Global Information Assurance Certification Paper

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

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 sent 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.

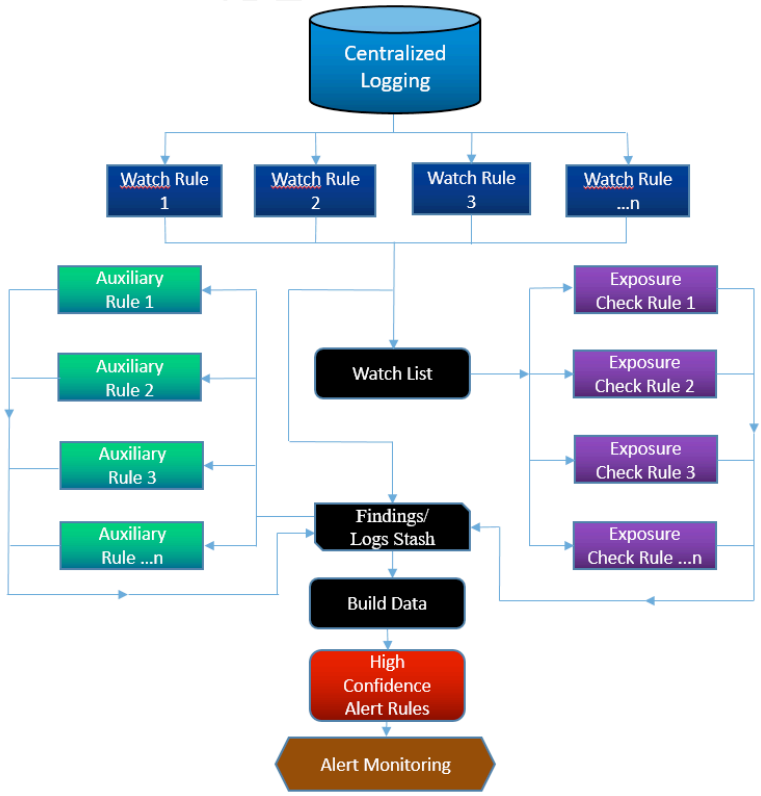


Figure 3.0.1: Automated Detection System Architecture

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

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 ②will be covered in section 4.

```
index=antivirus earliest=-30d name="a new process has been launch" `comment(rule configuration)`
| stats earliest(_time) as start latest(_time) as end dc(src_host_name) as dc_host_execution values(src_host_name)
values(src_address) values(src_user_name) latest(file_path) AS file_path count by hash_sha256
| eval cutoff=now()-7200
| where count < 50 AND end > cutoff
| rename values(*) as *
| mvexpand src_host_name
| mvexpand src_address
| mvexpand src_user_name
| `CALCULATE_EXPONENTIAL_DECAY(count,1,50,0,33)`
| rename confidence AS confidence_count
| `CALCULATE_TIME_DECAY(dc_host_execution,1,6,0,33)`
| rename confidence AS confidence_dc
| rex field=file_path "(?<file_name>\s*([^\\\\]+)$)"
| `ut_shannon(file_name)`
| `CALCULATE_EXPONENTIAL_GROWTH(ut_shannon,2,6,0,33)`
| rename confidence AS confidence_shannon
| eval confidence=confidence_count+confidence_dc+confidence_shannon
| `ASSET_INFO(src_address,src_host_name,src_user_name)`
| eval _time=now()
| eval alert_name="WATCH_RULE_AV_NEW_PROCESS_LAUNCH"
| sort confidence desc
| where confidence > 0
| head 50 `comment(send the asset information to the watch_list)`
| appendpipe
    [| eval action="add_watch"
    | table _time start end asset_address asset_host_name asset_owner action alert_name
    | collect index="automation_watch_list"]
| search NOT action="add_watch" `comment(send the events to the stash for later use)`
| eval action="collect"
| table start end src_host_name src_address src_user_name file_path asset_address asset_host_name asset_owner confidence file_path hash* *
| collect index="automation_stash"
```

20 Per Page ▾ Format Preview ▾

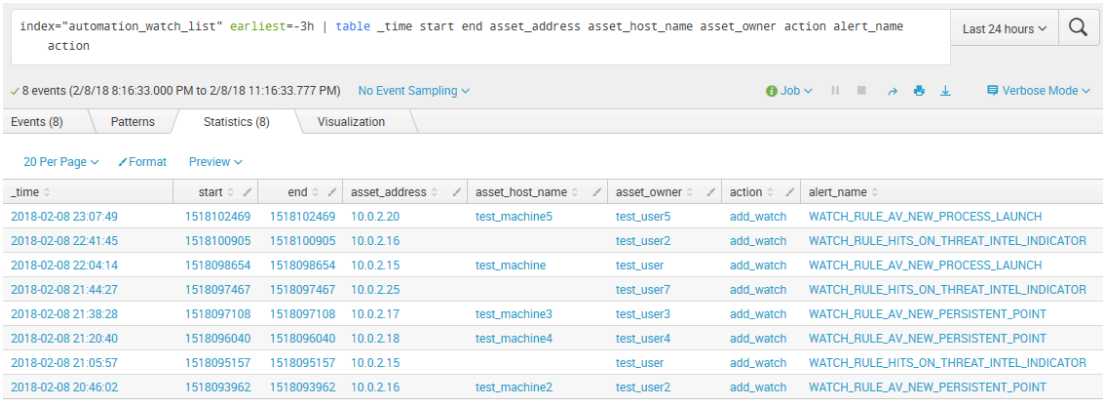
start	end	src_host_name	src_address	src_user_name	file_path	asset_address	asset_host_name	asset_owner	confidence	hash_sha256
1517748832	1517748832	test_machine	10.0.2.15	test_user	C:\Users\User\AppData\Local\Temp\prink.exe	10.0.2.15	test_machine	test_user	68.2897	b0c91214a3ed

Figure 3.1.1: Example of a Watch Rule Written using Splunk

The next half of the Splunk rule is to indicate what data should be sent to the watch list and stash. Point ③contains a self-written Splunk advanced search/function call `ASSET_INFO` (see appendix for actual code), which is to identify the information to pivot on. The function takes in an "IP address", "hostname" and "username" and stores them in asset address, asset_host_name and asset_owner. These are to standardise the fields to be used by other rules for pivoting. The last section of the code ④is to send the required fields to the "automation_watch_list" index. Finally, the copy of the trigger event details will be sent to the "automation_stash" for storage at ⑤with a calculated confidence level of 61.8% as seen in ⑥.

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.



