Global Information Assurance Certification Paper

Enhancing the security capabilities of the Ubiquiti UniFi Security Gateway (USG)

GIAC (GDSA) Gold Certification

Author: Tim Coakley, timcoakley@yahoo.co.uk
Advisor: Christopher Walker, CISSP, CCISO, CISA, GCED, GWEB
Accepted: October 1, 2020

Abstract

The UniFi Security Gateway (USG) is a popular security device manufactured by Ubiquiti; it is relatively unique within the marketplace for its affordability and adoption of use within both Enterprise and SOHO environments. The USG, at its core, provides a firewall, routing, and advanced security features for network protection, traffic management, and ease of integration. A balanced set of features come pre-packaged. However, advanced users and security practitioners seeking more granular detail may be disappointed with some of the box security reporting options.

This paper seeks to document, utilize, and extend upon the capabilities of the USG device to enhance monitoring, detection, and reporting of events within a cybersecurity context. This paper will benefit a broad range of security practitioners from Enterprise, Small business, and independent security researchers.

1. Introduction

America technology company Ubiquiti created the USG. It comes in two form factors, USG and USG Pro 4, sharing very similar feature sets differing primarily in throughput, memory, and processor requirements. Both models are enterprise solutions (Ubiquiti, 2020).

The Ubiquiti product range places a strong emphasis on modular and ease of integration with other Ubiquiti products, desirable in an enterprise environment, achieved through the use of controller software. The UniFi controller can be deployed on premise or off-premise to manage and integrate all devices from a central location collectively. On-premise controllers can be implemented as installed software on almost any client device or through a dedicated low power device called a UniFi Cloud Key. A UniFi Cloud Key connects directly to a UniFi switch port using Power over Ethernet with software hosted on an onboard SD Memory card. Off-premise controllers are typical via cloud-hosted environments and ideal for global enterprises with centralized Network Operations (Ubiquiti, 2020).

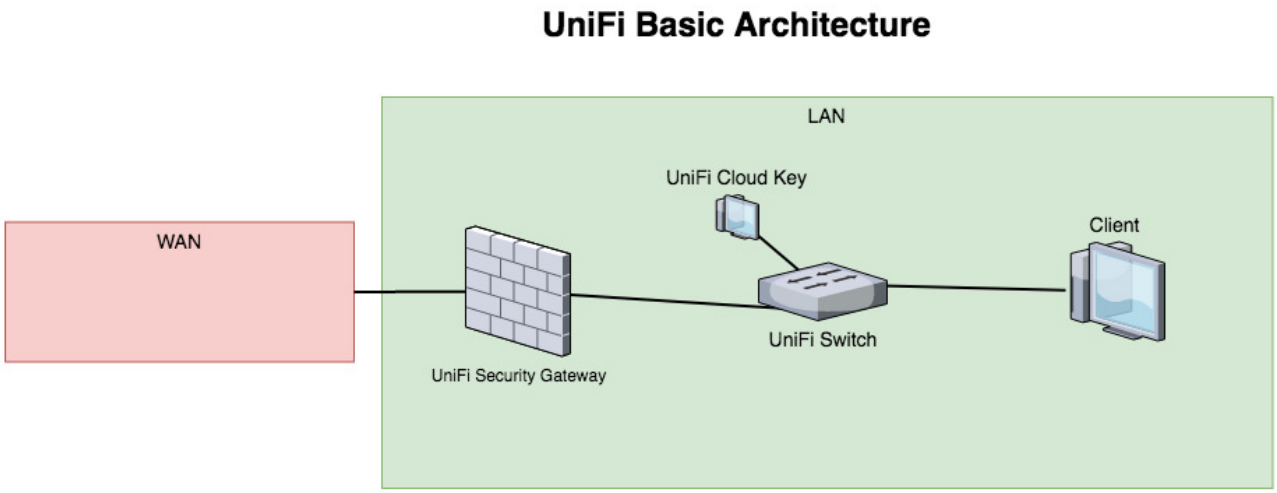

Illustration 1: Basic Architecture

This paper briefly delves into the UniFi Controller interface to enable features necessary to enhance the security functionality of a USG further. The majority of the research will focus on extracting value from the logging information created by a USG

Tim Coakley, timcoakley@yahoo.co.uk

during operation. It was noted logging information is not readily available to a user or accessible from within the factory set controller environment interfaces.

Initial review of the USG identified several shortcomings with the controller interface, including but are not limited to the inability to view, search, enrich, process, and report on the internal logging, which by default is disabled.

The researchers' goal is to use available technology to best process available logging from a UniFi USG device to be analyzed from a cybersecurity perspective and in a cost-effective manner suitable for the broadest audience.

