Automated Detection and Analysis using Mathematical Calculations

Lionel Teo
Automated Detection and Analysis using Mathematical Calculations

GIAC (GSEC) Gold Certification

Author: Lionel Teo, lionelteo87@gmail.com

Advisor: Dr. Johannes Ullrich

Accepted: May 5th, 2018

Abstract

A compromised system usually shows some form of anomalous behaviour. Examples include new processes, services, or outbound traffic. In an ideal environment, rules are configured to alert on such anomalies, where an analyst would perform further analysis to determine a possible compromise. However, the real-world situation is less than ideal; new processes, outbound traffic, or other anomalies often blend into legitimate activities. A large network can generate terabytes of data daily, causing the task of developing efficient detection capabilities a bit challenging. Mathematical calculations can enhance detection capability by emulating the human confidence level on assessment and analysis. Mathematical analysis can help understand the context of the event, establishing fidelity of the initial investigation automatically. By incorporating automated analysis to handle false positives, human errors and false negative can be avoided, resulting in a greater detection and monitoring capability.
1. Introduction

We have come to the age of technology where the future of cybersecurity lies in automation (Ayehu, 2016), machine learning, and artificial intelligence (AI) (Darktrace, 2017). Large networks are collecting terabytes of data daily, and cyber-attacks are continuously evolving to using more sophisticated and advanced techniques. Traditional detection capabilities on a large network can generate a large number of false positives, which there may not be sufficient resources to be able to handle every single alert thoroughly. (Violino, 2015).

An adversary can spend more than 200 days in systems before discovery; often a third of all compromises are discovered by internal processes, with the rest being discovered by external notification (Muncaster, 2015). The growing volume of security alarms generated (ESG Survey, 2017) puts the additional burden and requirement of security teams to ensure they have sufficient resources and coverage to review, respond, and validate alerts.

While it agreed that automation and machine learning is the best way to identify suspicious activity (bi-survey, 2017), the technology to facilitate this has yet to mature. The importance of applying automation and analytics for detection had been in constant discussion as recently as 2016 (Tom, Adnan, 2016). Also, unsupervised AI (an artificial intelligence with self-learning capability) has mostly seen success in the gaming industry with games such as Go (Deepmind, 2017) and Dota 2 (OpenAI, 2017). Given that the current technology landscape, both machine learning and artificial intelligence detection has yet to mature. Therefore, automation detection would be easier to implement. Once automation detection is in place, the firm can gradually move to machine learning and artificial intelligence as they mature.

2. Objective and Goals

The objective here is to automate the analysis of alerts to identify true
Automated Centralized Detection and Analysis using Mathematics Calculations

Lionel Teo, lionelteo87@gmail.com

positives from false positives. Similar to how alerts sent triggered information to active monitoring for manual assessment, rules that detect anomalies can instead send the alert to a watch list. The information gathered in the watch list is further processed by other rules. Since manual analysis usually involves pivoting searches using “asset address”, “asset owner” and “asset hostname”, the watch list would store this information for the same purpose.

To replicate the exposure checks performed by manual analysis, a different set of rules that specially designed for exposure check can retrieve the information from the watch list to further analyse the logs for unusual events. Each exposure check rule is configured to detect on a specific type of anomaly to cover the different possibilities of detection that can pick up via manual analysis.

The flow of how the rules work together described the concept of automating the extraction of logs for analysis. To complete the analysis, the rules will use mathematical calculations to help build the context required for assessment. These calculations will help to translate the event details to a confidence rating. The confidence is then added to the event before sending it to a stash for final collection. which is later used by one final rule to calculate the total confidence score for each asset. When the total confidence score hit over 90, the rule will then fire an alert to monitoring to be picked up by an analyst for further analysis.

3. Architecture Planning and Setup

Given the objective is to replicate common manual triaging steps with automation. The automation aspect of the checks are to be handled by the architecture, and the analysis aspect is to be handled using mathematics. The architecture designed had to facilitate some changes due to the difference between automation checks and manual assessment.

As previously mentioned, the alert monitoring channel is replicated using a
watch list and logs stash to keep track of information. Watch rules will help to identify possible compromised and send the asset information to the watch list for further analysis.

After adding the asset information in the watch list, other rule types which are exposure check and auxiliary rules will further process the asset information. Exposure check rules are meant to analyse the asset and identify possible anomalies, while auxiliary caters to data enrichment from other sources such as VirusTotal (VT) and internal ticketing systems. Figure 3.0.1 shows how the rules will flow and interact with each other in the architecture.

Figure 3.0.1: Automated Detection System Architecture

To replicate how an analyst would assess the results, all rules will incorporate mathematics (see section 4) to calculate the confidence level from the results. All rules part of the architecture will also send the triggered results to a logs stash, which was meant to serve as a logs collection area for later analysis. The collected logs will...

Lionel Teo, lionelteo87@gmail.com
then be processed periodically to calculate the total confidence level for each asset, which will trigger and send an alert when any asset hit over a total confidence level of over 90. To show how the concept would work, Splunk will be used to illustrate the implementation.

### 3.1 Watch Rules

Watch rules are similar to normal detection rules except they are meant to fire the alert into it's a watch list instead of the actual monitoring. Figure 3.1.1 shows an example of how a watch rule alert would work in Splunk. The screenshot below shows a simple detection to identify “new process launch” captured by the system antivirus as seen in point ①.

A new process launch on the system is a common event which can be generated by simply running an executable file. To identify if the hash is unique in the environment, a stats command is used to calculate the number of counts seen for past 30 days. The details of the mathematical calculations indicated at point ② will be covered in section 4.
3.2 Watch List

After populating the automation watch list index, the asset information will be used by other rules to pivot as searches into other logs sources to identify possible anomalies. Figure 3.1.2 shows an example of what the watch list would contain after being populated by the watch rules. The _time field represents the information added time, while the start and end time indicate the first seen and last seen the time of the event time.

