Using Elastic Cloud to Identify, Analyze, and Generate Alerts for Anomalous Network Traffic

Anomaly Detection Analysis

***Abstract*— Analyzing network traffic is an integral part of ensuring the security of a network. However, traditional signature-based detection does not protect against novel attacks. Through the use of an Elastic Stack deployment in Elastic Cloud (Software as a Service) composed of Elasticsearch, Logstash, and Kibana, we can use their machine learning nodes to detect anomalous traffic as it traverses a network. Additionally, features in the Elastic stack deployment allow us to generate alerts and supply recommended follow-up actions based on the severity and type of alert. However, Elastic is not a perfect one size fits all tool. We initially found that Elastic would only succeed at machine learning anomaly detection if the team setting up the detection had extensive knowledge of what they were looking for. After extensive research and trial/error, we did find that some success could be achieved with methods that require fewer resources and background knowledge (such as queries). Therefore, we determined that Elastic anomaly detection is a decent middle ground tool to bridge the gap between manual searches and queries, however, it needs certain data characteristics to meet its full potential.**

***Index Terms*—Anomaly detection, machine learning, Elastic, network traffic, cyberattacks, security, SIEM, Elastic stack, Elastic stack deployment, Mordor Project**

I. REQUIREMENTS

The project goal was to create an anomaly detection system using Elastic Cloud (Software as a Service) as a proof of concept for the Cybersecurity and Infrastructure Security Agency (CISA), within the Department of Homeland Security (DHS). This required us to either find public datasets or create our own focused on malicious network activity and use Elastic's anomaly detection algorithm to detect the anomalies. We needed to utilize the Elastic stack deployment, specifically Kibana, to analyze and visualize the results of the anomaly detection. Finally, we would create a functional alert system that would alert the SOCs of any anomalies. The underlying philosophy of the project was to identify pitfalls in the process as a whole to help CISA have more insight into their current work utilizing Elastic.

II. CONTEXT

CISA supports federal agencies by providing asset management, identity and access management, network security management, data protection management, and operational dashboards to host information regarding these capabilities. We were tasked with finding or generating data and creating a system that can identify malicious activity, which will help to spot unauthorized activity on systems belonging to federal agencies. The proposed system is hosted within the Elastic Cloud and will leverage its machine learning, anomaly detection, and visualization capabilities.

III. STAKEHOLDERS

This project is facilitated through the George Mason University Cyber Security Engineering degree. The sponsor for this project is CISA, which provides services for monitoring and mitigating vulnerabilities across the government and other organizations.

IV. CONCEPT OF OPERATIONS

The Concept of Operations for this project is based on three distinct sections: the data sources, the data analysis/ processing, and the stakeholders. The data sources in the production environment will consist of workstations, servers, and any network devices that the stakeholders want to monitor. These data sources will then transfer or have their logs, metrics, and other attributes moved to designated Logstash installations which will process, filter, and ingest all data into the appropriate Elastic stack deployment in Elastic Cloud. From there, they will be inside the data analysis and processing section, which will be based on Elastic Cloud deployments unique to each relevant department. Elastic, Elasticsearch, and Kibana will facilitate the storage, processing, analysis, anomaly detection, and data visualizations needed. This will be operated by an analyst group, where they will be able to use local Logstash installations to upload test or validation datasets when necessary. They will also be responsible for the maintenance and monitoring of Kibana and its dashboards. These dashboards will provide alerts to the analyst. Agencies will also be able to access their Kibana deployments directly as part of the stakeholder group. CISA will aggregate multiple agencies' and departments' dashboards and results to maintain a government-wide dashboard that can monitor large-scale patterns. CISA will then be able to direct the analyst groups to change, modify or set new goals for the Elastic deployments.

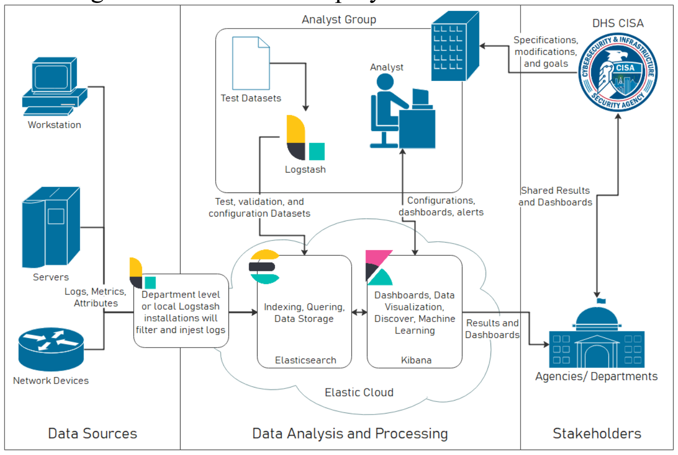

Analyst Group
Test Datasets
Analyst
Logstash
DHS CISA
Specifications, modifications, and goals
Workstation
Servers
Network Devices
Logs, Metrics, Attributes
Department level or local Logstash installations will filter and injest logs
Test, validation, and configuration Datasets
Configurations, dashboards, alerts
Shared Results and Dashboards
Indexing, Quering, Data Storage
Elasticsearch
Dashboards, Data Visualization, Discover, Machine Learning
Kibana
Results and Dashboards
Agencies/ Departments
Elastic Cloud
Data Sources
Data Analysis and Processing
Stakeholders

Fig. 1. (Product Concept of Operations) - Final Concept of Operations ~see Appendix A for full-sized

V. Design

Our design relies on Elastic Cloud for all of our infrastructure and data management systems. Using Elastic accounts provided by CISA, we have built a functional system utilizing Elasticsearch, Kibana, Logstash, Elastic anomaly detection jobs, data visualizer, and more. The core of our project is to use machine learning algorithms to detect anomalies in network traffic datasets for multiple MITRE ATT&CK® tactics categories. This experimentation will allow us to make recommendations as to when and where machine learning should and should not be used for anomaly detection. This is enabled by using Elastic's anomaly detection jobs on the datasets we have selected, where Elastic's included machine learning nodes host the algorithm. Our architecture [Fig. 2] consists of two groups: the local team and the Elastic Cloud. The members of the local team are the only ones allowed to access the Elastic Cloud deployment as they will be the only ones issued login credentials; all other users who would attempt to login would not be able to as they would not hold any credentials. These team members would be able to utilize various aspects of the Elastic deployment based on their account privileges. Any team member with a correctly configured local copy of Logstash will be able to ingest datasets into our Elastic deployment. In a production environment, Logstash would be placed on systems where we want to monitor the logs of said system, and it would run continuously. In our case, Logstash is used to ingest the datasets from the Mordor Project [1]. Within Elastic Cloud, we primarily use two services, Elasticsearch and Kibana. Elasticsearch is used to ingest and manage our datasets, while Kibana is used for anomaly detection jobs, creating data visualizations, and creating dashboards. Kibana would also be used in a production environment to monitor these dashboards for anomalies and alerts.

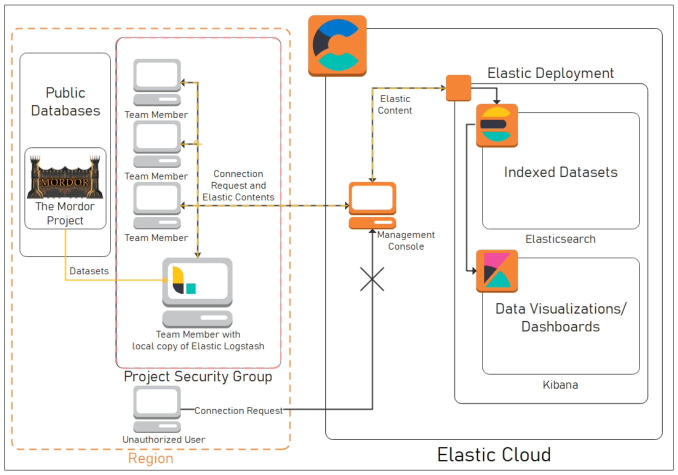


Fig. 2. (Product Architecture) - Our final products architecture model ~see Appendix A for full-sized

Our data model [Fig. 3] is relatively simple and is mostly contained within Elastic Cloud. Elastic Logstash is used locally on a team member's computer to ingest our selected datasets into Elasticsearch within our Elastic Cloud deployment. From there we can utilize our machine learning jobs and index the datasets so that they can be used by Kibana and our dashboards.

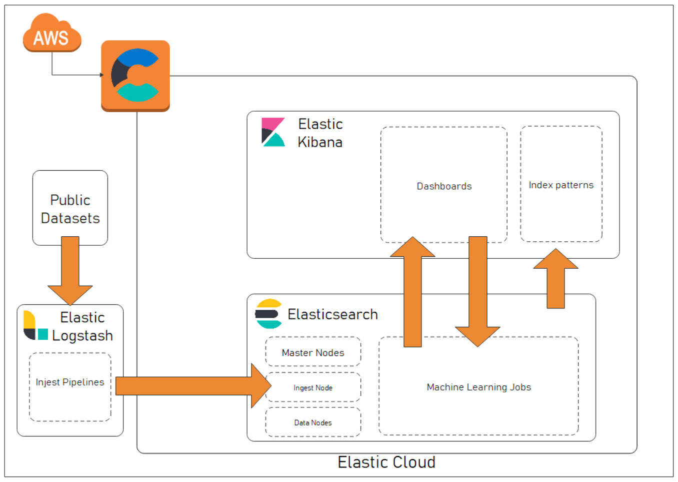


Fig. 3. (Product Data Model) - Our final products data model ~see Appendix A for full-sized

VI. Implementation

For our implementation, we had to take a ground-up approach to the project. While we had surrounding knowledge about the project such as log analysis, machine learning, AWS, and network traffic analysis, we had no experience with Elastic. After finding a direction we broke into two groups: one would be focused on learning the internals of Elastic using free training offered by Elastic, and the other would work on finding and/or generating datasets. This initial phase was not successful as the Elastic team struggled with finding relevant or useful training that wasn't just on making a barebones Elastic deployment. The dataset team also had no success with finding suitable datasets, nor with generating their own datasets. The dataset team initially tried to find traffic log datasets and was unsuccessful. The team then experimented with generating traffic and datasets also to no success. Having found little success, we were then pointed to the Mordor Project by CISA. At this point, we had already created individual Elastic cloud deployments which allowed us to look at smaller datasets in the data visualizer tool, up to 100 MB. We later learned that this was actually an artificial limit and could be changed under the advanced settings for Kibana. We then used the data visualizer tool to start to examine, ingest, and run 6 out of a total of 89 smaller datasets from the Mordor Project through our Elastic anomaly detection jobs. These datasets covered 5 out of the 8 tactic categories of the MITRE ATT&CK® framework that the Windows datasets on the Mordor Project cover. Specifically, these datasets cover Execution, Persistence, Privilege Escalation, Defense Evasion, and Lateral Movement. The datasets we used are as follows:

- Covenant Remote WMI Wbemcomn DLL Hijacking (https://mordordatasets.com/notebooks/small/windows/02_execution/SDWIN-201009173318.html)

- Remote Scheduled Task Modification (https://mordordatasets.com/notebooks/small/windows/02_execution/SDWIN-201219075059.html)
- Empire Invoke DCOM ShellWindows (https://mordordatasets.com/notebooks/small/windows/02_execution/SDWIN-190518211052.html)
- Remote Scheduled Task Modification (https://mordordatasets.com/notebooks/small/windows/03_persistence/SDWIN-201219075059.html)
- Remote Scheduled Task Creation (https://mordordatasets.com/notebooks/small/windows/03_persistence/SDWIN-201219070027.html)
- Bitsadmin Download Malicious File (https://mordordatasets.com/notebooks/small/windows/03_persistence/SDWIN-201023023651.html).

During this phase, each member of our team decided to take one of the datasets to perform anomaly detection analysis on. However, while we tried the different features of the anomaly detection jobs, such as single metric, multi-metric, and advanced anomaly detection, no one was able to generate a single anomaly from the datasets. Techniques such as running anomaly detection on filtered data were also used on these datasets. Still, the team could not find any success. After deliberation, we suspected that when dealing with smaller datasets, machine learning cannot be as successful at determining anomalistic data. Some of the largest logs of these datasets would only have up to fifteen minutes' worth of data. This would make it almost impossible for the anomaly detection to work because the bucket span, the time in between the comparisons, would be unreasonably short, and thus there would be no anomalous data detected. Additionally, the short time span meant that the anomaly detection job could not determine what "normal" data looked like. Because these logs were too short, we decided to move on from the smaller datasets and look for another dataset that would be larger.

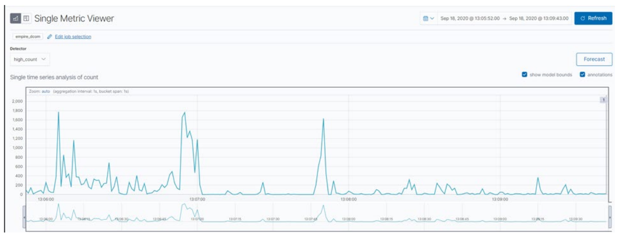


Fig. 4. Example of no anomalies using single metric viewer on smaller datasets ~see Appendix A for full-sized

After some time, a member of our team found some larger format datasets within Mordor, and one, in particular, was a dataset of APT 29 that had two datasets for two separate days of simulated attacks. The team identified this dataset as a strong candidate with which to develop our final product. After consultation with CISA, this dataset was agreed upon as the dataset we would build our final product around.

The following implementation is adaptable to many use-cases that follow datasets with similar data and features to analyze.

