splunk> turn data into doing

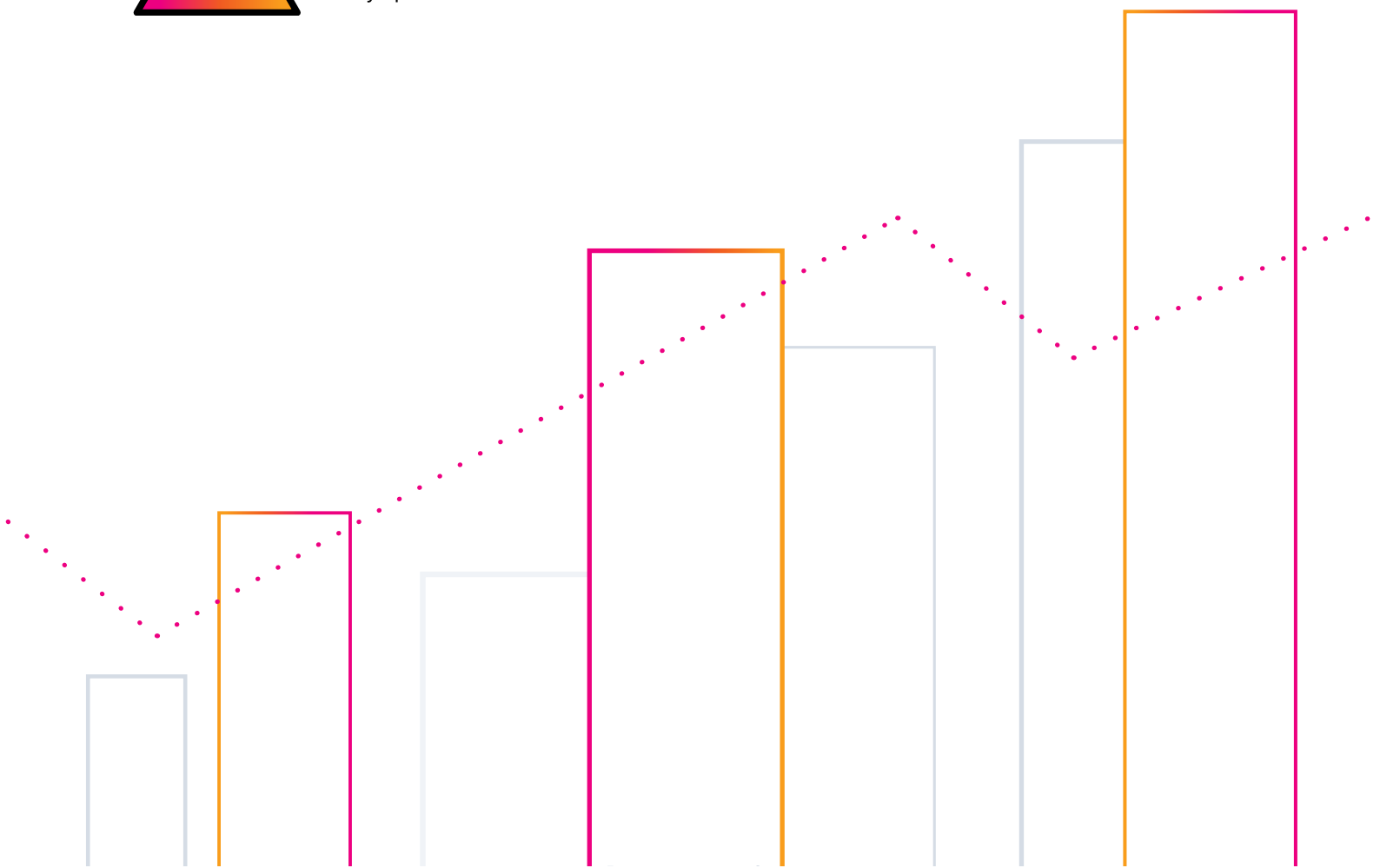
ホワイトペーパー

サプライチェーン攻撃の検出

SplunkとJA3/sハッシュを使用して悪質なアクティビティを検出する

Marcus LaFerrera、Ryan Kovar著

SURGe by Splunk



概要

SolarWinds[1]のような攻撃は、組織内のアプリケーションが未知の(悪質な可能性のある)ホストと通信を開始するのを検出することの難しさを証明しました。そして、その見落としは、深刻な「サプライチェーン攻撃」につながる可能性があります。このホワイトペーパーでは、ネットワークデータ、統計データ、JA3/JA3s[2]ハッシュ (Zeek[3]提供)、そしてSplunkを使用して、この問題に対応する方法について説明します。私たちが行った調査の主な目的は、ネットワークセキュリティ担当者が、従来なら見落としていた悪質なアクティビティを検出して攻撃者の優位に立てるようにすることです。

このホワイトペーパーでは、重要なサーバーで悪質なアクティビティを検出する方法をいくつか紹介します。現場の実務担当者に読んでいただきたい内容ですが、「はじめに」と「まとめ」のセクションはCISOや管理者の方にも参考になるでしょう。初めて観測されたデータポイントや発生頻度の低いデータポイントの検出など、時間をかけてテストしたクエリーについて詳しく説明します。また、アノマリ検出の技法を使って、悪質な可能性のあるプロセスを特定する方法も説明します。この調査の知見は、多くの面で、サプライチェーン攻撃を検出するための新しい手法や技法を求めるセキュリティ担当者に大いに役立つでしょう。