2. Research and Design

Several considerations were taken before conducting the study to include a high-level solution design, which greatly influenced this paper's outcome. The solution must rely on open source technology, be flexible, and process data of differing velocity, volume, and variety. The following high-level tasks were identified:

1. Build Infrastructure; for test and development
2. Configure the USG device
3. Analyze log samples
4. Develop log parsing pattern
5. Build Dashboard and Search Index (proposed final security enhancement)

2.1 Solution Diagram

The following illustration shows the USG, Log Collector, and ELK consisting of Elasticsearch, Logstash, and Kibana (Elastic, 2020) to include example IP addresses and port numbers. A fictitious but relevant IoT network is present in the design, as this may be a common challenge in many enterprises.

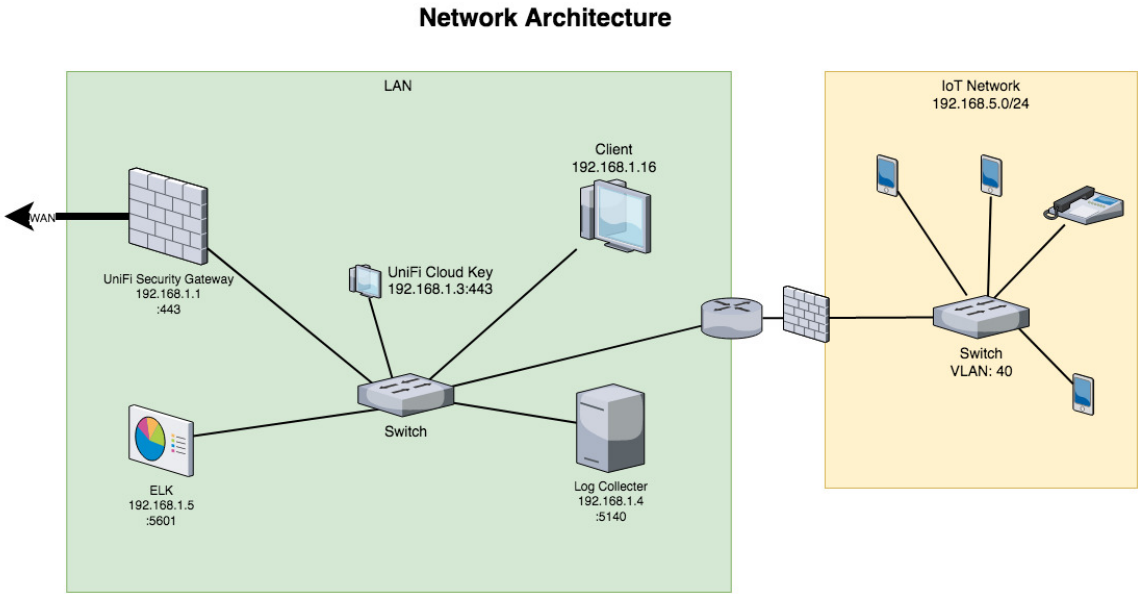

Illustration 2: Network Architecture

2.2 Components

The logical components in the following table were utilized for this research. The components represent minimum software requirements.

Item	Version	Description
UniFi Network Controller	5.13.32	Network Management Software
Fluentd	0.12.40	Log Agent
Filebeat	7.8.0	Log Forwarder
Elastic Search	6.8.10	Lucene based search engine
Log Stash	7.8.0	Data processing pipeline
Kibana	7.7.3	Data visualization
Memcached	1.6.7	Key-value store for enrichment

Table 1: Logical Components

Tim Coakley, timcoakley@yahoo.co.uk

2.3 USG Configuration – IoT Network Configuration

A fictitious network called IoT, abbreviated for the "Internet of Things", was created and was isolated from the corporate environment. The network was intended to route Internet connectivity to non-corporate mobile devices and prevent direct connectivity to corporate devices.

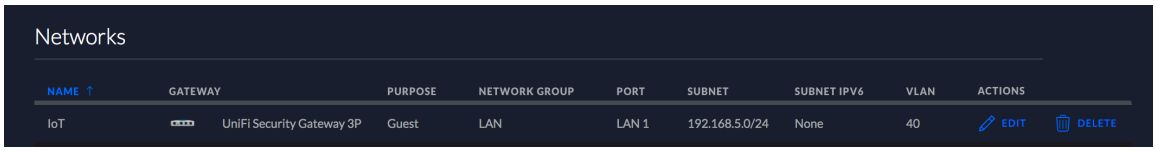

Illustration 3: IoT Network Configuration

2.4 USG Configuration – Firewall Rule Configuration

A firewall rule was created as "IoT OUTBOUND Logging", this test rule permitted all outbound traffic and log activity. The rule was critical to enable the USG device to generate the appropriate log data. The firewall rule only applied to the IoT network detailed in the previous section 2.3.