A. *APT 29 Dataset*

The APT 29 dataset was very large, and we were unable to upload the dataset like previous smaller datasets. Initially, a member of our team broke up the APT 29 Day 1 dataset into four partitions. From the separate partitions, we were able to upload these portions of the dataset directly within our deployment to begin exploring the dataset. This enabled the team to separate for a period of time so that everyone would be able to look at the dataset in unique and novel ways; the team explored the PCAP files to search for specific malicious activity, the sigma rules to identify indicators of compromise, and did further conventional exploration using Elastic's tools. The problem was that the partitioned datasets were small and would not allow for a full analysis, so the team had to find a way to ingest the data properly. With consultation from CISA, Elastic Logstash was selected as the tool we would use.

To provide context related to our selected dataset, APT 29 is a threat group linked to the Russian Federation that is known to target political organizations and think tanks. Their most widely known attack was against the Democratic National Committee (DNC) during the 2016 US Presidential Elections [5]. CISA has an interest in understanding this attack group and the payloads they use, which makes this dataset ideal for us to work on. The Mordor Dataset on APT 29 was created to emulate adversarial attacks and techniques for 2019 ATT&CK Evaluations of multiple organizations in Nashua, NH (Day 1) and Scranton, PA (Day 2). APT 29 payloads include CosmicDuke, MiniDuke, SeaDuke/SeaDaddy, CozyDuke/CozyCar, and Hammertoss [4]. Duke is a set of malware that has been used in attacks since 2008 beginning in the Chechnya Informational Center Incident and has affected many high-level organizations. APT 29 is thought to have originated these tools [3].

The APT 29 Dataset was developed after analyzing the threat group's historical attack pattern. Using the MITRE ATT&CK® Framework, the preattack on these datasets consisted of [4]:

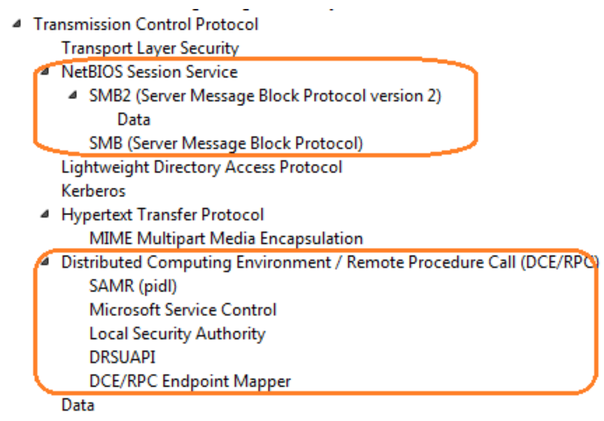
1. A spear-phishing campaign (Reconnaissance)
2. Determining specific file types from hosts (Reconnaissance)
3. Developing capability to exploit file-types using payloads and toolkits (Resource Development)

The attack consists of the following [4]:

1. Login to victim system and launch PUPY server (Initial Access)
2. Compress user login information files and export through PUPY (Credential Access)
3. Allows for reverse HTTPS from a malicious web server to drop files into a victim machine (Execution)
4. Hide and prevent detection (Defense Evasion)

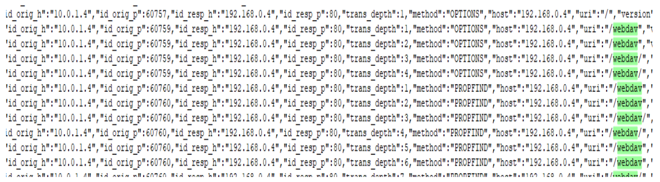
5. Gain access to credentials by stealing private keys (Credential Access)
6. Copy payload into a client (Lateral Movement)
7. Execute SeaDuke, a backdoor created by APT29 to continuously have access to the system (Persistence & Execution)

From this, we can deduce that the anomalistic data would be in features that consist of 1) attempts to log in, 2) connection with their web server, 3) processes to copy files to/from their server, and 4) malicious services that can relate to the SeaDuke payload.



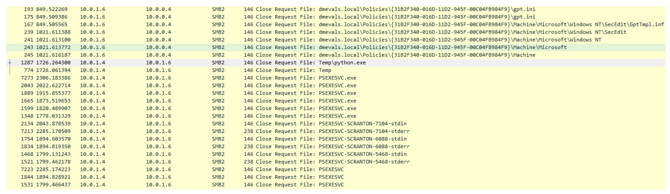
```
◢ Transmission Control Protocol
    Transport Layer Security
  ◢ NetBIOS Session Service
    ◢ SMB2 (Server Message Block Protocol version 2)
        Data
      SMB (Server Message Block Protocol)
    Lightweight Directory Access Protocol
    Kerberos
  ◢ Hypertext Transfer Protocol
      MIME Multipart Media Encapsulation
  ◢ Distributed Computing Environment / Remote Procedure Call (DCE/RPC)
      SAMR (pidl)
      Microsoft Service Control
      Local Security Authority
      DRSUAPI
      DCE/RPC Endpoint Mapper
    Data
```

Fig. 5. (PCAP Protocol Hierarchy) - Both PCAP files contain SMB (used for file transferring) and DCE/RPC traffic ~see Appendix A for full-sized



```
id_orig_h":"10.0.1.4","id_orig_p":60757,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":1,"method":"OPTIONS","host":"192.168.0.4","uri":"/","version'
'id_orig_h":"10.0.1.4","id_orig_p":60759,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":1,"method":"OPTIONS","host":"192.168.0.4","uri":"/webdav","
'id_orig_h":"10.0.1.4","id_orig_p":60759,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":2,"method":"OPTIONS","host":"192.168.0.4","uri":"/webdav","
'id_orig_h":"10.0.1.4","id_orig_p":60759,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":3,"method":"OPTIONS","host":"192.168.0.4","uri":"/webdav/",'
'id_orig_h":"10.0.1.4","id_orig_p":60759,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":4,"method":"OPTIONS","host":"192.168.0.4","uri":"/webdav/",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":1,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":2,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":3,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav/",
id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":4,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav/",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":5,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":6,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav",'
```

Fig. 6. (Dataset) - 122 connections to Webdav, which is a protocol that allows files in file servers to be modified and managed ~see Appendix A for full-sized



```
 193 849.522269   10.0.1.6   10.0.0.4   SMB2   146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\gpt.ini
 175 849.509386   10.0.1.6   10.0.0.4   SMB2   146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\gpt.ini
 167 849.505565   10.0.1.6   10.0.0.4   SMB2   146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft\Windows NT\SecEdit\GptTmpl.inf
 239 1021.611388  10.0.1.6   10.0.0.4   SMB2   146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft\Windows NT\SecEdit
 241 1021.613100  10.0.1.6   10.0.0.4   SMB2   146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft\Windows NT
 243 1021.613772  10.0.1.6   10.0.0.4   SMB2   146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft
 245 1021.616187  10.0.1.6   10.0.0.4   SMB2   146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine
1287 1726.264300  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: Temp\python.exe
 774 1726.061394  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: Temp
7273 2306.183386  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC.exe
2043 2022.622714  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC.exe
1889 1915.855377  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC.exe
1665 1873.519653  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC.exe
1599 1820.489907  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC.exe
1348 1778.031329  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC.exe
2134 2043.878539  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC-SCRANTON-7104-stdin
7213 2285.170509  10.0.1.4   10.0.1.6   SMB2   238 Close Request File: PSEXESVC-SCRANTON-7104-stderr
1754 1894.603570  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC-SCRANTON-6088-stdin
1834 1894.819350  10.0.1.4   10.0.1.6   SMB2   238 Close Request File: PSEXESVC-SCRANTON-6088-stderr
1468 1799.131243  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC-SCRANTON-5468-stdin
1521 1799.462178  10.0.1.4   10.0.1.6   SMB2   238 Close Request File: PSEXESVC-SCRANTON-5468-stderr
7223 2285.174223  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC
1844 1894.828921  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC
1531 1799.466437  10.0.1.4   10.0.1.6   SMB2   146 Close Request File: PSEXESVC
```

Fig. 7. (Dataset) - PsExeSVC.exe and python.exe references ~see Appendix A for full-sized

There are a couple of interesting things going on here. First, the GptTmpl.inf file can be copied and modified from the SMB file transfer protocol. This file controls group policies and can be exploited to overwrite important registry keys and inject malicious services [6]. The next thing is that there were write/close requests for \temp\python.exe. Why would it be modified remotely? Why is SMB getting its attribute information? And is there a correlation with PsExeSVC.exe?

PsExeSVC provides the ability to redirect outputs and start processes [7]. After transferring the remote execution program, the attacker then starts using the Service Control Manager (SCM) to start services (background processes, analogous to daemons) remotely. They create and start a service binary (PsExeSVC.exe) using this capability. After doing more research, we found that the \temp\python.exe file was known to be modified to become a malicious payload (most likely into a backdoor) and executed as a part of the SeaDuke software suite.

Exploiting the capabilities of PsExeSVC.exe in this way is a well-known lateral movement technique used to access and control other machines on a network after initial access/persistence. This helped us understand what was going on within the dataset and what features we need to determine to move forward.

B. *Machine Learning*

We also looked at some features in log entries that were suspicious:

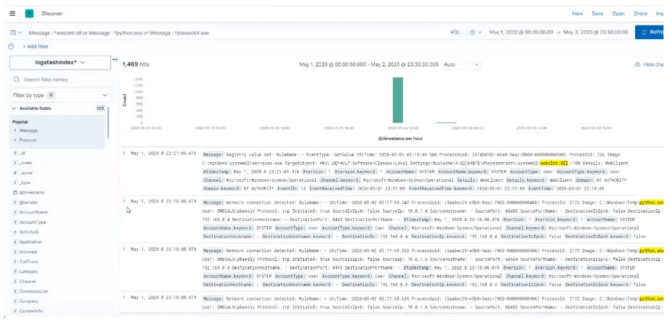
Network connection detected: RuleName: - UtcTime: 2020-05-02 03:16:54.700 ProcessGuid: {5aa8ec29-e5b8-5eac-7903-000000000400} ProcessId: 2172 Image: C:\Windows\Temp\python.exe User: DMEVALS\pbeesly Protocol: tcp Initiated: true SourceIsIpv6: false SourceIp: 10.0.1.6 SourceHostname: - SourcePort: XXXX SourcePortName: - DestinationIsIpv6: false DestinationIp: 192.168.0.4 DestinationHostname: - DestinationPort: 8443 DestinationPortName:

Registry value set: RuleName: - EventType: SetValue UtcTime: 2020-05-02 03:20:42.383 ProcessGuid: {47ab858c-e6ad-5eac-0b00-000000000500} ProcessId: 736 Image: C:\windows\system32\services.exe TargetObject: HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\4\52C64B7E\@%systemroot%\system32\webclnt.dll,-104 Details: WebDav Client Redirector Driver

From this information, we decided to look at some data points to determine malicious data:

Image: C:\Windows\Temp\python.exe
Image:C:\Program Files\SysinternalsSuite\PsExec64.exe
User: DMEVALS\pbeesly
Details: WebDav Client Redirector Driver
TargetObject: HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\4\52C64B7E\@%systemroot%\system32\webclnt.dl
ImageLoaded C:\Windows\System32\davclnt.dll

We used the corresponding features (Image, TargetObject, Details, User) to determine anomalistic references separately.


1,469 hits

May 1, 2020 @ 00:00:00.000 - May 2, 2020 @ 23:30:00.000

```
May 1, 2020 @ 23:21:09.410  Message: Registry value set: RuleName: - EventType: SetValue UtcTime: 2020-05-02 03:19:49.500 ProcessGuid: {47ab858c-e6ad-5eac-0b00-000000000500} ProcessId: 736 Image: C:\windows\system32\services.exe TargetObject: HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\4\52C64B7E\@%systemroot%\system32\webclnt.dll,-100 Details: WebClient @timestamp: May 1, 2020 @ 23:21:09.410 @version: 1 @version.keyword: 1 AccountName: SYSTEM AccountName.keyword: SYSTEM AccountType: User AccountType.keyword: User Channel: Microsoft-Windows-Sysmon/Operational Channel.keyword: Microsoft-Windows-Sysmon/Operational Details: WebClient Details.keyword: WebClient Domain: NT AUTHORITY Domain.keyword: NT AUTHORITY EventID: 13 EventReceivedTime: 2020-05-01 23:21:09 EventReceivedTime.keyword: 2020-05-01 23:21:09 EventTime: 2020-05-01 23:19:49
May 1, 2020 @ 23:18:00.874  Message: Network connection detected: RuleName: - UtcTime: 2020-05-02 03:17:59.342 ProcessGuid: {5aa0ec29-e5b8-5eac-7903-000000000400} ProcessId: 2172 Image: C:\Windows\Temp\python.exe User: DMEVALS\pbeesly Protocol: tcp Initiated: true SourceIsIpv6: false SourceIp: 10.0.1.6 SourceHostname: - SourcePort: 60403 SourcePortName: - DestinationIsIpv6: false DestinationIp: 192.168.0.4 DestinationHostname: - DestinationPort: 8443 DestinationPortName: - @timestamp: May 1, 2020 @ 23:18:00.874 @version: 1 @version.keyword: 1 AccountName: SYSTEM AccountName.keyword: SYSTEM AccountType: User AccountType.keyword: User Channel: Microsoft-Windows-Sysmon/Operational Channel.keyword: Microsoft-Windows-Sysmon/Operational DestinationHostname: - DestinationHostname.keyword: - DestinationIp: 192.168.0.4 DestinationIp.keyword: 192.168.0.4 DestinationIsIpv6: false DestinationIsIpv6.keyword: false
May 1, 2020 @ 23:18:00.874  Message: Network connection detected: RuleName: - UtcTime: 2020-05-02 03:17:59.352 ProcessGuid: {5aa0ec29-e5b8-5eac-7903-000000000400} ProcessId: 2172 Image: C:\Windows\Temp\python.exe User: DMEVALS\pbeesly Protocol: tcp Initiated: true SourceIsIpv6: false SourceIp: 10.0.1.6 SourceHostname: - SourcePort: 60404 SourcePortName: - DestinationIsIpv6: false DestinationIp: 192.168.0.4 DestinationHostname: - DestinationPort: 8443 DestinationPortName: - @timestamp: May 1, 2020 @ 23:18:00.874 @version: 1 @version.keyword: 1 AccountName: SYSTEM AccountName.keyword: SYSTEM AccountType: User AccountType.keyword: User Channel: Microsoft-Windows-Sysmon/Operational Channel.keyword: Microsoft-Windows-Sysmon/Operational DestinationHostname: - DestinationHostname.keyword: - DestinationIp: 192.168.0.4 DestinationIp.keyword: 192.168.0.4 DestinationIsIpv6: false DestinationIsIpv6.keyword: false
May 1, 2020 @ 23:18:00.874  Message: Network connection detected: RuleName: - UtcTime: 2020-05-02 03:17:58.639 ProcessGuid: {5aa0ec29-e5b8-5eac-7903-000000000400} ProcessId: 2172 Image: C:\Windows\Temp\python.exe User: DMEVALS\pbeesly Protocol: tcp Initiated: true SourceIsIpv6: false SourceIp: 10.0.1.6 SourceHostname: - SourcePort: 60402 SourcePortName: - DestinationIsIpv6: false DestinationIp:
```

Fig. 8. *(Elastic Discover) -Picture of querying dataset for "Message : *webclnt.dll or Message : *python.exe or Message : *psexec64.exe" ~see Appendix A for full-sized*

We found these fields by writing a query (using Kibana Query Language (KQL)) in Kibana's Discover module, which was instrumental in helping us find anomalies.