いずれの知見も、ソフトウェアサプライチェーン攻撃を検出するための特効薬というわけではありません。それでも、検出スピードを上げて攻撃を難しくするためには有効です(たとえば熊に遭遇したときに、熊より速く走れなくても、仲間より速く走れればリスクは減ります)。SOCの若手アナリストからベテランの脅威ハンターまで、セキュリティ担当者が早期に価値を実感できるよう、Zeekのような簡単に入手して導入できるツールと、すぐに実装できるクエリーを組み合わせた方法を考えました。私たちが目指したのは、いわば、干し草の山を徹底的に小さくすることで、その中の針をできるだけ見つけやすくすることです。

はじめに

多くのソフトウェア製品は、自動更新、コンテンツサブスクリプション、データアップロードなどをサポートするため、バックグラウンドで開発元と通信する「Phone home」機能を備えています。そして、それらのソフトウェアの多くは、組織のクラウド環境やオンプレミス環境内の特権的な場所に導入されます。たとえばSolarwinds Orionは、ネットワーク監視というその主要な用途をサポートするために、通常、ネットワークアクセスの点で完全な自由が与えられます。同様に、人気の高いコードカバレッジツールのCodecovは、継続的インテグレーションパイプラインに組み込まれ、ソースコード、パスワード、APIキー、証明書署名鍵などに自由にアクセスできます。一般的に、ソフトウェアの開発元が、Phone homeトラフィックの正規の通信先に使用するIPアドレス、ドメイン名、証明書を公開することはありません。公開するベンダーもありますが、多くの場合は顧客から開示要求があった場合のみで、情報自体も頻繁に変更されます。攻撃者は、こうした状況を踏まえて、この仕組みを積極的に悪用します。開発元のシステムに侵入し、製品のコードを改ざんして、正規の「Phone home」機能を装ってユーザーの機密データを別の場所にリダイレクトさせるのです。

ベンダーには当然、ソフトウェアサプライチェーンを構成する自社システムを保護する責任がありますが、ユーザー側でも、悪質なPhone homeアクティビティを検出するために何かできないでしょうか。多くのエキスパートが勧めるのが、正常なネットワークアクティビティのベースラインを設定し、そこから外れるアクティビティを検出したらアラートを出すという方法です。しかしこれは、口で言うほどたやすいことではありません。IPアドレスは頻繁に変わり、数分のうちに、多くの場合は別のユーザーに割り当て直されます。

この問題の解決策の1つが、高精度なデータポイントを収集して異常なアクティビティを検出することです。今回の調査を始めるにあたって、私たちは、意図的に目標を狭め、多くの制限を設けることで、設定やインフラの変更が最小限または皆無で済み、多くの読者が実践しやすいように心掛けました。大切なのは「今後」ではなく「今すぐ」使えることです。「今後」では遅すぎるかもしれないのです。収集対象として一般的で広くサポートされている高精度データポイントの1つがJA3ハッシュとJA3sハッシュです。ここではまとめて「JA3/s」と呼びます。このホワイトペーパーでは、JA3/sハッシュを高精度データポイントとして活用し、Splunkのコア機能を使って異常なアクティビティを突き止めます。

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/

JA3/sとは

JA3[4]は、SSL/TLSハンドシェイクプロセスで使用する特定のMD5ハッシュ値を生成するためのオープンソースの手法です。クライアントのハンドシェイクリクエストに含まれる主な属性がセッションから抽出され、連結されて、その文字列からMD5アルゴリズムによってハッシュが生成されます。具体的にはクライアント側の次の属性が抽出されます。

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

これらの値を連結し、その結果からハッシュを計算することにより、リクエストを行った特定のクライアントやライブラリ/バイナリの一貫したハッシュを生成できます。たとえば、Trickbotに感染したコンピューターからのセッションをこの方法で分析すると、TrickbotのバイナリのJA3ハッシュは、`6734f37431670b3ab4292b8f60f29984`になります。このハッシュは、Trickbotのバイナリからリクエストされ SSL/TLSセッションであれば、送信元と宛先のIPアドレスが変わっても常に同じです。攻撃者にとって、IPアドレスやドメイン名を変えることは比較的容易ですが、別のSSL/TLSライブラリを使用するようにマルウェアを改変することは容易ではありません。そのため、JA3を監視すれば、攻撃者に対してネットワーク接続を隠ぺいするハードルを上げることができます。

さらに、同様の方法でサーバーセッションのJA3ハッシュを計算できます。これは正式には「JA3sハッシュ」と言います。JA3ハッシュの生成プロセスは同じですが、サーバーセッションから抽出する主な属性が多少異なります。これは、クライアントのリクエストによってサーバーの応答が異なることがあるためです。具体的にはサーバー側の次の属性が抽出されます。

```
SSLVersion,Cipher,SSLExtension
```

JA3とJA3sのいずれも、さまざまなツールを使って簡単にネットワークトラフィックから取得できます。しかし、今回の調査の目標を踏まえて、ここではハッシュ生成ツールとしてZeekを使用します。

検出

今回の調査では、異常なアクティビティを検出する複数の方法を開発しました。目標は、Splunk Enterprise以外のコンポーネントを使用する必要がほとんどまたは全くないシンプルなクエリーを構築することでした。その結果、いくつかの注意事項と制限が生じたので、まずはそれらをお伝えします。

