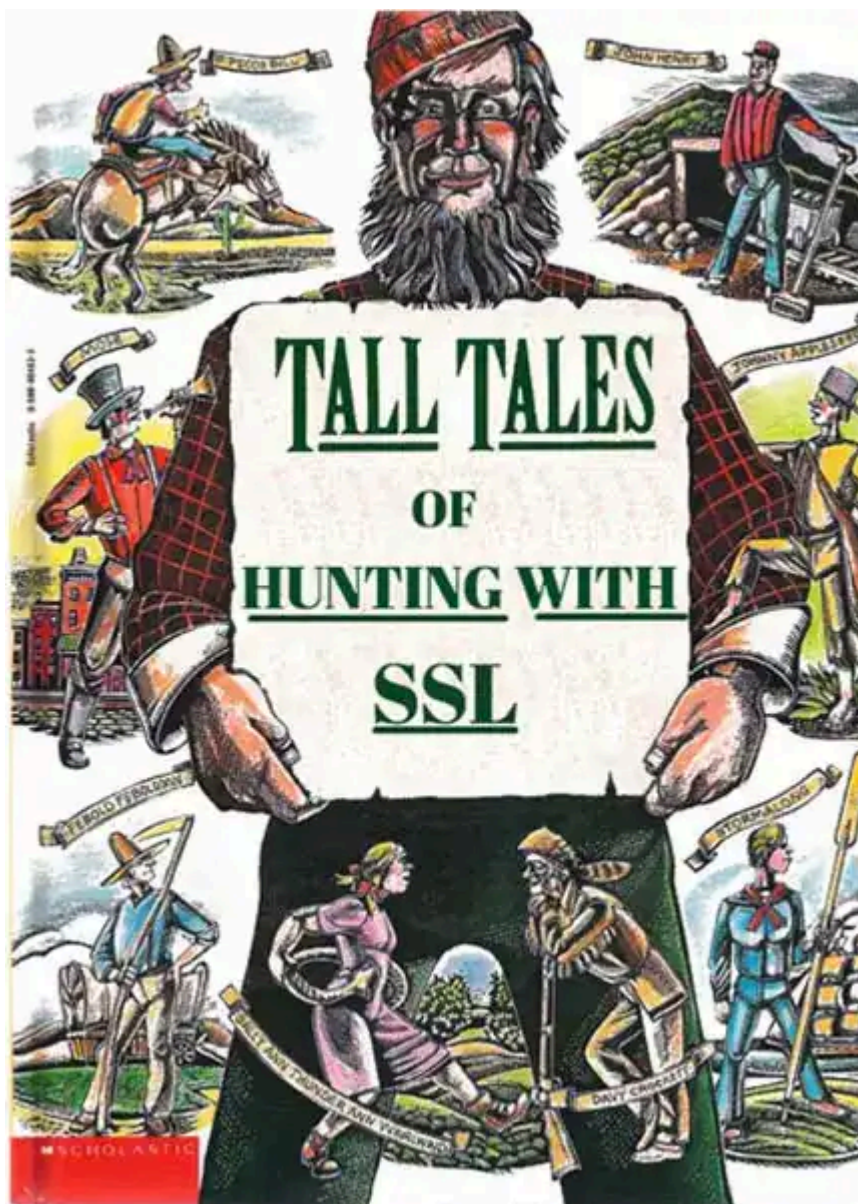


Threat Hunting with TLS/SSL Certificates | Splunk

By Lauren Stemler

Published: 2025-07-03 · Archived: 2026-04-05 22:26:06 UTC



In this article, we'll analyze how threat actors exploit TLS to hide their operations and how defenders can use exposed certificate metadata to detect them. We will discuss:

1. Why TLS certificate hunting matters
2. Which metadata fields remain visible
3. How Splunk can be used to identify suspicious TLS activity
4. Queries to enhance threat detection
5. Next steps for integrating TLS analysis into security operations.

Let's get started!

(This article is part of our [Threat Hunting with Splunk series](#). We've updated it recently to maximize your value.)