Through trial and error with the dataset, we determined that anomalistic data was best identified from using the Categorization, Population, and Advanced machine learning jobs. To create a machine learning job, we followed these steps:

1) Go to Elastic/Machine Learning/Anomaly Detection
2) Click "Create Job"
3) Select dataset
4) Click which type (Categorization, Population, Advanced)
5) Use full data
6) Depending on the job type:
    a) For Adv: Click Advanced/Detectors, use function: count, and use fields identified to be suspicious (CountBy and PartitionField TargetObject/Image) with influencers (TargetObject and Image)
    b) For Pop/Cat: Pick fields that were identified to be suspicious(Image/TargetObject/etc)
7) Bucket Span: 15s
8) Click "Next" all the way to the end.

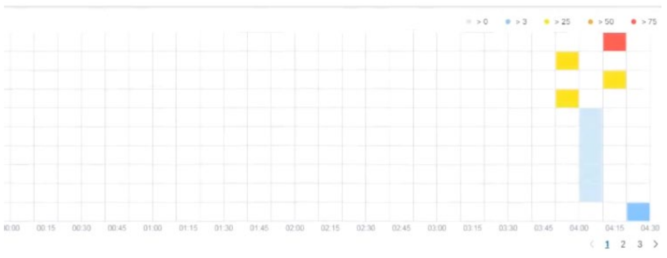
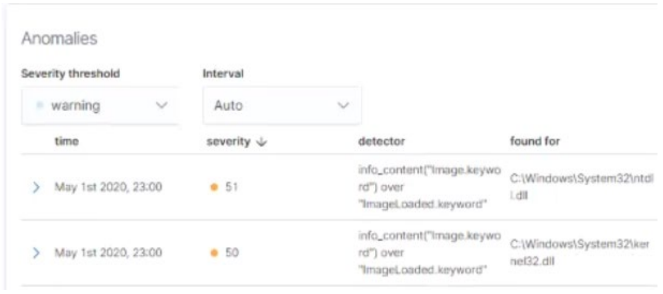


Fig. 9. *(Elastic Categorization) - Example of anomalies found for "Image" for specific services that have been run. ~see Appendix A for full-sized*


Anomalies

Severity threshold: warning
Interval: Auto

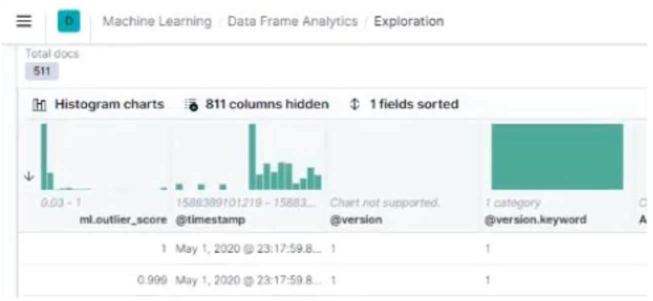
time	severity	detector	found for
May 1st 2020, 23:00	51	info_content("Image.keyword") over "ImageLoaded.keyword"	C:\Windows\System32\ntdll.dll
May 1st 2020, 23:00	50	info_content("Image.keyword") over "ImageLoaded.keyword"	C:\Windows\System32\kernel32.dll

Fig. 10. *(Elastic Advanced ML) - Example of anomalies found for count by "Image.keyword" with influencer"ImageLoaded" ~see Appendix A for full-sized*

These steps can be replicated when trying to detect anomalies within other datasets. If using the same features, the fields would not change. If the features are different, simply replace those fields. Machine Learning jobs can also be manually edited and copied using their JSON configuration. *See Appendix B Fig. 1.*

We also thought that determining the probability of critical outliers would be important, as it would help understand whether certain services may be anomalistic. To do this, we used the Data Frame Analytics machine learning job. To create:

1) Go to Elastic/Machine Learning/Data Frame Analytics
2) Create Job
3) Select "Outlier Detection"
4) Query "feature":"suspicious service" (Ex: Image: C:\\Windows\Temp\python.exe)
5) Click "continue" twice
6) Create Job ID
7) Click "Create"



ml.outlier_score	@timestamp	@version	@version.keyword
1	May 1, 2020 @ 23:17:59.8...	1	1
0.999	May 1, 2020 @ 23:17:59.8...	1	1

Fig. 11. *(Elastic Data Frame Analytics) -Picture of querying dataset Image.Keyword:*python.exe to understand whether instances of the program may be an outlier, and thus anomalistic. Criticality and timestamps on the left two columns. ~see Appendix A for full-sized*

With the Data Frame Analytics jobs, we were able to determine any outliers for specific features and validate machine learning job results. For example, after running an Anomaly Detection job and finding "psexesvc" as an anomalistic data point, we would run a job that uses a KQL query for "psexesvc" and determines outliers from the resulting dataset. This would help us cross-reference and determine if anomalistic data is truly valid. Of course, this method does have its weaknesses, such as only being able to confirm anomalistic data on a per-feature basis. Due to this, the process requires extensive manual review. This method could be refined more if we were able to fully query the dataset and create custom jobs for this, but due to time constraints, this was the method that was chosen.

These steps can be replicated for datasets that represent systems that have been attacked using the same methodology. This may not be relevant for datasets that are not representative of the APT 29 group.

You will be able to sort/filter out per feature of the dataset (Image, TargetObject, EventID) as well as a machine learning outlier score. It is a manual process to determine which columns should be shown in the final job.

C. *Logstash*

During this time a team member worked on using Logstash to create a stable, reproducible way to ingest large files to make it as close to production deployment as possible. After being shown the configuration that CISA was planning on basing their deployment on and creating an equivalent configuration, Logstash would not work. After two more weeks of work, a final, reproducible, configuration was found through extensive trial and error along with the use of the Logstash plugin repository [2]. The final configuration just needs two files, logstash.conf, and logstash.yml. A screenshot of both can be found in Appendix B under Fig. 2 and 3, respectively.

The most important aspect of these configurations is only applicable to our current use case. That being that Logstash is intended by default to ingest an active log file. This means that Logstash, unless told otherwise, will not shut down after ingesting a file; instead, it will ingest everything it has been pointed towards, whether that be a folder or an individual file, delete everything it has finished ingesting, and wait for more log files to appear. To stop this behavior, in the logstash.conf input section we have activated the 'mode => read' and the 'exit_after_read' plugins. This allows the team to just ingest individual datasets one at a time. This is not something that a final production version would utilize, but it is an essential feature when trying to build and develop said system from the ground-up. Some other notable roadblocks that we ran into was that Logstash treats a backslash character in the config files as an exit command, Logstash will not look at a file twice even when the sincedb_path is set to NUL, and that you have to use the Elasticsearch endpoint, something that can only be accessed on the overview screen of an Elastic Deployment.

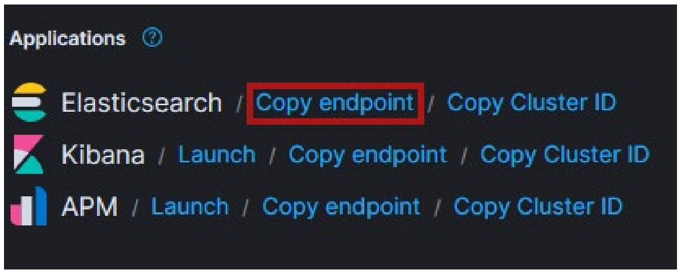

Applications
Elasticsearch / Copy endpoint / Copy Cluster ID
Kibana / Launch / Copy endpoint / Copy Cluster ID
APM / Launch / Copy endpoint / Copy Cluster ID

Fig. 12. (Elasticsearch endpoint) - The only location to get the correct endpoint for Logstash configuration.

After creating a step-by-step guide about creating a working Logstash model and ensuring that the team had working local Logstash deployments, the team was able to ingest both days of the APT 29 dataset. Once data was ingested into Elastic, the team utilized the research they all had conducted, along with the anomaly detection service built-in Elastic, to start. Using the categorization feature, the team created jobs to scan for anomalies within the logs, then displayed them through a Kibana dashboard.

D. *Kibana*

Within the Elastic Cloud ELK Stack, K - Kibana is a data visualization tool for creating dashboards for Elasticsearch. For this project, the visualizations and dashboard's audience is of an agency SOC analyst tasked with identifying anomalies. This tool offers visualization capabilities where users can create bar graphs, scatter plots, pie charts, and many more possibilities from large volumes of data. An additional function of Kibana is the ability to explore the data through Discover. Discover allows you to quickly search and filter your data, gain an understanding of the structure of the fields, and then transition into visualization.

When we split into our small teams, the Elastic team focused initially on free Elastic training that was offered on their website. Within these training sessions, they focused on various tools that could be utilized within Elastic. This included setting up an initial Elastic deployment and installing some sample data. This sample data additionally had pre-implemented visualizations and a dashboard, which was all to be used while going through the fundamental training.

After we completed various Elastic training lessons, we moved to begin ingesting datasets from the Mordor Project, which have pre-recorded security events that were generated in a controlled environment. We began ingesting small datasets from the Mordor Project and explored the dataset through the Discover tool to gain a better insight into the types of fields. After gaining an understanding of the datasets, we then transitioned to the visualization section of Kibana. From the fields, we were able to discover information that would be useful for a user to see visualized. The first dataset we explored had malicious activity in the form of bytes of data being transferred out of the network. A bar graph showcasing the various amounts of bytes being transferred and their occurrences was the first visualization made. This visualization was created on deployment version v7.10. The process for creating visualizations consisted of choosing the index and then choosing from various options within the two sections - Metrics and Buckets. For this first visualization focusing on bytes transferred out, the Metric was set to 'Count' and the field of 'Bytes Transferred Out' was chosen for Buckets. This then provided an output for this particular field with a bar graph.

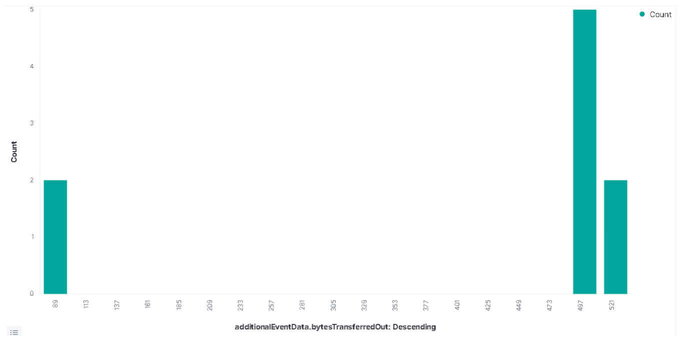


Fig. 13. (First Visualization) - Bar graph representing the number of bytes transferred out and the count ~see Appendix A for full-sized

Using the partitioned datasets that the team developed before the permanent pipeline was established, some initial test visualizations were created. These initial visualizations were examined by the team along with CISA to help guide further research and refinement to be used when we had ingested the full dataset. The successful implementation of Logstash soon after allowed for us as a team to ingest the entire dataset into our deployments with Logstash and move forward with the work being done in Kibana.

Now we have been successful in ingesting both days of data into our deployments. This allowed us to utilize the Discover module to understand the different fields within the dataset. After looking through the various fields, we began creating visualizations for each index. Some example visualizations were: Source IP, Destination IP, Host Names, etc. After creating these visualizations, we then transitioned to the dashboard section of Kibana where we choose which visualizations are to be included.