注意事項と制限

サプライチェーン攻撃に限らず、いずれのタイプの悪質なアクティビティでも、100%確実に検出する方法はありません。そのため、私たちの目標は常に、利用可能なツールを使って、異常なアクティビティをできるだけ多く突き止めることです。実際の企業データ[5]とテスト環境で生成したデータ[6]を用いたテストでは、異常なJA3/sハッシュから異常なアクティビティを高い確率で検出できることがわかりました。とはいえ、実際の効果は多くの要因に左右されます。誤検知の件数を減らすには、許可リストの作成はほぼ必須です。また、この調査ではJA3/sハッシュを使って異常なアクティビティを検出することに重点を置いているため、SSL/TLSで暗号化**されていない**ネットワーク接続には効果がありません。

さらに、これらの方法は組織内の適切なネットワークセグメントを対象に使用する必要があります。クエリーでは、アウトバウンド接続をする内部ホストが検出されます。そのため、組織内からのWebブラウズを仲介するホストや、SSL/TLSセッション経由でさまざまな外部サービスと頻繁に通信するホストには効果がありません。これらのクエリーは、クライアントとしてアウトバウンド接続することがあまりない内部ホストやネットブロックを対象に使用してください。

SSL/TLSインターセプトやインスペクションを使用する場合も効果がありません。SSL/TLSインターセプトを使用すると、リクエスト側のクライアントには、実際の外部サーバーの属性とは異なる属性が提示されます。そのため、JA3/sハッシュに基づいて異常なアクティビティを検出することは実質的に不可能です。同僚はこれを、私の名前を取って「LaFerreraパラドックス」と呼びます(私にとっては迷惑な話ですが)。一般的な検出手法ではサプライチェーン攻撃を検出できないことがわかるくらいの知識があるからこそ、それらの手法を無効にしなければ使えない緩和策を講じることになる、という皮肉です。

4. https://github.com/salesforce/ja3
5. これらのクエリーを実際のデータで試すため、Splunkの複数のお客様にご協力いただきましたことを感謝申し上げます。このご協力がなければ、今回の調査ははるかに困難なものになっていたでしょう。
6. https://github.com/mlaferrera/SEC1745/code

異常なアクティビティの検出

今回の調査を通じて、私たちはさまざまなアプローチで検出方法を開発しました。初めて観測されたアクティビティや発生頻度の低いアクティビティの検出のような従来の技法はもちろん、Splunk Enterpriseの`anomalydetection`[7]コマンド(頻度分析を用いてJA3sハッシュのようなカテゴリフィールドの通常ではありえない異常な値を検出するSPLコマンド)を使用するアプローチや、ルックアップ[8]とSPL[9]を使用する類似のアプローチなど、最新の戦略も含まれます。前述のとおり、多くの場合は、正常なネットワークトラフィックをサーチ結果から除外するための許可リストを作成する必要があります。

クエリー

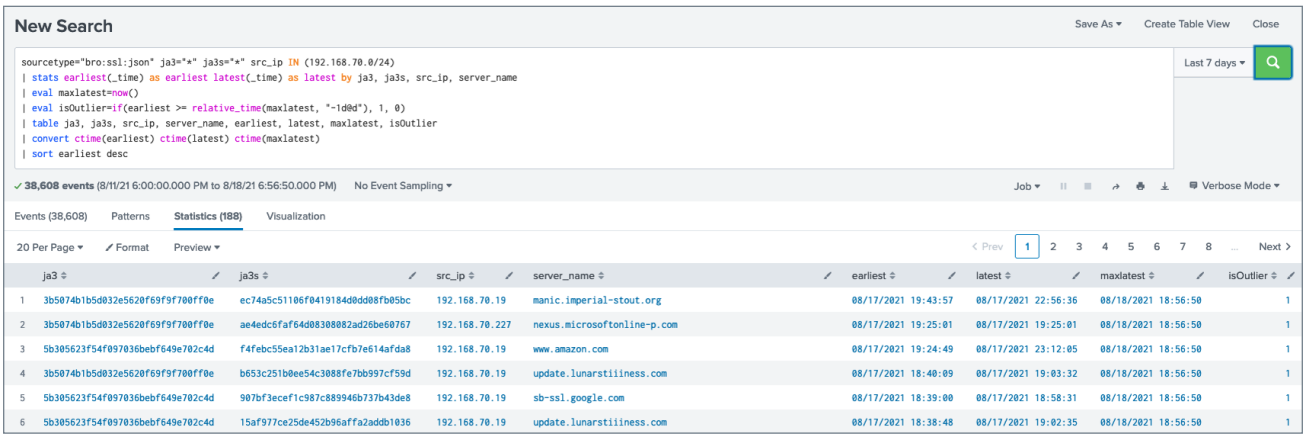
私たちは、大半のセキュリティ担当者が最小限のスキルや変更ですぐに導入できるシンプルな方法を探りました。また、幅広いデータソースで効果が実証されているクエリーを使用することも重視しました。いずれのクエリーも、悪質なアクティビティを100%確実に検出できるわけではありません。それでも、私たちの経験から言えば、目前の問題を緩和する最善の道は、シンプルで一定の効果のある解決策から始めることです。

以下のクエリーはすべて、異常なネットワークトラフィックを検出するためのもので、すでに実績があります。前述の制限があることはご留意ください。また、これらのクエリーを使用するときは常に、ネットワークアドレスを実際の環境に合わせて変更する必要があります。今回の調査と以下のクエリーの最新版は、GitHubリポジトリ[10]で公開しています。では、各クエリーの概要とSplunkでの使い方を順に説明していきます。

初めて観測されたアクティビティの検出(First Seenクエリー)[11]

