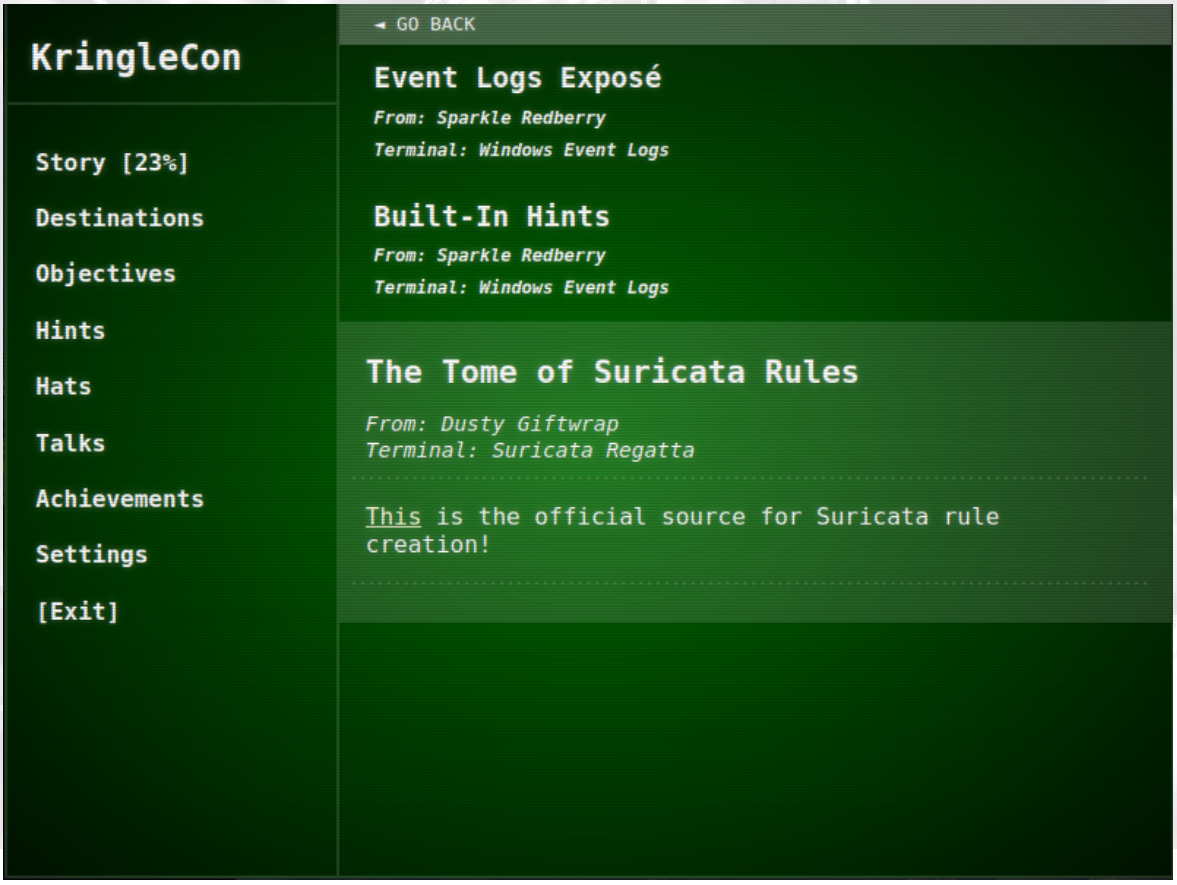
Write-up CTF
SANS - Holiday Hack Challenge 2022
Challenge: Windows Event Logs
Difficulty: Easy
https://github.com/brunopelegrini

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.