![Example of a Watch List showing a list of Added Asset for Investigation.](image_url)

Figure 3.2.1: Example of a Watch List showing a list of Added Asset for Investigation.
3.3 Exposure Check Rules

To pivot the data that from the watch list back to Splunk as a search, the format command is used to construct data from the watch list into a Splunk compatible search string. Figure 3.3.1 shows an example of how the output will be with the format command. The “asset_address”, “asset_host_name” and “asset_owner” is extracted from the watch list and extracted from a search string can that can manually copy and pasted as part of another Splunk search.

![Image of Splunk search query](image_url)

**Figure 3.3.1: Constructing the search query with the format command.**

Once the format command had been tested to be working, the next step is to simply enclose the search query with square brackets “[]” and parse it to the desired search. Figure 3.3.2 shows an example of how the information from the “automation watch list” is being used to pivot into another Splunk search for domain name service (DNS) logs.
Automated Centralized Detection and Analysis using Mathematics Calculations

Lionel Teo, lionelteo87@gmail.com

Figure 3.3.2: Searching the DNS logs with the information from the watch list.

However, there are several issues with the previous method discussed which requires further consideration. Given the entries in the automation watch list had different event start (first seen) and end (last seen) time, the exposure checks search range should be based on those timestamps.

Some of the malicious activity may require more data to be generated over time before a rule can detect it accurately. Therefore, exposure check rules that ran on newly added assets might not detect anything. Lastly, there should be a way to keep track which asset information and rule that had already been extracted from the watch list and used for exposure check to prevent duplicated checks.

Due to the complexity of the requirements to extract the asset for exposure checks, the code was implemented using the advanced search function in Splunk. Figure 3.3.3 illustrates the use of Splunk advanced search function name “CONSTRUCT_SEARCH_QUERY” to process the information from the “automation_watch_list” index. The search function takes in 6 arguments in the following order: “rule_name”, “group_time”, “start_time_construction”,

© 2018 The SANS Institute
Author retains full rights.
Automated Centralized Detection and Analysis using Mathematics Calculations

“latest_time_construction”, “minimum_waiting_time”, “maximum_waiting_time” to construct the search query.

```
makeresults | append ['CONSTRUCT_SEARCH_QUERY(rule_name, 15m, start, end=3600, -5h, -16)'] | table search
```

![Figure 3.3.3: Splunk function used for constructing exposure check search query.](image)

The argument “rule_name” is used to keep track of extracted assets from “automation_watch_list” index. When the assets are extracted from the “automation_watch_list” by the function, it will also add a new action entry “exposure_check” with the rule name and the extracted information. Adding the entry will help to indicate the asset had been extracted for exposure check before. During the next runtime, the function can then check if the asset has been extracted before and therefore skip it to the next list for extraction. Figure 3.3.4 shows an example of how the “automation watch list” index would look like after a few tests run using the “CONSTRUCT_SEARCH_QUERY” search function.

![Figure 3.3.4: Illustration to show how the searches are a track on automation watch list.](image)

The next argument, “group_time”, is used for grouping events with “start” and “end” time within proximity together during extraction. For example, “15m” will group all entries with a start and end time that is within 15 minutes. This argument will help all the event that occurs at the same time to be extracted together for

Lionel Teo, lionelteo87@gmail.com
Automated Centralized Detection and Analysis using Mathematics Calculations

The “start_time_construction” and “end_time_construction” argument will help to create the Splunk timestamp search variable (“earliest” and “latest”) in epoch format. When using the “earliest” and “latest” in a search, Splunk will only search in between given “earliest” and “latest” time range. Finally, the “minimum waiting time” and “maximum waiting time” modifier is used to determine if the records inside the watch list had met the required waiting time for exposure checks. Having a waiting period will prevent newly added assets in the watch list being extracted for exposure checks. If there is any interest for the actual Splunk query used for `CONSTRUCT_SEARCH_QUERY`, the code is in the appendix section for reference.

After running the function a second time, it will construct different search information (as shown in Figure 3.35). When used in an actual Splunk alert configuration, it will cycle through “automation watch list” to extract different asset information for exposure check during each runtime.

![Code snippet](https://example.com/code_snippet.png)

**Figure 3.3.5: Illustration to show what happens when calling the function a second time.**

Using domain name algorithm exposure check rule as an example, figure 3.3.6 shows the rule using `CONSTRUCT_SEARCH_QUERY` search function to extract different assets for exposure check. As the exposure check rule continues to extract different asset entries from the automation watch list index. Eventually, the exposure check rule will extract the compromise asset address 10.0.2.15 in our example and found to be beaconing to yvcjnrx.com. The rule then calculates a
confidence level of 39.9% out of 40% using the exponential growth formula with Bayesian formula (see section 4.6 for more information for mathematical calculation on domain name generated algorithms). Similar to watch rules, the event details from the search results are then sent to the automation stash using the collect command for further analysis.

Figure 3.3.6: An example of how an exposure check rule interacts with the watch list.

3.4 Auxiliary Rules

Investigation steps often involve searching internal ticketing systems or external sites for further information. The logs collected in the “automation_stash” can be further enriched using auxiliary rules. These rules are catered to enrich log data which had been collected in the logs stash by other rules.

