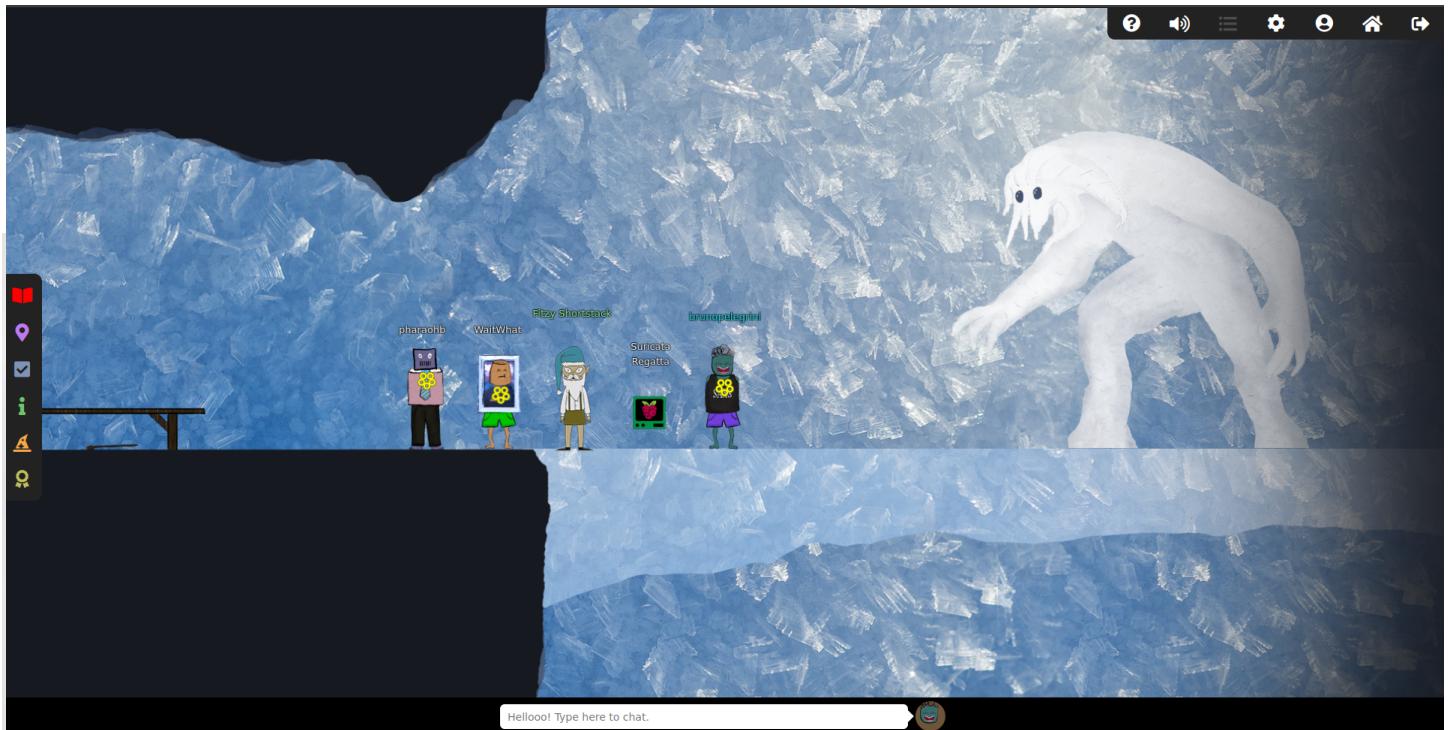


Write-up CTF
SANS - Holiday Hack Challenge 2022
Challenge: Windows Event Logs
Difficulty: Easy
<https://github.com/brunopelegini>

First of all, open the Objectives session in the left menu. For this challenge, we have the Suspicious.pcap file file to analyze. Download this file.