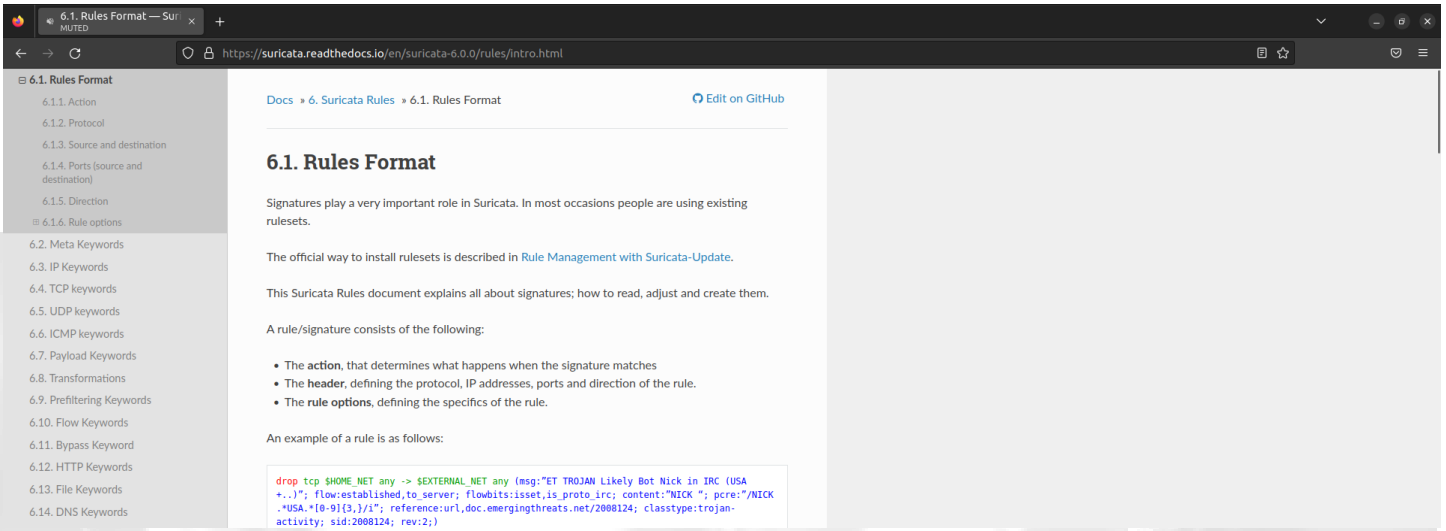
In the Kringlecon game, go to Tokien Ring tunnels and search for Suricata Regatta Terminal, then connect.