The importance of hunting in TLS certificates

First things first: this discussion will not delve into the cryptographic intricacies of [TLS and SSL](#). Instead, it's important to note that TLS (Transport Layer Security) is the modern, more secure successor to SSL (Secure Sockets Layer). For simplicity, the terms "SSL" and "TLS" will be used interchangeably throughout this article.

TLS has become today's standard for secure online communications, offering stronger encryption and enhanced security protocols. However, attackers can exploit TLS to conceal their malicious activities. While TLS effectively encrypts the content of communications, it does not entirely obscure metadata, such as the identities of the communicating parties.

As cyber threats evolve, attackers have adapted to advances in security. A decade ago, [malware command-and-control \(C2\) traffic](#) often relied on plaintext HTTP, making it straightforward for security teams to intercept and analyze. Today, attackers leverage TLS encryption to mask their traffic, complicating efforts to monitor and detect their activities.

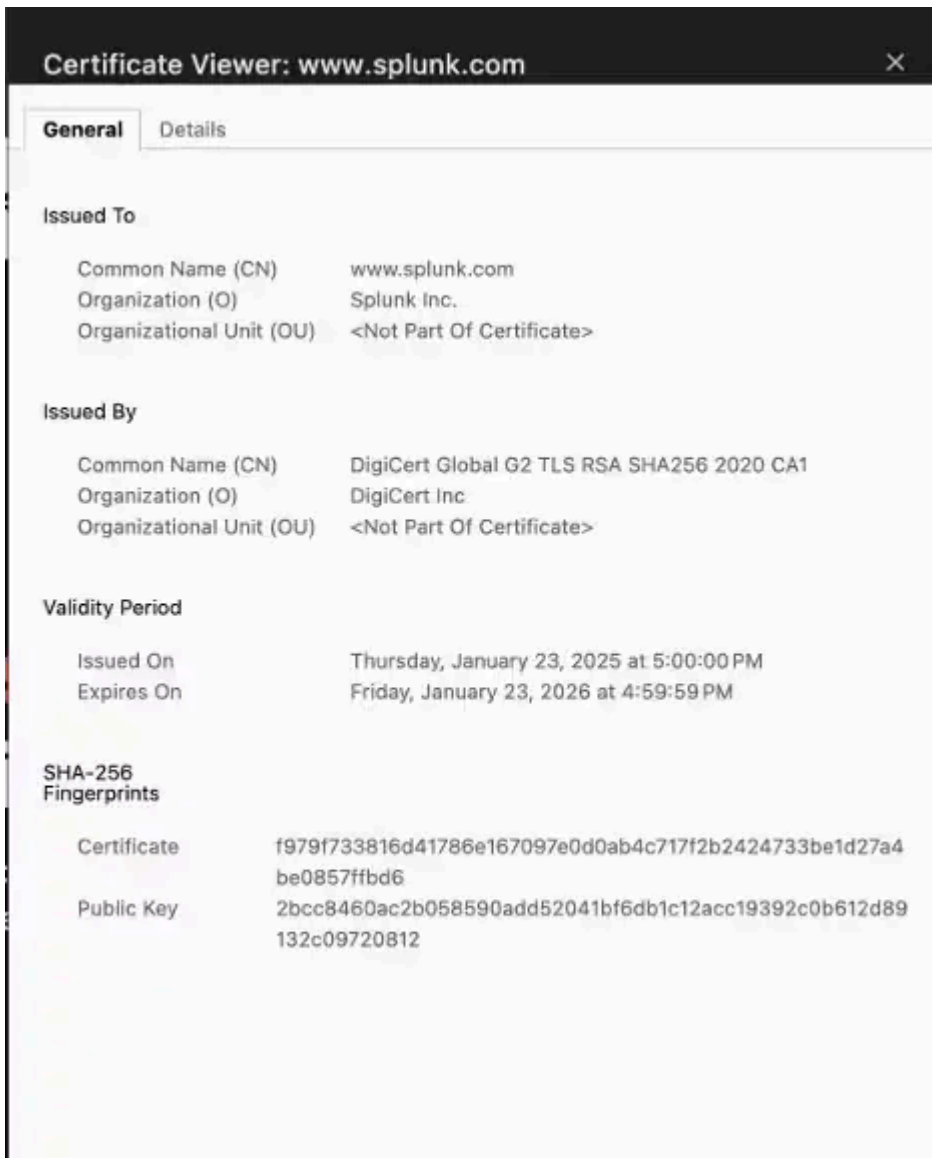
Despite these challenges, defenders are not without options. After all, we know that attackers are creatures of habit, they reuse infrastructure and leave data behind that defenders can use to trace their actions back to them. By identifying these clues, security teams can uncover malicious activity and link it back to its source.