First Seenクエリーでは、ネットワークアクティビティを詳細に把握していて、許可リストを使用する場合に、最も効果的に異常なアクティビティを検出できます。また、結果は特定の期間に限定されるため、指定する期間によって結果が大きく異なります。期間が極端に長いか短いと、異常なアクティビティが見落とされたり、正常なトラフィックと区別できなかったりすることがあります。調査では、多くの場合、期間を7日に設定すると最適な結果が得られ、ターゲットにした悪質なアクティビティがサーチ結果の上位20件に入りました。また、下の図にはありませんが、一般的なJA3sハッシュと`server_name`またはそのいずれかを記載した許可リストを追加して既知のエンティティを除外すると、精度が向上します。

```
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
```



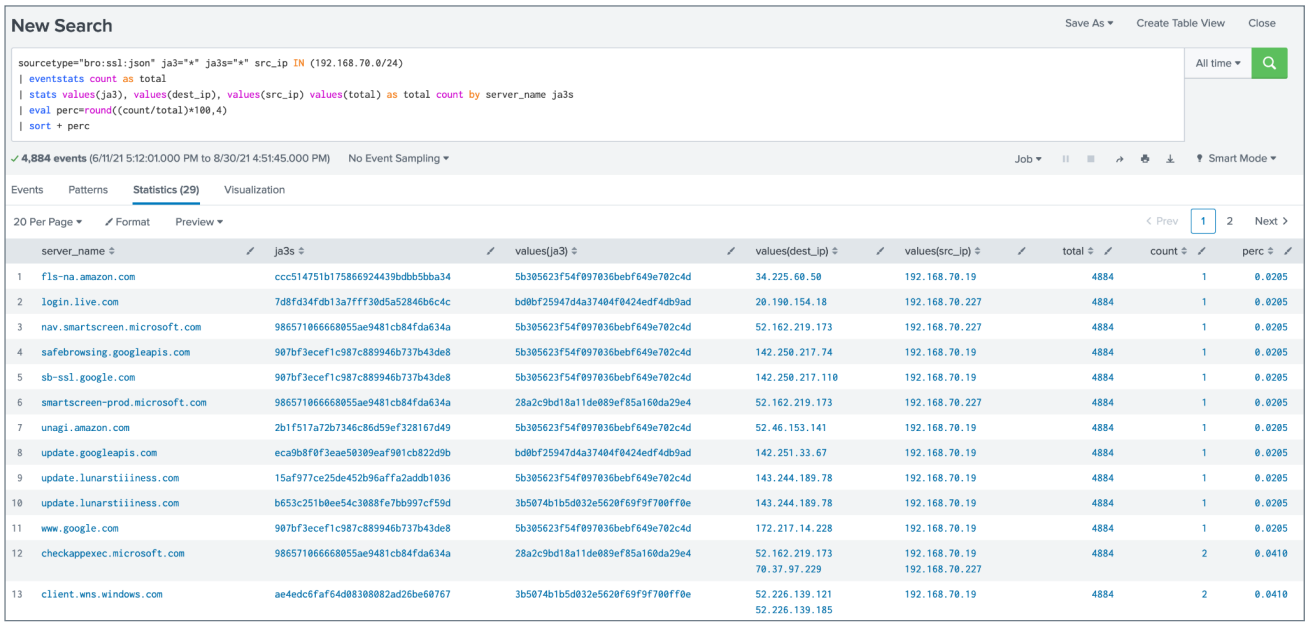
	ja3	ja3s	src_ip	server_name	earliest	latest	maxlatest	isOutlier
1	3b5074b1b5d032e5620f69f9f700ff0e	ec74a5c51106f0419184d0dd08fb05bc	192.168.70.19	manic.imperial-stout.org	08/17/2021 19:43:57	08/17/2021 22:56:36	08/18/2021 18:56:50	1
2	3b5074b1b5d032e5620f69f9f700ff0e	ae4edc6faf64d08308082ad26be60767	192.168.70.227	nexus.microsoftonline-p.com	08/17/2021 19:25:01	08/17/2021 19:25:01	08/18/2021 18:56:50	1
3	5b305623f54f097036bebf649e702c4d	f4febc55ea12b31ae17cfb7e614afda8	192.168.70.19	www.amazon.com	08/17/2021 19:24:49	08/17/2021 23:12:05	08/18/2021 18:56:50	1
4	3b5074b1b5d032e5620f69f9f700ff0e	b653c251b0ee54c3088fe7bb997cf59d	192.168.70.19	update.lunarstiiness.com	08/17/2021 18:40:09	08/17/2021 19:03:32	08/18/2021 18:56:50	1
5	5b305623f54f097036bebf649e702c4d	907bf3ecef1c987c889946b737b43de8	192.168.70.19	sb-ssl.google.com	08/17/2021 18:39:00	08/17/2021 18:58:31	08/18/2021 18:56:50	1
6	5b305623f54f097036bebf649e702c4d	15af977ce25de452b96affa2addb1036	192.168.70.19	update.lunarstiiness.com	08/17/2021 18:38:48	08/17/2021 19:02:35	08/18/2021 18:56:50	1

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/ja_jp/resources/search-processing-language.html
10. https://github.com/mlaferrera/SEC1745
11. https://github.com/mlaferrera/SEC1745/blob/main/queries/firstseen.txt

発生頻度の低いアクティビティの検出(Rarest)[12]