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

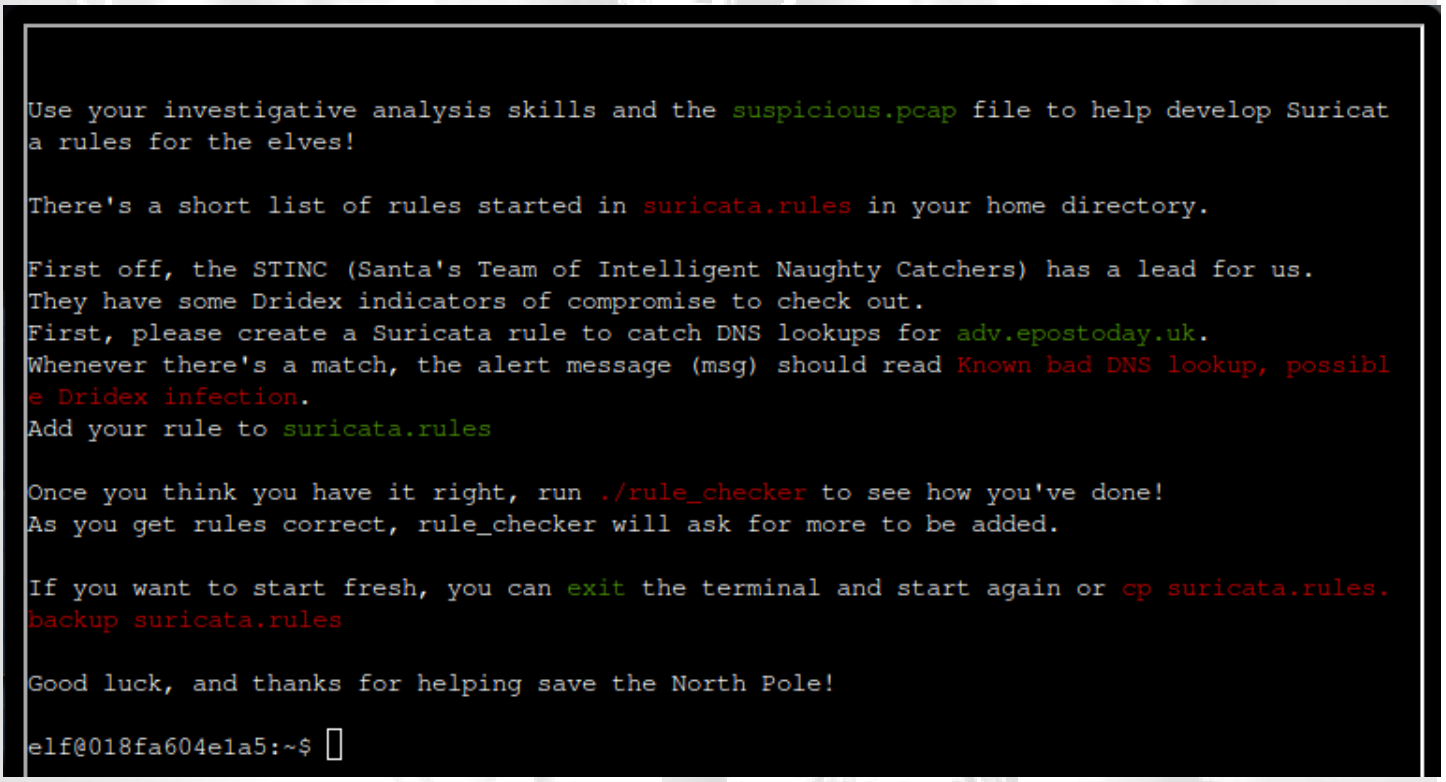


@badposeidon1
https://github.com/brunopelegrini

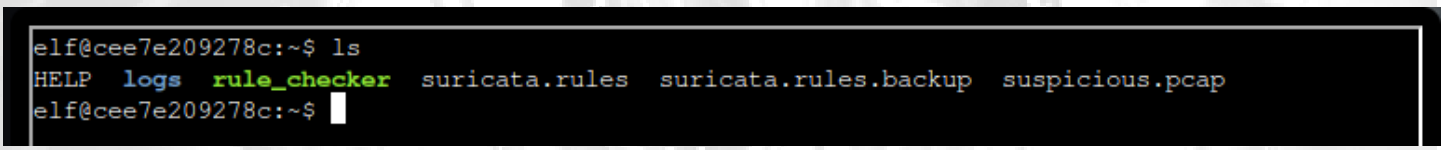
The hint is about the Official Documentation of Suricata, it's very important to read it!



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



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



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 imag
e 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; c
ontent:"Content-Type|3a 20|text/plain"; file.data; content:"|7f 45 4c 46|"; startswith; fas
t_pattern; isdataat:3000,relative; classtype:bad-unknown; sid:2032973; rev:1; metadata:upda
ted_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 DRO
P Listed Traffic Inbound group 27"; reference:url,www.spamhaus.org/drop/drop.lasso; thresho
ld: 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 $HOME_NET any -> any any (msg:"ET WEB_CLIENT Malicious Chrome Extension Domain Re
quest (stickies .pro in DNS Lookup)"; dns.query; content:"stickies.pro"; nocase; sid:202521
8; rev:4;)
alert tcp any any -> any any (msg:"SURICATA Applayer No TLS after STARTTLS"; flow:establish
ed; app-layer-event:applayer_no_tls_after_starttls; flowint:applayer.anomaly.count,+,1; cla
sstype:protocol-command-decode; sid:2260004; rev:2;)
alert pkthdr any any -> any any (msg:"SURICATA IPv4 total length smaller than header size";
 decode-event:ipv4.iplen_smaller_than_hlen; classtype:protocol-command-decode; sid:2200002;
 rev:2;)
alert udp any any -> any 123 (msg:"ET DOS Possible NTP DDoS Inbound Frequent Un-Authed GET_
RESTRICT Requests IMPL 0x02"; content:"|00 02 10|"; offset:1; depth:3; byte_test:1,!&,128,0
; byte_test:1,&,4,0; byte_test:1,&,2,0; byte_test:1,&,1,0; threshold: type both,track by_ds
t,count 2,seconds 60; classtype:attempted-dos; sid:2019021; rev:3; metadata:created_at 2014
_08_26, updated_at 2014_08_26;)
elf@cee7e209278c:~$
```

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:establish
ed; app-layer-event:applayer_no_tls_after_starttls; flowint:applayer.anomaly.count,+,1; cla
sstype: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_ds
t,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.epost
oday.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 m
ode
11/1/2023 -- 23:10:37 - <Notice> - all 3 packet processing threads, 4 management threads in
itialized, 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 byt
es
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 communi
cates with internal systems over HTTP.
When there's a match, the message (msg) should read Investigate suspicious connections, pos
sible 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 imag
e data, 1 x 1,"; sid:19; rev:1;)
alert http $EXTERNAL_NET any -> $HOME_NET any (msg:"ET HUNTING Possible ELF executable sent
 when remote host claims to send a Text File"; flow:established,from_server; http.header; c
ontent:"Content-Type|3a 20|text/plain"; file.data; content:"|7f 45 4c 46|"; startswith; fas
t_pattern; isdataat:3000,relative; classtype:bad-unknown; sid:2032973; rev:1; metadata:upda
ted_at 2021_05_18;)
elf@cee7e209278c:~$
```