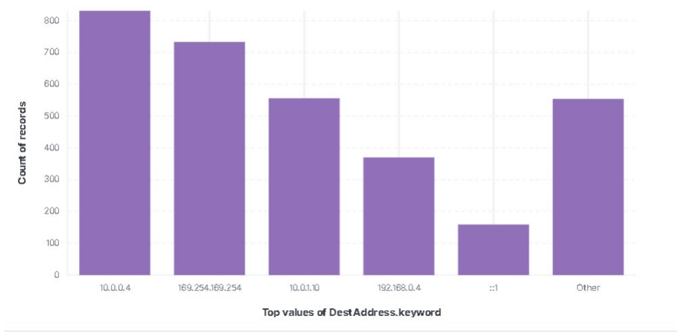


Fig. 14. (APT 29 Day 1Visualization) - Bar graph representing the top Destination IP addresses within the Day 1 dataset ~see Appendix A for full-sized

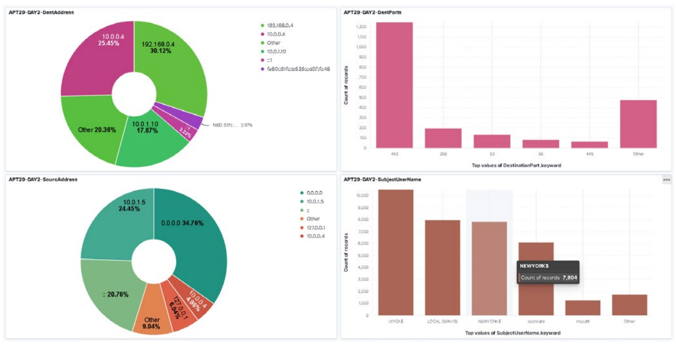


Fig. 15. (APT 29 Day 2 Dashboard) - Dashboard of 4 visualizations which consist of: Destination IP Addresses, Destination Ports, Source Addresses, and Subject Username ~see Appendix A for full-sized

E. Alert System

Over the course of the project, our goals and focus changed various times through discussions within our team and CISA. One of these changes was to delay the creation of an alerts system. Given the time constraints, we decided to focus more on the other aspects and deliverables of the project for two main reasons. The first being that we believed we needed more time to work on making our recommendation about the feasibility of machine learning anomaly detection, along with the creation of our Kibana dashboard. We felt that if we were to divide our time between these we would deliver substandard versions of both. The second reason being we felt that since we did not have a data stream source the creation of an alert system would have been necessitated to match one for one to the anomalies we detected. This artificial creation would not benefit our sponsors since it would have been done in a completely different context as they would develop theirs.

VII. Verification and Validation

A. Verification

The team verified the effectiveness of the anomaly detection and analysis using the APT 29 datasets. First, each team member created their own Logstash ingest pipeline to get the large dataset files into our respective Elastic deployments. The team then used Kibana's Discover module to ensure the integrity of the datasets and that the datasets contained the malicious activity found in the PCAP files. This was to ensure that the dataset was accurate, and contained all the information that we need. Then, anomaly detection jobs were created using Elastic's machine learning module, with job types including population, categorization, and advanced. The jobs created for our final product were ensured to follow the proper dataset indices and features.

B. Validation

The team validated the effectiveness of the anomaly detection and analysis using the APT 29 datasets. Because of the team's data exploration with Discover, we were able to determine which fields in the data set contained information that would be useful in anomaly detection jobs (E.G. TargetObject, Image, ImageLoaded). We then created anomaly detection jobs and data frame analytics using these fields as categories, identifiers, and influencers, as well as for filters. Initial anomalistic results were verified through cross-referencing Data Frame Analytic jobs with anomalistic data identified through our anomaly detection jobs. Outliers were compared with anomalies to get a better grasp of the data. A more in-depth review of the results was done manually to identify false positives. We hoped to use the previously identified sigma rules to generate KQL queries, but the feature set used for the sigma rules differed from the feature set used in the APT 29 dataset. After our manual review, we found that some results were indeed false positives, such as identifying an instance of the Microsoft Teams installer as anomalistic. Even so, the results did include many instances of data that were malicious programs or launched malicious services remotely, such as svchost.exe. In this sort of program, some false positives and false negatives are to be expected, but we were pleasantly surprised to see that proper instances of anomalistic data were discovered. This information was determined by research that was done beforehand on the APT 29 group. In the production environment, the dashboard will be the final validation method, used in conjunction with alerts the dashboard will allow those without first-hand research to see the context and validate the results.

VIII. Recommendations and Future Work

A. Recommendations

Based on our work on both the small datasets and on the APT 29 dataset, we cannot recommend machine learning anomaly detection as a sole data-analysis product. Instead, machine learning should be used as an additional resource on the tool belt of a security team that wants more verification and validation for their manual work. This is due to the large amount of investigation and analysis required for anomaly detection to work. However, our perspective is limited to the Mordor Project datasets analyzed, some of which were limited in size. What we can say is that machine learning-based anomaly detection works best when using large datasets with a small amount of contextual understanding required. For example, identifying one type of denial of service attack requires recognizing an extreme increase in traffic, which requires a very limited contextual understanding. Alternatively, identifying an attack that involves utilizing specific functions in a specific order requires a high contextual understanding. Given that trained analysts are much better at understanding the context of an attack, the attack in the highly contextual example could be more easily identified using a human-made query.

B. Future Work

Future work that needs to be completed in this project mainly stems from one major aspect of the project: sample size. Since the requirements and timeline of this project did not enable us to take the time needed to examine a sufficient amount of datasets, the number of datasets tested needs to be dramatically expanded. Another team using our setup could accomplish this fairly easily by examining the 83 other small datasets available from the Mordor Project, but they might be held back by the relatively small sizes of said datasets in the same way that we were. They could also spend the time working on the other large-scale dataset the Mordor Project has based on APT 3. Although we did not find a lot of success using single metric and multi-metric anomaly job features, future research can also be done on those features as they can be useful in other scenarios.

Acknowledgment

The team would like to thank the entire CISA team for their support, guidance as well as the opportunity to work on this project. We would like to thank the team at the Mordor Project for their work on their public datasets. Finally, we would like to thank our Capstone professor for his constant support and mentorship on this project.

References

[1] R. Rodriguez and J. L. Rodriguez, “Introduction,” Introduction - , 2020. [Online]. Available: https://mordordatasets.com/introduction.html. [Accessed: 12-Apr-2021].

[2] Elasticsearch B.V., “Logstash Introduction,” Elastic. [Online]. Available: https://www.elastic.co/guide/en/logstash/current/introduction.html. [Accessed: 12-Apr-2021].

[3] “The Dukes: 7 Years Of Russian Cyber-Espionage,” F-Secure, 17-Sep-2015. [Online]. Available: https://blog-assets.f-secure.com/wp-content/uploads/2020/03/18122307/F-Secure_Dukes_Whitepaper.pdf. [Accessed: 13-Apr-2021].

[4] Mitre-Attack, “mitre-attack/ attack-arsenal/adversary_emulation/APT29/Emulation_Plan/Day 1/,” GitHub, 23-Jun-2020. [Online]. Available: https://github.com/mitre-attack/attack-arsenal/tree/master/adversary_emulation/APT29/Emulation_Plan/Day%201. [Accessed: 13-Apr-2021].

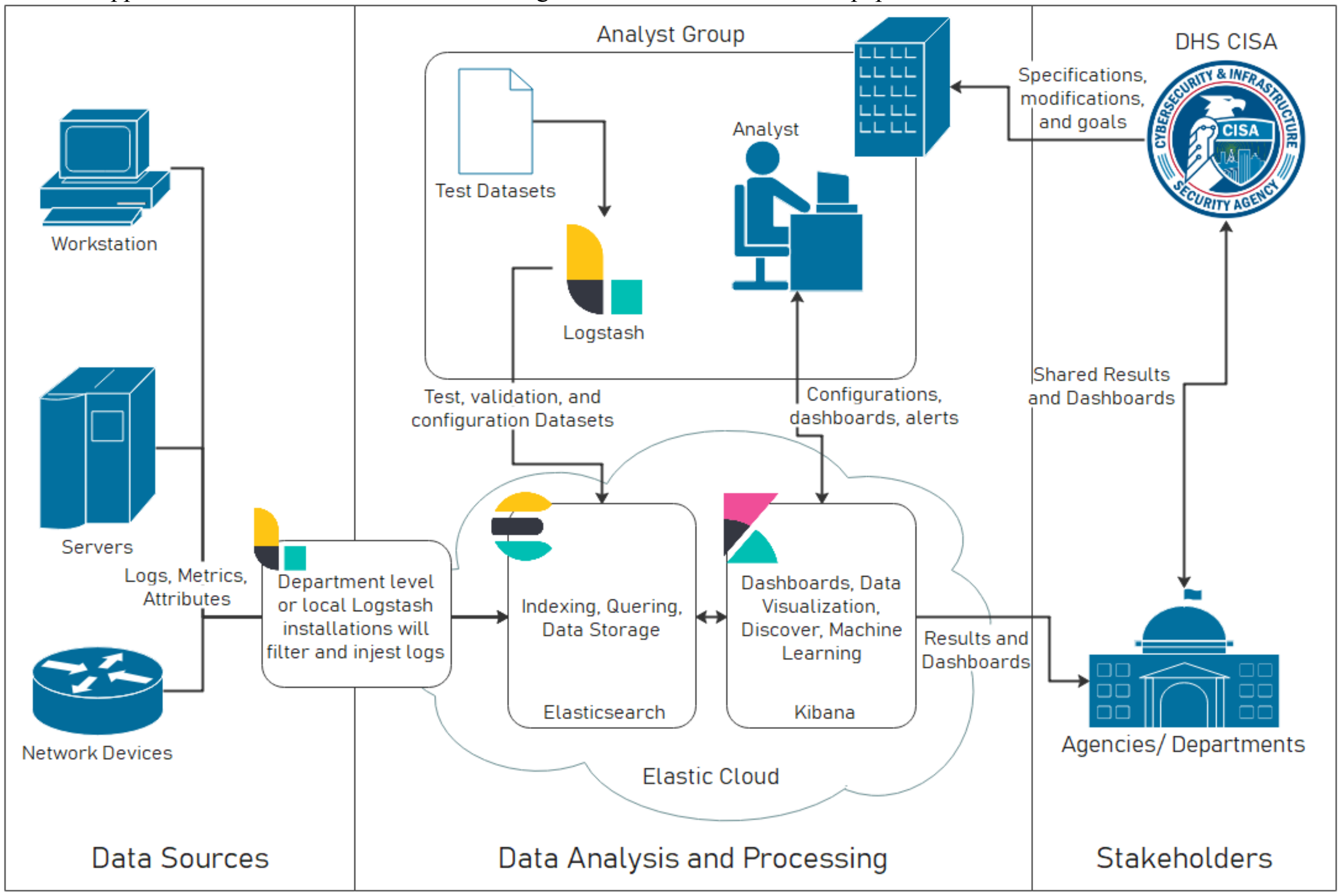
[5] Kaspersky Lab. , “What's behind APT29?,” Kaspersky MITRE ATT&amp;CK . [Online]. Available: https://www.kaspersky.com/enterprise-security/mitre/apt29. [Accessed: 13-Apr-2021].

[6] Core Security, “MS15-011 - Microsoft Windows Group Policy real exploitation via a SMB MiTM attack,” Core Security. [Online]. Available: https://www.coresecurity.com/core-labs/articles/ms15-011-microsoft-windows-group-policy-real-exploitation-via-a-smb-mitm-attack. [Accessed: 13-Apr-2021].

[7] M. Russinovich, “PsExec v2.33,” Windows Sysinternals | Microsoft Docs, 23-Mar-2021. [Online]. Available: https://docs.microsoft.com/en-us/sysinternals/downloads/psexec. [Accessed: 13-Apr-2021].

Appendix A

This Appendix includes full size versions of diagrams and models used in the paper.



Analyst Group
Test Datasets
Logstash
Analyst
Specifications, modifications, and goals
DHS CISA
CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY
CISA
Workstation
Servers
Logs, Metrics, Attributes
Network Devices
Department level or local Logstash installations will filter and injest logs
Test, validation, and configuration Datasets
Configurations, dashboards, alerts
Shared Results and Dashboards
Indexing, Quering, Data Storage
Elasticsearch
Dashboards, Data Visualization, Discover, Machine Learning
Kibana
Results and Dashboards
Agencies/ Departments
Elastic Cloud
Data Sources
Data Analysis and Processing
Stakeholders

