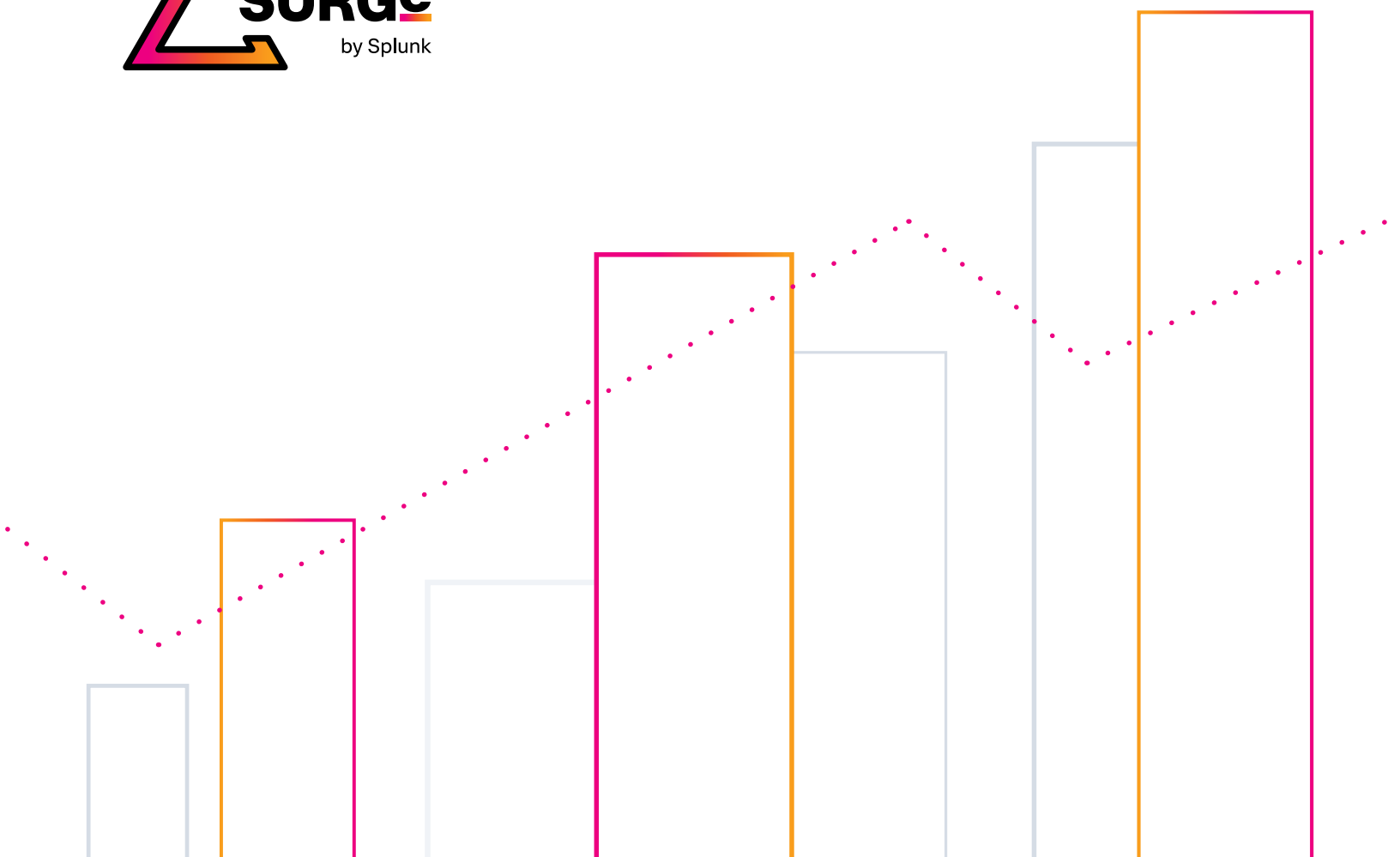
Detecting Supply Chain Attacks

Using Splunk and JA3/s hashes to detect malicious activity on critical servers

By Marcus LaFerrera and Ryan Kovar

SURGe by Splunk



Executive Summary

Attacks like SolarWinds[1] have shown that organizations have difficulty detecting when their internal appliances begin communicating to new external (possibly malicious) hosts. This lack of visibility contributes to the dreaded “supply chain compromise.” This paper provides a method for assisting with that problem by using network data, statistics, and JA3/JA3s[2] hashes powered by Zeek[3] and Splunk. Our primary goal throughout this research was to provide network defenders an added advantage in detecting malicious activity that would otherwise go undetected.

In this paper, we will go over a handful of methods that can be used to help detect malicious activity on critical servers. The primary audience of this white paper is for technical practitioners but CISOs and leaders will find value in the introductions and conclusions. We will explore time tested queries, such as identifying first seen and rarest data points. Additionally, we will walk through using anomaly detection techniques and identifying potentially malicious processes. In most cases, the findings we present will be most useful for network defenders looking for novel methods and techniques to add to their supply chain attack detection toolbox.

None of the findings in this paper will prove to be a silver bullet in detecting software supply chain compromises. Still, it will help increase the speed of detection and thus increase the cost of adversarial operations (when a fancy bear chases you, you don’t have to be the fastest, just faster than your peers). By using commonly found and easily configured tools like Zeek combined with queries that have a low barrier to entry, we hope that security professionals, from junior SOC analysts to grizzled threat hunters, will find quick value. In essence, we sought to drastically reduce the size of the proverbial haystack to minimize the effort required to find the ever-elusive needle.

Introduction

Many software products are designed with so-called "phone home" features to support automatic updates, content subscriptions, or data upload. These same products are often deployed in privileged locations within the customers' cloud and on-prem environments. For example, Solarwinds Orion is usually granted carte blanche from a network perspective to support its primary use case: network monitoring. Likewise, Codecov, a popular code coverage tool, is embedded into continuous integration pipelines with unfettered access to source code, passwords, API keys, certificate signing keys, etc. It is not a standard practice for software developers to publish lists of IP addresses, domain names, or certificates representing legitimate destinations for phone home traffic. While some vendors provide such information, it is just as common for customers to need to press for it, and the data itself often changes. Attackers have recognized this combination of factors, and they are actively exploiting it. In this type of attack, the adversary infiltrates the developers' systems and modifies their product to redirect customers' sensitive data to a different location under the guise of legitimate "phone home" functionality.