(Note: This article assumes that you have already captured TLS metadata in Splunk. If not, implementing metadata collection will need to be done before diving into these hunting techniques.)

What unencrypted information does SSL leave?

TLS certificates are like digital ID cards for websites and services, proving their authenticity while enabling encrypted information. But here's the catch — while the actual communication is encrypted, the certificate itself contains unencrypted metadata that attackers can't hide. Defenders like you can use this metadata to identify and track suspicious activity.

The fields within a TLS certificate are generated either by cryptographic algorithms or manually entered during the creation of the certificate.



How does this help defenders? Because these fields can be unique and are often shared by adversaries across their backend infrastructure. So... what are some unique fields? The ones that we are primarily interested in are:

- `ssl_cert_self_signed`
- `ssl_subject_country`
- `ssl_subject_state`
- `ssl_issuer`
- `ssl_subject_locality`
- `ssl_subject_unit`
- `ssl_subject_organization`
- `ssl_validity_end`
- `ssl_subject_common_name`
- `ssl_serialnumber`
- `ssl_cert_sha1`

These fields can help security teams detect anomalies, track [threat actors](#), and distinguish legitimate infrastructure from adversarial activity.

Now that we know TLS certificates leave behind valuable metadata, how do we actually use this information? Since adversaries are using TLS to encrypt command and control (C2) communications, it's important to look beyond the encryption itself and analyze the exposed metadata. Since we have a hypothesis that malware is beginning to use SSL to encrypt its communications, I believe it is also reasonable to assume that they would use SSL to encrypt communication on alternative ports!

Finding encrypted communications in SSL

For nearly all the following examples, we will be using SSL data captured in Stream. Let's look:

```
sourcetype=stream:tcp ssl_cert_sha1=* NOT (dest_port=443 OR dest_port=993 OR dest_port=995 OR dest_port=465 OR dest_port=9001) | stats VALUES(ssl_issuer) VALUES(dest_port) VALUES(ssl_subject_common_name) VALUES(dest_ip) count(ssl_subject_common_name) BY ssl_cert_sha1
```