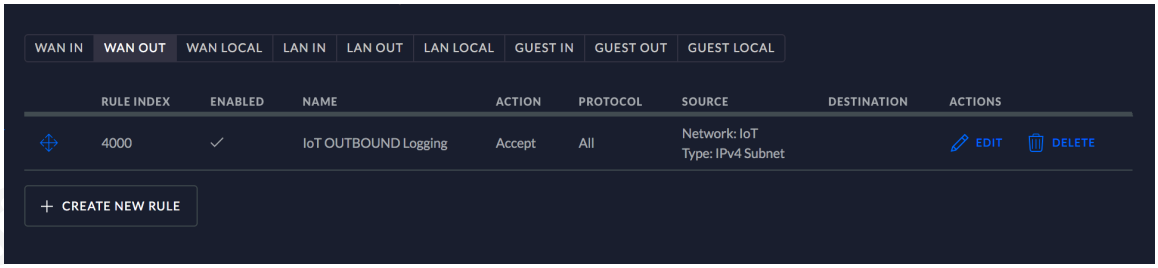

Illustration 4: Firewall Configuration

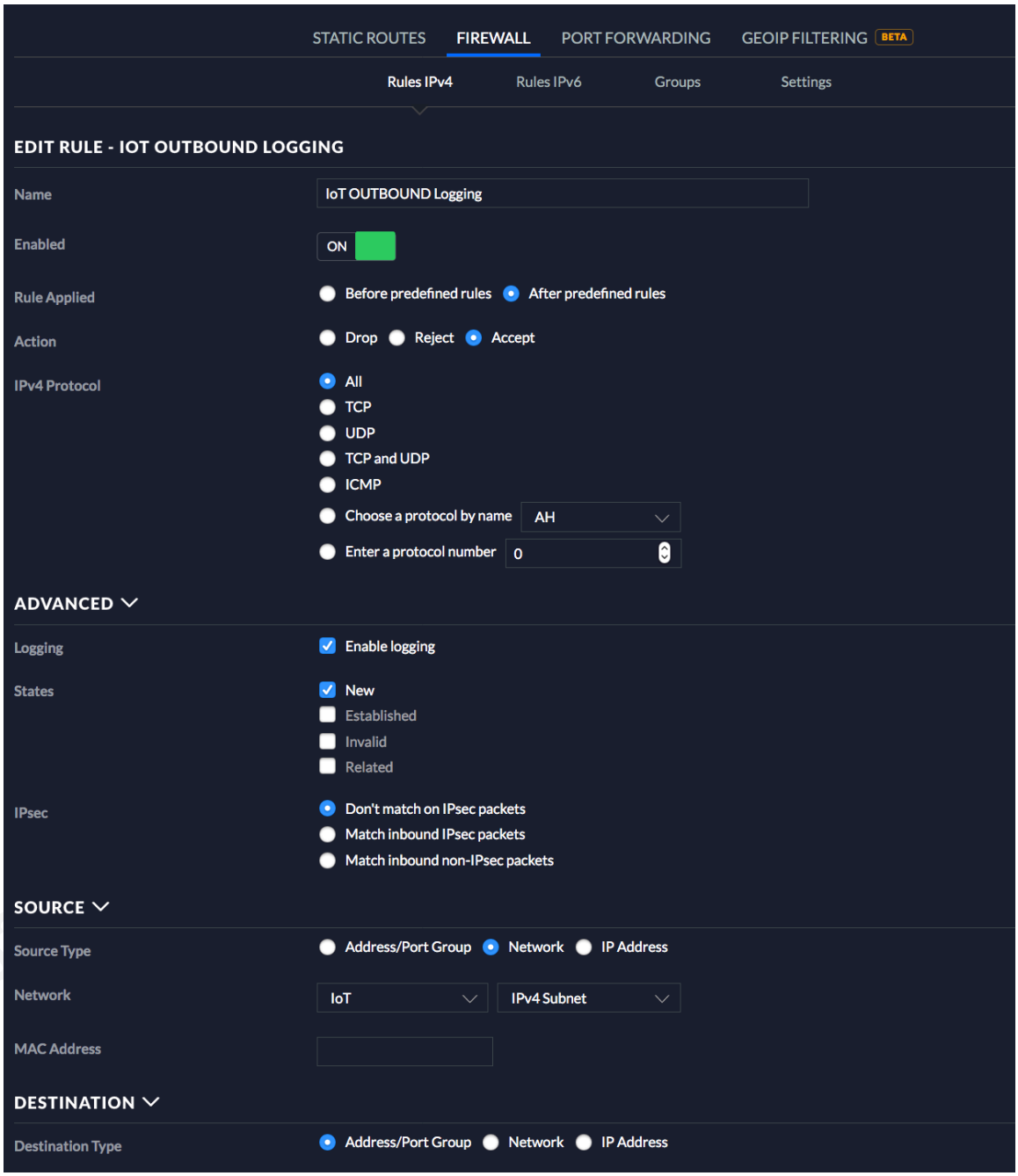

Illustration 5: IoT Outbound Logging Configuration

2.5 USG Configuration – Enable Logging

The setting to Enable remote Syslog server was enabled. This setting was enabled to send logs from the USG device to the remote log collector/aggregator. In this example, the collector was hosted at 192.168.1.4 on port 5140.