Vendors are responsible for protecting their systems that comprise the software supply chain, but what can end customers do to detect malicious phone home activity? Experts often advise organizations to create baselines of regular network activity and then alert when deviations are observed. This is easier said than done. IP addresses often change and can be re-allocated within minutes, often to different customers.

A potential solution to this problem would be to leverage a higher fidelity data point to detect anomalous activity. At the onset of our research, we purposefully defined very narrow goals and limitations to ensure our results were usable for most readers with little to no configuration or infrastructure modifications. In short, we sought to enable network defenders today, rather than tomorrow when it may be too late. One higher fidelity data point that is commonly collected and widely supported are JA3 and JA3s hashes. Collectively, we will refer to them as JA3/s. This paper will leverage JA3/s hashes as this higher fidelity data point and showcase core Splunk capabilities to bring anomalous activity close to the forefront.

1. https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html
2. https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-24736285596
3. https://zeek.org/

What is JA3/s?

JA3[4] is an open-source methodology that allows for creating an MD5 hash of specific values found in the SSL/TLS handshake process. Key attributes from the client's handshake request are extracted from the session, concatenated, then hashed with the MD5 algorithm. Specifically, the attributes extracted from the client-side of the session are:

```
SSLVersion,Cipher,SSLExtension,EllipticCurve,
EllipticCurvePointFormat
```

By joining these values and then hashing the result, one can generate a consistent hash of specific clients and the libraries/binary making the request. For example, using this methodology on a session captured from a Trickbot compromise, the Trickbot binary's JA3 hash was `6734f37431670b3ab4292b8f60f29984`. This hash would be consistent across all SSL/TLS sessions originating from that binary, regardless of source and destination IP address. While changing IP addresses and domain names is relatively easy for adversaries, modifying their malware to use different SSL/TLS libraries is not. In this way, JA3 monitoring increases the overall cost for the adversary to hide the network connectivity.

Additionally, there is a similar methodology for calculating the JA3 hash of a server session. This is aptly named JA3s. The process is identical to generating a JA3 hash; however, the key attributes extracted from the server's session are slightly different. This is because servers may respond differently, depending on the request sent by the client. The details extracted from the server-side of the session are:

```
SSLVersion,Cipher,SSLExtension
```

Both JA3 and JA3s are easily obtained from network traffic using various tools. For the purpose of this research, the tool leveraged for hash generation has been limited to Zeek.

4. https://github.com/salesforce/ja3
5. Several Splunk customers were very generous in helping generate these queries using real-world data. Without their help, our research would have been far more difficult.
6. https://github.com/mlaferrera/SEC1745/code

Detections

As part of this research, we've developed several methodologies to detect abnormal activity. Our goal was to ensure the developed queries are simple to leverage with little to no required components outside Splunk Enterprise. As a result, there are several caveats and limitations that should be highlighted.

Caveats and Limitations

There are no silver bullets in detecting supply chain attacks, nor in detecting malicious activity in general. Our goal has always been to help bring anomalous activity as close to the forefront as possible with the available tools. In our testing using real-world enterprise data[5], along with data generated[6] from our testing environments, the results showed it is highly probable anomalous activity can be detected via abnormal JA3/s hashes. However, your mileage may vary depending on many factors. In all likelihood, an allow list will be required to limit the number of perceived false positives. Because this research focuses on using JA3/s hashes to detect anomalous activity, none of this research will be effective against network connectivity that is **not** encrypted over SSL/TLS.

Additionally, a network defender knowing their network will ensure these methodologies target the correct internal network segments. The queries are designed to limit the analysis to just internal hosts that are making outbound connections. None of the concepts presented in this paper will work effectively against internal source hosts used for general web browsing or hosts that routinely reach out to a multitude of external services via SSL/TLS sessions. As such, all queries should be restricted to just the internal hosts or netblocks that have limited outbound connectivity as a client.

SSL/TLS interceptions or inspection will break all of the methodologies presented here. This is because SSL/TLS interception will show different characteristics than the actual external server to the client making the request. As such, JA3/s hashes will be potentially unusable for detecting anomalous activity. This has been called (quite annoyingly to the author) by my colleagues the *LaFerrera Paradox*, as in where a defender is advanced enough to know they cannot detect Supply Chain issues but, as such, have put in mitigations that prevent common methods of detection.

Detecting Anomalous Activity

Throughout this research, we took many approaches to develop detections. From more traditional techniques such as first seen or rarest to more advanced strategies such as leveraging Splunk Enterprise's `anomalydetection`[7] command, which is an SPL command that uses frequency analysis to detect unlikely(anomalous) values in categorical fields such as JA3s hashes, and creating a similar approach using lookup[8] and SPL[9]. As previously mentioned, in most cases, an allow list will be required to ensure expected network traffic is not included in the results.