ssl_cert_sha1	VALUES(ssl_issuer)	VALUES(dest_port)	VALUES(ssl_subject_common_name)	VALUES(dest_ip)	count(ssl_subject_common_name)
A48CF8A98B95A778980792039AAC2880954ACAC2	C = US, O = DigiCert Inc, CN = DigiCert SHA2 Secure Server CA	9061	webssl2.chinanetcenter.com	230.243.230.24	1
A789978E2697DADC00C9A1EA6ED1F48A94833970C	C = US, ST = California, L = San Francisco, O = OpenGarden, OU = FireChat, CN = firechat.opengarden.com, emailAddress = straya@opengarden.com	4201	firechat.opengarden.com	107.20.137.27	1
AA4C16D3A43630EC880197BE9008DA6C0CC0A77A	CN = www.d2mzcmdqk67p2.com	8080	www.victpg76.net	104.121.11.214	1
AC81308D7AF81FE7189B182066A8D9B184F0BADA4	CN = www.2bak76.com	8001	www.hgsh-gf2c.net	91.121.23.100	1
AF52B46AA82BA3D0BE24864C84D0F5F1AE228F32	C = US, ST = Arizona, L = Scottsdale, O = "GoDaddy.com, Inc.", OU = http://certs.godaddy.com/repository/, CN = GoDaddy Secure Certificate Authority - G2	8001	*.cradlepointcm.com	52.25.11.64	1
AF38458899E8ACB4666AC023D83779ED684DC9	C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3	6697	testtelus.ru	8.101.101.148	1
B01F0E4E0B7959EC38325348F22AA8B718063BA	CN = www.jnuw2ent.com	12246	www.crs57gshscrgenaj.net	188.166.222.39	1
B3229D66D0FCEA377244D0880667108F03EE3C7	CN = 60130107, O = Code42, OU = TEST, ST = MN, C = US	4287	60130107	216.17.8.7	1
B3B97177C153778FA599CEC76829A33871E5CC53	C = KR, O = SSS, CN = SuperCA	35000	self	112.106.134.93	1
B40A2FE4AD5C186801C284C7C1832E7F1423D52B	C = TW, ST = Taiwan, L = Taipei, O = Synology Inc., OU = Certificate Authority, CN = Synology Inc. CA, emailAddress = product@synology.com	63381	synology.com	160.70.88.210	1
B4129A16951696628AF71CB49A538EB29105E61	CN = www.d4trnhw6b7.com	110	www.fipse5nic2r3.net	81.175.221.207	1
B6E4FC5EFC8FDE1D799BCB11ABAEDC3647D9421	CN = www.5d4whuonra.com	9090	www.saw42k3upwmdak.net	217.79.190.25	1

BOOM! You'll notice the 3rd and 4th entry have very suspicious metadata and link to some nasty looking domains and IP addresses. A quick pivot to VirusTotal shows that there have been malicious files hosted on that IP address in the past:

The screenshot shows the VirusTotal interface for IP address 104.131.11.214. It includes sections for Geolocation (Country: US, Autonomous System: 393406), Passive DNS replication (domains: www.nullsky.info, nullsky.info), and Latest detected files that communicate with this IP address (listing file hashes and dates).

Finding unique values in SSL certificates

For our next search, let's try and find values in SSL certificates that are unique. We will do a little pre-parsing by showing only SSL certificates that are not part of the [Alexa 1 Million top sites](#). This is done with the UT_PARSE macro (which you can install with [URL Toolbox](#)) and the Alexa 1 Million list. You can download a sample of Alexa 1 Million from [this GitHub](#) as a properly formatted lookup table.

```
sourcetype=stream:tcp ssl_cert_sha1=*
| stats VALUES(ssl_issuer) AS ssl_issuer VALUES(ssl_subject_common_name) AS ssl_subject_common_name VALUES(dest_ip) AS dest_ip
| search count=1
| eval list=""
|\ut_parse(ssl_subject_common_name,list)\
| lookup alexa-1MM domain AS ut_domain
| fillnull value=NULL rank
| search rank=NULL
| stats VALUES(ssl_cert_sha1) VALUES(ssl_subject_common_name) AS ssl_subject_common_name VALUES(dest_ip) AS dest_ip
```

New Search

```
sourcetype=stream:tcp ssl_cert_sha1=*
| stats VALUES(ssl_issuer) AS ssl_issuer VALUES(ssl_subject_common_name) AS ssl_subject_common_name VALUES(dest_ip) AS dest_ip count
(ssl_subject_common_name) AS count BY ssl_cert_sha1
| search count=1
| eval list=""
| `ut_parse(ssl_subject_common_name,list)`
| lookup alexa-1MM domain AS ut_domain
| fillnull value=NULL rank
| search rank=NULL
| stats VALUES(ssl_cert_sha1) AS ssl_cert_sha1 VALUES(ssl_subject_common_name) AS ssl_subject_common_name VALUES(dest_ip) AS dest_ip VALUES
(ut_domain) AS ut_domain count(ssl_subject_common_name) AS count BY ssl_issuer
```