Fig. 1. *(Product Concept of Operations) - Final Concept of Operations*

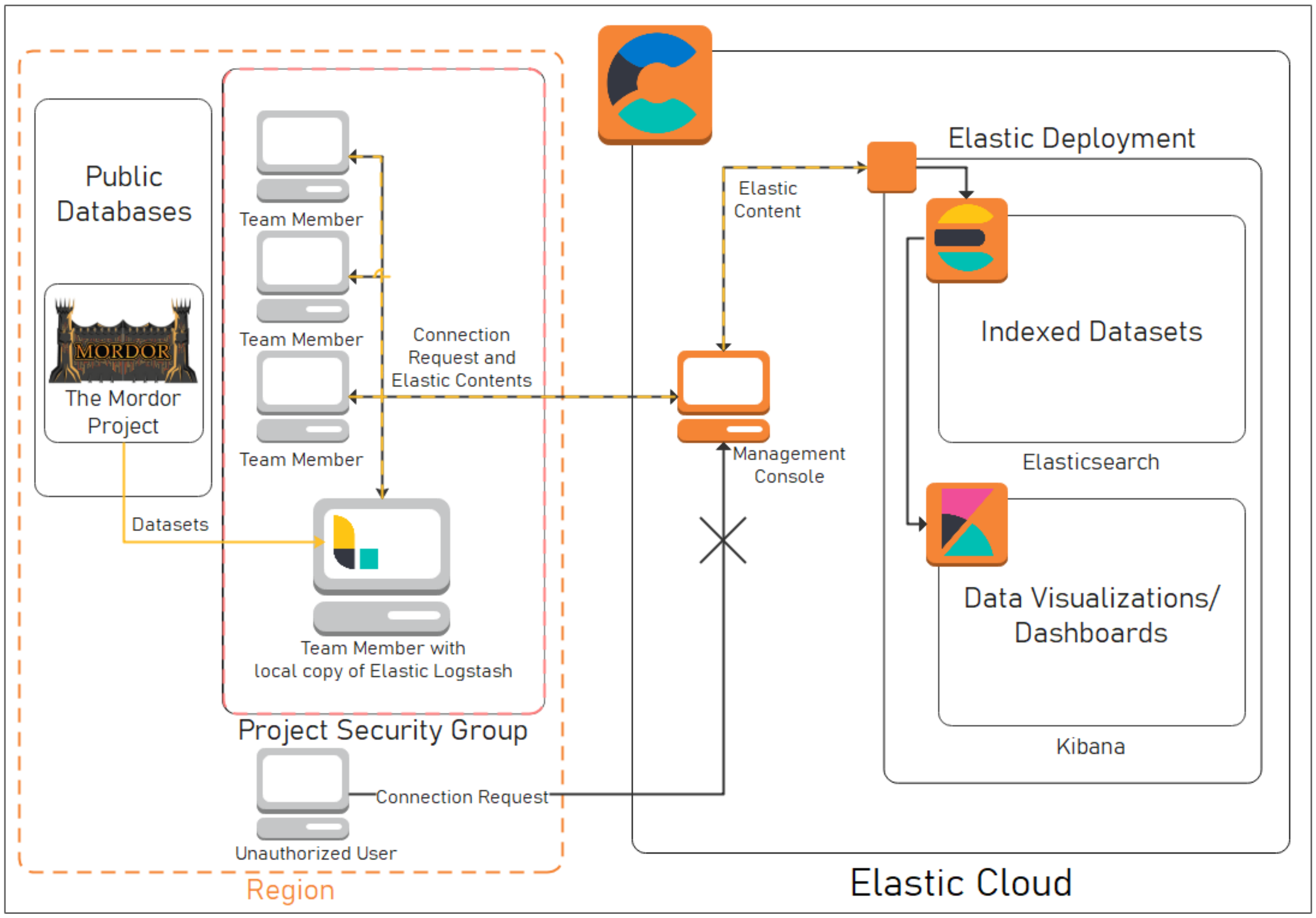


Fig. 2. (Product Architecture) - Our final products architecture model

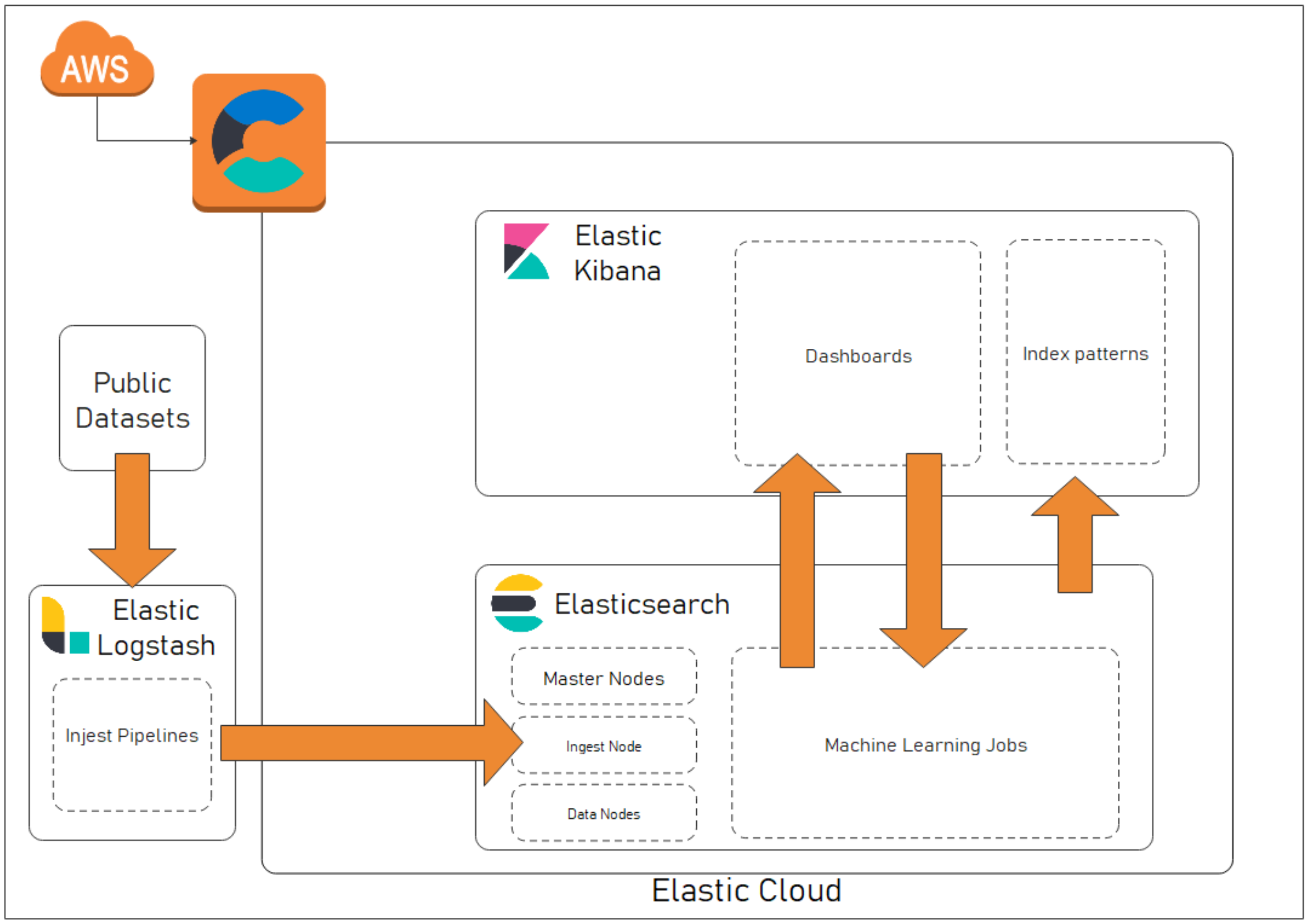


Fig. 3. *(Product Data Model) - Our final products data model*

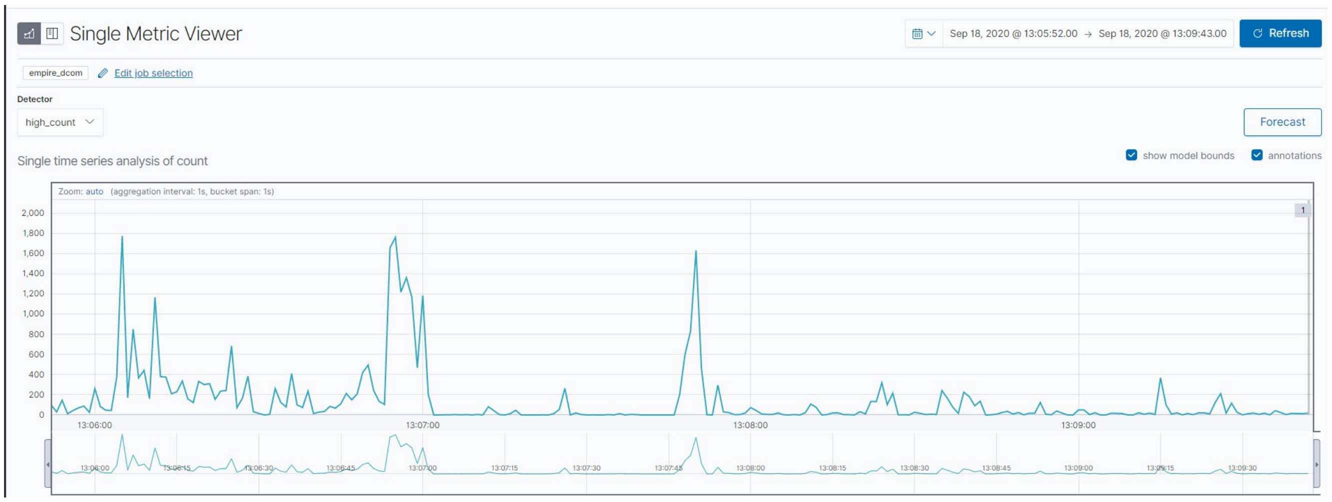
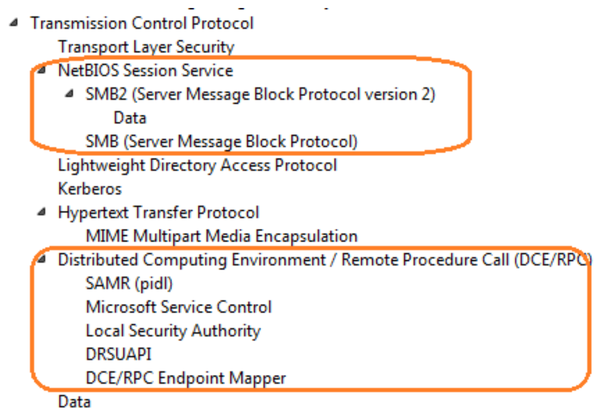
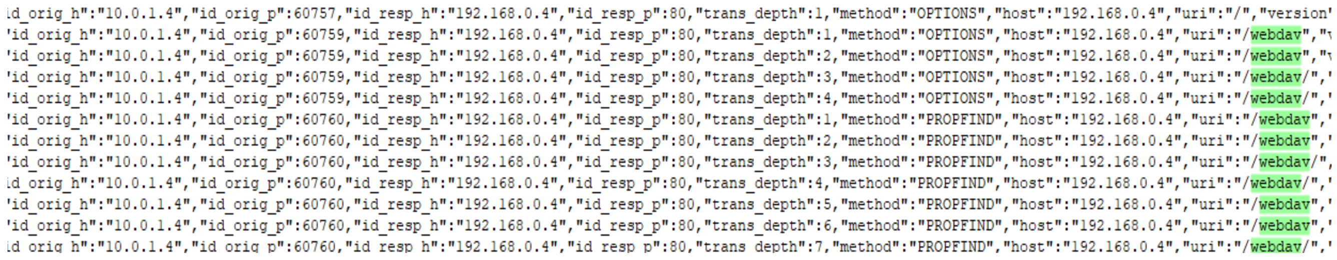


Fig. 4. *Example of no anomalies using single metric viewer on smaller datasets*



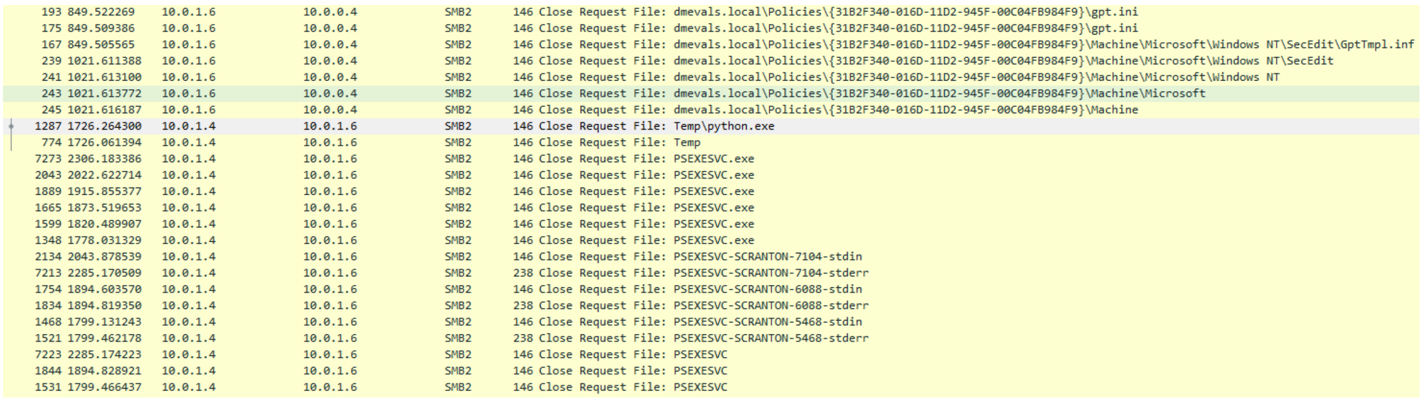
```
◢ Transmission Control Protocol
    Transport Layer Security
  ◢ NetBIOS Session Service
    ◢ SMB2 (Server Message Block Protocol version 2)
        Data
      SMB (Server Message Block Protocol)
    Lightweight Directory Access Protocol
    Kerberos
  ◢ Hypertext Transfer Protocol
      MIME Multipart Media Encapsulation
  ◢ Distributed Computing Environment / Remote Procedure Call (DCE/RPC)
      SAMR (pidl)
      Microsoft Service Control
      Local Security Authority
      DRSUAPI
      DCE/RPC Endpoint Mapper
    Data
```

Fig. 5. *(PCAP Protocol Hierarchy) - Both PCAP files contain SMB (used for file transferring) and DCE/RPC traffic*