Queries

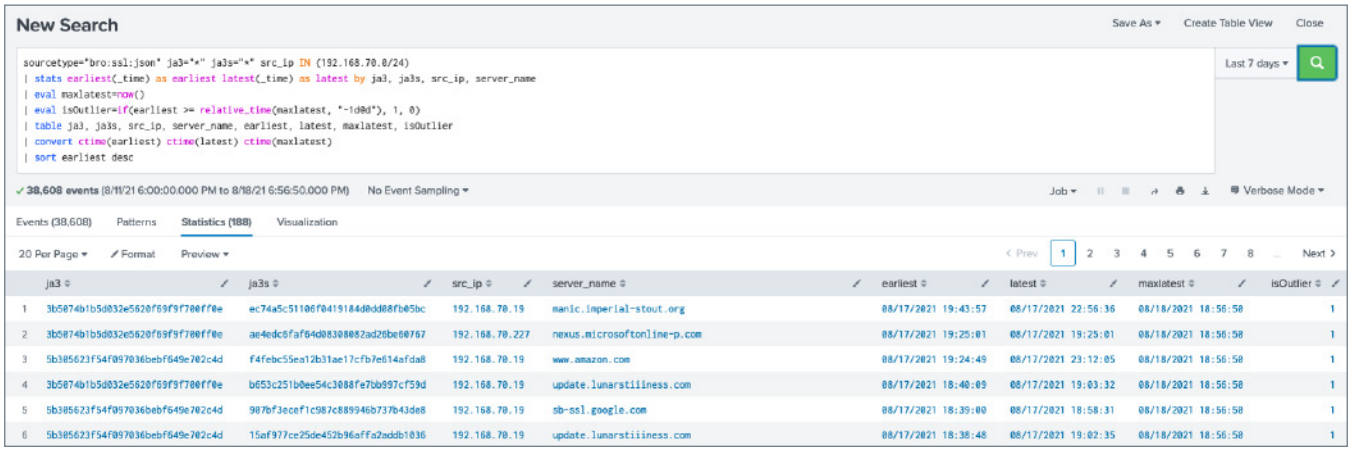
We focused on simple methodologies that the large majority of network defenders would be able to immediately leverage with minimal experience or modifications. We have also focused on query types that have been proven effective with a wide variety of data sources. None of the queries should be considered the silver bullet to detecting malicious activity. In our experience, however, starting with simple but effective solutions is the best way to help solve the problems of now.

All of the following queries have been used to identify potential abnormal network traffic and have been proven effective, with the aforementioned limitations in mind. In all scenarios, the queries will need to be modified to reflect your specific network addresses. The most up-to-date version of this research and the below queries can be found in the GitHub repository[10]. Each type of query is explained and then demonstrated using Splunk.

First Seen[11]

Detecting abnormal activity via a first seen query proved helpful when the analyst was familiar with network activity and leveraged an allow list. Additionally, the results are temporal, so the results can vary widely based on the timeframe specified. If the time window is too wide or narrow, potential malicious abnormal activity may be missed or blended with legitimate traffic. In many cases during our research, a time window of 7 days yielded the best results for finding the targeted malicious activity within the top 20 results. Finally, although not seen below, accuracy can be improved if an allow list of the most common JA3s hashes and/or `server_name` is added to remove known entities.

```
sourcetype="bro:ssl:json" ja3="*" ja3s="*" src_ip IN (192.168.70.0/24)
| stats earliest(_time) as earliest latest(_time) as latest by ja3, ja3s, src_ip, server_name
| eval maxlatest=now()
| eval isOutlier=if(earliest >= relative_time(maxlatest, "-1d@d"), 1, 0)
| table ja3, ja3s, src_ip, server_name, earliest, latest, maxlatest, isOutlier
| convert ctime(earliest) ctime(latest) ctime(maxlatest)
| sort earliest desc
```

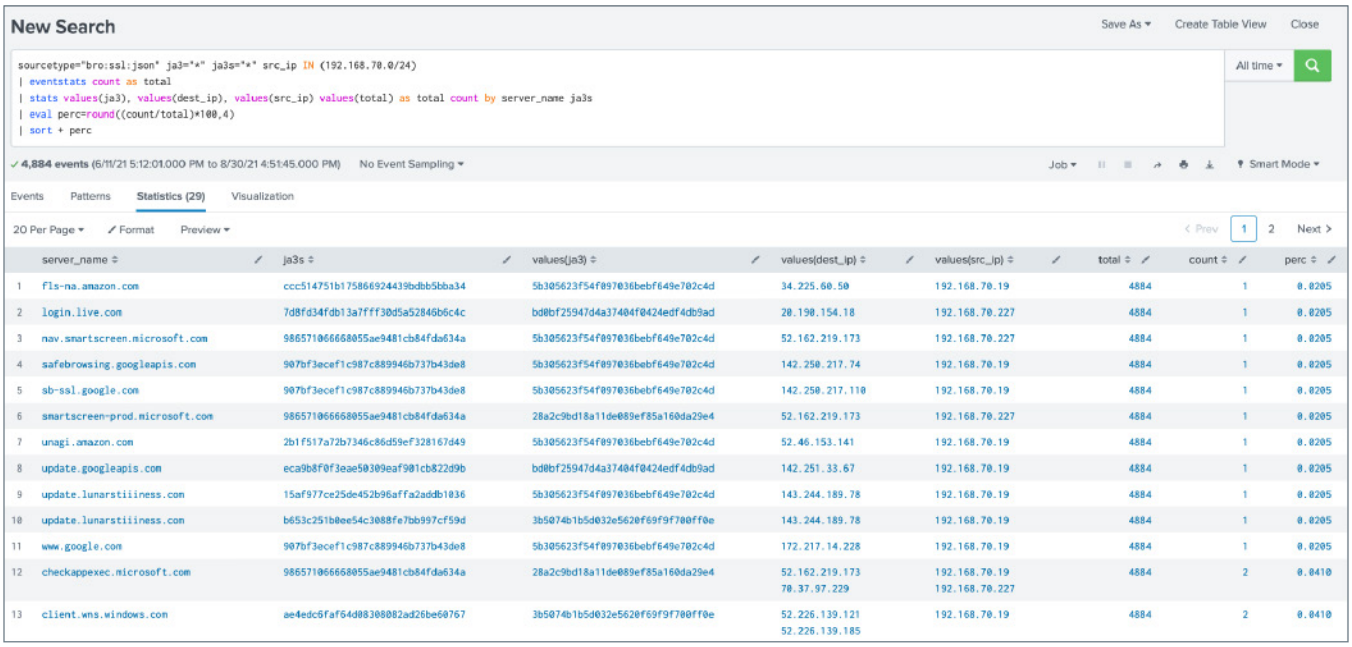


7. https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Anomalydetection.
8. https://docs.splunk.com/Documentation/Splunk/8.2.2/Knowledge/Aboutlookupsandfieldactions
9. https://www.splunk.com/en_us/resources/search-processing-language.html
10. https://github.com/mlaferrera/SEC1745
11. https://github.com/mlaferrera/SEC1745/queries/firstseen.txt