_time	start	end	asset_address	asset_host_name	asset_owner	action	alert_name
2018-02-08 23:07:49	1518102469	1518102469	10.0.2.20	test_machine5	test_user5	add_watch	WATCH_RULE_AV_NEW_PROCESS_LAUNCH
2018-02-08 22:41:45	1518100905	1518100905	10.0.2.16		test_user2	add_watch	WATCH_RULE_HITS_ON_THREAT_INTEL_INDICATOR
2018-02-08 22:04:14	1518098654	1518098654	10.0.2.15	test_machine	test_user	add_watch	WATCH_RULE_AV_NEW_PROCESS_LAUNCH
2018-02-08 21:44:27	1518097467	1518097467	10.0.2.25		test_user7	add_watch	WATCH_RULE_HITS_ON_THREAT_INTEL_INDICATOR
2018-02-08 21:38:28	1518097108	1518097108	10.0.2.17	test_machine3	test_user3	add_watch	WATCH_RULE_AV_NEW_PERSISTENT_POINT
2018-02-08 21:20:40	1518096040	1518096040	10.0.2.18	test_machine4	test_user4	add_watch	WATCH_RULE_AV_NEW_PERSISTENT_POINT
2018-02-08 21:05:57	1518095157	1518095157	10.0.2.15		test_user	add_watch	WATCH_RULE_HITS_ON_THREAT_INTEL_INDICATOR
2018-02-08 20:46:02	1518093962	1518093962	10.0.2.16	test_machine2	test_user2	add_watch	WATCH_RULE_AV_NEW_PERSISTENT_POINT

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.

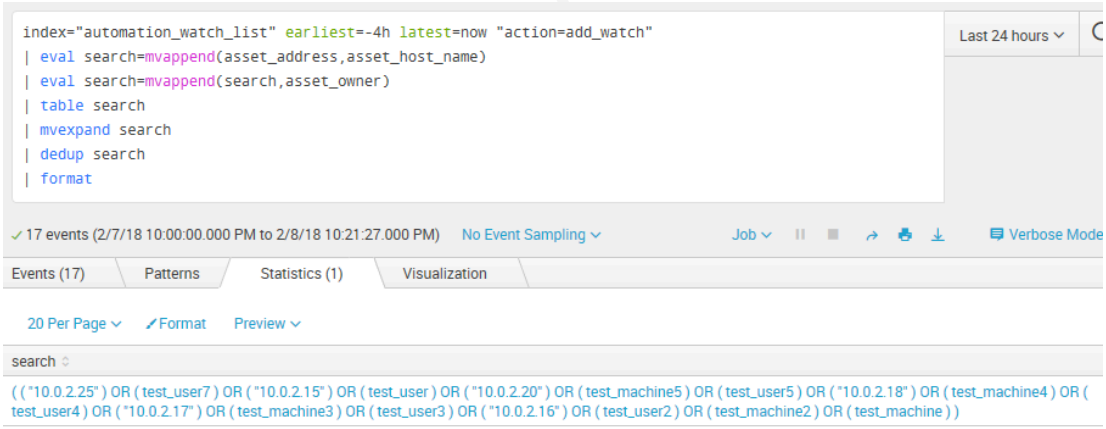


Figure 3.3.1: Constructing the search query with the format command.

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.

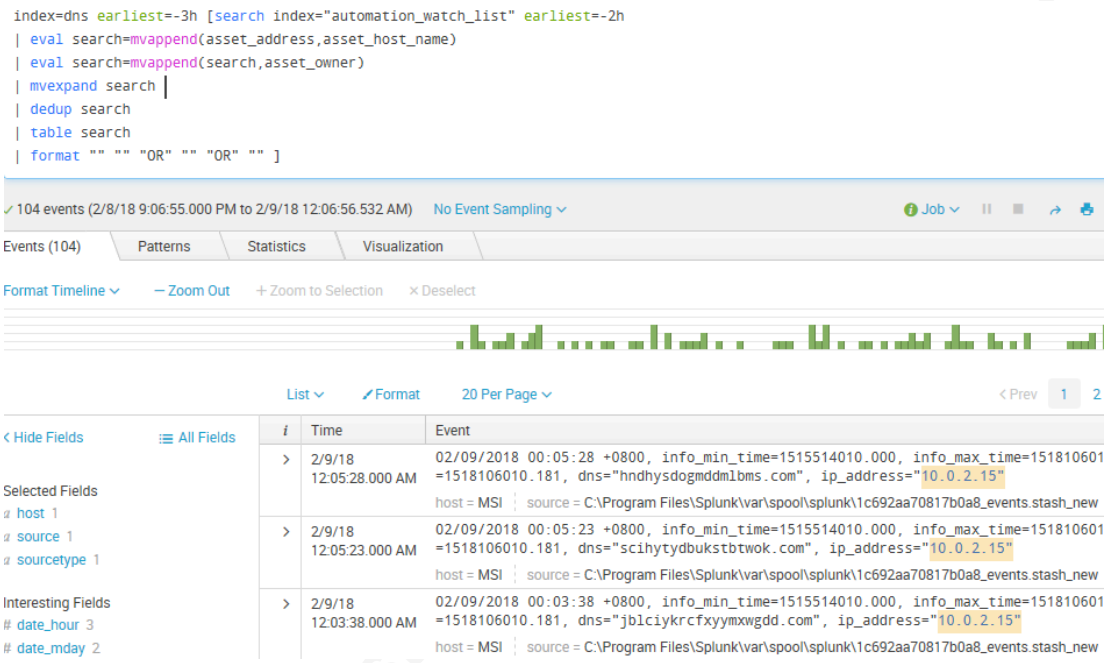


Figure 3.3.2: Searching the DNS logs with the information from the watch list.

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 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",

"latest_time_construction", "minimum_waiting_time", "maximum_waiting_time" to construct the search query.

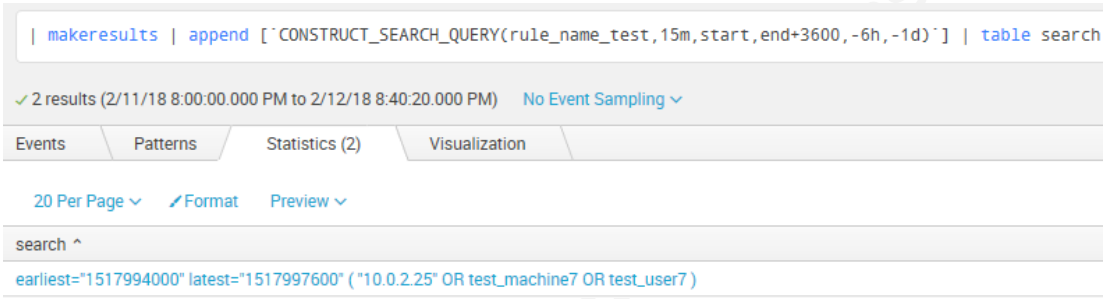


Figure 3.3.3: Splunk function used for constructing exposure check search query.