発生頻度の低い、server_nameごとのJA3sハッシュを検出するには、許可リストの作成がほぼ欠かせません。作成しなくても、既知の悪質なホストがサーチ結果の上位20件に入ることはありますが、常に検出できるとは限りません。結果は特定の期間に限定されるため、指定する期間によって結果が大きく異なります。期間が極端に長いか短いと、悪質な通信の頻度や期間によっては結果が偏ることがあります。そのため、このクエリーは、ここで紹介する他の方法の補足として使用するのが効果的です。

```
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
```



	server_name	ja3s	values(ja3)	values(dest_ip)	values(src_ip)	total	count	perc
1	fls-na.amazon.com	ccc514751b175866924439bdbb5bba34	5b305623f54f097036bebf649e702c4d	34.225.60.50	192.168.70.19	4884	1	0.0205
2	login.live.com	7d8fd34fdb13a7fff30d5a52846b6c4c	bd0bf25947d4a37404f0424edf4db9ad	20.190.154.18	192.168.70.227	4884	1	0.0205
3	nav.smartscreen.microsoft.com	9865710666668055ae9481cb84fda634a	5b305623f54f097036bebf649e702c4d	52.162.219.173	192.168.70.227	4884	1	0.0205
4	safebrowsing.googleapis.com	907bf3ecef1c987c889946b737b43de8	5b305623f54f097036bebf649e702c4d	142.250.217.74	192.168.70.19	4884	1	0.0205
5	sb-ssl.google.com	907bf3ecef1c987c889946b737b43de8	5b305623f54f097036bebf649e702c4d	142.250.217.110	192.168.70.19	4884	1	0.0205
6	smartscreen-prod.microsoft.com	9865710666668055ae9481cb84fda634a	28a2c9bd18a11de089ef85a160da29e4	52.162.219.173	192.168.70.227	4884	1	0.0205
7	unagi.amazon.com	2b1f517a72b7346c86d59ef328167d49	5b305623f54f097036bebf649e702c4d	52.46.153.141	192.168.70.19	4884	1	0.0205
8	update.googleapis.com	eca9b8f0f3eae50309eaf901cb822d9b	bd0bf25947d4a37404f0424edf4db9ad	142.251.33.67	192.168.70.19	4884	1	0.0205
9	update.lunarstiiiness.com	15af977ce25de452b96affa2addb1036	5b305623f54f097036bebf649e702c4d	143.244.189.78	192.168.70.19	4884	1	0.0205
10	update.lunarstiiiness.com	b653c251b0ee54c3088fe7bb997cf59d	3b5074b1b5d032e5620f69f9f700ff0e	143.244.189.78	192.168.70.19	4884	1	0.0205
11	www.google.com	907bf3ecef1c987c889946b737b43de8	5b305623f54f097036bebf649e702c4d	172.217.14.228	192.168.70.19	4884	1	0.0205
12	checkappexec.microsoft.com	9865710666668055ae9481cb84fda634a	28a2c9bd18a11de089ef85a160da29e4	52.162.219.173 70.37.97.229	192.168.70.19 192.168.70.227	4884	2	0.0410
13	client.wns.windows.com	ae4edc6faf64d08308082ad26be60767	3b5074b1b5d032e5620f69f9f700ff0e	52.226.139.121 52.226.139.185	192.168.70.19	4884	2	0.0410

アノマリ検出[13]

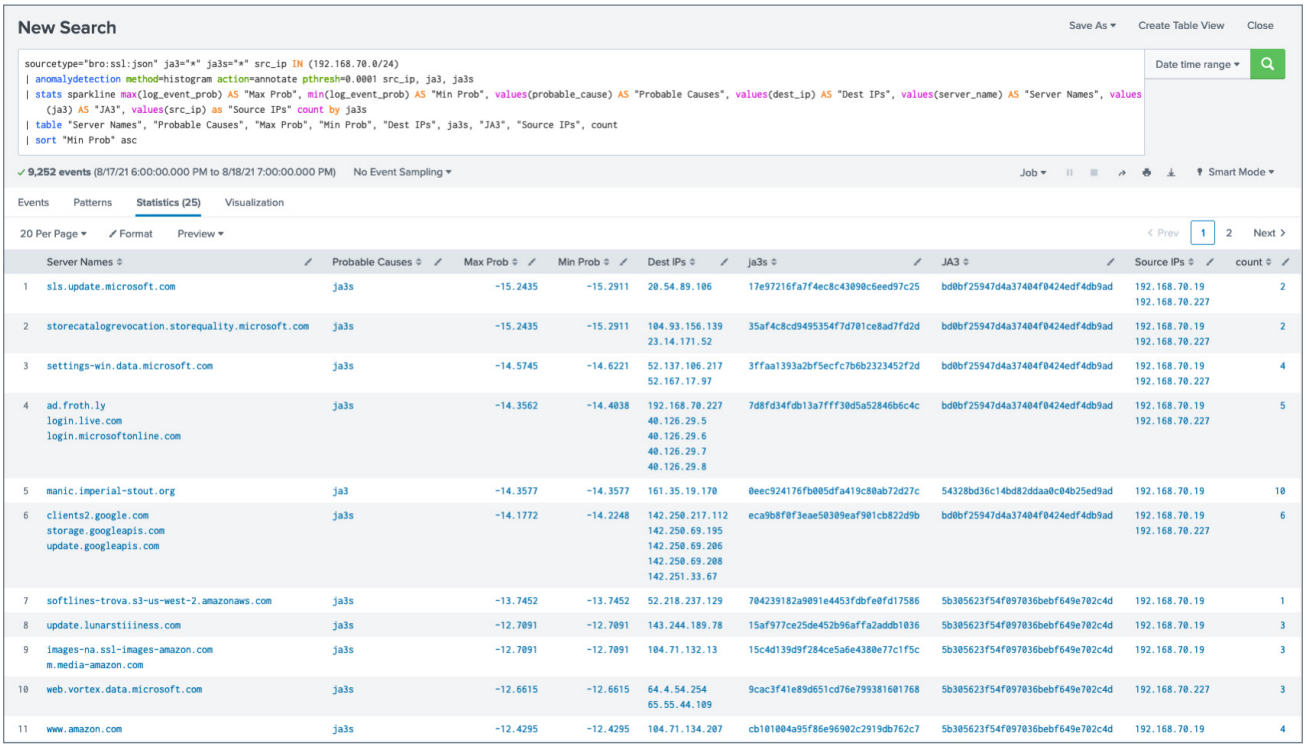
First SeenクエリーとRarestクエリーで不審なアクティビティを確認したら、anomalydetectionのヒストグラム機能を使用します。Splunkのこのネイティブコマンドは、データから異常なイベントを検出するのに役立ちます。結果に含まれる各イベントの発生確率を計算して、発生確率が異常に低いイベントを特定します。この方法は、クエリーで指定した期間内で異常なイベントを検出したいときに便利です。ただし、悪質なイベントの発生頻度が正常なトラフィックと似ている場合、それらが正常なアクティビティに見えることがある点に注意してください。