Rarest[12]

Identifying the least frequently occurring `JA3s` hash by `server_name` had limited utility without defining an allow list. In some cases, the known malicious hosts were found in the top 20 results; however, this was not always the case. The results were highly temporal, causing inconsistent findings based on the time frame chosen for the query. Time windows that are either too long or too short for analysis may return skewed results, depending on the frequency and duration of malicious connections. As such, this query is perhaps more useful as an addendum to other methods outlined in this research.

```
sourcetype="bro:ssl:json" ja3="*" ja3s="*" src_ip IN (192.168.70.0/24)
| eventstats count as total
| stats values(ja3), values(dest_ip), values(src_ip) values(total) as total count by server_name
ja3s
| eval perc=round((count/total)*100,4)
| sort + perc
```



Anomaly Detection[13]

After seeing initial success with "first seen" and "rarest" query methods, our research focused on using histogram function for `anomalydetection`. This Splunk native command helps to identify anomalous events in our data. It will compute a probability for each event in the results and then identify events with an unusually small probability. It can be useful for identifying abnormal events within the time window for a query. One thing to note is that even malicious events can seem like benign activity if the frequency of the events is similar to legitimate traffic.

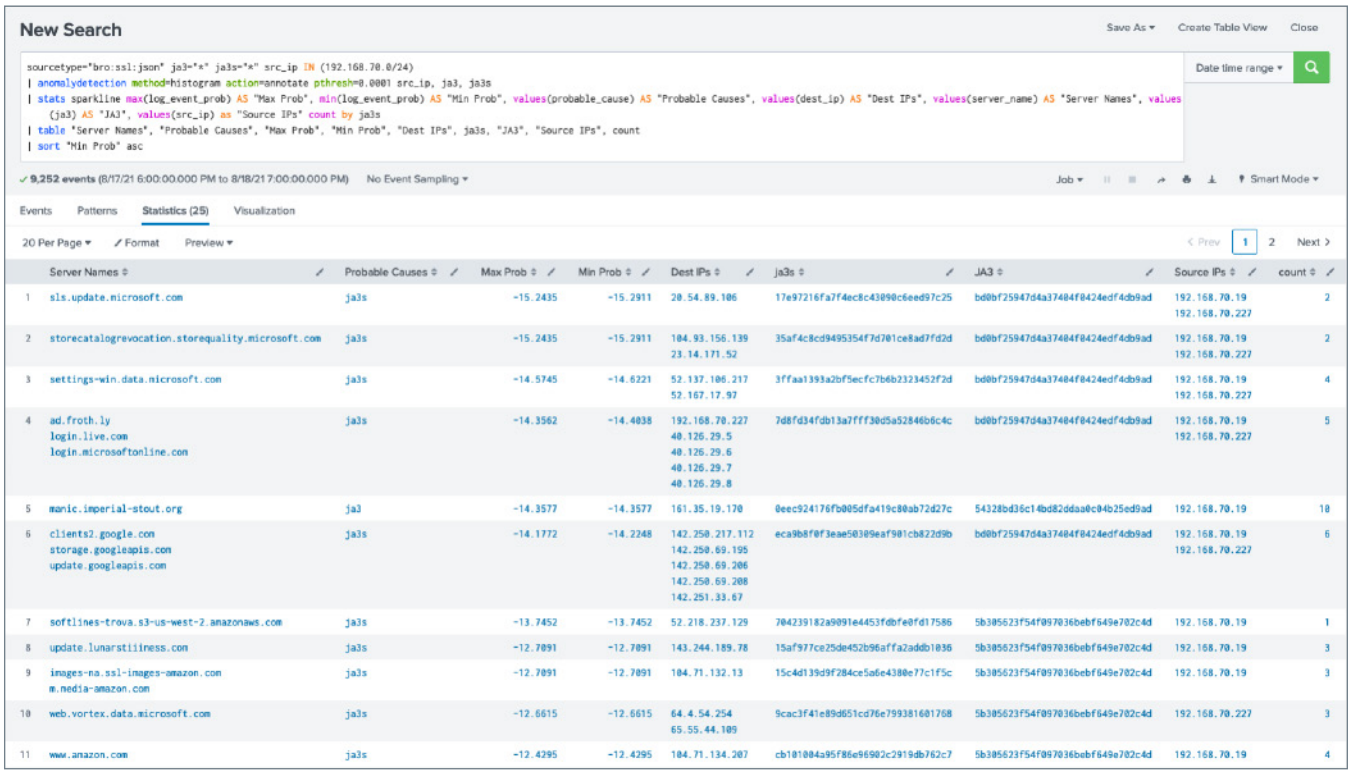
In our testing, modifying the probability threshold (`pthresh`) was required for fine-tuning the results and limiting benign results. The maximum effective `pthresh` value in our experiments was 0.001. However, this will most likely need to be adjusted based on the amount of data collected and the desired sensitivity to anomalous events.