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.

_time	start	end	action	alert_name	asset_address	asset_host_name
2018-02-08 23:41:27	1518103360	1518103360	add_watch	WATCH_RULE_AV_NEW_PROCESS_LAUNCH	10.0.2.15	test_machine
2018-02-08 23:36:45	1518103360	1518103360	add_watch	WATCH_RULE_AV_NEW_PROCESS_LAUNCH	10.0.2.15	test_machine
2018-02-08 23:35:38	1517997600	1518004800	exposure_check	rule_name_test	10.0.2.15 10.0.2.20	- test_machine5
2018-02-08 23:35:38	1518001200	1518004800	exposure_check	TEST	10.0.2.17 10.0.2.18	test_machine3 test_machine4
2018-02-08 23:35:38	1517997600	1518001200	exposure_check	TEST	10.0.2.15 10.0.2.20	- test_machine5
2018-02-08 23:30:49	1518103360	1518103360	add_watch	WATCH_RULE_AV_NEW_PROCESS_LAUNCH	10.0.2.15	test_machine
2018-02-08 23:30:08	1518103360	1518103360	add_watch	WATCH_RULE_AV_NEW_PROCESS_LAUNCH	10.0.2.15	test_machine

Figure 3.3.4: Illustration to show how the searches are a track on automation watch list.

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

checking.

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.

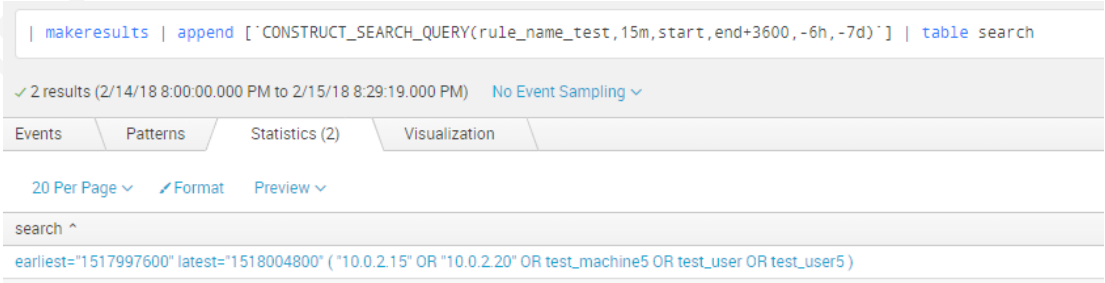
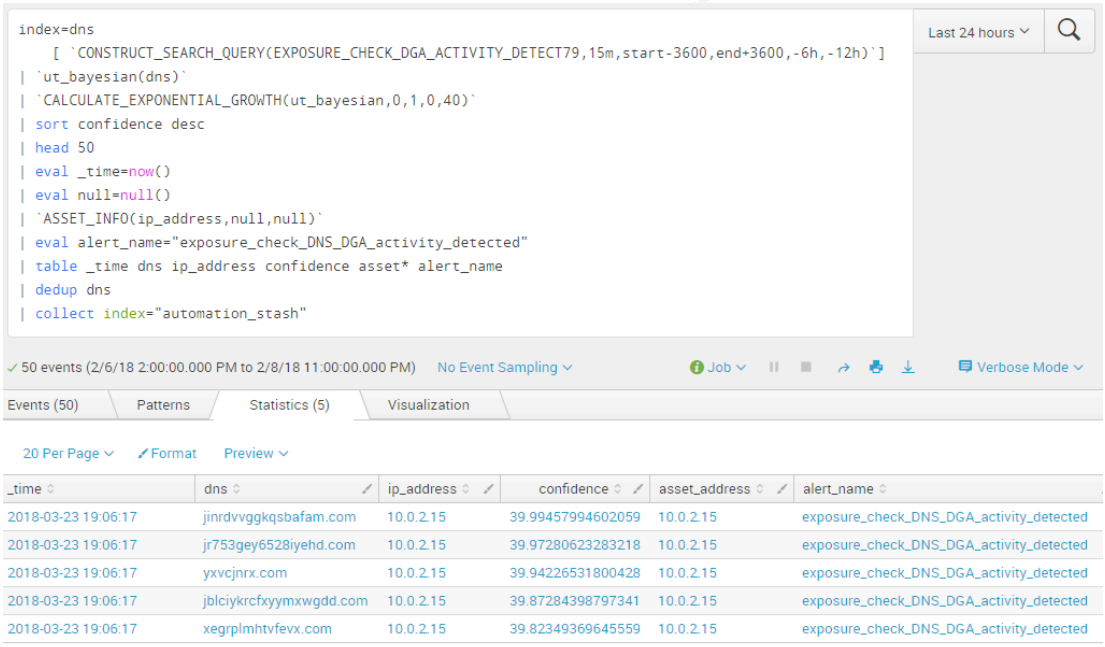


Figure 3.3.5: Illustration to show what happens when calling the function a second time.

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.



```
index=dns
    [ `CONSTRUCT_SEARCH_QUERY(EXPOSURE_CHECK_DGA_ACTIVITY_DETECT79,15m,start-3600,end+3600,-6h,-12h)`]
| `ut_bayesian(dns)`
| `CALCULATE_EXPONENTIAL_GROWTH(ut_bayesian,0,1,0,40)`
| sort confidence desc
| head 50
| eval _time=now()
| eval null=null()
| `ASSET_INFO(ip_address,null,null)`
| eval alert_name="exposure_check_DNS_DGA_activity_detected"
| table _time dns ip_address confidence asset* alert_name
| dedup dns
| collect index="automation_stash"
```

Last 24 hours

✓ 50 events (2/6/18 2:00:00.000 PM to 2/8/18 11:00:00.000 PM) No Event Sampling Job Verbose Mode

Events (50) Patterns Statistics (5) Visualization

20 Per Page Format Preview

_time	dns	ip_address	confidence	asset_address	alert_name
2018-03-23 19:06:17	jinrdvvggkqsbafam.com	10.0.2.15	39.99457994602059	10.0.2.15	exposure_check_DNS_DGA_activity_detected
2018-03-23 19:06:17	jr753gey6528iyehd.com	10.0.2.15	39.97280623283218	10.0.2.15	exposure_check_DNS_DGA_activity_detected
2018-03-23 19:06:17	yxvcjnrx.com	10.0.2.15	39.94226531800428	10.0.2.15	exposure_check_DNS_DGA_activity_detected
2018-03-23 19:06:17	jblciykrcfxyymxwgdd.com	10.0.2.15	39.87284398797341	10.0.2.15	exposure_check_DNS_DGA_activity_detected
2018-03-23 19:06:17	xegrplmhtvfevx.com	10.0.2.15	39.82349369645559	10.0.2.15	exposure_check_DNS_DGA_activity_detected