テストでは、結果を調整して適切な結果のみに絞り込むために、確率のしきい値(pthresh)を変更する必要がありました。その際、最も効果が高かったpthresh値は0.001でした。しかし実際に使用する場合は、収集するデータ量と、異常なイベントをどの程度の感度で検出したいかによって、この値を独自に調整する必要があるでしょう。

anomalydetectionコマンドを使用する方法は、24 ～ 48時間の期間で異常なアクティビティを検出する場合に高い効果を発揮します。期間がそれよりも長いと、効果が下がります。テストでは、サブネットマスク「/24」のネットブロック1つで構成される小規模ネットワークの場合、許可リストなしでも、悪質なアクティビティが一貫して上位30件に含まれました。一方、複数のネットブロックを含むネットワークや、ネットブロックの範囲が広いネットワークでは、あまり一貫した結果は得られませんでした。既知の悪質なアクティビティを検出することはできても、上位30件に入らないことがありました。この場合、悪質なアクティビティが上位30件に入るようにするには、正常なホストを記載した許可リストを指定するのが効果的です。

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

```
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
```



	Server Names	Probable Causes	Max Prob	Min Prob	Dest IPs	ja3s	JA3	Source IPs	count
1	sls.update.microsoft.com	ja3s	-15.2435	-15.2911	20.54.89.106	17e97216fa7f4ec8c43090c6eed97c25	bd0bf25947d4a37404f0424edf4db9ad	192.168.70.19 192.168.70.227	2
2	storecatalogrevocation.storequality.microsoft.com	ja3s	-15.2435	-15.2911	104.93.156.139 23.14.171.52	35af4c8cd9495354f7d701ce8ad7fd2d	bd0bf25947d4a37404f0424edf4db9ad	192.168.70.19 192.168.70.227	2
3	settings-win.data.microsoft.com	ja3s	-14.5745	-14.6221	52.137.106.217 52.167.17.97	3ffaa1393a2bf5ecfc7b6b2323452f2d	bd0bf25947d4a37404f0424edf4db9ad	192.168.70.19 192.168.70.227	4
4	ad.froth.ly login.live.com login.microsoftonline.com	ja3s	-14.3562	-14.4038	192.168.70.227 40.126.29.5 40.126.29.6 40.126.29.7 40.126.29.8	7d8fd34fdb13a7fff30d5a52846b6c4c	bd0bf25947d4a37404f0424edf4db9ad	192.168.70.19 192.168.70.227	5
5	manic.imperial-stout.org	ja3	-14.3577	-14.3577	161.35.19.170	0eec924176fb005dfa419c80ab72d27c	54328bd36c14bd82ddaa0c04b25ed9ad	192.168.70.19	10
6	clients2.google.com storage.googleapis.com update.googleapis.com	ja3s	-14.1772	-14.2248	142.250.217.112 142.250.69.195 142.250.69.206 142.250.69.208 142.251.33.67	eca9b8f0f3eae50309eaf901cb822d9b	bd0bf25947d4a37404f0424edf4db9ad	192.168.70.19 192.168.70.227	6
7	softlines-trova.s3-us-west-2.amazonaws.com	ja3s	-13.7452	-13.7452	52.218.237.129	704239182a9091e4453fdbfe0fd17586	5b305623f54f097036bebf649e702c4d	192.168.70.19	1
8	update.lunarstiiiness.com	ja3s	-12.7091	-12.7091	143.244.189.78	15af977ce25de452b96affa2addb1036	5b305623f54f097036bebf649e702c4d	192.168.70.19	3
9	images-na.ssl-images-amazon.com m.media-amazon.com	ja3s	-12.7091	-12.7091	104.71.132.13	15c4d139d9f284ce5a6e4380e77c1f5c	5b305623f54f097036bebf649e702c4d	192.168.70.19	3
10	web.vortex.data.microsoft.com	ja3s	-12.6615	-12.6615	64.4.54.254 65.55.44.109	9cac3f41e89d651cd76e799381601768	5b305623f54f097036bebf649e702c4d	192.168.70.227	3
11	www.amazon.com	ja3s	-12.4295	-12.4295	104.71.134.207	cb101004a95f86e96902c2919db762c7	5b305623f54f097036bebf649e702c4d	192.168.70.19	4

ルックアップによるアノマリ検出[14、15、16]