The screenshot shows the KringleCon interface. On the left, there's a sidebar with the following menu items: Story [23%], Destinations, Objectives (which is selected and highlighted in blue), Hints, Hats, Talks, Achievements, Settings, and [Exit]. The main panel has a title 'GO BACK' at the top. Below it, there's a hint: 'terminal or offline. Get hints for this challenge by typing `hint` in the upper panel of the Windows Event Logs terminal.' Under the 'Objectives' section, there are four listed: 'Find the Next Objective' (completed with a green checkmark), 'Suricata Regatta' (completed with a green checkmark), 'Recover the Elfen Ring' (not completed, no checkmark), 'Recover the Web Ring' (not completed, no checkmark), 'Recover the Cloud Ring' (not completed, no checkmark), and 'Recover the Burning Ring of Fire' (not completed, no checkmark). Each objective has a small ring icon to its right.

In the Kringlecon game, go to Tokien Ring tunnels and search for Suricata Regatta Terminal, then connect.



Looking at the Hints panel, we found the hint about Suricata Rules, read it!

The image shows the KringleCon menu interface. On the left, a sidebar lists various options: Story [23%], Destinations, Objectives, Hints (which is currently selected), Hats, Talks, Achievements, Settings, and [Exit]. The main content area is titled "Event Logs Exposé" and includes information from "Sparkle Redberry" via "Windows Event Logs". Below this, the "Built-In Hints" section also lists "Sparkle Redberry" and "Windows Event Logs". The final section, "The Tome of Suricata Rules", is attributed to "Dusty Giftwrap" via "Suricata Regatta". A note in this section states: "This is the official source for Suricata rule creation!"

The hint is about the Official Documentation of Suricata, it's very important to read it!

A screenshot of a web browser window displaying the official Suricata documentation. The URL is <https://suricata.readthedocs.io/en/suricata-6.0.0/rules/intro.html>. The page title is "6.1. Rules Format — Suricata 6.0.0 documentation". On the left, there is a sidebar with a navigation tree for the rules format, including sections like Action, Protocol, Source and destination, Ports, Direction, Rule options, Meta Keywords, IP Keywords, TCP keywords, UDP keywords, ICMP keywords, Payload Keywords, Transformations, Prefiltering Keywords, Flow Keywords, Bypass Keyword, HTTP Keywords, File Keywords, and DNS Keywords. The main content area is titled "6.1. Rules Format" and discusses signatures, rule management, and provides an example rule:

```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET TROJAN Likely Bot Nick in IRC (USA +..)"; flow:established,to_server; flowbitsisset,proto.irc; content:"NICK "; pcre:"/NICK .*USA.*[0-9]{3,}/"; reference:url,doc.emerginthreats.net/2008124; classtype:trojan-activity; sid:2008124; rev:2;)
```

The terminal showed the instructions for this challenge. Now the first question about the challenge is shown.

A terminal window with the following text:

```
Use your investigative analysis skills and the suspicious.pcap file to help develop Suricat a rules for the elves!

There's a short list of rules started in suricata.rules in your home directory.

First off, the STINC (Santa's Team of Intelligent Naughty Catchers) has a lead for us. They have some Dridex indicators of compromise to check out. First, please create a Suricata rule to catch DNS lookups for adv.eposttoday.uk. Whenever there's a match, the alert message (msg) should read Known bad DNS lookup, possible Dridex infection.

Add your rule to suricata.rules

Once you think you have it right, run ./rule_checker to see how you've done! As you get rules correct, rule_checker will ask for more to be added.

If you want to start fresh, you can exit the terminal and start again or cp suricata.rules backup suricata.rules

Good luck, and thanks for helping save the North Pole!
```

elf@018fa604e1a5:~\$ █

I was listing the directory to find files and found rule_checker and suricata.rules. We'll use it!

```
elf@cee7e209278c:~$ ls
HELP  logs  rule_checker  suricata.rules  suricata.rules.backup  suspicious.pcap
elf@cee7e209278c:~$ █
```