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 m
ode
11/1/2023 -- 23:28:35 - <Notice> - all 3 packet processing threads, 4 management threads in
itialized, 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 byt
es
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, possibl
e 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!

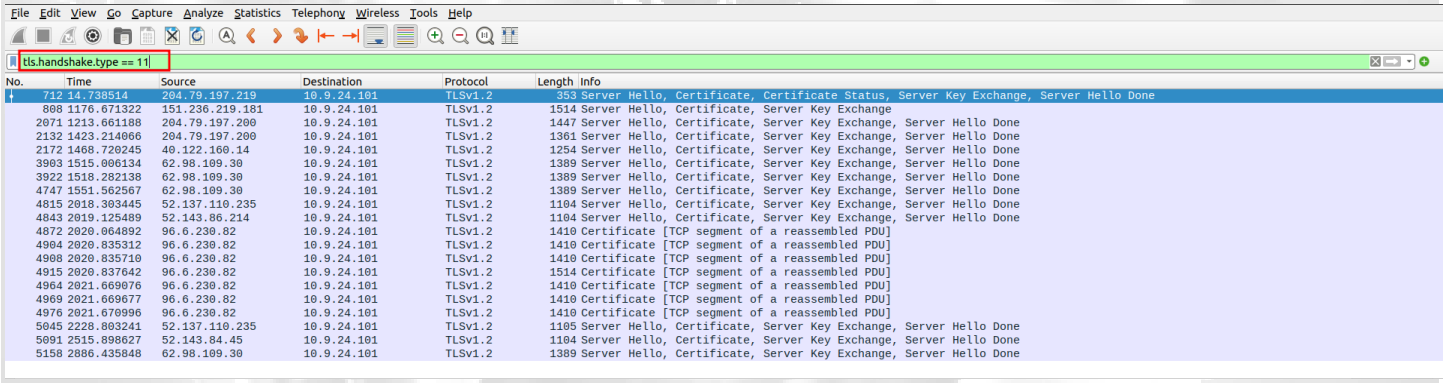
6.15.3. tls.cert_serial

Match on the serial number in a certificate.