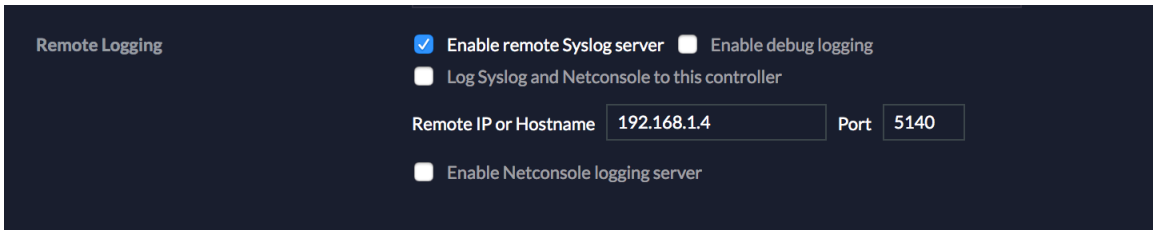

Illustration 6: Enable Remote Logging

2.6 Use Cases

The research scenario was to monitor outbound traffic to the Internet from devices connected to the IoT network. With no prior experience or business priorities, exploratory use cases to understand and baseline activity are detailed in the following table.

#	**Description**
1	Identify the most common ports by destination
2	Identify the least common ports by destination
3	Identify the activity based on the least common destination country
4	Identify the activity based on the least common destination region
5	Identify the activity of destination country by frequency
6	Identify the most common destination IP address
7	Display geographic map activity
8	Enhanced enrichment of malware threat lookup

Table 2: Use Cases

More complex use cases were considered, including but not limited to frequency and length of connections. An enrichment process was created, see use case number seven, of the above table with geographic representation of IP address data. An additional enrichment process requiring an accurate and up to date threat feed was also

created to detect devices connecting to malicious destination addresses, see use case eight.

2.7 Tags

Tags are an essential method of putting data into related groups, useful for an operator tasked with managing or analyzing large data sets (Defining Alerts, 2020). In this research, data is partitioned using the following groups:

- Logs which have not parsed correctly (tag: _grokparsefailure)
- Logs derived from the IoT network and correctly parsed (tag: IoT)
- Logs which have not been successfully enriched tag: _geoip_lookup_failure).

Tagging complements filtering and analysis techniques by allowing large datasets to be significantly reduced before performing more memory-intensive filtering operations such as Boolean conditions and full-text searching. Tagging strategies were also useful for detecting and troubleshooting during log parsing development or through unexpected changes in logging format.

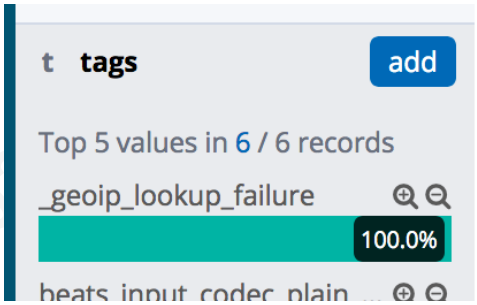

Illustration 7: Tagging

3. Tools and Techniques

3.1 GROK

Logstash GROK is a pattern-matching language for parsing arbitrary text and structure it for processing; Logstash includes over 100 default patterns (Elastic, 2020). A custom grok pattern for matching USG logs was required to parse events from USG logging correctly. The complete grok pattern is included within the sample Logstash configuration, lines 10 through to 17 in Appendix A.

The development phase, creating the appropriate grok pattern to match all fields while discarding non-matching fields, was found to be the most problematic and time-consuming. Rudimentary debugging was available through the command line execution using Logstash and an online pattern debugger grokdebug (GrokDebug, 2020). Improvements in development techniques would increase awareness and adoption of this pattern matching methodology.

3.2 Kibana Index Pattern and Elastic Indexes

From within the management interface of the Kibana instance, an index pattern was defined using the pattern logstash-usgfirewall-* (Index patterns and fields, 2020). During development, it was noted that the prepended Logstash- is mandatory for the built-in GeoIP enrichment to function correctly.

The Index Management interface for Elasticsearch confirmed the Logstash configuration and Kibana index pattern worked as designed with an index created for each day of logging received. As defined in the Logstash configuration file, an example index was created logstash-usgfirewall-2020.07.12. All parsed fields were indexed, allowing ingested logs to be searched, filtered, and reported.