Figure 3.4.1 shows an example of how an auxiliary rule would enrich a hash captured in the “automation_stash” with Virustotal (VT) results. The rule uses a custom “vtlookup” command to lookup the hash from VT and retrieves the results. The hash (c8ac3d375d9780cd8d117bd5de85bfe7) used in the example had a score of 45/68 hits from VT. The score result is further calculated using the logarithmic growth...
formula (see section 4.1 for more details) to derive the confidence of 90.42%.

If the hash has a score of 0 hits on VT, the rule will instead assign negative confidence of “-30”. The negative confidence can help to identify that this activity is a false positive during the calculation of the total score. Similar to “add_watch” and “exposure_check” rules, the rule a will create new action “auxiliary” entry to the “automation_watch_list” and send the results to “automation_stash”.

### 3.5 Build Collected Data Rule

After collecting the results in “automation stash”, they had to be processed periodically to identify which event details belong to the same asset. The idea here is to replicate the manual thought process of correlating multiple anomalies to conclude the findings. Therefore each asset should only use the max confidence score from each alert. Once the build data rule had identified the events that belong to the same asset, it can then proceed to calculate the total confidence level for each asset correctly. Finally, the rule will send the processed data to the “automation_build” index, which is to later retrieve one final time during the triggering of the alert.

Also, the build data rule should also take in consideration of past triggered

---

**Figure 3.4.1: An example of how an auxiliary rule lookup VT to calculate confidence.**

<table>
<thead>
<tr>
<th>Time</th>
<th>Start</th>
<th>End</th>
<th>Asset address</th>
<th>Asset host name</th>
<th>Asset owner</th>
<th>Hits</th>
<th>Total</th>
<th>Confidence</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-02-20 20:34:54</td>
<td>1518103965</td>
<td>1518103965</td>
<td>test_machine</td>
<td>test_user</td>
<td>45</td>
<td>66</td>
<td>90.42%</td>
<td>90.42%</td>
<td>10059121436</td>
</tr>
</tbody>
</table>

© 2018 The SANS Institute
Author retains full rights.
alerts. To determine which alert should be selected, the rule will first depreciate the confidence based on the time when the alert trigger. After depreciating the confidence level, the rule can then proceed to determine which alert has the highest confidence to calculate the total confidence score.

Figure 3.5.1 shows a snippet of how the build data rule would apply confidence depreciation on Splunk. ① Due to how events related to an incident can span over the course of few days, the rule will extract up to 5 days of collected data for analysis. The auxiliary rules are not extracted for confidence depreciation as the data used is not affected by time (example: “virustotal” and internal ticketing results). ② A whitelist is also being set up here due to the build data rule being the choke point for all the rules before firing the alert. ③ The confidence depreciation is then applied based on the time difference at the runtime, with only the highest confidence selected (capped at 5000). The confidence will only start depreciation after 86400 seconds, which equates to 24 hours. Meaning that newly added rules are not affected by the depreciation. ④ The auxiliary rule events are added to the results only after depreciating the confidence.

```
| index="Automation_stash" earliest=-5d latest=-1h action=collect alert_name="auxiliary" NO: ①  
| inputlookup automation_whitelist.csv ②  
| table value  
| rename value as search  
| format ]
| eval old_confidence=confidence  
| eval time_difference=now()-epoch_time~86400  
| eval time_difference=if(time_difference<0,0,time_difference)  
| calculate TIME_DECAY(time_difference,0.604800,0,old_confidence)  
| fillnull asset_owner asset_host_name asset_address value="-"  
| dedup asset_owner asset_host_name asset_address value=sort by +confidence  
| sort confidence desc  
| head 5000 ③  
| append | search index="automation_stash" earliest=-5d latest=-1h action=collect alert_name="auxiliary"  
| sort time desc | head 5000 ④ |
```

Figure 3.5.1: An example of how to apply confidence depreciation on the collected data.

The next step of the rule is to build the data to select a key to calculate the
total confidence. Due to inconsistent asset information from different logs, it is not possible to simply calculate the total confidence without identifying the key field correctly. Figure 3.5.2 shows the inconsistency between each entry for asset information collected from different logs. Although it is possible to calculate the total confidence based on asset address, this may cause accuracy issues if an asset host was using multiple addresses. Instead, the rule should identify the appropriate key for use. The workaround is to look up the “automation_watch_list” index to join back the information and select the ideal key to calculate the total confidence.

![Figure 3.5.2: An example of collected data having inconsistent information between data.](image)

Figure 3.5.2 shows the continuation of the Splunk query which will join the asset information from “automation_watch_list”. The Splunk query will select the key field to calculate the total confidence level. ① The query would first discard any information that has less than five characters to prevent matching on incorrect data. The idea is that any data which are too short can be uncleaned data, hence matching this data can cause errors.

Next, ② the query then continue to join and retrieve back various information from “automation_watch_list”. The “join” command is run three times with similar conditions and saved into separate variables, which are asset_host_name1, asset_host_name2 and asset_host_name3.

③ After extracting the variables, the coalesce command will match the first non-null value and stored it into the “KEY” variable. This command will populate the “KEY” field if an asset_host_name field is present. Otherwise, it will try to select

Lionel Teo, lionelteo87@gmail.com
Automated Centralized Detection and Analysis using Mathematics Calculations

from the variables returned by the join command. If the “asset_host_name” is not available for selection, the query will instead select the “asset_address” or “asset_owner” as the “KEY” field.

**Figure 3.5.3**: An example query to select the Key fields for calculating total confidence.

![Example query](image)

After selecting the KEY fields to calculate the total confidence on, the query can then proceed to select highest confidence from each alert to do the calculations. Figure 3.5.4 shows the rest of the build data rule query in Splunk for summing up the confidence and sending the information to “automation_build”.

