:LEFT;VERTICAL-ALIGN:TOP}}

  .TG .TG-PSP6{{BACKGROUND-COLOR:#FFFFC7;BORDER-COLOR:INHERIT;COLOR:#00D2CB;TEXT-ALIGN:LEFT;VERTICAL-ALIGN:TOP}}

  .TG .TG-6FX8{{BACKGROUND-COLOR:#FFFFC7;BORDER-COLOR:INHERIT;COLOR:#9B9B9B;TEXT-ALIGN:LEFT;VERTICAL-ALIGN:TOP}}

  </STYLE>

  <TABLE CLASS="TG" STYLE="UNDEFINED;TABLE-LAYOUT: FIXED; WIDTH: 792PX">

  <COLGROUP>

  <COL STYLE="WIDTH: 162PX">

  <COL STYLE="WIDTH: 117PX">

  <COL STYLE="WIDTH: 111PX">

  <COL STYLE="WIDTH: 125PX">

  <COL STYLE="WIDTH: 277PX">

  </COLGROUP>

  <THEAD>

    <TR>

      <TH CLASS="TG-78UE">TIMESTAMP</TH>

      <TH CLASS="TG-BOLJ">USER</TH>

      <TH CLASS="TG-BOLJ">SOURCE IP</TH>

      <TH CLASS="TG-BOLJ">AGENT</TH>

      <TH CLASS="TG-BOLJ">MESSAGE</TH>

    </TR>

  </THEAD>

  <TBODY>

    <TR>

      <TD CLASS="TG-0LAX">{0}</TD>

      <TD CLASS="TG-23C7">{1}</TD>

      <TD CLASS="TG-PSP6">{2}</TD>

      <TD CLASS="TG-0S4R">{3}</TD>

      <TD CLASS="TG-6FX8">{4}</TD>

    </TR>

  </TBODY>

  </TABLE>
```

And these are examples of HTML alerts generated for the lockout and IPS events:

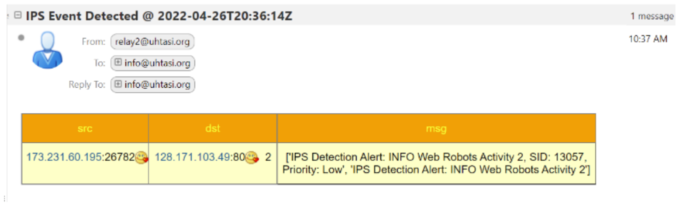


Lockout Event Detected from ['DCON2$', 'roman'] 1 message

From: relay2@uhtasi.org 11:38 AM
To: info@uhtasi.org
Reply To: info@uhtasi.org

Timestamp	User	Source IP	Agent	Message
2022-04-26T21:38:08.324Z	['DCON2$', 'roman']		DCON2	User Account Management

Fig. 10. The lockout alert. Note that the Source IP field is empty because this particular log has no source IP. Also the Message field needs to be reconfigured.



IPS Event Detected @ 2022-04-26T20:36:14Z 1 message

From: relay2@uhtasi.org 10:37 AM
To: info@uhtasi.org
Reply To: info@uhtasi.org

src	dst	msg
173.231.60.195:26782😘	128.171.103.49:80😘 2	['IPS Detection Alert: INFO Web Robots Activity 2, SID: 13057, Priority: Low', 'IPS Detection Alert: INFO Web Robots Activity 2']

Fig. 11. The IPS alert. Note that the emoji in the src and dst fields was automatically generated, replacing the 'X' character in the actual field.

CONCLUSION

I am happy to conclude that the project was successful. I managed to install and configure Elasticsearch, Logstash, and Kibana along with ElastAlert2. I am able to ingest TASI-PHIDC logs and analyze and report on them through Kibana. Overall, this implementation improves the organization's security posture. Despite the apparent success of the project, there are still improvements to be made. One improvement is the mounted directory. Though the indices are able to successfully be stored in the mount, it is troublesome to re-index them back into Elasticsearch. This will require additional research and may require spinning up a new Logstash pipeline exclusively for the mounted files. Another problem with the mounted directory is that it actually reached max capacity fairly quickly. Once the mount is at complete capacity, Logstash fails. It gives the error "Exception in flushing files" which refers to the output to the mount being at full storage and unable to accept more data. A workaround could be to develop scripts that will delete old files in the mount, but that sort of defeats the purpose of storing ELK back-ups. An option is to use Kibana's snapshot and restore feature. I initially avoided this because I thought it required payment, but it looks like it may be free.

Aside from this improvement, I would also like to discuss future plans with the project. One idea is to alert on software installations completed on the organization's workstations. This would require installing winlogbeat on these workstations, as well as adding a public network interface for the ELK server. This is because we would want to alert on installations even when the workstation is at home, meaning it is off the VPN. This would also require additional security precautions to be taken due to the network interface going from private to public.

ACKNOWLEDGMENT

I would like to acknowledge and thank TASI-PHIDC for employing me and giving me the opportunity and training needed to accomplish this project. I would also like to thank the Department Information and Computer Science at UH Manoa for providing me with a strong computer science background.

REFERENCES

[1] Owa, Kosho, and Mark Walkom. "Earthquake Data with the Elastic Stack." *Elastic Blog*, 18 Apr. 2019, https://www.elastic.co/blog/earthquake-data-with-the-elastic-stack.

[2] Quintal, Becky. "How Versatile Is the Elastic Stack? Ask Walmart, NASA, or Airbus." *Elastic Blog*, 29 July 2021, https://www.elastic.co/blog/how-versatile-is-the-elastic-stack-ask-walmart-nasa-airbus.

[3] "US Geological Survey: Tectonic Shifts, Tweets, & Tactical Data Analysis." *Elastic*, 3 July 2020, https://www.elastic.co/elasticon/conf/2015/sf/tectonic-shifts-tweets-and-tactical-data-analysis.

[4] "What Is a Vsphere Hypervisor?: Free Hypervisor." *VMware*, 6 Apr. 2022, https://www.vmware.com/products/vsphere-hypervisor.html.

[5] "7.1.string – Common string operations" *Python*, 27 Apr. 2022, https://docs.python.org/2/library/string.html#format-string-syntax.