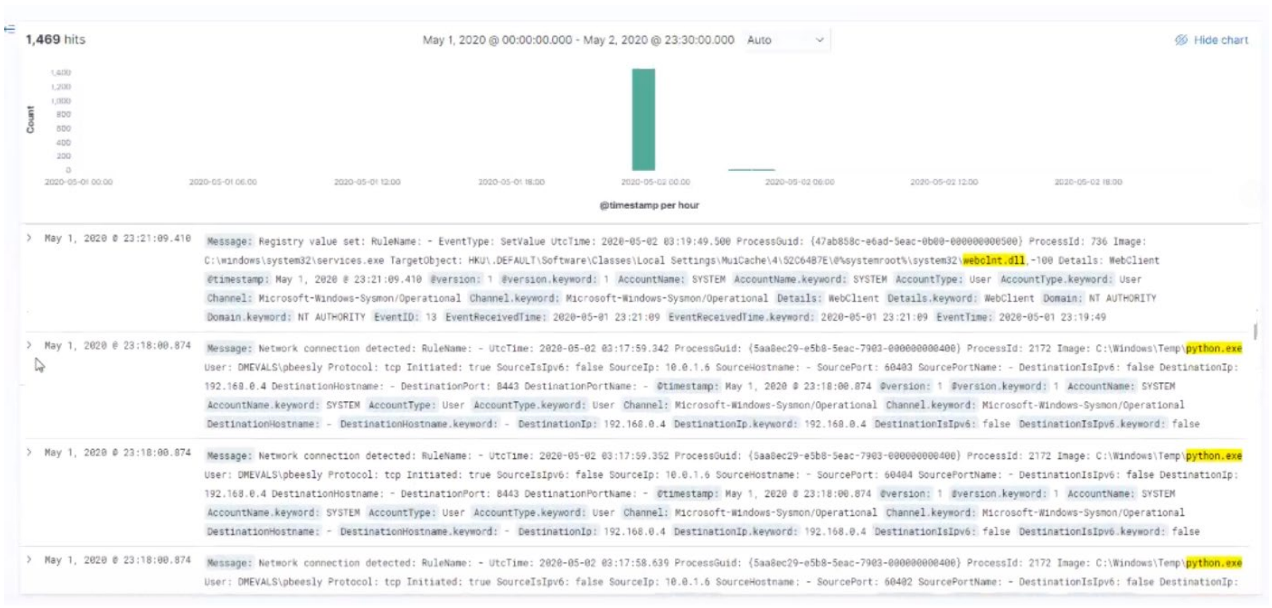
```
id_orig_h":"10.0.1.4","id_orig_p":60757,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":1,"method":"OPTIONS","host":"192.168.0.4","uri":"/","version'
'id_orig_h":"10.0.1.4","id_orig_p":60759,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":1,"method":"OPTIONS","host":"192.168.0.4","uri":"/webdav","
'id_orig_h":"10.0.1.4","id_orig_p":60759,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":2,"method":"OPTIONS","host":"192.168.0.4","uri":"/webdav","
'id_orig_h":"10.0.1.4","id_orig_p":60759,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":3,"method":"OPTIONS","host":"192.168.0.4","uri":"/webdav/",'
'id_orig_h":"10.0.1.4","id_orig_p":60759,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":4,"method":"OPTIONS","host":"192.168.0.4","uri":"/webdav/",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":1,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":2,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":3,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav/",
id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":4,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav/",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":5,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav",'
'id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":6,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav",'
id_orig_h":"10.0.1.4","id_orig_p":60760,"id_resp_h":"192.168.0.4","id_resp_p":80,"trans_depth":7,"method":"PROPFIND","host":"192.168.0.4","uri":"/webdav/",'
```

Fig. 6. *(Dataset) - 122 connections to Webdav, which is a protocol that allows files in file servers to be modified and managed*



```
 193 849.522269    10.0.1.6    10.0.0.4    SMB2    146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\gpt.ini
 175 849.509386    10.0.1.6    10.0.0.4    SMB2    146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\gpt.ini
 167 849.505565    10.0.1.6    10.0.0.4    SMB2    146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft\Windows NT\SecEdit\GptTmpl.inf
 239 1021.611388   10.0.1.6    10.0.0.4    SMB2    146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft\Windows NT\SecEdit
 241 1021.613100   10.0.1.6    10.0.0.4    SMB2    146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft\Windows NT
 243 1021.613772   10.0.1.6    10.0.0.4    SMB2    146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft
 245 1021.616187   10.0.1.6    10.0.0.4    SMB2    146 Close Request File: dmevals.local\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine
1287 1726.264300   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: Temp\python.exe
 774 1726.061394   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: Temp
7273 2306.183386   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC.exe
2043 2022.622714   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC.exe
1889 1915.855377   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC.exe
1665 1873.519653   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC.exe
1599 1820.489907   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC.exe
1348 1778.031329   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC.exe
2134 2043.878539   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC-SCRANTON-7104-stdin
7213 2285.170509   10.0.1.4    10.0.1.6    SMB2    238 Close Request File: PSEXESVC-SCRANTON-7104-stderr
1754 1894.603570   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC-SCRANTON-6088-stdin
1834 1894.819350   10.0.1.4    10.0.1.6    SMB2    238 Close Request File: PSEXESVC-SCRANTON-6088-stderr
1468 1799.131243   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC-SCRANTON-5468-stdin
1521 1799.462178   10.0.1.4    10.0.1.6    SMB2    238 Close Request File: PSEXESVC-SCRANTON-5468-stderr
7223 2285.174223   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC
1844 1894.828921   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC
1531 1799.466437   10.0.1.4    10.0.1.6    SMB2    146 Close Request File: PSEXESVC
```

Fig. 7. *(Dataset) - PSExeSVC.exe and python.exe references*



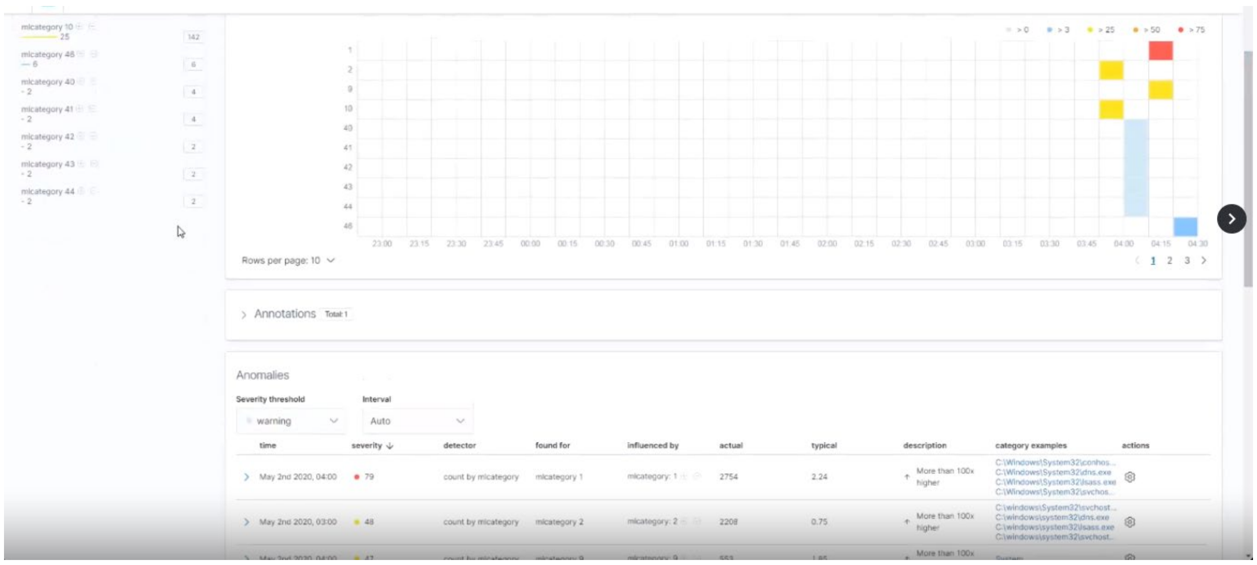
```
> May 1, 2020 @ 23:21:09.410   Message: Registry value set: RuleName: - EventType: SetValue UtcTime: 2020-05-02 03:19:49.500 ProcessGuid: {47ab858c-e6ad-5eac-0b00-000000000500} ProcessId: 736 Image:
                               C:\windows\system32\services.exe TargetObject: HKU\.DEFAULT\Software\Classes\Local Settings\MuiCache\4\52C64B7E\@%systemroot%\system32\webclnt.dll,-100 Details: WebClient
                               @timestamp: May 1, 2020 @ 23:21:09.410 @version: 1 @version.keyword: 1 AccountName: SYSTEM AccountName.keyword: SYSTEM AccountType: User AccountType.keyword: User
                               Channel: Microsoft-Windows-Sysmon/Operational Channel.keyword: Microsoft-Windows-Sysmon/Operational Details: WebClient Details.keyword: WebClient Domain: NT AUTHORITY
                               Domain.keyword: NT AUTHORITY EventID: 13 EventReceivedTime: 2020-05-01 23:21:09 EventReceivedTime.keyword: 2020-05-01 23:21:09 EventTime: 2020-05-01 23:19:49

> May 1, 2020 @ 23:18:00.874   Message: Network connection detected: RuleName: - UtcTime: 2020-05-02 03:17:59.342 ProcessGuid: {5aa8ec29-e5b8-5eac-7903-000000000400} ProcessId: 2172 Image: C:\Windows\Temp\python.exe
                               User: DMEVALS\pbeesly Protocol: tcp Initiated: true SourceIsIpv6: false SourceIp: 10.0.1.6 SourceHostname: - SourcePort: 60403 SourcePortName: - DestinationIsIpv6: false DestinationIp:
                               192.168.0.4 DestinationHostname: - DestinationPort: 8443 DestinationPortName: - @timestamp: May 1, 2020 @ 23:18:00.874 @version: 1 @version.keyword: 1 AccountName: SYSTEM
                               AccountName.keyword: SYSTEM AccountType: User AccountType.keyword: User Channel: Microsoft-Windows-Sysmon/Operational Channel.keyword: Microsoft-Windows-Sysmon/Operational
                               DestinationHostname: - DestinationHostname.keyword: - DestinationIp: 192.168.0.4 DestinationIp.keyword: 192.168.0.4 DestinationIsIpv6: false DestinationIsIpv6.keyword: false

> May 1, 2020 @ 23:18:00.874   Message: Network connection detected: RuleName: - UtcTime: 2020-05-02 03:17:59.352 ProcessGuid: {5aa8ec29-e5b8-5eac-7903-000000000400} ProcessId: 2172 Image: C:\Windows\Temp\python.exe
                               User: DMEVALS\pbeesly Protocol: tcp Initiated: true SourceIsIpv6: false SourceIp: 10.0.1.6 SourceHostname: - SourcePort: 60404 SourcePortName: - DestinationIsIpv6: false DestinationIp:
                               192.168.0.4 DestinationHostname: - DestinationPort: 8443 DestinationPortName: - @timestamp: May 1, 2020 @ 23:18:00.874 @version: 1 @version.keyword: 1 AccountName: SYSTEM
                               AccountName.keyword: SYSTEM AccountType: User AccountType.keyword: User Channel: Microsoft-Windows-Sysmon/Operational Channel.keyword: Microsoft-Windows-Sysmon/Operational
                               DestinationHostname: - DestinationHostname.keyword: - DestinationIp: 192.168.0.4 DestinationIp.keyword: 192.168.0.4 DestinationIsIpv6: false DestinationIsIpv6.keyword: false

> May 1, 2020 @ 23:18:00.874   Message: Network connection detected: RuleName: - UtcTime: 2020-05-02 03:17:58.639 ProcessGuid: {5aa8ec29-e5b8-5eac-7903-000000000400} ProcessId: 2172 Image: C:\Windows\Temp\python.exe
                               User: DMEVALS\pbeesly Protocol: tcp Initiated: true SourceIsIpv6: false SourceIp: 10.0.1.6 SourceHostname: - SourcePort: 60402 SourcePortName: - DestinationIsIpv6: false DestinationIp:
```

Fig. 8. *(Elastic Discover) -Picture of querying dataset for "Message : *webclnt.dll or Message : *python.exe or Message : *psexec64.exe"*



mlcategory 10 - 25	142
mlcategory 48 - 6	6
mlcategory 40 - 2	4
mlcategory 41 - 2	4
mlcategory 42 - 2	2
mlcategory 43 - 2	2
mlcategory 44 - 2	2

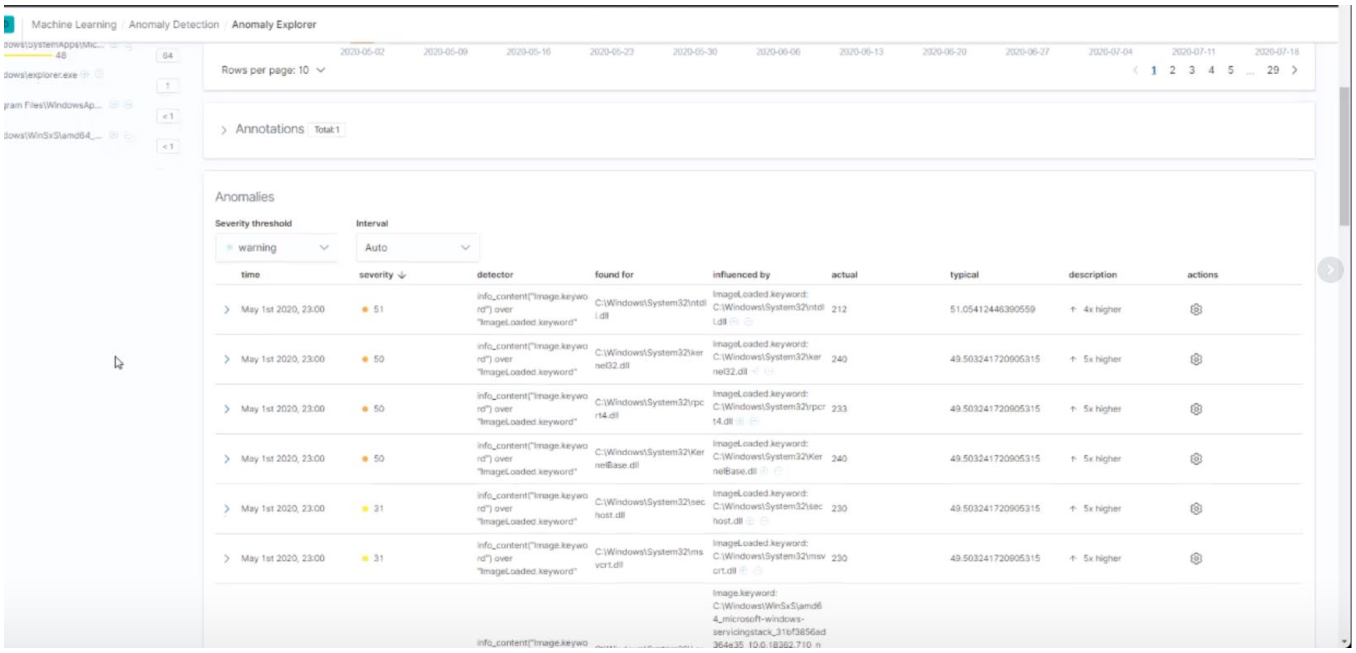
Annotations Total: 1

Anomalies

Severity threshold: warning
Interval: Auto

time	severity	detector	found for	influenced by	actual	typical	description	category examples	actions
May 2nd 2020, 04:00	● 79	count by mlcategory	mlcategory 1	mlcategory: 1	2754	2.24	More than 100x higher	C:\Windows\System32\conhos... C:\Windows\System32\dns.exe C:\Windows\System32\lsass.exe C:\Windows\System32\svchos...	
May 2nd 2020, 03:00	● 48	count by mlcategory	mlcategory 2	mlcategory: 2	2208	0.75	More than 100x higher	C:\windows\System32\svchost... C:\windows\system32\dns.exe C:\windows\system32\lsass.exe C:\windows\system32\svchost...	
May 2nd 2020, 04:00	● 47	count by mlcategory	mlcategory 9	mlcategory: 9	553	1.85	More than 100x	System	