**Figure 3.5.4**: Illustration to show how the build data rule calculates the total confidence.

![Build data rule query](image)

The “eventstats” command will calculate the total confidence to a new column field name “sum_confidence”. Any events with “sum_confidence” that is less...
than 90 will be removed with a “where clause” statement. Once the search results are ready, the rule will send them to the “automation_build” index. The high confidence alert rule will then be able to use them for alerting. Figure 3.5.5 shows how the processed data will look in the “automation_build” index.

![Figure 3.5.5: An illustration to show the results from the build data rule.](image)

3.6 High Confidence Alert Rule

The high confidence alert rule will extract the data from “automation_build” index and fire the alert to active monitoring. Figure 3.6.1 shows the Splunk search query for the high alert confidence rule.

![Figure 3.6.1 Illustration of how a high confidence alert rule would work in Splunk](image)

1 The rule will first perform some filtering to omit duplicated or incorrectly processed data. The use of the “head 1” command will allow only one asset to trigger
for alerting each time. If the “KEY” had not been alerted on before, it would create an entry with the action=“alert” in the “automation_watch_list” index. Once the rule determines the asset to alert on, the format command will construct the search information based on the “KEY”. The search query will return to the main search to retrieve the necessary information for alerting. Figure 3.6.2 will shows how the result will look like when it was running. After the rule is ready, it is saved as an alert to run every 30 minutes and send an email to lionelteo87@gmail.com for further analysis if the alert were to trigger.

<table>
<thead>
<tr>
<th>end</th>
<th>asset_address</th>
<th>asset_host_name</th>
<th>asset_user_name</th>
<th>alert_name</th>
<th>confidence</th>
<th>total_confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1519129210</td>
<td>10.0.2.15</td>
<td>test_machine</td>
<td></td>
<td>AUXILIARY_VT_LOOKUP</td>
<td>90.42</td>
<td>199.7187</td>
</tr>
<tr>
<td>1519129210</td>
<td>10.0.2.15</td>
<td></td>
<td></td>
<td>EXPOSURE_CHECK_DGA_DETECTED</td>
<td>40</td>
<td>199.7187</td>
</tr>
<tr>
<td>1519129412</td>
<td>10.0.2.15</td>
<td></td>
<td></td>
<td>WATCH_RULE_HITS_ON_THREAT_INTEL_INDICATOR</td>
<td>0</td>
<td>199.7187</td>
</tr>
<tr>
<td>1519129350</td>
<td>10.0.2.15</td>
<td>test_machine</td>
<td></td>
<td>WATCH_RULE_AV_NEW_PROCESS_LAUNCH</td>
<td>68.2897</td>
<td>199.7187</td>
</tr>
</tbody>
</table>

**Figure 3.6.2 Results from a high confidence alert rule.**

**Figure 3.6.3 Alert settings to run the query and send email to monitoring recipient.**

4. Mathematical Use Cases and Applicability

After having a working architecture to automate the searches, the analysis of the data is assessed using mathematical calculations. The idea here is to emulate how
Automated Centralized Detection and Analysis using Mathematics Calculations

Lionel Teo, lionelteo87@gmail.com

an analyst assessment of the data using mathematical functions to calculate the confidence level. Taking VirusTotal (VT) lookup as a use case to illustrate, consider a hash is having a score of 30 out of 65 hits on VT. While 30/65 hits on VT is considered as high fidelity to human interpretation, applying a standard mathematical percentage calculation would only result in a confidence rating of 46% (30/65*100). When calculating the same value using a logarithmic growth equation (illustration seen in figure 4.1.1), it would result in a confidence score of 81.9% (see table 4.0.1 for calculations).

<table>
<thead>
<tr>
<th>Percentage Calculation</th>
<th>Logarithmic Growth Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>30/65*100=46</td>
<td>constant=100/ln(65+1)</td>
</tr>
<tr>
<td></td>
<td>constant=23.8683152091</td>
</tr>
<tr>
<td></td>
<td>confidence=23.8683152091*ln(30+1)</td>
</tr>
<tr>
<td></td>
<td>confidence=81.9634890207</td>
</tr>
</tbody>
</table>

Table 4.0.1: difference in calculations for the confidence level

Since the high confidence rule requires more than 90% confidence to trigger an alert, the remaining 9% can come from the initial watch rule that picks up the program execution on the system itself. It is reasonable to have the total confidence to be able to go over 100, similar to how an analyst being very confidence when performing analysis assessment during the investigation.

Then again, why would rule detection go to the length of using mathematical calculations and not alert based on a VT hit count threshold? Mathematical calculations can help build the context as the number of VT hit count gets lower. Using a different example, if a system was running an executable which VT lookup returns the results of 4/60 hits, traditional rules may not meet the required threshold and fail to alert. A rule can instead use a logarithmic mathematical calculation to retain the calculated confidence of 39.1% (see table 4.0.2 for calculations), which can
use together with another rule to build a total confidence level.

\[
\text{constant} = \frac{100}{\ln(60+1)} \\
\text{constant} = 24.3257281308 \\
\text{confidence} = 24.3257281308 \times \ln(4+1) \\
\text{confidence} = 39.1507491013
\]

Table 4.0.2: Mathematical calculation for a file on VT with 4/60 hits

### 4.1 Logarithmic Growth

The logarithmic growth model will start off with a rapid confidence increase with each count increment at the lower range but follow by a slower growth per increment at higher count range. Figure 4.1.1 shows how the confidence level would grow with each increment of the count value. Logarithmic Growth formula is especially useful to determine a particular activity that would benefit greatly in confidence for each subsequent count. One mathematical use case example as previously mentioned would be applying this to calculate confidence level from VT lookup results.