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%.

```
index="automation_stash" earliest=-1h hash!=""
| vtlookup
| `CALCULATE_LOG_GROWTH(hits,0,total,0,100)`
| eval _time=now()
| eval alert_name="AUXILIARY_VT_LOOKUP"
| eval confidence=if(hits=="0",-30,confidence)
| table _time start end asset_address asset_host_name asset_owner hits total confidence hash file_path src_host_name src_address src
| `ASSET_INFO(asset_address,asset_host_name,asset_owner)`
| appendpipe
    [ eval action="auxiliary"
    | table _time start end asset_address asset_host_name asset_owner action alert_name
    | collect index="automation_watch_list" ]
| search NOT action="auxiliary"
| collect index="automation_stash"
```

✓ 16 events (2/17/18 8:34:54.000 PM to 2/20/18 8:34:55.418 PM) No Event Sampling

Events (16) Patterns Statistics (16) Visualization

20 Per Page ✓ Format Preview

_time	start	end	asset_address	asset_host_name	asset_owner	hits	total	confidence	hash
2018-02-20 20:34:54	1518103360	1518103360	10.0.2.15	test_machine	test_user	45	68	90.42383304085706	b0c91214a3e

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

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

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* NOT  1
    [| inputlookup automation_whitelist.csv                                                  2
    | table value
    | rename value as search
    | format ]
| eval old_confidence=confidence                                                            3
| 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 sortby +confidence
| sort confidence desc
| head 5000
| append                                                                                   4
    [ search index="automation_stash" earliest=-5d latest=-1h action=collect alert_name=*auxiliary*
    | sort _time desc
    | head 5000]
```

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.

action	alert_name	asset_address	asset_host_name	asset_owner	confidence
collect	WATCH_RULE_AV_NEW_PROCESS_LAUNCH	10.0.2.15	test_machine		68.2897
collect	WATCH_RULE_HITS_ON_THREAT_INTEL_INDICATOR	10.0.2.15		test_user	0
collect	EXPOSURE_CHECK_DGA_DETECTED	10.0.2.15			40
collect	AUXILIARY_VT_LOOKUP	10.0.2.15	test_machine		90.42

Figure 3.5.2: An example of collected data having inconsistent information between data.

Figure 3.5.3 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. ①The 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, ②the 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.

③After 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

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.

```
| eval asset_host_name=if(len(asset_host_name)<5,null(),asset_host_name)
| eval asset_owner=if(len(asset_owner)<5,null(),asset_owner)
| join asset_address asset_owner type=outer
    [ search index=automation_watch_list earliest=-5d action=add_watch
    | sort _time desc
    | where isnotnull(asset_address) and isnotnull(asset_owner)
    | stats latest(asset_address) latest(asset_owner) by asset_host_name
    | rename latest(*) as * asset_host_name as asset_host_name1
    | fields asset_address asset_owner asset_host_name1]
| join asset_owner type=outer
    [ search index=automation_watch_list earliest=-5d action=add_watch
    | sort _time desc
    | where isnotnull(asset_owner)
    | stats latest(asset_owner) by asset_host_name
    | rename latest(*) as * asset_host_name as asset_host_name2
    | fields asset_owner asset_host_name2]
| join asset_address type=outer
    [ search index=automation_watch_list earliest=-5d action=add_watch
    | sort _time desc
    | where isnotnull(asset_address)
    | stats latest(asset_address) by asset_host_name
    | rename latest(*) as * asset_host_name as asset_host_name3
    | fields asset_address asset_host_name3]
| eval KEY=coalesce(asset_host_name,asset_host_name1,asset_host_name2,asset_host_name3,asset_address
    ,asset_owner)
```

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

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".

```
| dedup KEY alert_name sortby +confidence +_time
| eventstats max(_time) as time sum(confidence) as sum_confidence values(alert_name) as alert_name by KEY
| sort 0 sum_confidence > 90
| fields - _raw
| head 500
| eval _time=now()
| eval detection_time=time
| eval action=build
| collect index="automation_build"
```

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

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.

alert_name	asset_address	asset_host_name	asset_owner	confidence	KEY	sum_confidence
WATCH_RULE_AV_NEW_PROCESS_LAUNCH	10.0.2.15	test_machine		68.2897	test_machine	198.7187
WATCH_RULE_HITS_ON_THREAT_INTEL_INDICATOR	10.0.2.15		test_user	0	test_machine	198.7187
EXPOSURE_CHECK_DGA_DETECTED	10.0.2.15			40	test_machine	198.7187
AUXILIARY_VT_LOOKUP	10.0.2.15	test_machine		90.42	test_machine	198.7187

Figure 3.5.5: An illustration to show the results from the build data rule.

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.