今回の調査では、SPLの`anomalydetection`コマンドを再現して、結果をルックアップテーブルに保存することで、サーチを拡張する方法も開発しました。このクエリーでは、イベントの`src_ip`、`ja3`、`ja3s`を一組として同様の発生確率を計算し、`outputlookup`コマンドを使って結果をCSV形式のルックアップテーブルに保存します。

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

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

Events Patterns **Statistics (21)** Visualization

100 Per Page ▾ Format Preview ▾

	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	-0.42596873227228116	-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	-0.42596873227228116
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	-0.42596873227228116	-0.9030899869919435
8	fe4ed1c080a692befd172f4501467368	5b305623f54f097036bebf649e702c4d	907bf3ecef1c987c889946b737b43de8	192.168.70.19	5	1	0.625	0	-0.2041199826559248
9	ce67167bb8123d0e4fd11aaf38799830	28a2c9bd18a11de089ef85a160da29e4	9865710666668055ae9481cb84fda634a	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

ルックアップテーブルを生成したら、lookupコマンドを使ったクエリーを実行して、異常なアクティビティを検出します。lookupコマンドによる2つ目のクエリーで指定する期間は、**outputlookup**を使用する1つ目のクエリーの対象期間に収まる範囲にするのが最適です。テストでは、**outputlookup**で過去7日間のデータからテーブルを生成した後、過去最大48時間の範囲で異常なイベントを検索しました。

```
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_cumulative_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
```

New Search

Save As ▾ Create Table View Close

```
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_cumulative_like, log_hash_pair_like
| table _time, src_ip, ja3s, server_name, dest_ip, ja3, log_cumulative_like, log_hash_pair_like, count, total_host_count
| sort log_hash_pair_like
```

from Aug 17 through Aug 20, 2021 ▾

✓ **4,884 events** (8/17/21 12:00:00.000 AM to 8/21/21 12:00:00.000 AM) No Event Sampling ▾ Job ▾ Smart Mode ▾

Events Patterns **Statistics (4,884)** Visualization

100 Per Page ▾ Format Preview ▾ ‹ Prev 1 2 3 4 5 6 7 8 … Next ›

	_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	-0.42596873227228116	-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	-0.60205999132796 24	-0.9030899869919435	1
6	2021-08-19 17:52:03	192.168.70.19	ae4edc6faf64d08308082ad26be60767	client.wns.windows.com	52.226.139.121	3b5074b1b5d032e5620f69f9f700ff0e	-0.42596873227228116	-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	-0.42596873227228116	-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	-0.42596873227228116	3
11	2021-08-19 18:29:34	192.168.70.19	dd638b91d791c45c599b83addf922232	vortex.data.microsoft.com	65.55.44.109	3b5074b1b5d032e5620f69f9f700ff0e	0	-0.42596873227228116	3
12	2021-08-19 19:29:40	192.168.70.19	dd638b91d791c45c599b83addf922232	vortex.data.microsoft.com	64.4.54.254	3b5074b1b5d032e5620f69f9f700ff0e	0	-0.42596873227228116	3
13	2021-08-19 17:46:24	192.168.70.227	678aeaf909676262acfb913ccb78a126	s1.adhybridhealth.azure.com	40.126.26.19	3b5074b1b5d032e5620f69f9f700ff0e	-0.35105885776420 61	-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	-0.35105885776420 61	-0.3510588577642061	86

最後に、常に最新の確率を得るため、ルックアップテーブルの情報を最新にするためのクエリーを実行します。このクエリーは、1つ目のoutputlookupクエリーを少し変更するだけで作成できます。たとえば、元のoutputlookupクエリーでは過去7日間を対象としましたが、ここでは、クエリーを24時間ごとに実行して、過去24時間分のデータを対象に含めます。元のクエリーの結果を追加し、前回のクエリーが完了した後で開始するように期間を制限します。

```
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
```

この方法の結果は、効果の点でも検索時間の点でも、anomalydetectionコマンドを使用する方法と比べて遜色ありません。ただし一般的に、毎日使用する場合では、lookupを使ったクエリーよりもanomalydetectionの方が約100倍高速であることがテストでわかりました。また、対象となるネットワークが拡大したときは許可リストも必須です。テストでは、許可リストを指定すると、既知の悪質なアクティビティが一貫して上位30件に含まれました。

JA3sとSysmon[17,18]

ネットワークデータとローカルプロセスの実行データの組み合わせは、脅威検出において非常に価値のあるデータソースです。SplunkでSysmon[19]データを収集すれば、外部と通信しているプロセスを特定して、JA3/sハッシュと相関付けることができます。これにより、Windowsプロセス、JA3/sハッシュ、server_nameを結び付けることができます。たとえば、外部ホストに接続するpowershell.exeプロセスを検出できます。この関連データを収集するには、Network Connection Initiatedイベント(イベントコード3)を収集するようにSysmonを設定します。Olaf Hartong氏[20]は、Sysmonをモジュール式で設定するためのユーティリティを開発し、オープンソース化しています。これを使用すると、必要なデータをすばやく収集できます。