![Logarithmic Growth Graph Example](image)

Figure 4.1.1: Logarithmic Growth Graph Example

Given that the mathematical equation is frequently by different rules to calculate confidence score, hence it is ideal to implement it as a function call. Figure 4.1.2 shows an example of the implementation using the Splunk “search macro” function. The function will take in 5 arguments. The first argument would be the “variable to measure”, while the next two arguments are “minimum value” and

Lionel Teo, lionelteo87@gmail.com
“maximum value” of the measured variable. The last two arguments are the “minimum” and “maximum” confidence. Calling the function will store the calculated results in the variable “confidence”.

![Figure 4.1.2a Link to advanced search as seen from Splunk UI.](image)

**Figure 4.1.2a** Link to advanced search as seen from Splunk UI.

![Figure 4.1.2b Logarithmic Growth formula implementation using Splunk.](image)

**Figure 4.1.2b** Logarithmic Growth formula implementation using Splunk.

The saved “search macro” can be used as part of any Splunk search query. Figure 4.1.3 shows an example of using the “search macro” to calculate the confidence level using the results from a VT hash lookup script. The script returns positives of 45 out of 68 hits, which calculated with a confidence level of 90.21% using the logarithmic growth mathematical formula.
Automated Centralized Detection and Analysis using Mathematics Calculations

Lionel Teo, lionelteo87@gmail.com

4.2 Exponential Growth

This mathematical calculation is useful for measurements on variables which the confidence level starts as very low initially, but growth increases exponentially after reaching beyond a threshold value. Figure 4.2.1 shows an example of how the confidence level increases with each incremental count. This mathematical calculation is for activity that will consider high confidence only when the measured variable value is high enough. A few examples are excessing beaconing traffic or the randomness of domain names that are generated using domain generated algorithm (DGA).

Figure 4.2.1: Exponential Growth Graph Example

The mathematical formula can be added to Splunk similarly to how the previous formula was implemented using the advanced search function. Figure 4.2.2 shows the implementation of the mathematical formula in Splunk. The search function will take in the same set of arguments (variable to measure, minimum value, maximum value, minimum confidence, maximum confidence), and will calculate the confidence level using the exponential growth formula when called.
Using DNS DGA detection as an example, figure 4.2.3 shows how the formula calculates a 65% confidence score using the exponential growth formula. The rule uses the Bayesian formula (see section 4.6 for ut_bayesian) to identify and counts the number of potential bad domains that occur in a short period. When the same logic applies to traffic going to only Twitter and Google, the formula calculates a much significant confidence level of 1.5%.
4.3 Exponential Decay

This formula is the opposite of exponential growth, except that the confidence decreases as the variables measured count increases. This mathematical formula (provided in figure 4.3.2) is best used to measure if a particular process or service is a unique occurrence on the system. Activity captured for the first time will start with an initial high confidence level, but the confidence will gradually decrease drastically for each occurrence that has occurred in the past.

![Exponential Decay Graph Example](image)

**Figure 4.3.1: Exponential Decay Graph Example**

The exponential decay formula is useful for counting the number of occurrences seen in the past, with each subsequent occurrence resulting in a lower confidence value. When the occurrence reaches beyond a certain threshold, it effectively provided an automated whitelist effect. Although repeated occurrences resulted in a calculated lower confidence score, the subsequent exposure check and
auxiliary rules can still detect the activity as malicious.

Figure 4.3.3 shows an example of using the exponential decay formula to measure how some occurrences of the hash seen in the environment for the past 30 days. The max confidence argument of 33 percent is provided to the function. Since the hash is the only was seen only one in past 30 days, hence the formula returns the max confidence of 33%.

Figure 4.3.3: An example of an alert using exponential decay calculation.

To further show how the confidence would decrease as the number of count increases, a further test is done with the file “count.csv” with numbers from 1 to 10 using the mathematical formula (shown in figure 4.3.4).

Figure 4.3.4: Confidence level decreases gradually as count increases with exponential decay.
4.4 Time Decay

Time decay is the opposite of logarithmic growth. Confidence level decreases much slowly over time (graph illustrated in 4.4.1 with implementation in 4.4.2). This mathematical use case can help to build the context to assess if there is a massive production update on the network, providing a confidence rating based on the number of occurrence of distinct host seen.

Figure 4.4.1: Time Decay Graph Example

Figure 4.4.2: Implementation of the time decay calculation in Splunk.

Figure 4.4.3 shows an example of how the time decay is used to measure if the event is an occurrence is unique on a single host. The stats distinct count command “stats dc(src_host_name)” to count how many hosts were captured with the process of this file by hash. The mathematical calculation was supply with arguments
calculating the constant using the maximum measurement of 6 distinct hosts, with confidence level range from the minimum negative 33% to maximum positive 33%.

Due to this being the only occurrence, the confidence will be calculated with the maximum score of 33% as provided by the arguments. The confidence decreases gradually as the distinct occurrence increases. Therefore, any distinct count more than 6 will start to return negative confidence. Figure 4.3.4 illustrates what the confidence level would be as the count value increases with each increment by 1.

**Figure 4.4.3:** Time decay used to measure unique occurrence on the host.

**Figure 4.4.4:** Illustration to show the confidence level relationship with the count variable.
4.5 Adjusting Confidence Levels Using Static Values

Static values are used to make a slight adjustment to the overall confidence. For example, a verified change ticket or a clean file reported by VT can be assigned with a negative value to lower the overall confidence slightly. Having a negative value will only make the asset less prone to reaching the threshold when calculating the total confidence, but still, keep the possibility that the asset can trigger an alert if other rules pick up the asset exhibiting multiple anomalies. See section 3.4 for more reference on the negative confidence implemented in the suggested “virustotal lookup” auxiliary rule.