```
index="automation_build" 3
[ search index="automation_build" earliest=-4h 1
| dedup KEY alert_name sortby +sum_confidence +_time
| where sum_confidence > 90
| eval add_watch_check=if(match(alert_name,"WATCH_RULE"),"true",null())
| stats max(sum_confidence) as confidence values(add_watch_check) as add_watch_check count by KEY
| search add_watch_check="true"
| search NOT
  [ search index="automation_watch_list" action="alert"
  | table KEY
  | dedup KEY
  | format "(" "(" "OR" ")" "OR" ")"]
| sort sum_confidence desc
| head 1
  | eval _time=now() 2
  | eval action="alert"
  | eval alert_name="AUTOMATION_ALERT_HIGH_CONFIDENCE"
  | collect index="automation_watch_list"
  | table KEY
  | format "" "" "" "" "" ""]
| dedup KEY alert_name sortby +sum_confidence +_time
| rename sum_confidence AS total_confidence
| table _time start end asset_address asset_host_name asset_user_name alert_name confidence total_confidence
```

Figure 3.6.1 Illustration of how a high confidence alert rule would work in Splunk

①The 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. ②If 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". ③The 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.

end	asset_address	asset_host_name	asset_user_name	alert_name	confidence	total_confidence
1519129210	10.0.2.15	test_machine		AUXILIARY_VT_LOOKUP	90.42	198.7187
1519129071	10.0.2.15			EXPOSURE_CHECK_DGA_DETECTED	40	198.7187
1519129412	10.0.2.15			WATCH_RULE_HITS_ON_THREAT_INTEL_INDICATOR	0	198.7187
1519129356	10.0.2.15	test_machine		WATCH_RULE_AV_NEW_PROCESS_LAUNCH	68.2897	198.7187

Figure 3.6.2 Results from a high confidence alert rule.

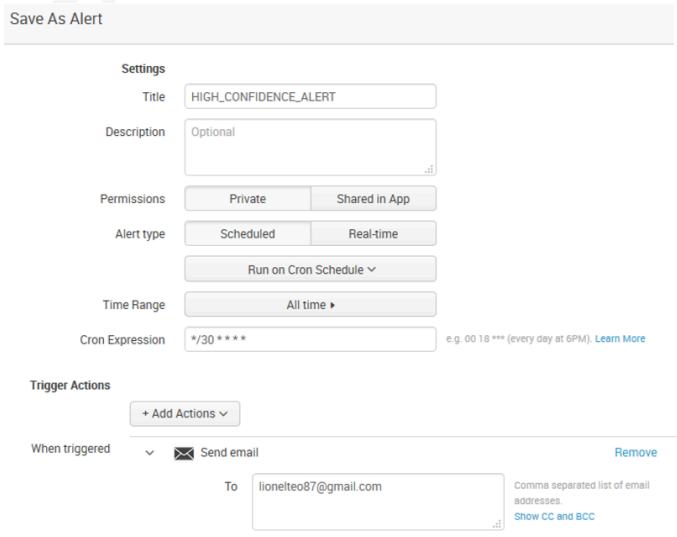


Figure 3.6.3 Alert settings to run the query and send email to monitoring recipient.

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

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).