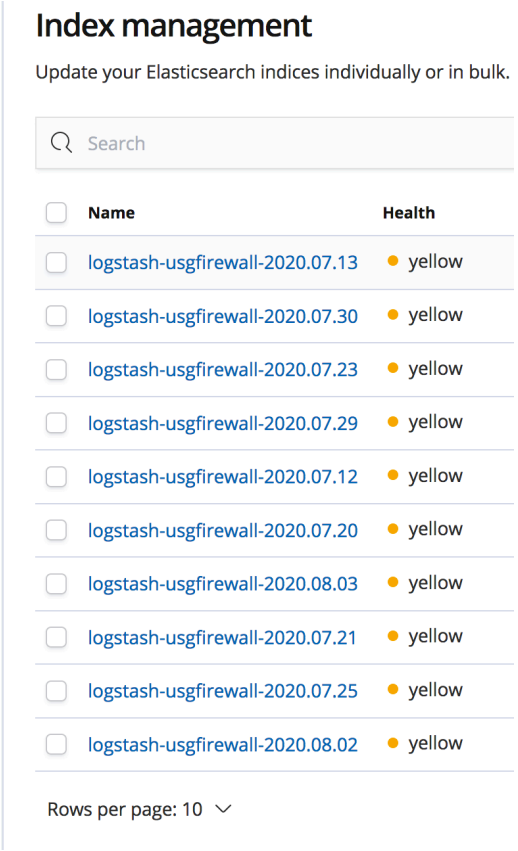

Illustration 8: Elasticsearch Indexes generated daily

3.3 Kibana Date Time Settings

During development, date time stamps from the logs displayed incorrectly as the system local date-time. Log source date-time was stored as UTC by the USG device. The research concluded that the Kibana interface would default to the user time zone's current browser setting, which may misrepresent the presented log records. The Kibana Advanced Settings for time zone was modified from 'browser' to 'UTC'.

Illustration 9: Kibana Timezone Property

Tim Coakley, timcoakley@yahoo.co.uk

3.4 Dashboard

Visual elements have been created according to the use cases documented in section 2.6. The visual creation process is trivial, given the Graphical User Interface provided using Kibana (Visualize, 2020). For brevity, procedural steps are not included; a selection of screen capture illustrates the simple process of creating unique visuals to form a single dashboard containing multiple visuals.