Looking at suricata.rules file it is possible to find an example for alerting to requests of DNS.

```
elf@cee7e209278c:~$ cat suricata.rules
alert http any any -> any any (msg:"FILE tracking PNG (1x1 pixel) (1)"; filemagic:"PNG image data, 1 x 1,"; sid:19; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET HUNTING Possible ELF executable sent when remote host claims to send a Text File"; flow:established,from_server; http.header; content:"Content-Type|3a 20|text/plain"; file.data; content:"|7f 45 4c 46|"; startswith; fast_pattern; isdataat:3000,relative; classtype:bad-unknown; sid:2032973; rev:1; metadata:updated_at 2021_05_18;)
alert ip any any -> any any (msg:"SURICATA IPv4 invalid checksum"; ipv4-csum:invalid; class type:protocol-command-decode; sid:2200073; rev:2;)
alert ip [199.184.82.0/24,199.184.223.0/24] any -> $HOME_NET any (msg:"ET DROP Spamhaus DROPPED Listed Traffic Inbound group 27"; reference:url,www.spamhaus.org/drop/drop.lasso; threshold: type limit, track_by_src, seconds 3600, count 1; classtype:misc-attack; flowbits:set,ET.Evil; flowbits:set,ET.DROPIP; sid:2400026; rev:3398; metadata:updated_at 2022_10_06;)
alert dns $HOME_NET any -> any any (msg:"ET WEB_CLIENT Malicious Chrome Extension Domain Request (stickies .pro in DNS Lookup)"; dns.query; content:"stickies.pro"; nocase; sid:2025218; rev:4;)
alert tcp any any -> any any (msg:"SURICATA Applayer No TLS after STARTTLS"; flow:established; app-layer-event:applayer_no_tls_after_starttls; flowint:applayer.anomaly.count,+,1; classtype:protocol-command-decode; sid:2260004; rev:2;)
alert pkthdr any any -> any any (msg:"SURICATA IPv4 total length smaller than header size"; decode-event:ipv4.iplen_smaller_than_hlen; classtype:protocol-command-decode; sid:2200002; rev:2;)
alert udp any any -> any 123 (msg:"ET DOS Possible NTP DDoS Inbound Frequent Un-Authed GET_RESTRICT Requests IMPL 0x02"; content:"|00 02 10|"; offset:1; depth:3; byte_test:1,!&,128,0; byte_test:1,&,4,0; byte_test:1,&,2,0; byte_test:1,&,1,0; threshold: type both,track_by_dst,count 2,seconds 60; classtype:attempted-dos; sid:2019021; rev:3; metadata:created_at 2014_08_26, updated_at 2014_08_26;)
elf@cee7e209278c:~$
```

Using the reference available I search for DNS Requests, so then I developed a similar rule.

```
alert dns $HOME_NET any -> any any (msg:"Know bad DNS lookup, e Dridex Infection (adv.epostoday.uk in DNS Lookup)"; dns.query; content:"adv.epostoday.uk"; nocase;sid:1234; rev:1;)
```

In this example, **red** is the action, **green** is the header and **blue** are the options.

```
alert tcp any any -> any any (msg:"SURICATA Applayer No TLS after STARTTLS"; flow:established; app-layer-event:applayer_no_tls_after_starttls; flowint:applayer.anomaly.count,+,1; classtype:protocol-command-decode; sid:2260004; rev:2;) alert pkthdr any any -> any any (msg:"SURICATA IPv4 total length smaller than header size"; decode-event:ipv4.iplen_smaller_than_hlen; classtype:protocol-command-decode; sid:2200002; rev:2;) alert udp any any -> any 123 (msg:"ET DOS Possible NTP DDoS Inbound Frequent Un-Authed GET_RESTRICT Requests IMPL 0x02"; content:"|00 02 10|"; offset:1; depth:3; byte_test:1,!,&,128,0; byte_test:1,&,4,0; byte_test:1,&,2,0; byte_test:1,&,1,0; threshold: type both,track by_dst,count 2,seconds 60; classtype:attempted-dos; sid:2019021; rev:3; metadata:created_at 2014_08_26, updated_at 2014_08_26;) alert dns $HOME_NET any -> any any (msg:"Know bad DNS lookup, e Dridex Infection (adv.epostoday.uk in DNS Lookup)"; dns.query; content:"adv.epostoday.uk"; nocase;sid:1234; rev:1;) elf@cee7e209278c:~$
```

After saving the suricata.rules file, I ran the ./rule_checker.

```
elf@cee7e209278c:~$ ./rule_checker
```