4.6 Other Mathematical Use Cases

Other additional mathematical use cases should actively consider implementing for detection. Some of the mathematical functions seen in previous illustrations came from a Splunk application know as Url Toolbox. While the application features are to parse URLs and complicated top-level domains (TLDs), it also included a series of useful mathematical functions such as Shannon entropy, counting, suites, meaning ratio and Bayesian analysis (Cedric, 2016). Figure 4.6.1 shows a list of added functions added by the Url Toolbox application.

```
ut_bayesian(1)
lookup ut_bayesian lookup world as $word$ word

ut_countert(2)
lookup ut_countertlookup world as $word$ set as $set1$ word as $set2$ word

ut_levenstem(2)
lookup ut_levenstem lookup world as $word$ set as $set1$ word as $set2$ word

ut_meaning(1)
lookup ut_meaning lookup word as $word$ word

ut_parse(2)
lookup ut_parse extendedurl $url$ $list$

ut_parse_extended(2)
lookup ut_parse_extended lookup url as $url$ list as $list$ url $url$

ut_parse_simple(1)
lookup ut_parse_simple lookup url as $url$ url

ut_shannon(1)
lookup ut_shannon lookup world as $word$ word

ut_suites(2)
lookup ut_suites lookup words as $word$ set as $set1$ $word$ $set2$ word $suites$ word $set$
```

**Figure 4.6.1: A list of functions added by URL Toolbox.**

The most notable function here would be “ut_shannon”, which is based on the Shannon Entropy formula invented by Claude Shannon. The Shannon Entropy
algorithm can help to discover the statistical structure of a word (Tricauld, 2016). Figure 4.6.2 shows the code from the “ut_shannon.py” script for calculating the entropy score. The formula literate through the consideration of each character in a word and determine if the word is a probable output statistically, which means that the domain “aaaaaaaaaa.com” will result in a low entropy score as the formula will determine that this is a probable output. As seen in figure 4.6.3, the function would calculate a low entropy score for “sans.com”, “google.com” and “aaaaaaaaaa.com” and a high entropy score for an unlikely domain such as “akjlcshiu-zykuidaixhxic.com”. Therefore, this formula can detect domain generated algorithm callback domains (DGA) due to the randomness of the characters used for DGA domains. (Kovar, 2015)

```
def shannon(word):
    entropy = 0.0
    length = len(word)

    occ = {}
    for c in word:
        if not c in occ:
            occ[c] = 0
            occ[c] += 1
    for (k, v) in occ.items():
        p = float(v) / float(length)
        entropy -= p * math.log(p, 2) # Log base 2
    return entropy
```

Figure 4.6.2: Code used by “ut_shannon” to measure the entropy of a given word.

```
# results (4/20/18 11:00:00 AM/4-21/18 11:34:48 400 AG)
Events | Patterns | Statistics | Visualization
<table>
<thead>
<tr>
<th>Time</th>
<th>u_channon</th>
<th>word</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-04-21 11:34:48</td>
<td>5.0585625556041</td>
<td>aaaaaaaaaa.com</td>
</tr>
<tr>
<td>2018-04-21 11:34:48</td>
<td>3.1292860498736</td>
<td>abcdag.com</td>
</tr>
<tr>
<td>2018-04-21 11:34:48</td>
<td>2.73</td>
<td>sans.com</td>
</tr>
<tr>
<td>2018-04-21 11:34:48</td>
<td>2.645435244671015</td>
<td>google.com</td>
</tr>
<tr>
<td>2018-04-21 11:34:48</td>
<td>3.9647351787239047</td>
<td>akjlcshiu-zykuidaixhxic.com</td>
</tr>
</tbody>
</table>
```

Figure 4.6.3: Using the `ut_shannon` to measure the entropy of some sample domains.

Other than using “ut_shannon” to measure for DGA callback domains. There
are two other functions from URL Toolbox (“ut_bayesian” and “ut_meaning”) which can help to calculate if a domain name makes humanly readable sense.

The mathematical function “ut_meaning” uses the wordlist in “meaning.dic” to measure if a given word makes sense. Based on the comments and codes from the “ut_meaning” script itself in figure 4.6.2, the formula computes a ratio using the word length and the length of its known composing word from the dictionary to derive a score. The function would calculate a meaning ratio score from the range 0 to 1. If the given word does not match the known composing word from the dictionary, the function will calculate a low meaning ratio score close to 0.

```python
""
Module that computes a ratio between the word length and the length of its known composing word. Use the ut_meaning.py script's arguments ( --url_dictionary meaning.dic --score_range 0 1 ). This list is loaded once per batch of 50,000
events (pklzfile Internet behavior on course search commands).

- ut_bayesian: micro = soft (2 words in meaning.dic) -> ratio = 1
- ut_bayesian: micro = app (1 word in meaning.dic) -> ratio = 0.95 (len("micro")/len("microsoft"))

This is a very naive algorithm, this should be improved.
""

```python

def loadWordlist():
    f_path = os.path.join(os.path.dirname(__file__), "meaning.dic")
    f_in = open(f_path, 'r')
    WORDLIST = {
        line: f_in.readline()
        while line:
            line = line.lower().strip()
            lw = len(line)
            # Do we have a word of the same size already?
            if lw in WORDLIST:
                WORDLIST[lw] += 1
            else:
                WORDLIST[lw] = 1
            # Check required in case of duplicated words.
            if lw in WORDLIST:
                WORDLIST[lw] += 1
                WORDLIST[lw].append(re.compile(line))
            line = f_in.readline()
        f_in.close()
    return WORDLIST