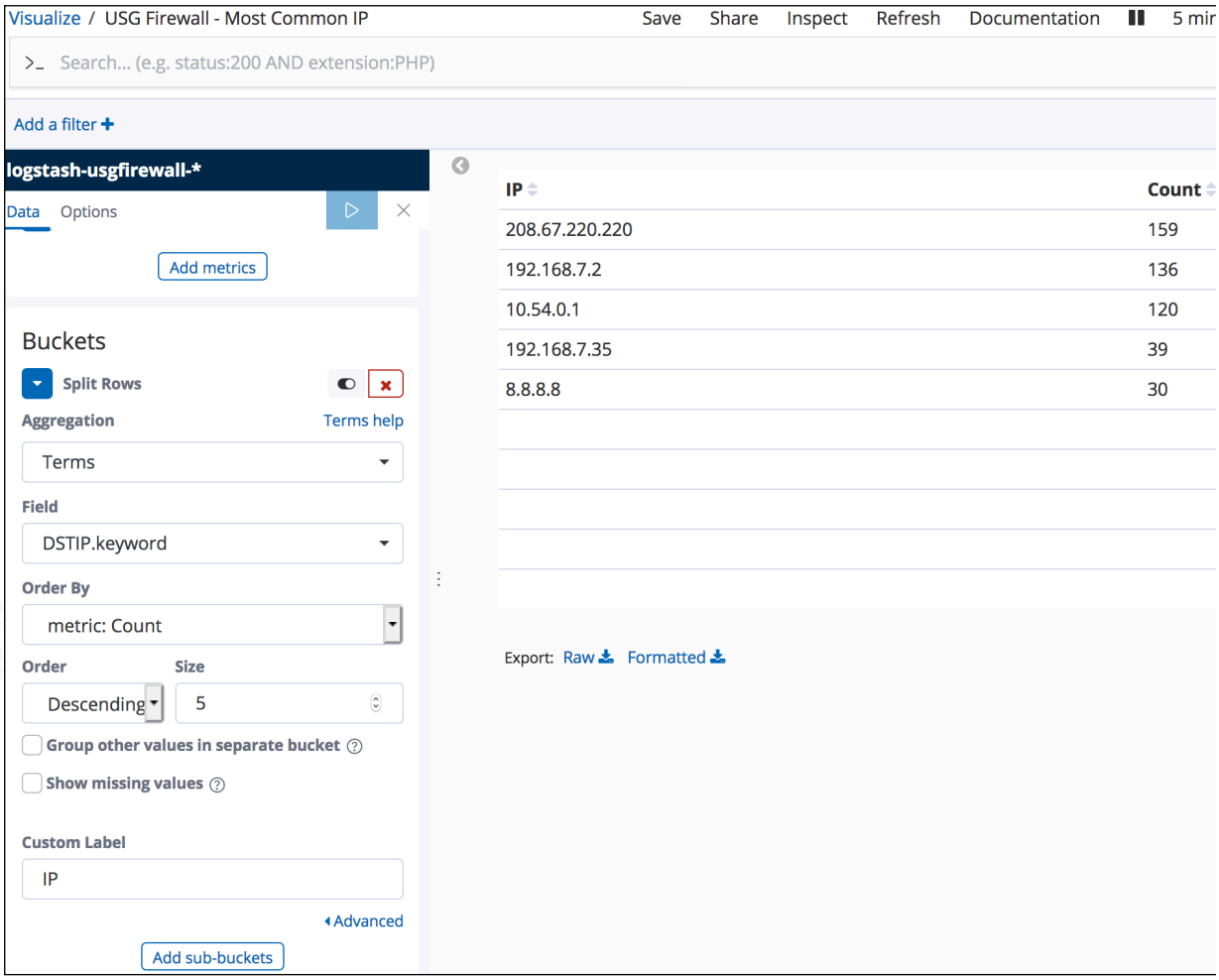

Illustration 10: Creating Visuals

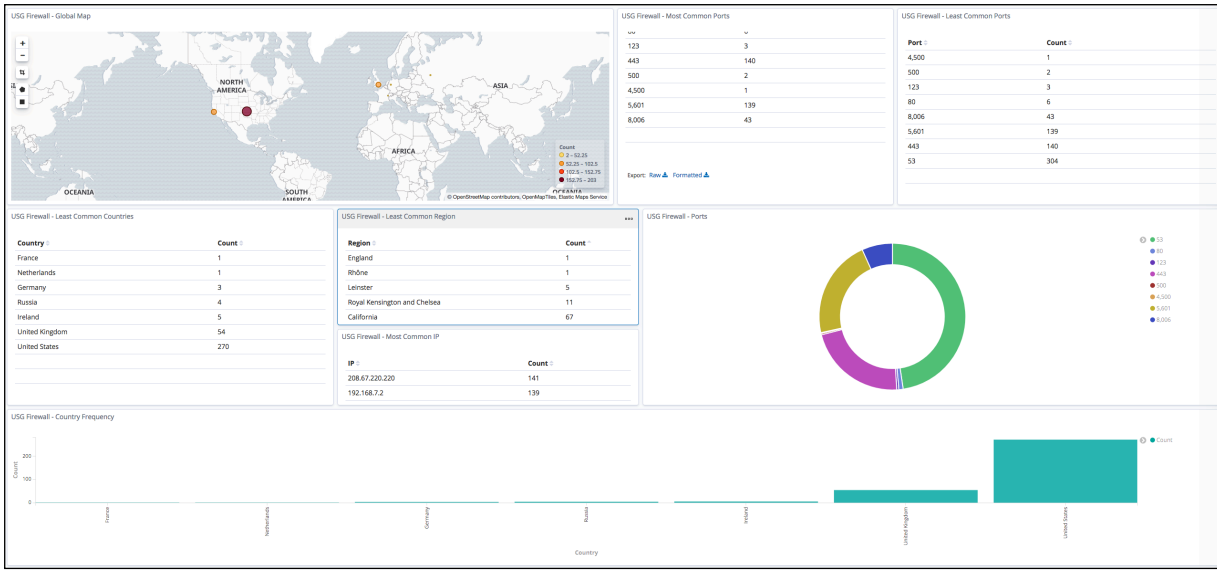

Illustration 11: Creating a Dashboard from Visuals

3.5 Enhanced Enrichment Threat Data

To further enhance the security capabilities and fulfill the proposed use cases, additional detective capability was developed. Destination addresses suspected of containing malware was maintained in a managed list and used as a lookup referred to as a threat data lookup. The threat data lookup in this research was simulated to avoid including real malicious addresses.

The software Memcached was required as a key-value pair lookup; this was an open-source distributed memory object caching service (MemCached, 2020). The software was installed onto the same platform containing the ELK technology; however, Memcached can be installed on a dedicated system for greater scalability.

Entries consisting of malicious IP address key pairs were entered into the Memcached instance before assessment for the automated comparison to function correctly.

Additional configuration of the logstash.conf file was required, the following code was included for Logstash to connect to the running Memcached instance. A get request was also needed; the get request requires the destination IP address from each log file entry as a parameter. If the destination IP address exists within the Memcached store, the ELK field malware_ip is populated with the value 'malware'.

```
memcached {
        hosts => ["127.0.0.1:11211"]
        get => {"%{DSTIP}" => "[malware_ip]"}
}
```

Confirmation of no match within the threat feed was also required, where an entry in the threat feed was not present the ELK field malware_ip is populated with the value 'no'.

```
if ![malware_ip] {
    mutate {
       add_field=> {"[malware_ip]" => "no"}
    }
}
```

The addition of the threat feeds allows any external data source to be compared against corporate network traffic. Threat feeds were found to provide additional situation context and detective capabilities. The further enrichment would give extra assurance to the business, showing resources are likely not to have been negatively impacted over a period of time. Threat feeds also allow the business to respond through prevention and incident response capabilities in a timelier manner, where detection is successful.