Fig. 9. *(Elastic Categorization) - Example of anomalies found for "Image" for specific services that have been run.*

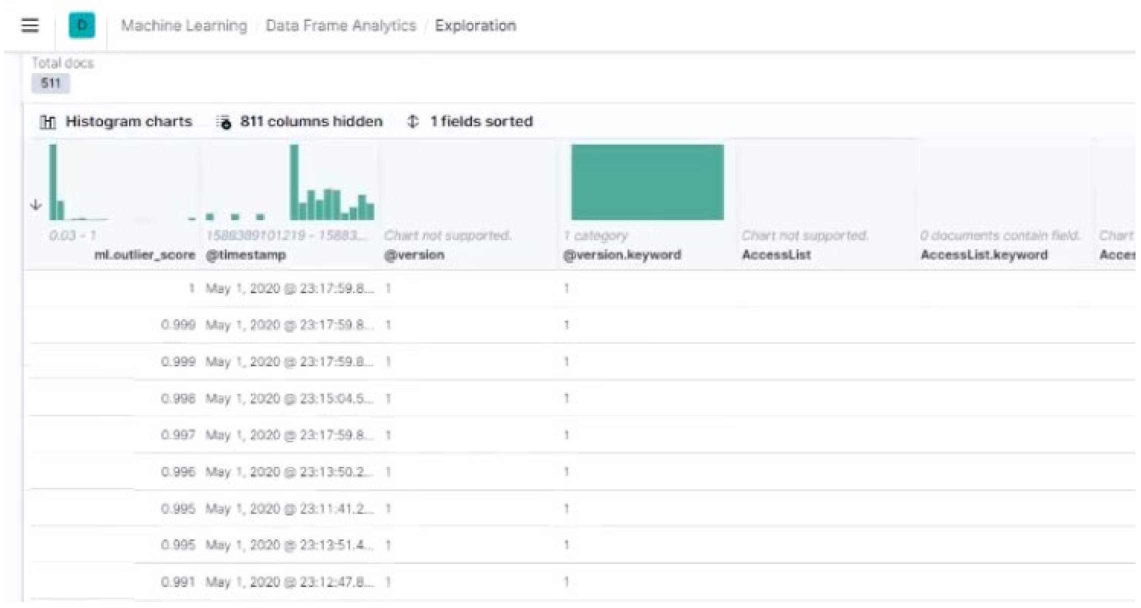


Anomalies

Severity threshold: warning
Interval: Auto

time	severity	detector	found for	influenced by	actual	typical	description	actions
May 1st 2020, 23:00	51	info_content("Image.keyword") over "ImageLoaded.keyword"	C:\Windows\System32\ntdll.dll	ImageLoaded.keyword: C:\Windows\System32\ntdll.dll	212	51.05412446390559	↑ 4x higher	
May 1st 2020, 23:00	50	info_content("Image.keyword") over "ImageLoaded.keyword"	C:\Windows\System32\kernel32.dll	ImageLoaded.keyword: C:\Windows\System32\kernel32.dll	240	49.503241720905315	↑ 5x higher	
May 1st 2020, 23:00	50	info_content("Image.keyword") over "ImageLoaded.keyword"	C:\Windows\System32\rpcrt4.dll	ImageLoaded.keyword: C:\Windows\System32\rpcrt4.dll	233	49.503241720905315	↑ 5x higher	
May 1st 2020, 23:00	50	info_content("Image.keyword") over "ImageLoaded.keyword"	C:\Windows\System32\KernelBase.dll	ImageLoaded.keyword: C:\Windows\System32\KernelBase.dll	240	49.503241720905315	↑ 5x higher	
May 1st 2020, 23:00	31	info_content("Image.keyword") over "ImageLoaded.keyword"	C:\Windows\System32\sechost.dll	ImageLoaded.keyword: C:\Windows\System32\sechost.dll	230	49.503241720905315	↑ 5x higher	
May 1st 2020, 23:00	31	info_content("Image.keyword") over "ImageLoaded.keyword"	C:\Windows\System32\msvcrt.dll	ImageLoaded.keyword: C:\Windows\System32\msvcrt.dll	230	49.503241720905315	↑ 5x higher	
		info_content("Image.keywo		Image.keyword: C:\Windows\WinSxS\amd64_microsoft-windows-servicingstack_31bf3856ad364e35_10.0.18362.710_n				

Fig. 10. (Elastic Advanced ML) - Example of anomalies found for count by "Image.keyword" with influencer"ImageLoaded"



ml.outlier_score	@timestamp	@version	@version.keyword	AccessList	AccessList.keyword
1	May 1, 2020 @ 23:17:59.8...	1	1		
0.999	May 1, 2020 @ 23:17:59.8...	1	1		
0.999	May 1, 2020 @ 23:17:59.8...	1	1		
0.998	May 1, 2020 @ 23:15:04.5...	1	1		
0.997	May 1, 2020 @ 23:17:59.8...	1	1		
0.996	May 1, 2020 @ 23:13:50.2...	1	1		
0.995	May 1, 2020 @ 23:11:41.2...	1	1		
0.995	May 1, 2020 @ 23:13:51.4...	1	1		
0.991	May 1, 2020 @ 23:12:47.8...	1	1		

*Fig. 11. (Elastic Data Frame Analytics) -Picture of querying dataset Image.Keyword:*python.exe to understand whether instances of the program may be an outlier, and thus anomalistic. Criticality and timestamps on the left two columns.*

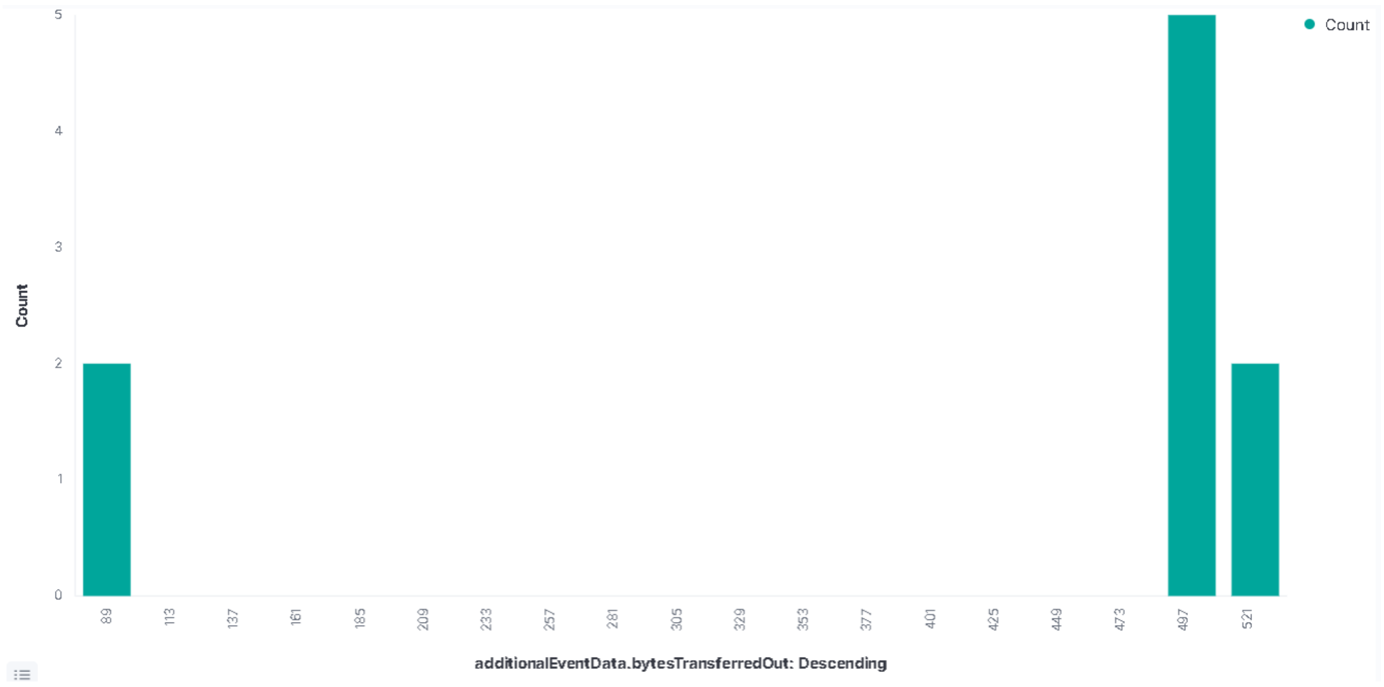


Fig. 13. *(First Visualization) - Bar graph representing the number of bytes transferred out and the count*

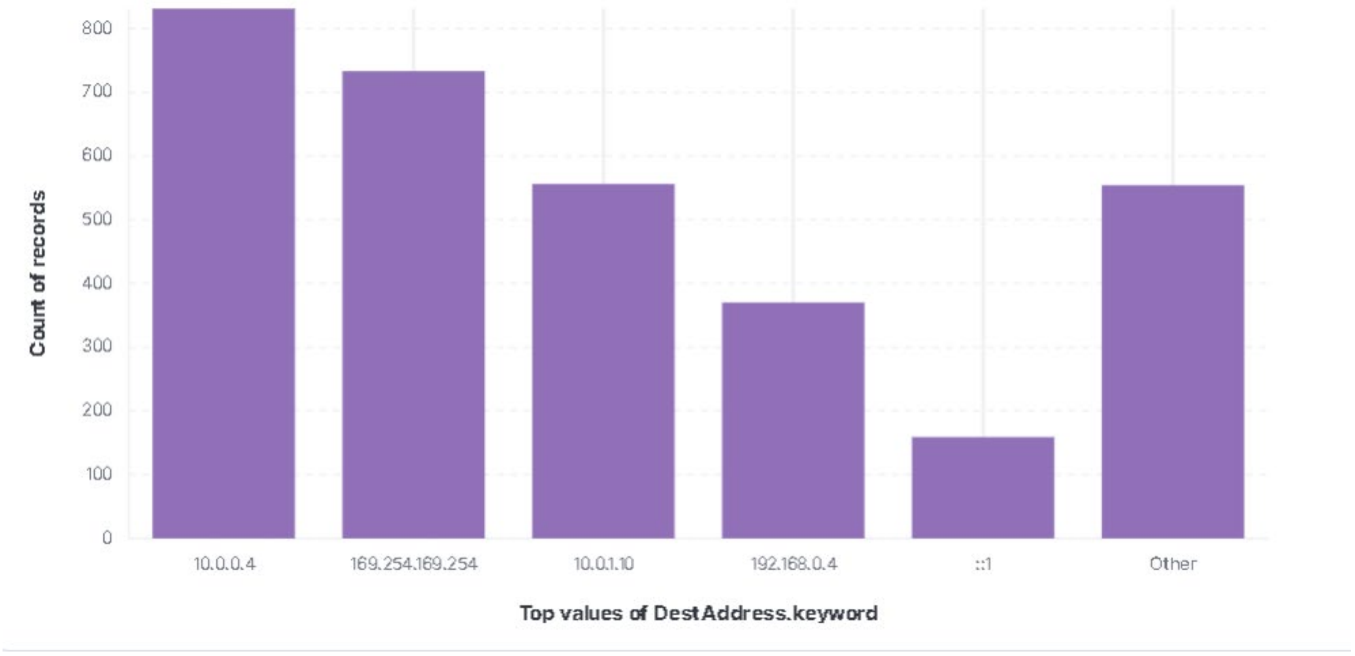


Fig. 14. *(APT 29 Day 1Visualization) - Bar graph representing the top Destination IP addresses within the Day 1 dataset*