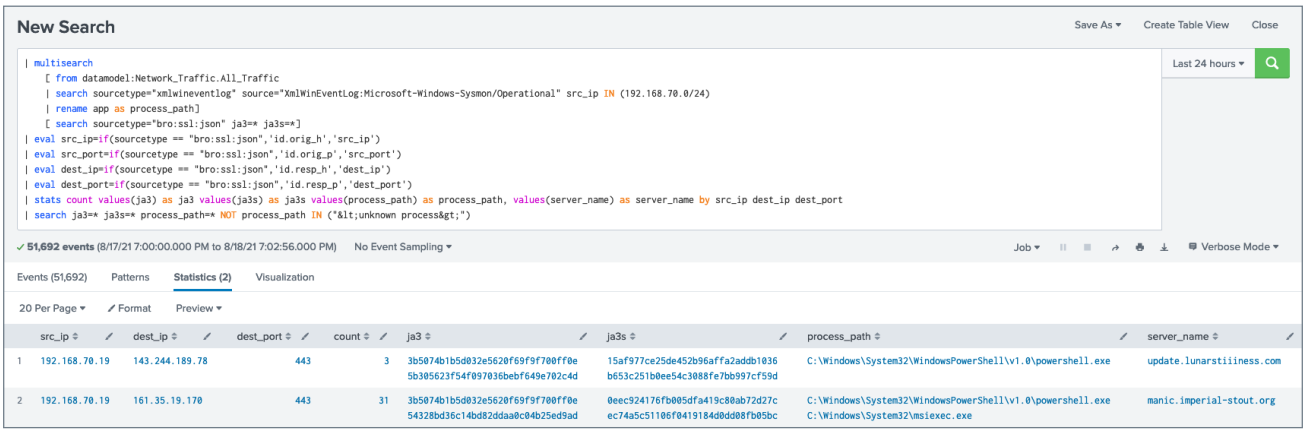
私たちは、問題の確認後にJA3sとSysmonの結果を相関付けるための2つの方法を開発しました。1つ目が次に示す方法で、SysmonデータとJA3/ネットワークデータを検索しますが、より効率的なSplunkデータモデル[21]は使用していません。

```
(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')
| 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 ("<unknown process>")
```

17. https://github.com/mlaferrera/SEC1745/blob/main/queries/sysmon-simple.txt
18. https://github.com/mlaferrera/SEC1745/blob/main/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

2つ目の方法は、データモデルを使用してパフォーマンスを高めています。いずれの方法も結果は同じです。ただし、テストでは、データモデルを使用するクエリーの方が約4倍高速でした。

```
| 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 ("<unknown process>")
```



環境によっては、これらのクエリーで、悪質な可能性のあるアクティビティのトリアージはできても、異常なアクティビティの特定はできない場合があります。テストでは、前述の方法で悪質な可能性のある異常なアクティビティを特定してから、JA3sとSysmonを使用するクエリーを使ってイベントをトリアージするという使い方が最も効果的でした。また、`server_name`を取得できる場合はその値が含まれるため、異常なアクティビティや不審なアクティビティを手動で特定する場合も役立つでしょう。

まとめ

JA3/sを使用して悪質なアクティビティを検出するのは、決して完璧な方法ではありません。いくつかの注意事項や制限を受け入れる必要があります。それでも、制限内でデータを収集して分析すれば、他の方法では特定できない悪質なアクティビティを検出できる可能性があります。

JA3/sハッシュの生成方法を考えると、悪質なアクティビティを高い信頼度で検出するのには問題があります。しかし、アクセスが制限された重要なネットワークセグメントやホストで異常なアクティビティを検出する分には、詳細調査をすべきと判断できるレベルの信頼度を確保できます。私たちは今回の調査を通じて、ネットワークセンサーのデータセットで一般的なデータを活用するためのシンプルで新しい方法を探りました。SSL/TLSフィンガープリンティングの技法をより有効に活用する方法は、当然、ほかにもあるでしょう。この調査の知見が、組織で実行できる対策をより深く理解し、さらに高度な対策を探求するために役立てば幸いです。これらの知見を最大限に活用するために、ここで紹介した方法をぜひ実際に試してみてください。こちらのデータは、**https://bots.splunk.com** のインタラクティブなSplunkワークショップでご利用いただけます。

要点

このホワイトペーパーでは、Splunkコマンドやクエリーを使って異常なSSL/TLS通信を検出する方法を紹介しました。これらのクエリーが実際の環境で効果を発揮するかどうかは、さまざまな要因に左右されます。どのクエリーも、多くの場合、最適に機能させるには、ある程度の調整や変更が必要です。また、SSL/TLSイベントを大量に生成するサーバー /ホストには効果がありません。一定の効果が保証されるのは、クライアントとして外部と頻繁に通信することのない、重要度の高いホストやネットブロックに限られます。

多くの環境では、`anomalydetection`コマンドを使用する方法が効果的ですが、この方法は拡張性の面で限界があります。拡張性が必要な場合は、ルックアップによるアノマリ検出の方法が妥当で効率的かもしれません。さらに、サーチ結果の件数を絞り込み、悪質なアクティビティをより確実に検出するには、承認済みの証明書、ドメイン、JA3/sハッシュを記載した許可リストを作成することが欠かせません。

謝辞

最後に、今回の調査にご協力いただいた皆様に感謝を申し上げます。以下の方々には、技法の開発の支援から、クエリーの作成とトラブルシューティングの支援、テスト用インフラの構築、データサイエンスの概念や用語の説明、アイデアやコンセプトの相談まで、私たちの仮説を証明(または反証)するためにビデオチャットで何時間もお付き合いいただきました。

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

SURGeアラートにぜひ**ご登録ください**

splunk>

営業へのお問い合わせはこちら：https://www.splunk.com/ja_jp/talk-to-sales.html
〒100-0004 千代田区大手町1-1-1 大手町パークビルディング 8階

www.splunk.com/ja_jp
splunkjp@splunk.com

21-21294-Splunk-Detecting Supply Chain Attacks-A4-WP-JA-202112