Leveraging the `anomalydetection` command proved to be highly effective at identifying malicious abnormal activity over a 24 to 48 hour period. Periods longer than this reduced the effectiveness of the query. In experiments of smaller networks with a single /24 netblock, the known malicious activity was consistently identified without an allow list in the top 30 events. However, in networks with multiple or more extensive netblocks, this was not the case. Though it

12. https://github.com/mlaferrera/SEC1745/queries/rarest.txt.
13. https://github.com/mlaferrera/SEC1745/queries/anomalydetection.txt

did identify known malicious activity, they were not consistently in the top 30 events. An allow list of benign hosts was beneficial in this scenario, ultimately identifying malicious anomalous activity within the top 30 events.

```
sourcetype="bro:ssl:json" ja3="*" ja3s="*" src_ip IN (192.168.70.0/24)
| anomalydetection method=histogram action=annotate pthresh=0.0001 src_ip, ja3, ja3s
| stats sparkline max(log_event_prob) AS "Max Prob", min(log_event_prob) AS "Min Prob",
values(probable_cause) AS "Probable Causes", values(dest_ip) AS "Dest IPs", values(server_name)
AS "Server Names", values(ja3) AS "JA3", values(src_ip) as "Source IPs" count by ja3s
| table "Server Names", "Probable Causes", "Max Prob", "Min Prob", "Dest IPs", ja3s, "JA3",
"Source IPs", count
| sort "Min Prob" asc
```



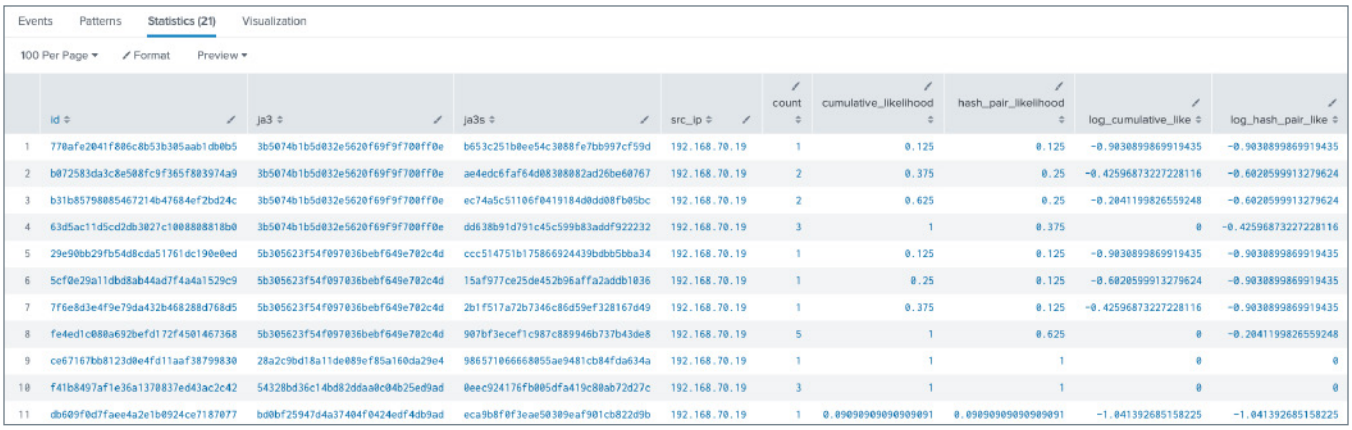
Anomaly Detection via Lookups[14,15,16]

Our research also focused on replicating the `anomalydetection` command in SPL and storing the results in a lookup table for better scalability. In this query, we calculate a similar frequency likelihood of the event's `src_ip`, `ja3`, and `ja3s` tuple, then store our results in a lookup table CSV via the `outputlookup` command.

```
sourcetype="bro:ssl:json" ja3="*" ja3s="*" src_ip IN (192.168.70.0/24)
| eval id=md5(src_ip+ja3+ja3s)
| stats count by id,ja3,ja3s,src_ip
| eventstats sum(count) as total_host_count by src_ip,ja3
| eval hash_pair_likelihood=exact(count/total_host_count)
| sort src_ip ja3 hash_pair_likelihood
| streamstats sum(hash_pair_likelihood) as cumulative_likelihood by src_ip,ja3
```

14. https://github.com/mlaferrera/SEC1745/queries/outputlookup.txt
15. https://github.com/mlaferrera/SEC1745/queries/inputlookup.txt
16. https://github.com/mlaferrera/SEC1745/queries/outputlookup-update.txt

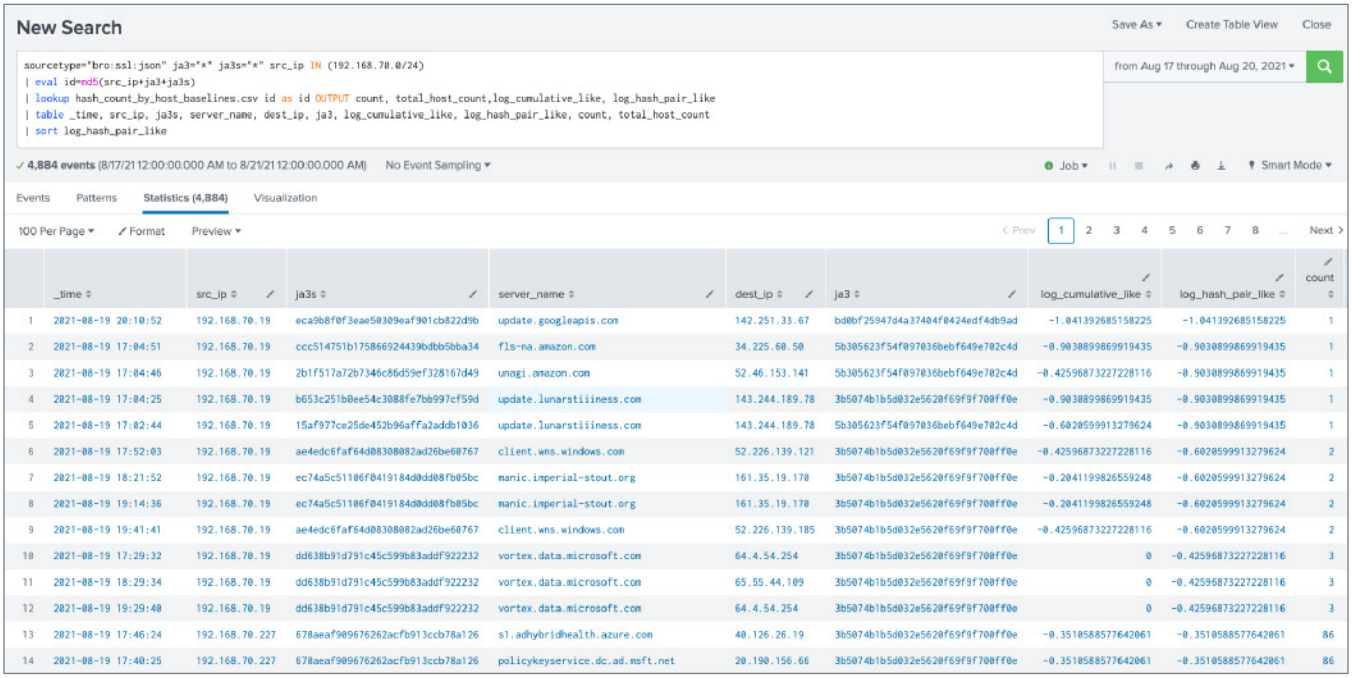
```
| eval log_cumulative_like=log(cumulative_likelihood)
| eval log_hash_pair_like=log(hash_pair_likelihood)
| outputlookup hash_count_by_host_baselines.csv
```