Look, the first rule looks good! Now, we have developed a Suricata rule that alerts whenever the infected IP address 192.185.57.242 communicates with an internal system over HTTP.

```
elf@cee7e209278c:~$ ./rule_checker
rm: cannot remove '/home/elf/logs/*': No such file or directory
11/1/2023 -- 23:10:37 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER mode
11/1/2023 -- 23:10:37 - <Notice> - all 3 packet processing threads, 4 management threads initialized, engine started.
11/1/2023 -- 23:10:37 - <Notice> - Signal Received. Stopping engine.
11/1/2023 -- 23:10:38 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
First rule looks good! ←
STINC thanks you for your work with that DNS record! In this PCAP, it points to 192.185.57.242.
Develop a Suricata rule that alerts whenever the infected IP address 192.185.57.242 communicates with internal systems over HTTP.
When there's a match, the message (msg) should read Investigate suspicious connections, possible Dridex infection

For the second indicator, we flagged 0 packet(s), but we expected 681. Please try again!
elf@cee7e209278c:~$
```

Looking at the suricata.rules it is possible to identify two examples about requests http.

```
elf@cee7e209278c:~$ cat suricata.rules | grep http
alert http any any -> any any (msg:"FILE tracking PNG (1x1 pixel) (1)"; filemagic:"PNG image data, 1 x 1,"; sid:19; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET HUNTING Possible ELF executable sent when remote host claims to send a Text File"; flow:established,from_server; http.header; content:"Content-Type|3a 20|text/plain"; file.data; content:"|7f 45 4c 46|"; startswith; fast_pattern; isdataat:3000,relative; classtype:bad-unknown; sid:2032973; rev:1; metadata:updated_at 2021_05_18;)
elf@cee7e209278c:~$
```

So, reading the documentation, I developed the rule below, notice the operator for this rule is "<>" its means the traffic is bi-direcional.

```
alert http 192.185.57.242 any <> $HOME_NET any (msg:"Investigate suspicious connections, possible Dridex infection"; sid:10000002; rev:1;)
```

In this example, red is the action, green is the header and blue are the options.

After saving the suricata.rules file, I ran the ./rule_checker.

```
elf@cee7e209278c:~$ ./rule_checker
```

The second rule looks good! Now, We heard that some naughty actors are using TLS certificates with a specific CN.

Develop a Suricata rule to match and alert on an SSL certificate for heardbellith.Icanwepeh.nagoya.

When your rule matches, the message (msg) should read Investigate bad certificates, possible Dridex infection.

```
elf@cee7e209278c:~$ ./rule_checker
11/1/2023 -- 23:28:35 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER mode
11/1/2023 -- 23:28:35 - <Notice> - all 3 packet processing threads, 4 management threads initialized, engine started.
11/1/2023 -- 23:28:35 - <Notice> - Signal Received. Stopping engine.
11/1/2023 -- 23:28:36 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
First rule looks good!

Second rule looks good! ←

We heard that some naughty actors are using TLS certificates with a specific CN.
Develop a Suricata rule to match and alert on an SSL certificate for heardbellith.Icanwepeh.nagoya.
When your rule matches, the message (msg) should read Investigate bad certificates, possible Dridex infection

For the third indicator, we flagged 0 packet(s), but we expected 1. Please try again!
```

Reading the Suricata official documentation, I found the topic about `tls.cert_serial`, then I tried it!

6.15.3. `tls.cert_serial`

Match on the serial number in a certificate.

Example:

```
alert tls any any -> any any (msg:"match cert serial"; \
    tls.cert_serial; content:"5C:19:B7:B1:32:3B:1C:A1"; sid:200012;)
```

For our rule number 3, we need to open the file `suspicious.pcap` on Wireshark, on Wireshark we need to find the requests to the bad certificate. For it, we used the filter “`tls.handshake.type==11`”.