Percentage Calculation	Logarithmic Growth Calculation
30/65*100=46	constant=100/ln(65+1) constant=23.8683152091 confidence=23.8683152091*ln(30+1) confidence=81.9634890207

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.

constant=100/ln(60+1)
constant=24.3257281308
confidence=24.3257281308*ln(4+1)
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.

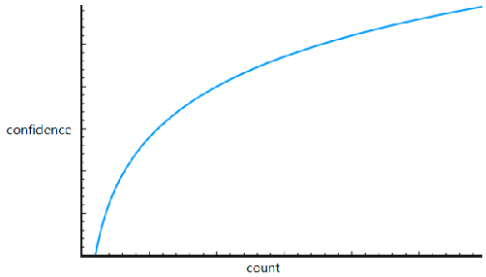


Figure 4.1.1: Logarithmic Growth Graph Example

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

"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".

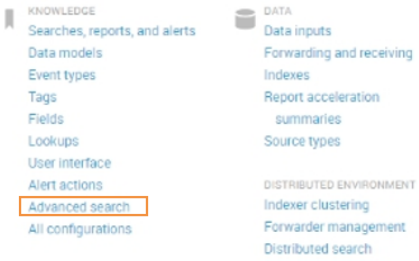


Figure 4.1.2a Link to advanced search as seen from Splunk UI.

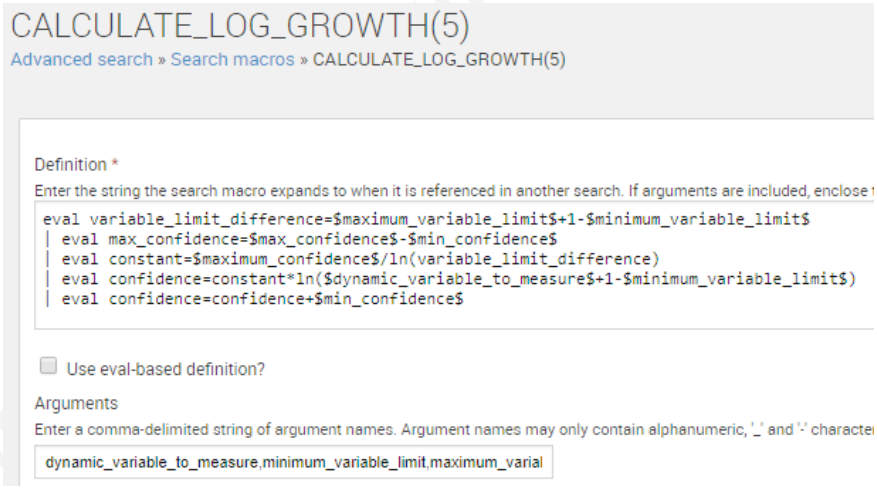


Figure 4.1.2b Logarithmic Growth formula implementation using Splunk.

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.

Figure 4.1.3 Calculating confidence level from VT results using log growth.

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).

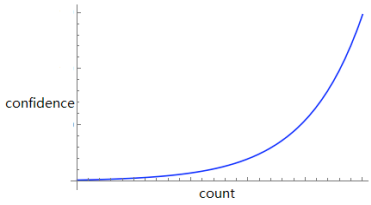


Figure 4.2.1: Exponential Growth Graph Example

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.

Lionel Teo, lionelteo87@gmail.com

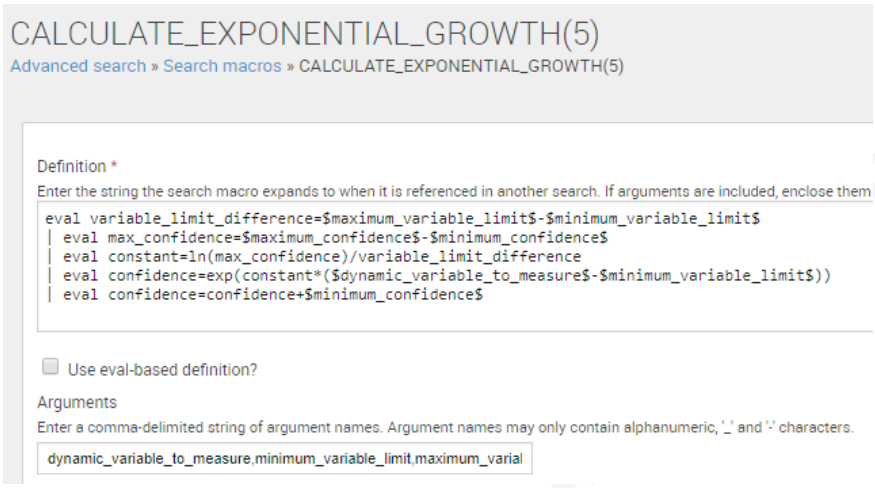


Figure 4.2.2: Exponential Growth Calculation Implemented using Splunk.

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%.

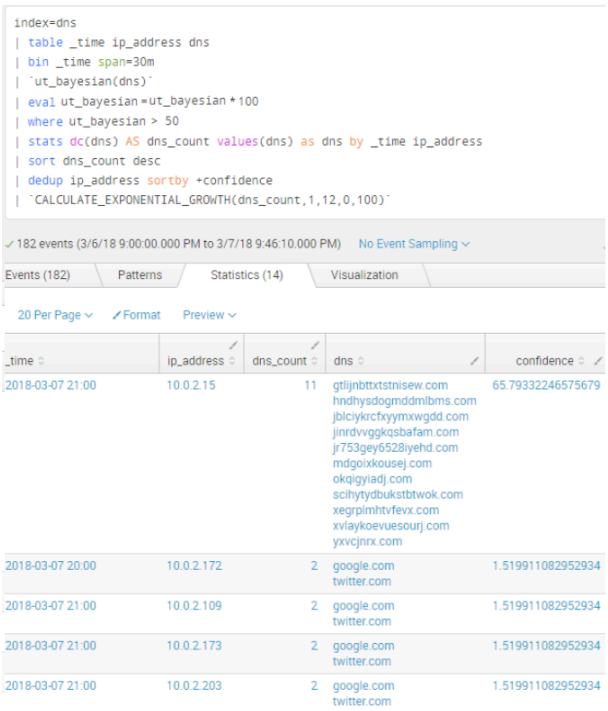


Figure 4.2.3 An example query using an exponential calculation to measure beaconing activity.

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.

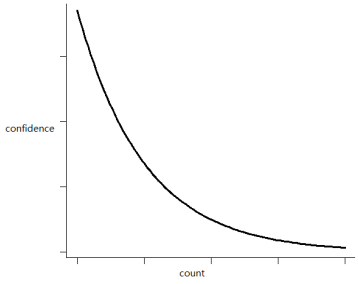


Figure 4.3.1: Exponential Decay Graph Example

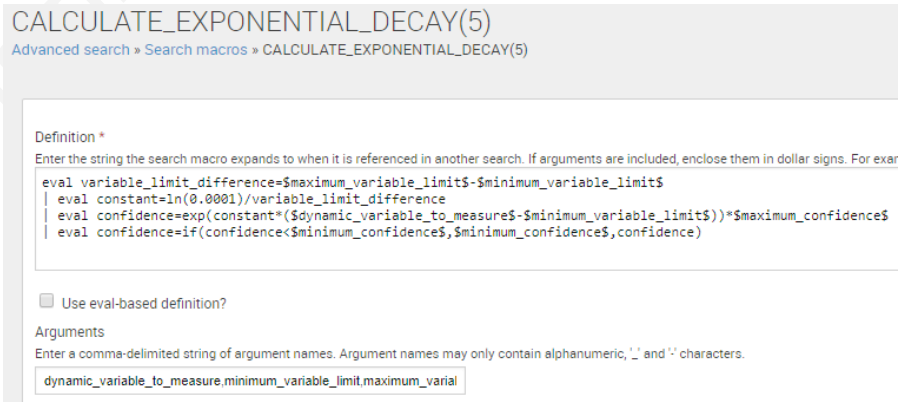


Figure 4.3.2: Exponential Decay Calculation

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%.

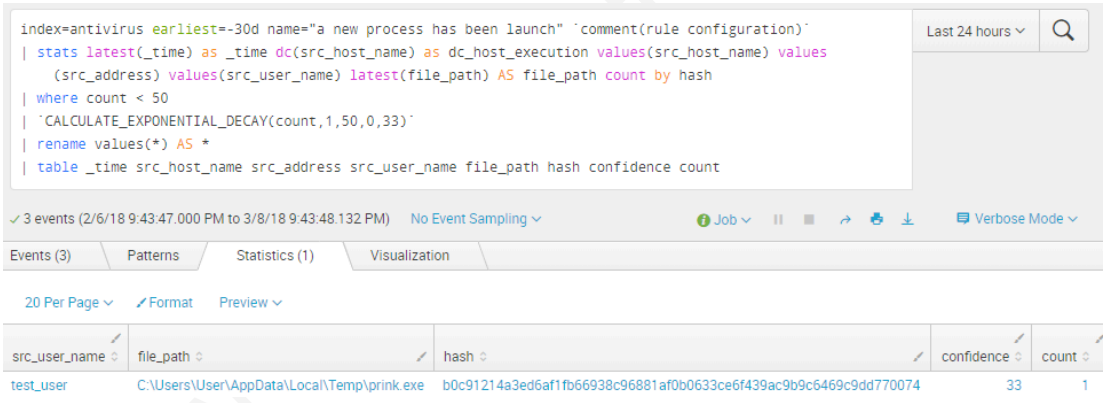


Figure 4.3.3: An example of an alert using exponential decay calculation.