73,163 events (Partial results for before 9/16/17 11:58:20.000 PM) No Event Sampling

Events Patterns Statistics (189) Visualization

20 Per Page Format Preview

ssl_issuer	ssl_cert_sha1	ssl_subject_common_name	dest_ip	ut_domain	count
C = AU, ST = 3t2i3rgeg, L = 2ehedqsdxfjh, O = 3wrwsts, OU = wfwstwe645qfhuy, CN = fg2eq34df	69D69D6DEEC4EFA2C8EA37698D1570B6A03CCE0A	fg2eq34df	80.79.114.179	None	1
C = BE, O = GlobalSign nv-sa, CN = AlphaSSL CA - SHA256 - G2	00E16ECD4D5F220DF1D8DE09DB7313912C62D8D504021642C0C158135C3D955212DA222C0874395F9DAD5C395E48053A8EAEAFB0E8209E3EBAE1041FE8BCE93843D96D031041C04C4EAD54B44898B142	*.aasky.net *.now.sh *.pelmorex.com *.schemaverse.com	142.46.208.28 50.57.126.149 52.52.75.31 54.192.136.58	aasky.net now.sh pelmorex.com schemaverse.com	4
C = BE, O = GlobalSign nv-sa, CN = GlobalSign CloudSSL CA - SHA256 - G3	136B16EF21AF72FAF667268914BAC57E9EAE1A21472C0760FD48F75803ECC282F4AF80A8D496E28ABB305E8816E69289A80B2007BAFD3004D9A2C3FC17CC3A20C015499EA213296CAF15DB4F23513DF1A7F3E4458AC589B366D72C849992765741D8C9	f4.shared.global.fastly.net iheart.map.fastly.net m.ssl.fastly.net n.ssl.fastly.net ticketmaster.map.fastly.net	151.101.1.178 151.101.1.204 151.101.2.110 151.101.53.13 151.101.61.5	fastly.net	5

Lets take a look

Right away we see some very odd values in the `ssl_issuer` and `ssl_subject_common_name` fields. If we search for that SHA1 certificate value (`ssl_cert_sha1`), we find that various websites have identified that SHA1 hash as malicious.

Google

69D69D6DEEC4EFA2C8EA37698D1570B6A03CCE0A

All Maps Videos Images Shopping More Settings Tools

3 results (0.36 seconds)

69d69d6deec4efa2c8ea37698d1570b6a03cce0a - SSL Blacklist
<https://sslbl.abuse.ch/intel/69d69d6deec4efa2c8ea37698d1570b6a03cce0a>
Oct 31, 2016 - Fingerprint (SHA1): 69d69d6deec4efa2c8ea37698d1570b6a03cce0a. Status: Blacklisted (Reason: TrickBot C&C, Listing date: 2016-10-31 ...
You've visited this page 2 times. Last visit: 9/16/17