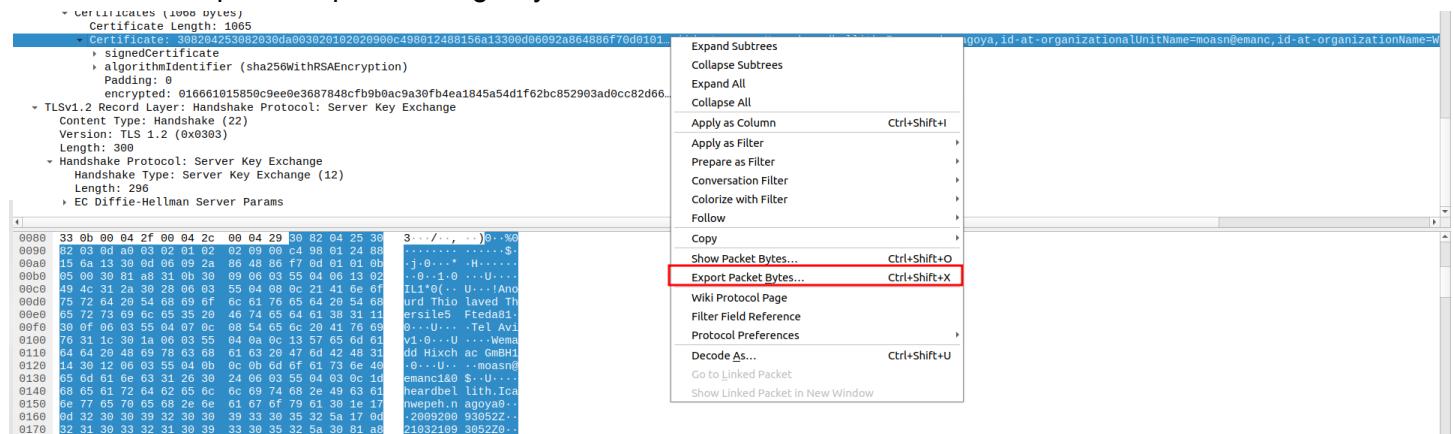
No.	Time	Source	Destination	Protocol	Length	Info
712	14.738514	204.79.197.219	16.9.24.101	TLSv1.2	353	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
808	1176.671322	151.236.219.181	16.9.24.101	TLSv1.2	1514	Server Hello, Certificate, Server Key Exchange
2071	1213.661188	204.79.197.200	16.9.24.101	TLSv1.2	1447	Server Hello, Certificate, Server Key Exchange, Server Hello Done
2132	1423.214066	204.79.197.200	16.9.24.101	TLSv1.2	1361	Server Hello, Certificate, Server Key Exchange, Server Hello Done
2172	1468.720245	49.122.160.14	16.9.24.101	TLSv1.2	1254	Server Hello, Certificate, Server Key Exchange, Server Hello Done
3903	1515.006134	62.98.109.30	16.9.24.101	TLSv1.2	1389	Server Hello, Certificate, Server Key Exchange, Server Hello Done
3922	1518.282138	62.98.109.30	16.9.24.101	TLSv1.2	1389	Server Hello, Certificate, Server Key Exchange, Server Hello Done
4747	1551.562567	62.98.109.30	16.9.24.101	TLSv1.2	1389	Server Hello, Certificate, Server Key Exchange, Server Hello Done
4815	2018.303445	52.137.110.235	16.9.24.101	TLSv1.2	1184	Server Hello, Certificate, Server Key Exchange, Server Hello Done
4843	2019.125481	52.143.86.214	16.9.24.101	TLSv1.2	1184	Server Hello, Certificate, Server Key Exchange, Server Hello Done
4872	2020.064899	96.6.239.82	16.9.24.101	TLSv1.2	1410	Certificate [TCP segment of a reassembled PDU]
4964	2020.835312	96.6.239.82	16.9.24.101	TLSv1.2	1410	Certificate [TCP segment of a reassembled PDU]
4998	2020.835712	96.6.239.82	16.9.24.101	TLSv1.2	1410	Certificate [TCP segment of a reassembled PDU]
4931	2020.836462	96.6.239.82	16.9.24.101	TLSv1.2	1514	Certificate [TCP segment of a reassembled PDU]
4964	2021.669076	96.6.239.82	16.9.24.101	TLSv1.2	1410	Certificate [TCP segment of a reassembled PDU]
4989	2021.689977	80.0.24.101	16.9.24.101	TLSv1.2	1410	Certificate [TCP segment of a reassembled PDU]
4976	2021.670996	80.6.239.82	16.9.24.101	TLSv1.2	1410	Certificate [TCP segment of a reassembled PDU]
5045	2220.803241	52.137.110.235	16.9.24.101	TLSv1.2	1185	Server Hello, Certificate, Server Key Exchange, Server Hello Done
5091	2515.898627	52.143.84.45	16.9.24.101	TLSv1.2	1184	Server Hello, Certificate, Server Key Exchange, Server Hello Done
5158	2886.435848	62.98.109.30	16.9.24.101	TLSv1.2	1389	Server Hello, Certificate, Server Key Exchange, Server Hello Done

Now, we need find to the `heardbellith.Icanwepeh.nagoya` bad certificates.