	id	ja3	ja3s	src_ip	count	cumulative_likelihood	hash_pair_likelihood	log_cumulative_like	log_hash_pair_like
1	770afe2041f806c8b53b305aab1db0b5	3b5074b1b5d032e5620f69f9f700ff0e	b653c251b0ee54c3088fe7bb997cf59d	192.168.70.19	1	0.125	0.125	-0.9030899869919435	-0.9030899869919435
2	b072583da3c8e508fc9f365f803974a9	3b5074b1b5d032e5620f69f9f700ff0e	ae4edc6faf64d08308082ad26be60767	192.168.70.19	2	0.375	0.25	-0.425968732272281116	-0.6020599913279624
3	b31b85798085467214b47684ef2bd24c	3b5074b1b5d032e5620f69f9f700ff0e	ec74a5c51106f0419184d0dd08fb05bc	192.168.70.19	2	0.625	0.25	-0.2041199826559248	-0.6020599913279624
4	63d5ac11d5cd2db3027c1008808818b0	3b5074b1b5d032e5620f69f9f700ff0e	dd638b91d791c45c599b83addf922232	192.168.70.19	3	1	0.375	0	-0.425968732272281116
5	29e90bb29fb54d8cda51761dc190e0ed	5b305623f54f097036bebf649e702c4d	ccc514751b175866924439bdbb5bba34	192.168.70.19	1	0.125	0.125	-0.9030899869919435	-0.9030899869919435
6	5cf0e29a11dbd8ab44ad7f4a4a1529c9	5b305623f54f097036bebf649e702c4d	15af977ce25de452b96affa2addb1036	192.168.70.19	1	0.25	0.125	-0.6020599913279624	-0.9030899869919435
7	7f6e8d3e4f9e79da432b468288d768d5	5b305623f54f097036bebf649e702c4d	2b1f517a72b7346c86d59ef328167d49	192.168.70.19	1	0.375	0.125	-0.425968732272281116	-0.9030899869919435
8	fe4ed1c080a692befd172f4501467368	5b305623f54f097036bebf649e702c4d	907bf3ecef1c987c889946b737b43de8	192.168.70.19	5	1	0.625	0	-0.2041199826559248
9	ce67167bb8123d8e4fd11aaf38799830	28a2c9bd18a11de089ef85a160da29e4	986571066668055ae9481cb84fda634a	192.168.70.19	1	1	1	0	0
10	f41b8497af1e36a1370837ed43ac2c42	54328bd36c14bd82ddaa0c04b25ed9ad	0eec924176fb005dfa419c80ab72d27c	192.168.70.19	3	1	1	0	0
11	db609f0d7faee4a2e1b0924ce7187077	bd0bf25947d4a37404f0424edf4db9ad	eca9b8f0f3eae50309eaf901cb822d9b	192.168.70.19	1	0.09090909090909091	0.09090909090909091	-1.041392685158225	-1.041392685158225

Once the lookup table is generated, another query can be run with the lookup command to identify anomalous activity. Ideally, the query that produces the `outputlookup` should be run over a period outside the secondary query's bounds with the lookup command. Our testing focused on generating an `outputlookup` over the previous seven days' worth of data, then querying for anomalous events from up to the last 48 hours.

```
sourcetype="bro:ssl:json" ja3="*" ja3s="*" src_ip IN (192.168.70.0/24)
| eval id=md5(src_ip+ja3+ja3s)
| lookup hash_count_by_host_baselines.csv id as id OUTPUT count, total_host_count,log_cumula-
tive_like, log_hash_pair_like
| table _time, src_ip, ja3s, server_name, subject, issuer, dest_ip, ja3, log_cumulative_like, log_
hash_pair_like, count, total_host_count
| sort log_hash_pair_like
```