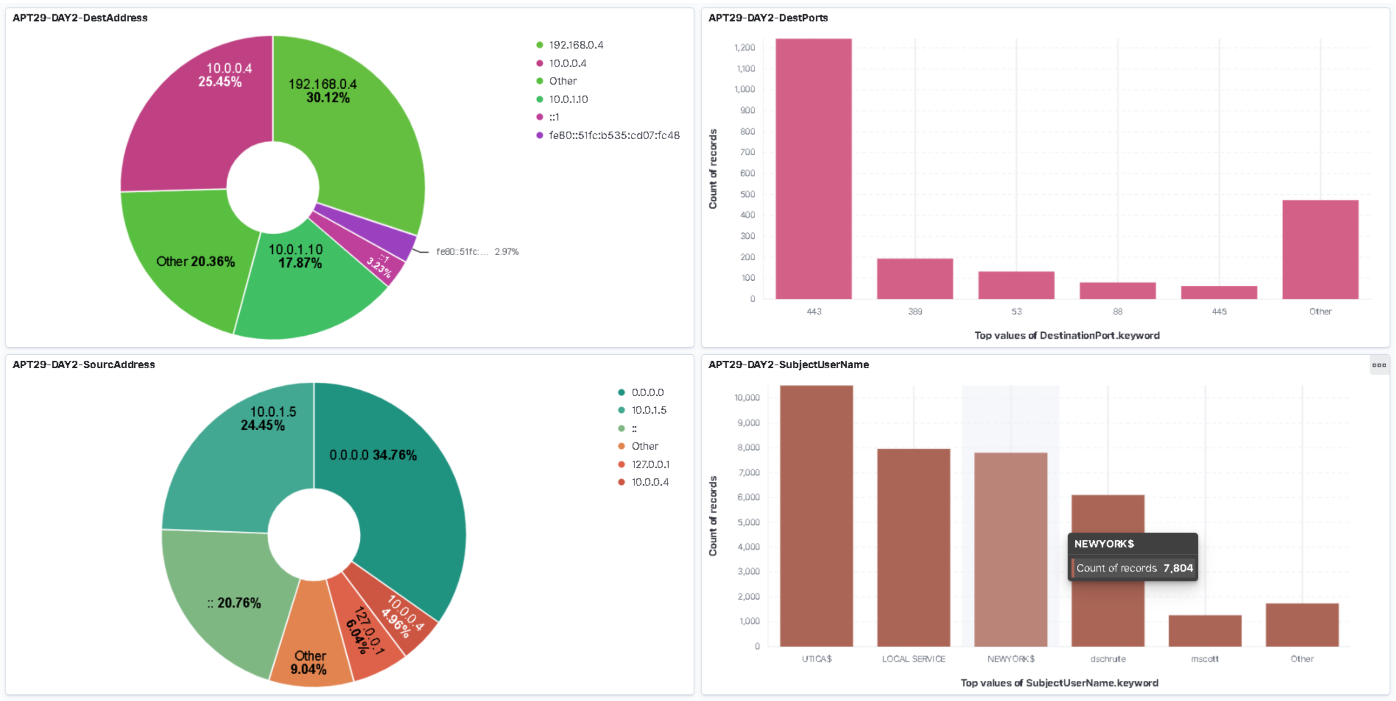
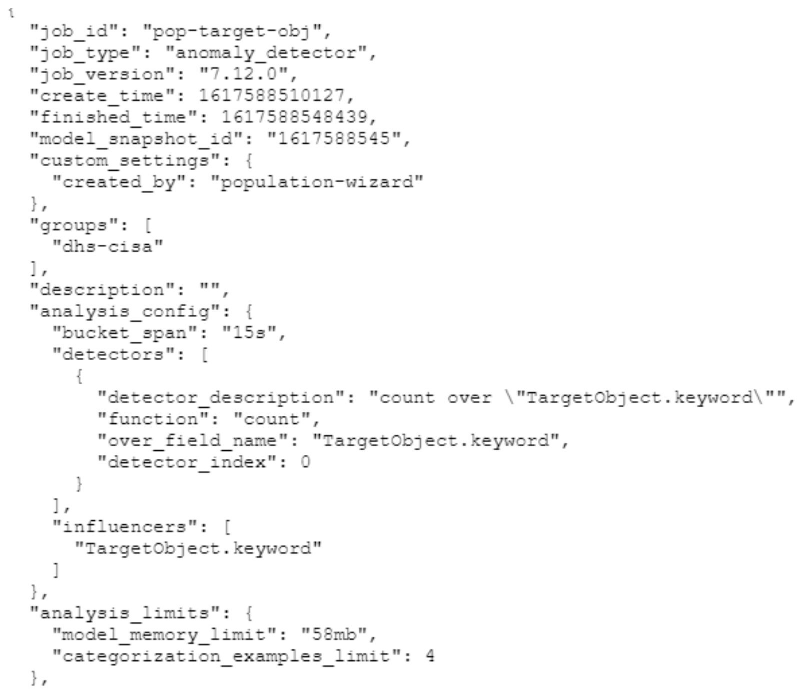


Fig. 15. (APT 29 Day 2 Dashboard) - Dashboard of 4 visualizations which consist of: Destination IP Addresses, Destination Ports, Source Addresses, and Subject Usernam

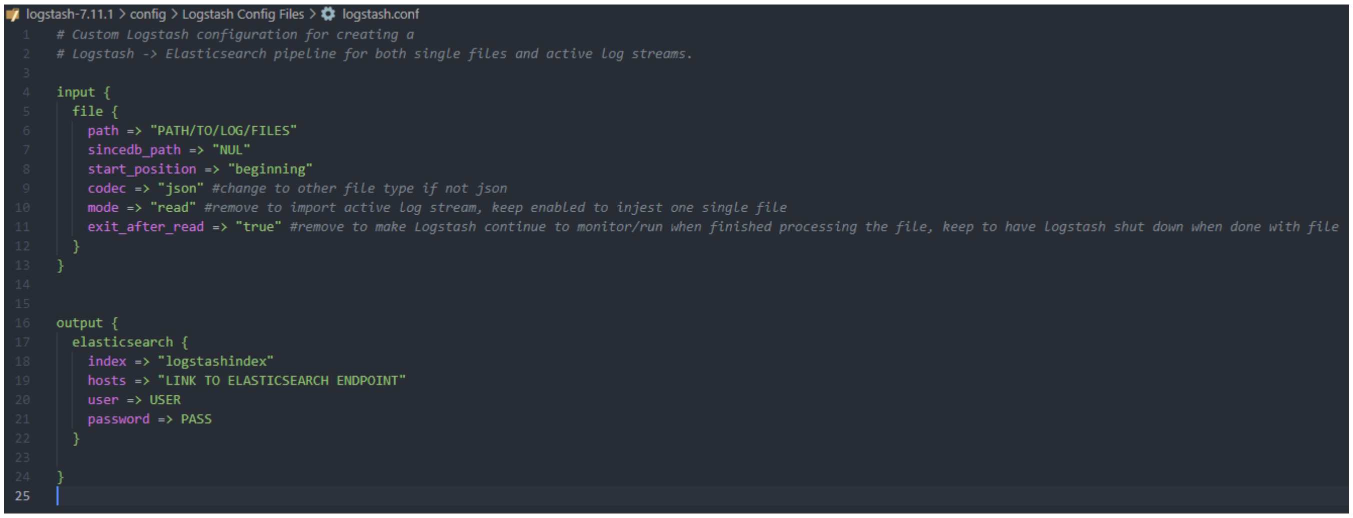
Appendix B

This Appendix includes figures mentioned in the paper but not included inline in the paper.



```
{
  "job_id": "pop-target-obj",
  "job_type": "anomaly_detector",
  "job_version": "7.12.0",
  "create_time": 1617588510127,
  "finished_time": 1617588548439,
  "model_snapshot_id": "1617588545",
  "custom_settings": {
    "created_by": "population-wizard"
  },
  "groups": [
    "dhs-cisa"
  ],
  "description": "",
  "analysis_config": {
    "bucket_span": "15s",
    "detectors": [
      {
        "detector_description": "count over \"TargetObject.keyword\"",
        "function": "count",
        "over_field_name": "TargetObject.keyword",
        "detector_index": 0
      }
    ],
    "influencers": [
      "TargetObject.keyword"
    ]
  },
  "analysis_limits": {
    "model_memory_limit": "58mb",
    "categorization_examples_limit": 4
  },
```

Fig. 1. (pop-targetobj.json) - Machine learning jobs that were created for anomaly detection and data frame analytics were exported into json files. The code can be used when replicating jobs. These files are located under the ML Config directory that was a part of this delivery. Files will need to be modified accordingly.



logstash-7.11.1 > config > Logstash Config Files > logstash.conf

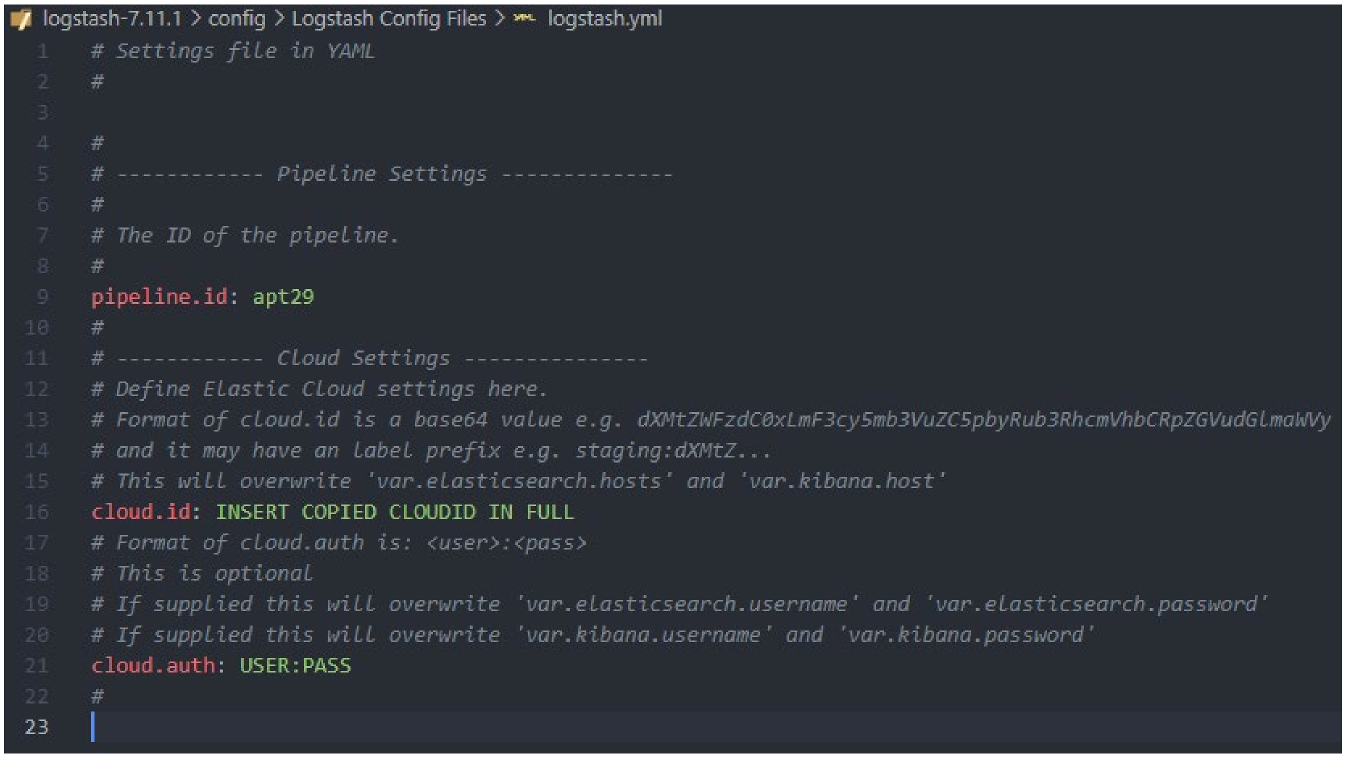
```
# Custom Logstash configuration for creating a
# Logstash -> Elasticsearch pipeline for both single files and active log streams.

input {
  file {
    path => "PATH/TO/LOG/FILES"
    sincedb_path => "NUL"
    start_position => "beginning"
    codec => "json" #change to other file type if not json
    mode => "read" #remove to import active log stream, keep enabled to injest one single file
    exit_after_read => "true" #remove to make Logstash continue to monitor/run when finished processing the file, keep to have logstash shut down when done with file
  }
}


output {
  elasticsearch {
    index => "logstashindex"
    hosts => "LINK TO ELASTICSEARCH ENDPOINT"
    user => USER
    password => PASS
  }

}
```

Fig. 2. (logstash.conf) - This conf file can be found under the config folder under the default Logstash install folder. This config can be made usable by inserting the applicable information from an Elastic deployment.



logstash-7.11.1 > config > Logstash Config Files > logstash.yml

```
# Settings file in YAML
#

#
# ------------ Pipeline Settings --------------
#
# The ID of the pipeline.
#
pipeline.id: apt29
#
# ------------ Cloud Settings ---------------
# Define Elastic Cloud settings here.
# Format of cloud.id is a base64 value e.g. dXMtZWFzdC0xLmF3cy5mb3VuZC5pbyRub3RhcmVhbCRpZGVudGlmaWVy
# and it may have an label prefix e.g. staging:dXMtZ...
# This will overwrite 'var.elasticsearch.hosts' and 'var.kibana.host'
cloud.id: INSERT COPIED CLOUDID IN FULL
# Format of cloud.auth is: <user>:<pass>
# This is optional
# If supplied this will overwrite 'var.elasticsearch.username' and 'var.elasticsearch.password'
# If supplied this will overwrite 'var.kibana.username' and 'var.kibana.password'
cloud.auth: USER:PASS
#

```

Fig. 3. (logtash.yml) - This yml file can be found under the config folder under the default Logstash install folder. This yml file has had all settings that have not been edited removed to allow all the new configurations to be shown in one screenshot. Everything not shown in this screenshot is kept as default.