Figure 4.6.4: Algorithm used by “ut_meaning” to measure the ratio of a given word.
```

The mathematical function “ut_bayesian” (based on the Bayes Theorem) which took multiple possibilities of a condition and converted it into a probability (Boone, 2016). The “ut_bayesian” command will attempt to calculate the probability if an input domain is bad using the wordlist from “bayesian_good.dic” and “bayesian_bad.dic”, as seen from the script itself in figure 4.6.3. If a domain name measured using the “ut_bayesian” The function would calculate a Bayesian score from the range “0” to “1”, “0” being a good domain and “1” being a bad domain.
Automated Centralized Detection and Analysis using Mathematics Calculations

Lionel Teo, lionelteo87@gmail.com

Figure 4.6.5: Algorithm used by “ut_bayesian” to measure the probability of the bad domain.

After understanding the applicability of the algorithm for “ut_shannon”, “ut_bayesian” and “ut_meaning”, the next step is to test the accuracy of the formula in detecting malicious domains. Figure 4.6.4 shows the result of how the three formulas scores against common and malicious domains. The malicious domains are generated from a malicious file (hash e8ac3d375d9780cd8d117bd5de85bfe7). Based on the results, “ut_bayesian” had better accuracy in identifying bad domains over among all three formulas.

Figure 4.6.6: Measuring domain names using mathematical formulas.
In addition to identifying DGA domains, URL Toolbox included another mathematical function that is useful for identifying possible spoof domains. The “ut_levenshtein” function was based on the Levenshtein distance formula invented by Vladimir Levenshtein. The algorithm compares between two strings and computes the minimum number of single-character differences (Homsom, 2017). Figure 4.6.5 shows the used of “ut_levenshtein” to compare between a good site “www.sans.org” and “www.5ans.org”. The score was then passed onto a time decay formula to calculate the score of 100 confidence.

![Figure 4.6.5: Detecting possible spoof domains using the Levenshtein distance formula](image)

5. Conclusion

With increasing volume of logs contributing to false positives, traditional monitoring would gradually face an increasing challenge to provide feasible monitoring. Organizations should push towards using methodologies and technologies that are capable of automating data analytics for detection, analysis and assessment. By incorporating mathematical calculations in traditional rules monitoring, it creates the possibility to emulate human analysis.

Automated detection and analysis would be the next frontier to handle data analytics in large networks. Automated analysis can help to review and validate the bulk of false positives generated by systems more thoroughly without human errors, expanding the coverage for the scope of monitoring effectively. The resource can be
better allocated to concentrate on priority alerts, and the efficiency of the monitoring team would overall increase to meet the business goal of reducing cyber risk.
References

Maclean, Don (2017, May 5th) The key to identifying the next big threat: Data analytics & cybersecurity
Retrieve from:

Ayehu (2016, May 3). Why automation is the key to the future of cyber security.
Retrieved from:

Darktrace (2017, Apr 17) Cyber threats are growing more serious, and artificial intelligence could be the key to security.
Retrieve from:

Bob Violino (2015, Nov 2) Security tools' effectiveness hampered by false positives
Retrieve from:

Bi-Survey (2017) Big Data Security Analytics
Retrieve from:
https://bi-survey.com/big-data-security-analytics

Muncaster, Phil (2017, Feb 24) Hackers Spend 200+ Days Inside Systems Before Discovery
Retrieve from:
ESG Survey (2017, July 13) *Security analytics and operations are becoming more difficult*

Retrieve from:

https://www.helpnetsecurity.com/2017/07/13/security-analytics-operations-difficult/

Tom Davenport, Adnan Amjad (2016, Sep 26) *The future of cybersecurity Analytics and automation are the next frontier*

Retrieve from:


Deepmind (2017) *AlphaGo*

Retrieve from:

https://deepmind.com/research/alphago/

OpenAI (2017, Aug) *Dota2: We’ve created a bot which beats the world’s top professionals at 1v1 matches of Dota 2 under standard tournament rules*

Retrieve From:

https://blog.openai.com/dota-2/


Retrieve From:


Sebastien Tricaud (2016, April 21) *When entropy meets Shannon*

Retrieve From:

https://www.splunk.com/blog/2016/04/21/when-entropy-meets-shannon.html

Ryan Kovar (2015, Oct 01) *Random Words on Entropy and DNS*

Retrieve from:

https://www.splunk.com/blog/2015/10/01/random-words-on-entropy-and-dns.html

Cedric Le Roux (2016, April) *URL Toolbox*
Automated Centralized Detection and Analysis using Mathematics Calculations

Retrieve from:

https://splunkbased.splunk.com/app/2734/

Kevin Boone (2016, May) *Bayesian statistics for dummies*

Retrieve from:

http://www.kevinboone.net/bayes.html

Stephen Hosom (2017, Jan) Detecting Phishing Attacks with Bro IDS

Retrieve from:

https://medium.com/@0xhosom/detecting-phishing-attacks-with-bro-ids-f04cf886f695
Appendix

Splunk Search Functions Code

CALCULATE_EXPONENTIAL_DECAY(5)
Arguments:
dynamic_variable_to_measure, minimum_variable_limit, maximum_variable_limit, minimum_confidence, maximum_confidence

| eval variable_limit_difference=$maximum_variable_limit$-$minimum_variable_limit$
| eval constant=ln(0.0001)/variable_limit_difference
| eval confidence=exp(constant*($dynamic_variable_to_measure$-$minimum_variable_limit$))*$maximum_confidence$
| eval confidence=if(confidence<$minimum_confidence$,$minimum_confidence$,confidence)

CALCULATE_EXPONENTIAL_GROWTH(5)
Arguments:
dynamic_variable_to_measure, minimum_variable_limit, maximum_variable_limit, minimum_confidence, maximum_confidence

| eval variable_limit_difference=$maximum_variable_limit$-$minimum_variable_limit$
| eval max_confidence=$maximum_confidence$-$minimum_confidence$
| eval constant=ln(max_confidence)/variable_limit_difference
| eval confidence=exp(constant*($dynamic_variable_to_measure$-$minimum_variable_limit$))
| eval confidence=confidence+$minimum_confidence$

CALCULATE_LOG_GROWTH(5)
Arguments:
dynamic_variable_to_measure, minimum_variable_limit, maximum_variable_limit, minimum_confidence, maximum_confidence
```
| eval variable_limit_difference=$maximum_variable_limit$+1-$minimum_variable_limit$
| eval max_confidence=$maximum_confidence$-$minimum_confidence$
| eval constant=max_confidence/ln(variable_limit_difference)
| eval confidence=constant*ln($dynamic_variable_to_measure$+1-$minimum_variable_limit$)
| eval confidence=confidence+$minimum_confidence$
```

`CALCULATE_TIME_DECAY(5)`

**Arguments:**
dynamic_variable_to_measure, minimum_variable_limit, maximum_variable_limit, minimum_confidence, maximum_confidence