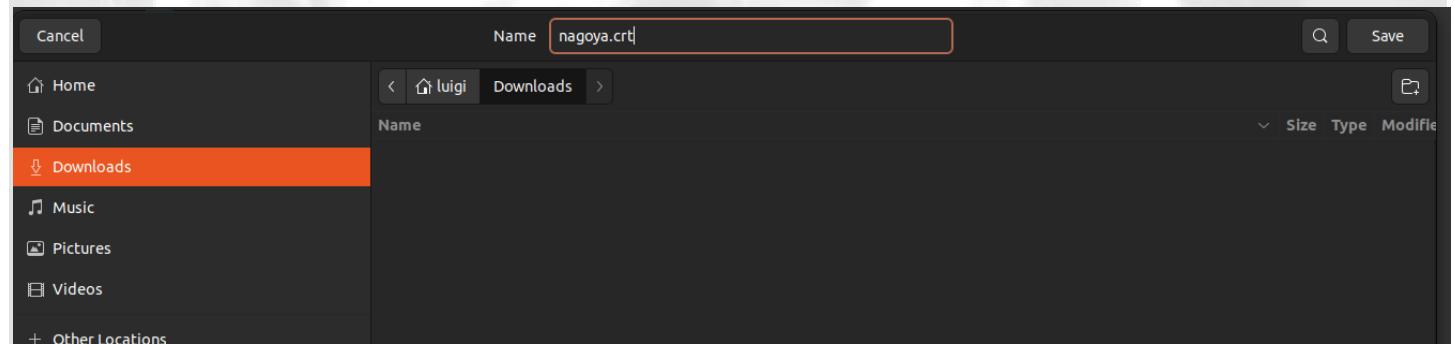
No.	Time	Source	Destination	Protocol	Length	Info
712	14.738514	204.79.197.219	16.9.24.101	TLSv1.2	353	Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hello Done
808	1176.671322	151.236.219.181	16.9.24.101	TLSv1.2	Server Hello, Certificate, Server Key Exchange	
2071	1213.661188	204.79.197.200	16.9.24.101	TLSv1.2	Server Hello, Certificate, Server Key Exchange, Server Hello Done	
2132	1423.214066	204.79.197.200	16.9.24.101	TLSv1.2	Server Hello, Certificate, Server Key Exchange, Server Hello Done	
2172	1468.720245	49.122.160.14	16.9.24.101	TLSv1.2	Server Hello, Certificate, Server Key Exchange, Server Hello Done	
3903	1515.006134	62.98.109.30	16.9.24.101	TLSv1.2	Server Hello, Certificate, Server Key Exchange, Server Hello Done	
3922	1518.282138	62.98.109.30	16.9.24.101	TLSv1.2	Server Hello, Certificate, Server Key Exchange, Server Hello Done	
4747	1551.562567	62.98.109.30	16.9.24.101	TLSv1.2	Server Hello, Certificate, Server Key Exchange, Server Hello Done	

Handshake Type: Certificate (11)
Length: 1071
Certificates Length: 1068
Certificates (1088 bytes)
Certificate Length: 1065
- Certificate: 308042530802030da003020102020900c498012488156a13300d06092a864886f70d010... (id-at-commonName=`heardbellith.Icanwepeh.nagoya` id-at-organizationalUnitName=`moasn@manc`, id-at-organizationName=`W`)
 + signedCertificate
 + algorithmIdentifier (`sha256WithRSAEncryption`)
 + Padding: 0
 + encrypted: 016661015850c9ee0e3087848cfb980ac9a30fb4ea1845a54d1f62bc852903ad0cc82d66...
- TLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)