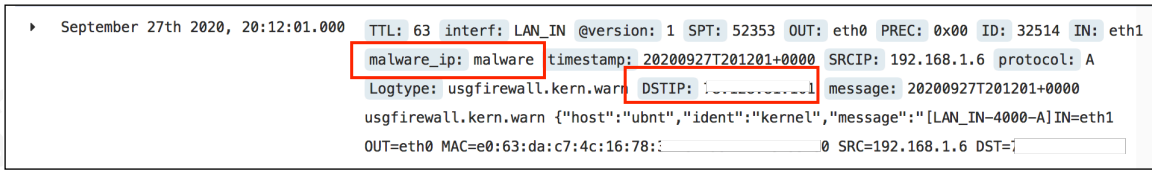

Illustration 12: Enriched Malware Log Entry marked as malware

Enriched log entries within the ELK, indicating malicious resources can be filtered and added to the dash boarding and alerting process.

4. Improvements and Recommendations

During the research and development process, several improvements and recommendations were documented and considered when implementing the proposed research.

In building the Infrastructure, the implementation should aim to make systems at least CIS Benchmarks Level 1. Benchmarks provide a suitable level of systems security hardening, dependent on business requirements (CIS, 2020).

Further enrichment of existing logging data is possible and can be implemented. Examples include but are not limited to incident indicators of compromise lookups or DNS reverse lookup to identify any domain names hosted on an IP address.

The development of more advanced use cases is recommended. This research applied to new network connections only. Existing connections should be included to identify the length of connections, essential to identify persistent remote access throughout of a given time.

Correlation of logs with other log sources is strongly recommended, for example assessing daily activity against previous activity for repeated suspicious behavior. In particular, logs should be correlated against incidents that are ongoing or incidents recently closed and still within the Recovery and Lessons Learned phases of incident response.

Alerts and Incidents generated from logs can automatically be forwarded to a suitable incident response platform. This recommendation would be as part of the automated security incident response process. Suspect events are converted to alerts/incidents and tasks automatically assigned to analysts to investigate and enrich beyond automated means.

Several log types were identified during research, many of which were not immediately used for the project goals, additional investigation is recommended. The research focused on log type usgfirewall.kern.warn only, a complete list of identified vendor undocumented log types as a result of this research is listed in Appendix D.

The timezone of log sources is recommended to be set to UTC and ensure any log review platform, including Kibana, is correctly displaying log date and time. Without this recommendation, Timestamp indexing in Kibana would provide misleading information to an analyst.

The configured 'IoT' network used for this project utilized IPv4 only; consideration should be given for devices requiring IPv6.

5. Excluded Items

The research relied upon several essential development stages, which, for brevity, are not included in detail within the paper but are instead summarized in this section.

Log collection and aggregation is a crucial precursor to any security enhancement strategy of this type. With the breadth of available logging agents available, the decision of the agent is left to the reader. This research utilized to-agent but is by no means considered the de-facto standard (Fluentd, 2020). The relevant section of the sample config file used for the chosen agent is presented in Appendix B.

Log forwarding is a crucial precursor to this research, which utilized FileBeat for log forwarding to the ELK instance (FileBeat, 2020). The reader should consider the appropriate forwarding agent for their specific use cases. A complete sample filebeat configuration file created for this project is presented in Appendix C.

ELK (Elastic Logstash Kibana) instance installation, build, and configuration is not covered in detail as part of the research. Complexity based on business needs and a variety of release version and host operating systems makes concise documentation challenging to achieve.

6. Conclusion

The combination of USG configuration supported by open source technology allows security practitioners to monitor, detect, and report suspect events based on business defined use cases.

Research into enhancing the security capabilities of the USG was successful. It highlighted other areas of research and development to expand on the findings. Significant cost savings may be made to any enterprise that chooses to build in-house security monitoring solutions.

Enrichment, including but not limited to threat feed assessments, provide the business and security management with an ongoing level of assurance. Enrichment is pivotal to improving enhanced detective capabilities and providing management with a continued sense of confidence, improvement and that security is a business enabler.

Development time may be substantial; therefore, a planned strategy and skilled workforce are required for an enterprise to create a solution that adds security value and is sustainable. One suitable approach would be to offload existing monitoring of less critical business areas were vendor costs are charged based on log throughput.

Improvement in personnel development and up-skilling may be increased where in-house development is encouraged, and business resource exists. Areas include but are not limited to system development, code development, security engineering, security rule development, and more intimate knowledge of security systems used in an enterprise.

During log analysis of the USG logs, the log type usgfirewall.kern.warn was identified as the most suitable log type generated by the USG for firewall monitoring. All log types were found to be undocumented. Analysis of the additional log types may yield additional value from a security analyst perspective.

The ELK technology stack proved to be a suitable solution for both small and large datasets limited only by the available hardware. Elasticsearch indexing was found to be more than adequate, visualization and dashboard features were found to be

satisfactory. Further work, not conducted as part of this research, is required to integrate the ELK technology stack with an automated Incident Response platform.

The USG provides a useful security feature set that is cost-effective for a broad user base. It would be possible for the vendor UniFi to incorporate some of the findings presented from this research directly into the USG device. Currently, no firewall logging data, visuals, or dashboards are introduced to a user directly from the device.