	_time	src_ip	ja3s	server_name	dest_ip	ja3	log_cumulative_like	log_hash_pair_like	count
1	2021-08-19 20:10:52	192.168.70.19	eca9b8f0f3eae50309eaf901cb822d9b	update.googleapis.com	142.251.33.67	bd0bf25947d4a37404f0424edf4db9ad	-1.041392685158225	-1.041392685158225	1
2	2021-08-19 17:04:51	192.168.70.19	ccc514751b175866924439bdbb5bba34	fls-na.amazon.com	34.225.60.50	5b305623f54f097036bebf649e702c4d	-0.9030899869919435	-0.9030899869919435	1
3	2021-08-19 17:04:46	192.168.70.19	2b1f517a72b7346c86d59ef328167d49	unagi.amazon.com	52.46.153.141	5b305623f54f097036bebf649e702c4d	-0.425968732272281116	-0.9030899869919435	1
4	2021-08-19 17:04:25	192.168.70.19	b653c251b0ee54c3088fe7bb997cf59d	update.lunarstiiiness.com	143.244.189.78	3b5074b1b5d032e5620f69f9f700ff0e	-0.9030899869919435	-0.9030899869919435	1
5	2021-08-19 17:02:44	192.168.70.19	15af977ce25de452b96affa2addb1036	update.lunarstiiiness.com	143.244.189.78	5b305623f54f097036bebf649e702c4d	-0.6020599913279624	-0.9030899869919435	1
6	2021-08-19 17:52:03	192.168.70.19	ae4edc6faf64d08308082ad26be60767	client.wns.windows.com	52.226.139.121	3b5074b1b5d032e5620f69f9f700ff0e	-0.425968732272281116	-0.6020599913279624	2
7	2021-08-19 18:21:52	192.168.70.19	ec74a5c51106f0419184d0dd08fb05bc	manic.imperial-stout.org	161.35.19.170	3b5074b1b5d032e5620f69f9f700ff0e	-0.2041199826559248	-0.6020599913279624	2
8	2021-08-19 19:14:36	192.168.70.19	ec74a5c51106f0419184d0dd08fb05bc	manic.imperial-stout.org	161.35.19.170	3b5074b1b5d032e5620f69f9f700ff0e	-0.2041199826559248	-0.6020599913279624	2
9	2021-08-19 19:41:41	192.168.70.19	ae4edc6faf64d08308082ad26be60767	client.wns.windows.com	52.226.139.185	3b5074b1b5d032e5620f69f9f700ff0e	-0.425968732272281116	-0.6020599913279624	2
10	2021-08-19 17:29:32	192.168.70.19	dd638b91d791c45c599b83addf922232	vortex.data.microsoft.com	64.4.54.254	3b5074b1b5d032e5620f69f9f700ff0e	0	-0.425968732272281116	3
11	2021-08-19 18:29:34	192.168.70.19	dd638b91d791c45c599b83addf922232	vortex.data.microsoft.com	65.55.44.109	3b5074b1b5d032e5620f69f9f700ff0e	0	-0.425968732272281116	3
12	2021-08-19 19:29:40	192.168.70.19	dd638b91d791c45c599b83addf922232	vortex.data.microsoft.com	64.4.54.254	3b5074b1b5d032e5620f69f9f700ff0e	0	-0.425968732272281116	3
13	2021-08-19 17:46:24	192.168.70.227	678aeaf909676262acfb913ccb78a126	s1.adhybridhealth.azure.com	40.126.26.19	3b5074b1b5d032e5620f69f9f700ff0e	-0.3510588577642061	-0.3510588577642061	86
14	2021-08-19 17:40:25	192.168.70.227	678aeaf909676262acfb913ccb78a126	policykeyservice.dc.ad.msft.net	20.190.156.66	3b5074b1b5d032e5620f69f9f700ff0e	-0.3510588577642061	-0.3510588577642061	86

Lastly, to ensure the probabilities are always up-to-date, we must run an additional query to ensure the latest information is in the lookup table. This can be done by simply modifying the original `outputlookup` query with a few different methods. For example, the initial `outputlookup` query should have a time window of the previous seven days, and this update query should run every 24 hours during the last 24 hours' worth of data. We will append the content from the previous query and restrict the time window to start when the last one is completed.

```
sourcetype="bro:ssl:json" ja3="*" ja3s="*" src_ip IN (192.168.70.0/24)
| eval id=md5(src_ip+ja3+ja3s)
| stats count by id,ja3,ja3s,src_ip
| append
    [| inputlookup hash_count_by_host_baselines.csv]
| stats sum(count) as count by id,ja3,ja3s,src_ip
| eventstats sum(count) as total_host_count by src_ip,ja3
| eval hash_pair_likelihood=exact(count/total_host_count)
| sort src_ip ja3 hash_pair_likelihood
| streamstats sum(hash_pair_likelihood) as cumulative_likelihood by src_ip,ja3
| eval log_cumulative_like=log(cumulative_likelihood)
| eval log_hash_pair_like=log(hash_pair_likelihood)
| outputlookup hash_count_by_host_baselines.csv
```

Results from this methodology proved to be of similar effectiveness and an equivalent amount of time for the queries to complete when compared to using the `anomalydetection` command. However, in general, day-to-day usage, testing indicates that it is approximately 100x faster when compared with the secondary lookup query. An allow list was also a necessity when tested with more extensive networks. With an allow list, the known malicious anomalous activity was consistently identified within the top 30 events.

JA3s with Sysmon[17,18]

As always, network data combined with local process executions is a very valuable data source for threat hunters. If collecting Sysmon[19] data within Splunk, it is possible to identify the processes communicating outbound and correlate them with their JA3/s hashes. This will allow for correlating Windows processes with JA3/s hashes along with the `server_name`. For instance, we will be able to identify a powershell.exe process connecting to an external host. In order to collect the relevant data, Sysmon must be configured to collect *network connection initiated (EventCode 3)* events. Olaf Hartong[20] has written and open-sourced a utility to modularly configure Sysmon, which may be the easiest way to collect the required data quickly.

After reviewing the problem, we devised two approaches to correlate JA3s with Sysmon. The first method, shown below, is searching across Sysmon and JA3/network data but is not using the more efficient Splunk datamodel[21].