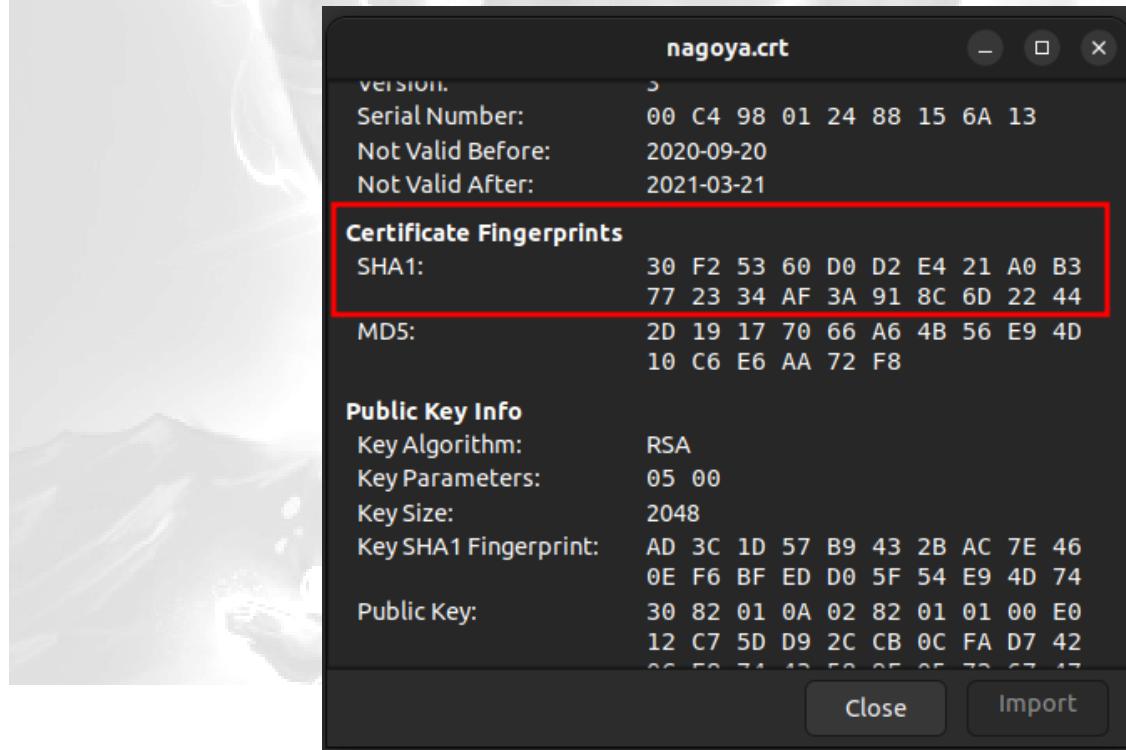
To develop the rule, we need to export the certificate and get the fingerprint, for it to click with the right button and select the option “Export Package Bytes”.



Save the certificate .crt in the folder of your preference.



Now, open the certificate properties, then search for Certificate Fingerprints, we'll use it in a rule.



I developed the rule below for tls which contained the fingerprint of the Nagoya certificate.

```
alert tls any any -> any any (msg:"Investigate bad certificates, possible Dridex infection"; tls.cert_fingerprint; content:"30:f2:53:60:d0:d2:e4:21:a0:b3:77:23:34:af:3a:91:8c:6d:22:44"; sid:200023;)
```

In this example, red is the action, green is the header and blue are the options.

```
elf@53e55de01d9a:~$ alert tls any any -> any any (msg:"Investigate bad certificates, possible Dridex infection"; tls.cert_fingerprint; content:"30:f2:53:60:d0:d2:e4:21:a0:b3:77:23:34:af:3a:91:8c:6d:22:44"; sid:200023;)
```

After saving the suricata.rules file, I ran the ./rule_checker.

```
elf@cee7e209278c:~/rule_checker
```

The third rule looks good! Now, OK, one more to rule them all and in the darkness find them.
Let's watch for one line from the JavaScript: let byteCharacters = atob
Oh, and that string might be GZip compressed - I hope that's OK!
Just in case they try this again, please alert on that HTTP data with message Suspicious JavaScript function, possible Dridex infection.

```
elf@53e55de01d9a:~$ ./rule_checker
rm: cannot remove '/home/elf/logs/*': No such file or directory
13/1/2023 -- 11:24:58 - <Notice> - This is Suricata version 6.0.8 RELEASE running in USER mode
13/1/2023 -- 11:24:58 - <Notice> - all 3 packet processing threads, 4 management threads initialized, engine started.
13/1/2023 -- 11:24:58 - <Notice> - Signal Received. Stopping engine.
13/1/2023 -- 11:24:58 - <Notice> - Pcap-file module read 1 files, 5172 packets, 3941260 bytes
First rule looks good!

Second rule looks good!

Third rule looks good! ←
```