References

UniFi Security Gateway, Enterprise Gateway Router Datasheet. Retrieved July 27, 2020, from https://dl.ubnt.com/datasheets/unifi/UniFi_Security_Gateway_DS.pdf

Elastic, Grok Filters, Retrieved August 7, 2020, from https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html

GrokDebug, Online Grok Debugger. Retrieved July 27, 2020, from grokdebug.herokuapp.com/

CIS (Center for Internet Security). Retrieved July 28, 2020, from https://www.cissecurity.org/cis-benchmarks/

ELK (Elastic Logstash Kibana) Technology Stack. Retrieved July 28, 2020, from https://www.elastic.co/what-is/elk-stack

Defining alerts. Retrieved July 28, 2020, from https://www.elastic.co/guide/en/kibana/7.9/defining-alerts.html

Index patterns and fields. Retrieved July 28, 2020, from https://www.elastic.co/guide/en/kibana/7.8/managing-fields.html

Visualize. Retrieved July 28, 2020, from https://www.elastic.co/guide/en/kibana/current/visualize.html

Fluentd. Retrieved July 28, 2020, from https://www.fluentd.org/download

Filebeat. Retrieved July 28, 2020, from https://www.elastic.co/beats/filebeat

Memcached. Retrieved September 27, 2020, from https://www.memcached.org/

Tim Coakley, timcoakley@yahoo.co.uk

Appendix

A. Sample Logstash Configuration File (logstash.conf)

```
# Define the input source, filebeats using port 5044, usgfirewall for identification
input {
  beats {
    port => 5044
    type => usgfirewall
  }
}
filter {
   # Match logs specific to the USG device
   grok {
      match => { "message" =>
"^%{NOTSPACE:timestamp}%{SPACE}%{NOTSPACE:Logtype}%{SPACE}\{\"host\":\"ubnt\",\"ident\":\"kernel\",\"m
essage\":\"\[%{WORD:interf}-%{WORD:ruleindex}-%{WORD:protocol}\]IN=%{NOTSPACE:IN}
OUT=%{NOTSPACE:OUT}%{SPACE}MAC=%{NOTSPACE:MAC}%{SPACE}SRC=%{NOTSPACE:SRCIP}%{SPACE}DST=%{NOTSPACE:DSTI
P}%{SPACE}LEN=%{WORD:Len}%{SPACE}TOS=%{WORD:TOS}%{SPACE}PREC=%{WORD:PREC}%{SPACE}TTL=%{WORD:TTL}%{SPAC
E}ID=%{WORD:ID}%{SPACE}(%{NOTSPACE:PROTO}| DF
PROTO)=%{WORD:DF}%{SPACE}SPT=%{WORD:SPT}%{SPACE}DPT=%{WORD:DPT}" }
   }
   # convert text to integer, useful to sort destination port numbers
   mutate {
    convert => { "DPT" => "integer" }
   }
   # set the timestamp parsed from logs as default timestamp field
    date {
      match => ["timestamp", "yyyyMMdd'T'HHmmssZ"]
      target => "@timestamp"
    }
   # enrich the DSTIP using geoip to provide geographic information based on IP address
   geoip {
     source => "DSTIP"
    }
  # enhanced enrichment threat feed assessment
  memcached{
     hosts => ["127.0.0.1:11211"]
     get => {"%{DSTIP}" => "[malware_ip]"}
   }
  # populate field if no match is found
  if ![malware_ip] {
      mutate {
         add_field => {"[malware_ip]" => "no"}
      }
  }
  # A a tag where a log originates from ethernet interface 1, VLAN 40 (IoT Network)
  if [IN] == "eth1.40" {
    mutate {
      add_tag => ["IoT"]
    }
  }
}
# Output to elastic search index specific to USG device
```

Tim Coakley, timcoakley@yahoo.co.uk

```
output {
  if [type] == "usgfirewall" {
    elasticsearch {
      hosts => ["http://192.168.1.5:9200"]
      index => "logstash-usgfirewall-%{+YYYY.MM.dd}"
    }
  }
}
```

B. Sample Fluentd Configuration File (td-agent.conf)

```
# syslog
<source>
  @type syslog
  port 5140
  bind 0.0.0.0
  tag usgfirewall
  path /var/log/td-agent/usg_firewall
</source>
```

C. Sample Filebeats Configuration File (filebeat.yml)

```
filebeat.inputs:
- type: log
  enabled: true
  paths:
    - /var/log/td-agent/usg_firewall*.*
  exclude_files: ['.gz$']

output.logstash:
  hosts: ["192.168.1.5:5044"]
```

D. Identified LogTypes

Undocumented log types identified during this research:

1	usgfirewall.kern.warn
2	usgfirewall.daemon.info
3	usgfirewall.user.info
4	usgfirewall.daemon.notice
5	usgfirewall.kern.info
6	usgfirewall.cron.info
7	usgfirewall.authpriv.info
8	usgfirewall.daemon.warn
9	usgfirewall.daemon.err

Tim Coakley, timcoakley@yahoo.co.uk

10	usgfirewall.syslog.err
11	usgfirewall.user.notice

Table 3: Identified LogTypes