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).

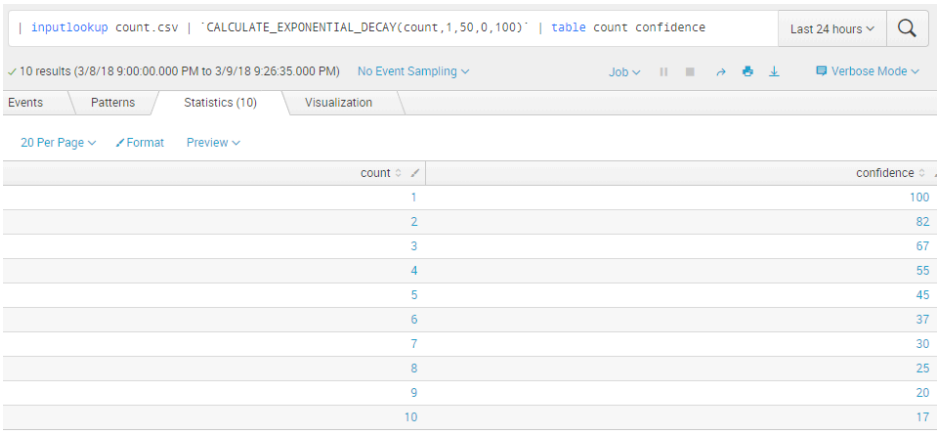


Figure 4.3.4: Confidence level decreases gradually as count increases with exponential decay.

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.

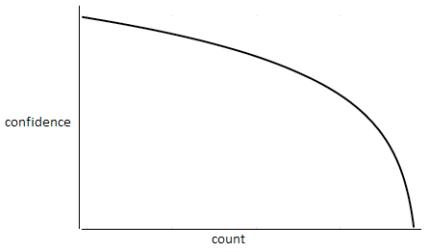


Figure 4.4.1: Time Decay Graph Example

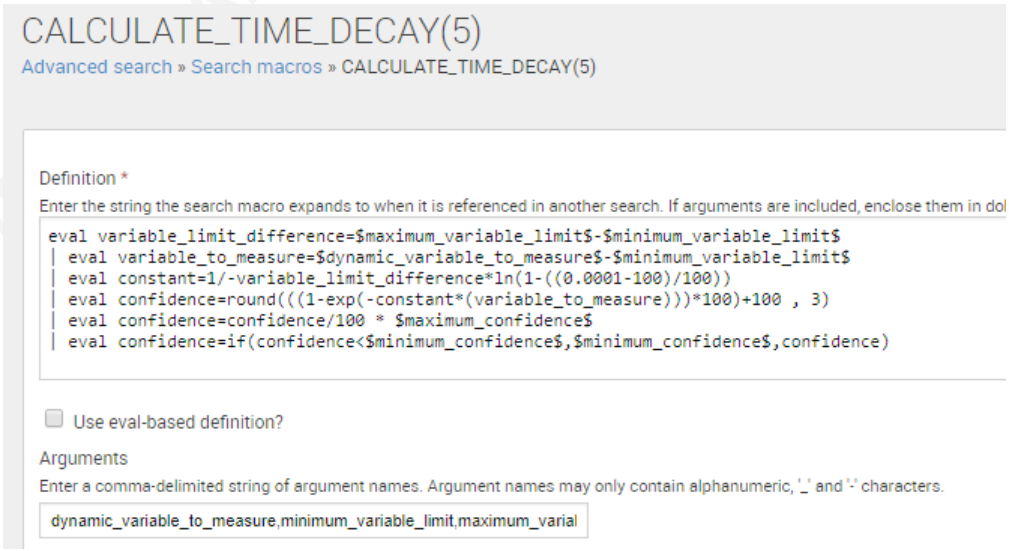


Figure 4.4.2: Implementation of the time decay calculation in Splunk.

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.

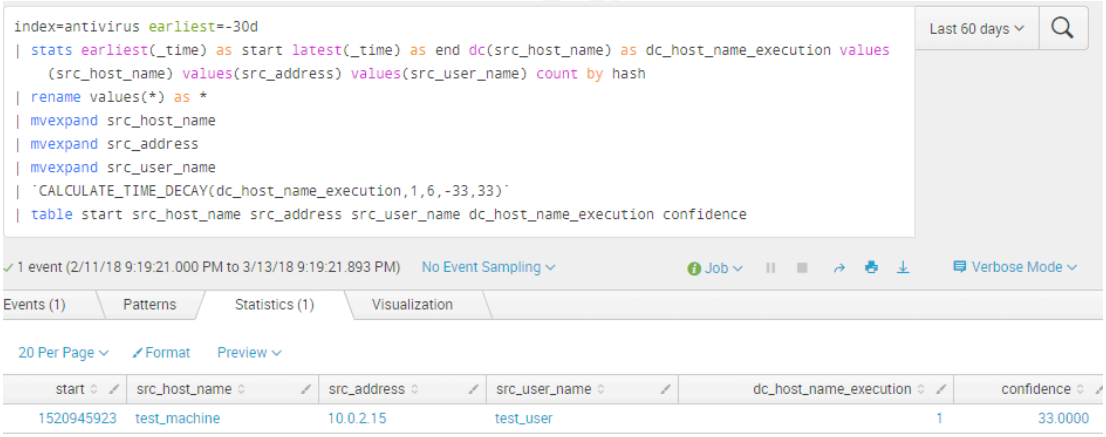


Figure 4.4.3: Time decay used to measure unique occurrence on the host.

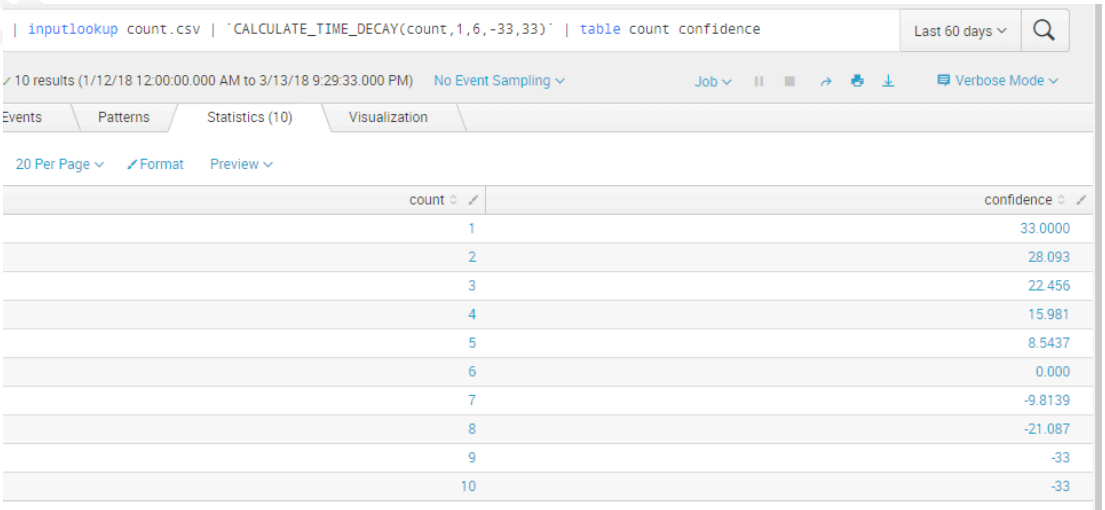


Figure 4.4.4: Illustration to show the confidence level relationship with the count variable.

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 word as $word$	word
ut_countset(2)	lookup ut_countset_lookup word as $word$ set as set	word, set
ut_levenshtein(2)	lookup ut_levenshtein_lookup word1 as $w1$ word2 as $w2$	w1, w2
ut_meaning(1)	lookup ut_meaning_lookup word as $word$	word
ut_parse(2)	`ut_parse_extended($url$, $list$)`	url, list
ut_parse_extended(2)	lookup ut_parse_extended_lookup url as url list as $list$ | spath input=ut_subdomain_parts | fields - ut_subdomain_parts	url, list
ut_parse_simple(1)	lookup ut_parse_simple_lookup url as url	url
ut_shannon(1)	lookup ut_shannon_lookup word as $word$	word
ut_suites(2)	lookup ut_suites_lookup word as $word$ set as set | spath input=ut_suites | fields - ut_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 "aaaaaaaaa.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 "aaaaaaaaa.com" and a high entropy score for an unlikely domain such as "akjlcshiu-zxy-ykiudxhzlicx.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.iteritems():
            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.