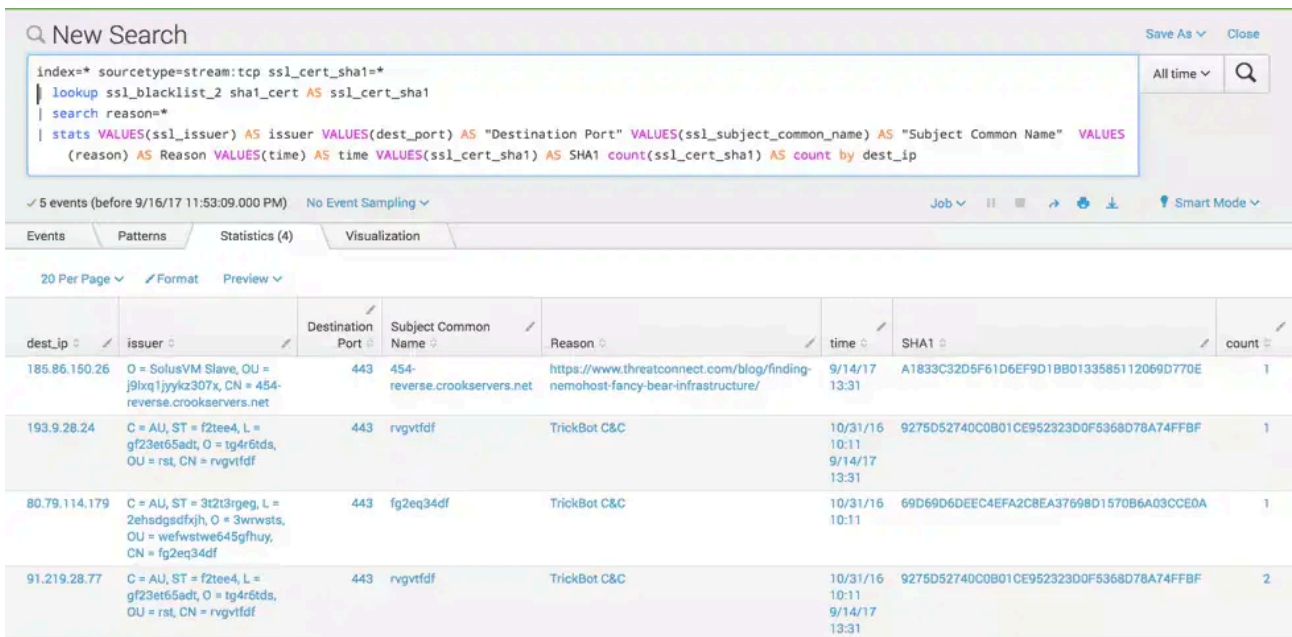
Alert 69d69d6deec4efa2c8ea37698d1570b6a03cce0a (2016-10-31 ...
<https://map.httpcs.com/alert/160131>
Alert #160131: "69d69d6deec4efa2c8ea37698d1570b6a03cce0a (2016-10-31 10:11:39)" on HTTPCS Interactive cyber-attack map : Real time Website attacks, ...
You visited this page on 9/16/17.

urlscan.io - sslbl.abuse.ch
<https://urlscan.io/result/4f8d2528-d6bd-4db2-b2b8-200cf3dcf4c2/dom/>
Aug 16, 2017 - backgroundColor=#D8D8D8;"> <td>2016-10-31 10:11:39</td> <td><a href="/intel/69d69d6deec4efa2c8ea37698d1570b6a03cce0a" ...
You visited this page on 9/9/17.

Detecting bad SSL certificates

Finally, what about detecting certificates that are “known bad”? The most common repository of externally identified certificates is from <https://ssllbl.abuse.ch/>. With minimal effort, you can turn that list of malicious SHA1 SSL certificates into a lookup table in Splunk. The example below assumes that you have done that already, but if you want an example, visit my [GitHub](#) and download the `ssl_blacklist_2.csv` file.

```
index=* sourcetype=stream:tcp ssl_cert_sha1=*
| lookup ssl_block_list sha1_cert AS ssl_cert_sha1
| search reason=*
| stats VALUES(ssl_issuer) AS issuer VALUES(dest_port) AS "Destination Port" VALUES(ssl_subject_common_name) AS
```



The results below show that we have five hits against our data and a quick explanation of each malicious certificate.

Next steps in security operations

Finding malicious SSL/TLS in your environment is just the first step. Just like finding malicious IP addresses or file hashes you will need to pivot and research using tools like

- www.passivetotal.com
- www.censys.io
- <https://www.circl.lu/services/passive-ssl/>

Once you understand how to use these certificates, try creating workflow actions in Splunk to pivot “automagically”.

If you’d like an example, check out my sample workflow action for www.censys.io located on my [github page](#). If you are interested in more information on hunting and pivoting with TLS certificates, check out this great talk from my good friend Mark Parsons on [hunting a TLS certificate](#). Until then...

Happy hunting!

Source: https://www.splunk.com/en_us/blog/security/tall-tales-of-hunting-with-tls-ssl-certificates.html