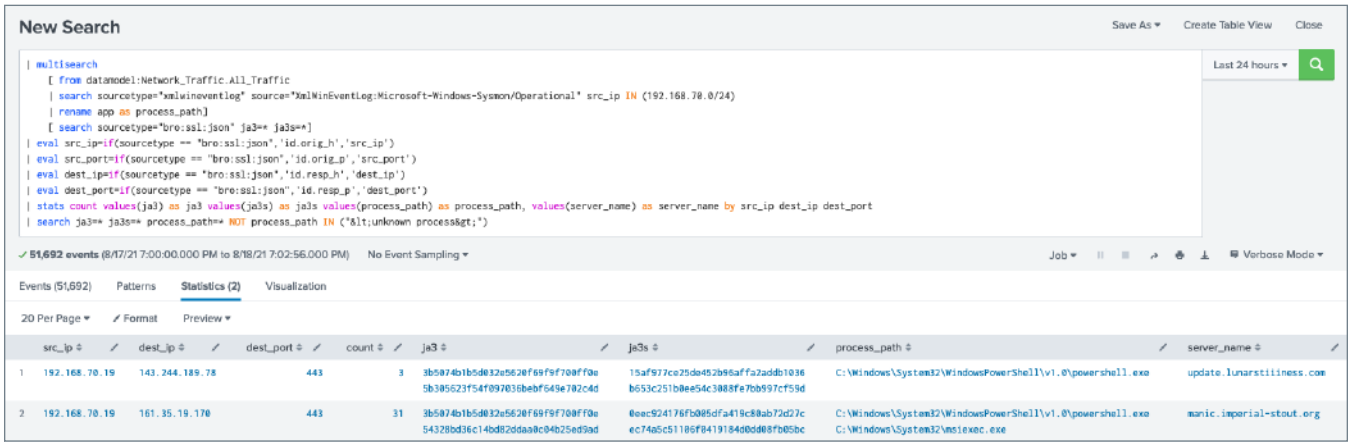
```
(source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=3 src_ip IN
(192.168.70.0/24))
       OR
(sourcetype="bro:ssl:json" ja3=* ja3s=*)
| eval src_ip=if(sourcetype == "bro:ssl:json",'id.orig_h','src_ip')
| eval src_port=if(sourcetype == "bro:ssl:json",'id.orig_p','src_port')
| eval dest_ip=if(sourcetype == "bro:ssl:json",'id.resp_h','dest_ip')
```

17. https://github.com/mlaferrera/SEC1745/queries/Sysmon-simple.txt
18. https://github.com/mlaferrera/SEC1745/queries/Sysmon-multisearch.txt
19. https://docs.microsoft.com/en-us/sysinternals/downloads/Sysmon
20. https://github.com/olafhartong/Sysmon-modular
21. https://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Aboutdatamodels

```
| eval dest_port=if(sourcetype == "bro:ssl:json",'id.resp_p','dest_port')
| stats values(ja3) as ja3 values(ja3s) as ja3s values(process_path) as process_path
values(server_name) as server_name by src_ip dest_ip dest_port
| search ja3=* ja3s=* process_path=* NOT process_path IN ("&lt;unknown process&gt;")
```

The second method is more performant and designed for use with datamodels. Both will return identical results. However, in our testing, the query leveraging datamodels was approximately 4x faster than the one without.

```
| multisearch
      [ from datamodel:Network_Traffic.All_Traffic
      | search sourcetype="xmlwineventlog" source="XmlWinEventLog:Microsoft-Windows-Sysmon/
Operational" src_ip IN (192.168.70.0/24)
      | rename app as process_path]
      [ search sourcetype="bro:ssl:json" ja3=* ja3s=*]
| eval src_ip=if(sourcetype == "bro:ssl:json",'id.orig_h','src_ip')
| eval src_port=if(sourcetype == "bro:ssl:json",'id.orig_p','src_port')
| eval dest_ip=if(sourcetype == "bro:ssl:json",'id.resp_h','dest_ip')
| eval dest_port=if(sourcetype == "bro:ssl:json",'id.resp_p','dest_port')
| stats count values(ja3) as ja3 values(ja3s) as ja3s values(process_path) as process_path,
values(server_name) as server_name by src_ip dest_ip dest_port
| search ja3=* ja3s=* process_path=* NOT process_path IN ("&lt;unknown process&gt;")
```



Depending on the environment, these queries may only be useful for triaging potential malicious activity rather than identifying anomalous activity. The most advantageous use case in our testing was using a previously identified method to identify possible malicious abnormal activity, then triage the event using the JA3s with Sysmon query. However, because the `server_name` is also included where it is available, it may be helpful to identify abnormal or suspicious activity manually.

Conclusion

Detecting anomalous malicious activity with JA3/s is by no means a perfect method. Some limitations and caveats must be taken into account. However, suppose an organization can accommodate data collection and analytics around these limitations. Then, the methodologies discussed here could help detect malicious activity that may not have been seen otherwise.

Due to how JA3/s hashes are generated, there are issues with using it to identify malicious activity with a high degree of confidence. However, using it to detect abnormal activity on a highly restricted and critical network segment or hosts can increase the level of confidence that could be cause for further investigation. Throughout our research, we sought to identify novel yet straightforward methods for leveraging data commonly found in network sensor datasets. There are undoubtedly other methods that could be developed to better leverage SSL/TLS fingerprinting techniques. We hope that this research allows organizations a better understanding of what is within the realm of possibility and inspires others to take these findings and explore additional avenues of research. Furthermore, we believe that the best way to experience this research is by trying it yourself. We have packed and hosted the data in an interactive Splunk workshop at https://bots.splunk.com.

Key Takeaways

We've explored several methodologies for identifying potential abnormal SSL/TLS communications using multiple Splunk commands and queries. In the end, numerous variables will determine how successful these queries are in your environments. Each query will almost certainly require some fine-tuning or modifications to work optimally. None of these methodologies would be useful against servers/hosts that generate large volumes of SSL/TLS events. All queries have been developed to be limited to only those hosts or netblocks of high criticality and do not generate large volumes of outbound client-side connections.

In many environments, the `anomalydetection` command will provide valuable results but may also be limited due to scaling considerations. In those cases, the Anomaly detection methodology utilizing lookups may prove to be the most relevant and efficient. Additionally, generating allow lists of approved certificates, domains, and/or JA3/s hashes will be essential to limiting the number of benign results and increasing the likelihood of detecting truly anomalous and malicious activity.

Special thanks

We want to take a moment to thank all of the individuals that helped in a multitude of ways throughout our research. From helping to identify potential techniques, helping to build and troubleshoot queries, building test infrastructure, explaining data science concepts and terminology, to just being a sounding board for ideas and concepts, and spending hours on video chat proving (and disproving) our assumptions.

- Josh Cowling
- Lily Lee
- Phillip Drieger
- Shannon Davis
- Dave Herrald
- Drew Church
- John Stoner
- Johan Bjerke
- John Lankau
- Nick Driver
- Drew Hunt
- Chris Morris