```
| makeresults | eval word="aaaaaaaaa.com"
| append[| makeresults | eval word="abcdeg.com"]
| append[| makeresults | eval word="sans.com"]
| append[| makeresults | eval word="google.com"]
| append[| makeresults| eval word="akjlcshiu-zxy-ykiudxhzlicx.com"]
| `ut_shannon(word)`
```

✓ 5 results (4/20/18 11:00:00.000 AM to 4/21/18 11:34:48.000 AM) No Event Sampling Job

Events Patterns Statistics (5) Visualization

20 Per Page ✓Format Preview

_time	ut_shannon	word
2018-04-21 11:34:48	1.505876255604107	aaaaaaaaa.com
2018-04-21 11:34:48	3.121928094887362	abcdeg.com
2018-04-21 11:34:48	2.75	sans.com
2018-04-21 11:34:48	2.6464393446710157	google.com
2018-04-21 11:34:48	3.9647351787255047	akjlcshiu-zxy-ykiudxhzlicx.com

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 ration 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.

```
"""
Module that compute a ratio between the word length and the length of it's known composing words
Use the wordlist meaning.dic (one word per line). This list is loaded once per batch of 50,000
events (Splunk Internal behavior on custom search commands).

microsoft = micro + soft (2 words in meaning.dic) => ratio = 1
microxyze = micro + xyze (1 word in meaning.dic)  => ratio = 5/9 (len('micro')/len('microsoft'))

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

def loadWordlist():
    f_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), "meaning.dic")

    f_in  = open(f_path, "r")
    WORDLIST = {}

    line = f_in.readline()
    while line:
        line = line.lower().strip()
        le = len(line)

        # do we have a words of the same size already?
        if not le in WORDLIST:
            WORDLIST[ le ] = []

        # check required in case of duplicated words.
        if not line in WORDLIST[ le ] :
            WORDLIST[ le ].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.

```
def bayescore(word, n_good, set_good, n_bad,  set_bad, gram_size=2):
    """
    Compute the naive bayesian score - Probability that the domain is a bad one knowing it's ngrams.
    Actually limited to 2-grams for now
    """

    score      = 0.0
    P_Total    = 1.0
    P_TotalInv = 1.0

    grams = ngramsplit(word, gram_size)
    for g in grams:

        if not g in set_bad or not g in set_good :
            continue

        # probability that the gram X appears in bad domains
        P_G_B = (100.0/n_bad) * set_bad[ g ]

        # probability that the gram X appears in good domains
        P_G_G = (100.0/n_good) * set_good[ g ]

        # probability that a domain is a bad one, knowing that the gram X is in it;
        P = P_G_B / (P_G_B + P_G_G)

        P_Total    *= P
        P_TotalInv *= (1 - P)

    return (P_Total/(P_Total+P_TotalInv))
```

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 c8ac3d375d9780cd8d117bd5de85bfe7). Based on the results, "ut_bayesian" had better accuracy in identifying bad domains over among all three formulas.

```
| inputlookup append=t dns.csv
| dedup dns
| table dns
| `ut_bayesian(dns)`
| `ut_meaning(dns)`
| `ut_shannon(dns)`
| `CALCULATE_EXPONENTIAL_GROWTH(ut_bayesian,0,1,0,100)`
| rename confidence as confidence_ut_bayesian
| table dns ut_shannon ut_meaning_ratio ut_bayesian confidence*
```

Last 24 hours

✓ 19 results (3/22/18 5:00:00.000 PM to 3/23/18 5:39:42.000 PM) No Event Sampling Job Verbose Mode

Events Patterns Statistics (19) Visualization

20 Per Page Format Preview

dns	ut_shannon	ut_meaning_ratio	ut_bayesian	confidence_ut_bayesian
linkedin.com	3.2516291673878226	0.5	0.3822709128824774	5.814894313371530
rgaonnkejei.com	3.5068905956085183	0.26666666666666666	0.6463756312075747	19.62236104900774
doueven.click	3.392747410448785	0.8461538461538461	0.6695641812301352	21.83375146823416
facebook.com	3.0220552088742005	0.6666666666666666	0.7121069520362239	26.55913363075081
instagram.com	3.392747410448785	0.5384615384615384	0.7535736815416274	32.14751170228141
google.com	2.6464393446710153	0.2	0.7659409297425657	34.03156015355566
youtube.com	3.0957952550009344	0.6363636363636364	0.8644443178636837	53.56592830867705
twitter.com	3.0271691184406193	0.18181818181818182	0.8695393526894205	54.83763358740396
hndhysdogmddmlbms.com	3.3446983751597124	0.23809523809523808	0.9814642869637766	91.81815756906842
scihytydbukstbtwok.com	3.6978458230844122	0.18181818181818182	0.986435853511104	93.9445752130073
okqigyiadj.com	3.521640636343319	0.35714285714285715	0.9917582283584809	96.27564940054414
gtlijnbttxtstnisew.com	3.658993415253805	0.045454545454545456	0.9946172821073376	97.55163758170251
xvlaykoevuesourj.com	3.7841837197791883	0.3	0.9971215789433041	98.68318499713841
mdgoixkousej.com	3.577819531114783	0.375	0.9985469801055674	99.33309336945180

Figure 4.6.6: Measuring domain names using mathematical formulas.

Lionel Teo, lionelteo87@gmail.com

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 4.6.7: Detecting possible spoof domains using the Levenshtein distance formula

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:

http://mil-embedded.com/guest-blogs/the-key-to-identifying-the-next-big-threat-data-analytics-cybersecurity/

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

Retrieved from:

https://www.networkworld.com/article/3065296/security/why-automation-is-the-key-to-the-future-of-cyber-security.html

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

Retrieve from:

https://www.cnbc.com/2017/04/17/darktrace-on-why-artificial-intelligence-is-key-in-cybersecurity.html

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

Retrieve from:

https://www.csoonline.com/article/2998839/data-protection/security-tools-effectiveness-hampered-by-false-positives.html

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:

https://www.infosecurity-magazine.com/news/hackers-spend-over-200-days-inside/

Lionel Teo, lionelteo87@gmail.com

ESG Survey (2017, July 13) *Security analytics and operations are becoming more difficult*

Retrieve from:

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

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

Retrieve from:

https://dupress.deloitte.com/dup-us-en/topics/analytics/future-of-cybersecurity-in-analytics-automation.html

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/

NIST (2006) *Guide to Computer Security Log Managemen*t

Retrieve From:

http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-92.pdf

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*

Lionel Teo, lionelteo87@gmail.com

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

Lionel Teo, lionelteo87@gmail.com

```
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(1-((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<$minimum_confidence$,$minimum_confidence$,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," ")
```

```
| 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)
```

```
| 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.