```
| eval variable_limit_difference=$maximum_variable_limit$-$minimum_variable_limit$
| eval variable_to_measure=$dynamic_variable_to_measure$-$minimum_variable_limit$
| eval constant=1/-variable_limit_difference*ln((0.0001-100)/100))
| eval confidence=round(((1-exp(-constant*(variable_to_measure)))*100)+100 , 3)
| eval confidence=confidence/100 * $maximum_confidence$
| eval confidence=if(confidence<$minimum_confidence$,confidence)
```

`CONSTRUCT_SEARCH_QUERY(6)`

**Arguments:**
rule_name, time_span, earliest_search_range, latest_search_range, minimum_waiting_time, maximum_waiting_time

```
search index=automation_watch_list (action="add_watch" OR action="exposure_check")
earliest=$maximum_waiting_time$ latest=$minimum_waiting_time$
| fillnull value="-" asset_address asset_host_name asset_owner
| eval asset_host_name=split(asset_host_name,"")
```
Automated Centralized Detection and Analysis using Mathematics Calculations

Lionel Teo, lionelteo87@gmail.com

mvexpand asset_host_name
| eval asset_address=split(asset_address," ")
| mvexpand asset_address
| eval asset_owner=split(asset_owner," ")
| mvexpand asset_owner
| eval start=coalesce(start,_time,now())
| eval end=coalesce(end,_time,now())
| stats values(_time) AS _time values(alert_name) AS alert_name min(start) AS start
| max(end) AS end by asset_address asset_host_name asset_owner
| search alert_name!="$rule_name$"
| sort 0 _time start asc
| search NOT (asset_address="-" asset_host_name="-" asset_owner="-")
| bin start span="$time_span$"
| bin end span="$time_span$"
| eval search_range_diff=end-start
| where search_range_diff < 259200
| stats avg(_time) as _time values(*) as * count by start end
| eval start=$earliest_search_range$
| eval end=$latest_search_range$
| eval action="exposure_check"
| sort count desc
| head 1
| eval alert_name="$rule_name$"
| table _time start end action alert_name asset_address asset_host_name asset_owner
| collect index="automation_watch_list"
| eval search=mvappend(asset_address,asset_host_name)
| eval search=mvappend(search,asset_owner)
| mvexpand search
| eval search=if(len(search)<5,null(),search)
Lionel Teo, lionelteo87@gmail.com

| where search!="-" AND search!="0.0.0.0" AND search!="127.0.0.1"
| eval earliest=start
| eval latest=end
| stats values(search) AS search by earliest latest
| stats min(earliest) AS earliest max(latest) AS latest values(search) AS search
| format "" "" "" "" "" OR "" ""
| table search

**ASSET_INFO(3)**

**Arguments:** arg1,arg2,arg3

eval asset_address=$arg1$
| eval asset_host_name=$arg2$
| eval asset_owner=$arg3$
Disclaimer

The topic of this research is to explore the concept of automation analysis in a large environment. While the content discussed in this research may be Used as a reference for creation and modification on automated detection and analysis, the content in this white paper is an independent research and is not affiliated with any organization, company, association or firm.

The data used in this article is from my personal machine and is not affiliated with any organization, company, association or firm.
<table>
<thead>
<tr>
<th>Event Name</th>
<th>Location</th>
<th>Dates</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Atlanta 2019</td>
<td>Atlanta, GA</td>
<td>May 28, 2019 - Jun 2, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Zurich June 2019</td>
<td>Zurich, CH</td>
<td>Jun 3, 2019 - Jun 8, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Munich July 2019</td>
<td>Munich, DE</td>
<td>Jul 1, 2019 - Jul 6, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SEC450 Security Ops-Analysis Beta 1</td>
<td>Crystal City, VA</td>
<td>Jul 8, 2019 - Jul 13, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Rocky Mountain 2019</td>
<td>Denver, CO</td>
<td>Jul 15, 2019 - Jul 20, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>DFIR Summit &amp; Training 2019</td>
<td>Austin, TX</td>
<td>Jul 25, 2019 - Aug 1, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Riyadh July 2019</td>
<td>Riyadh, SA</td>
<td>Jul 28, 2019 - Aug 1, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Krakow May 2019</td>
<td>OnlinePL</td>
<td>May 27, 2019 - Jun 1, 2019</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>Books &amp; MP3s Only</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>