OK, one more to rule them all and in the darkness find them.
Let's watch for one line from the JavaScript: let byteCharacters = atob
Oh, and that string might be GZip compressed - I hope that's OK!
Just in case they try this again, please alert on that HTTP data with message Suspicious JavaScript function, possible Dridex infection

For the fourth indicator, we flagged 0 packet(s), but we expected 1. Please try again!

```
elf@53e55de01d9a:~$
```

For question number 4, I search in Suricata Official Documentations for some things, but I can't find them. Then I search on Github and I find some examples.

In this example we have an alert for tcp with content inspection by string “POST”, then I have an idea to develop the rule using it!

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"ET TROJAN Possible Vundo Trojan Variant reporting to Controller"; flow:established,to_server; content:"POST "; depth:5; uricontent:"/frame.html?"; urilen:>80; classtype:trojan-activity; reference:url,doc.emergingthreats.net/2009173; reference:url,www.emergingthreats.net/cgi-bin/cvsweb.cgi/sigs/VIRUS/TROJAN_Vundo; sid:2009173; rev:2;)
```

The difference between `http.uri` and `uricontent` is the syntax:

My rule of Suricata was developed, the content string search in this case is “let byteCharacters = `atob`”, then I tried it!

```
alert tcp any any -> any any (msg: "Suspicious JavaScript function, possible Dridex infection"; file_data; content: "let byteCharacters = atob"; sid:10001234; rev:1;)
```

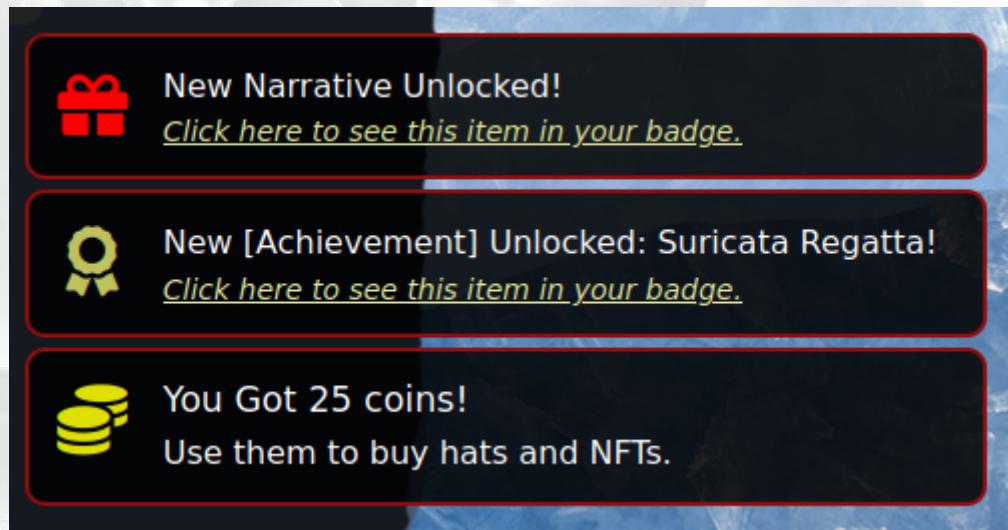
In this example, red is the action, green is the header and blue are the options.

```
elf@53e55de01d9a:~$ cat suricata.rules | grep atob
alert tcp any any -> any any (msg: "Suspicious JavaScript function, possible Dridex infection"; file_data; content: "let byteCharacters = atob"; sid:10001234; rev:1;)
elf@53e55de01d9a:~$ S
```

After saving the suricata.rules file, I ran the `./rule_checker`.

```
elf@cee7e209278c:~$ ./rule_checker
```

We finished this challenge.



Congratulations! You have completed
the Suricata Regatta challenge!

 [Tweet This!](#)