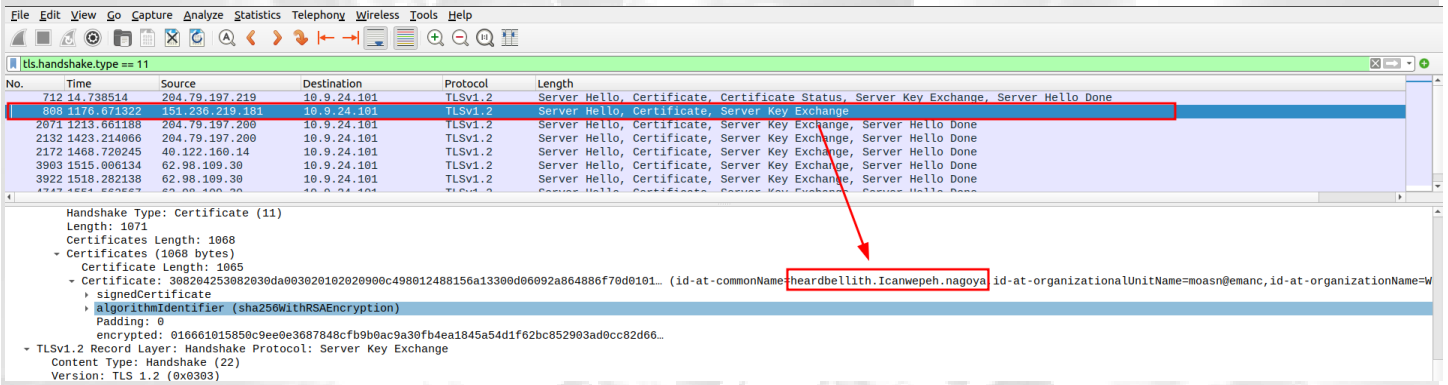
Example:

```
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”.

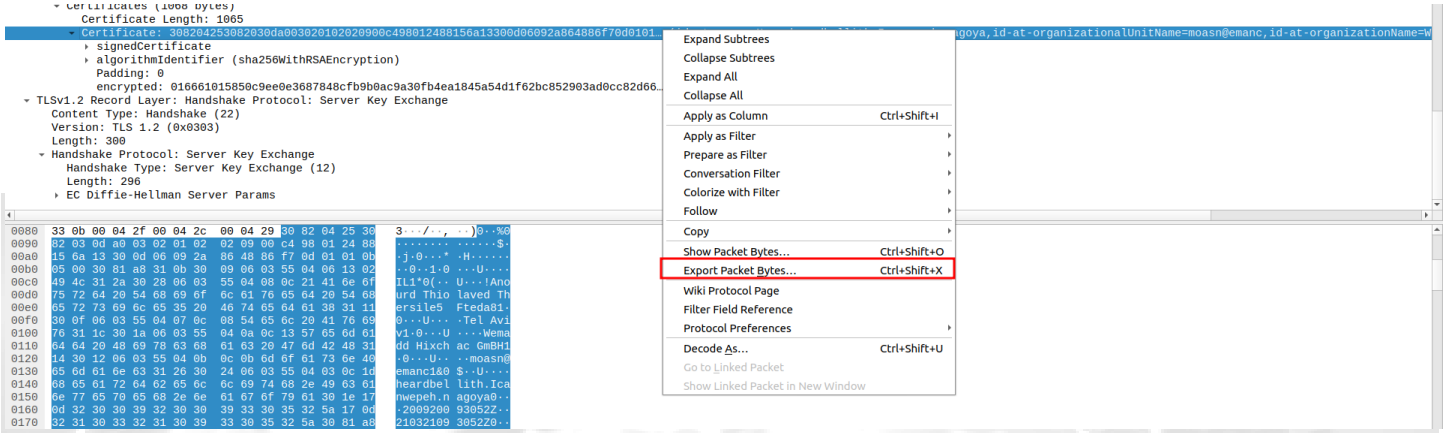


Now, we need find to the heardbellith.Icanwepeh.nagoya bad certificates.

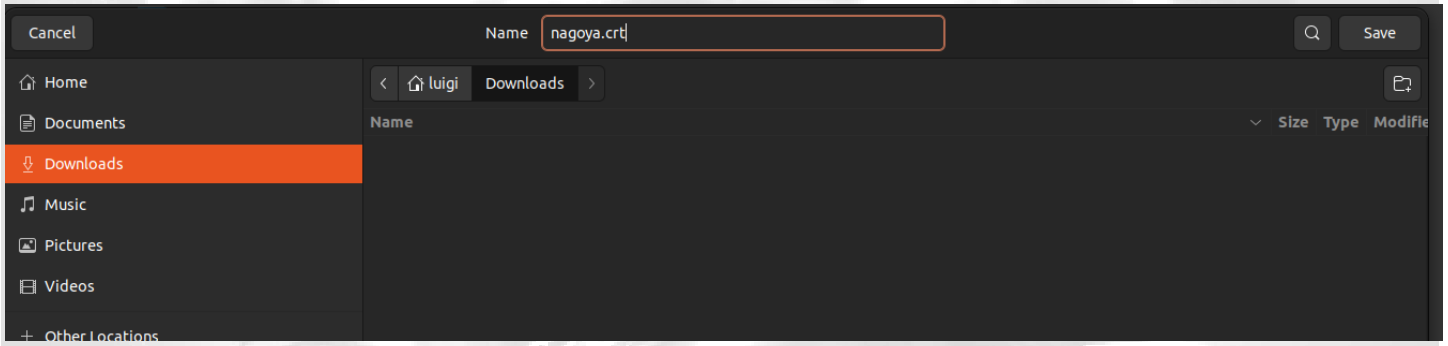


@badposeidon1
https://github.com/brunopelegrini

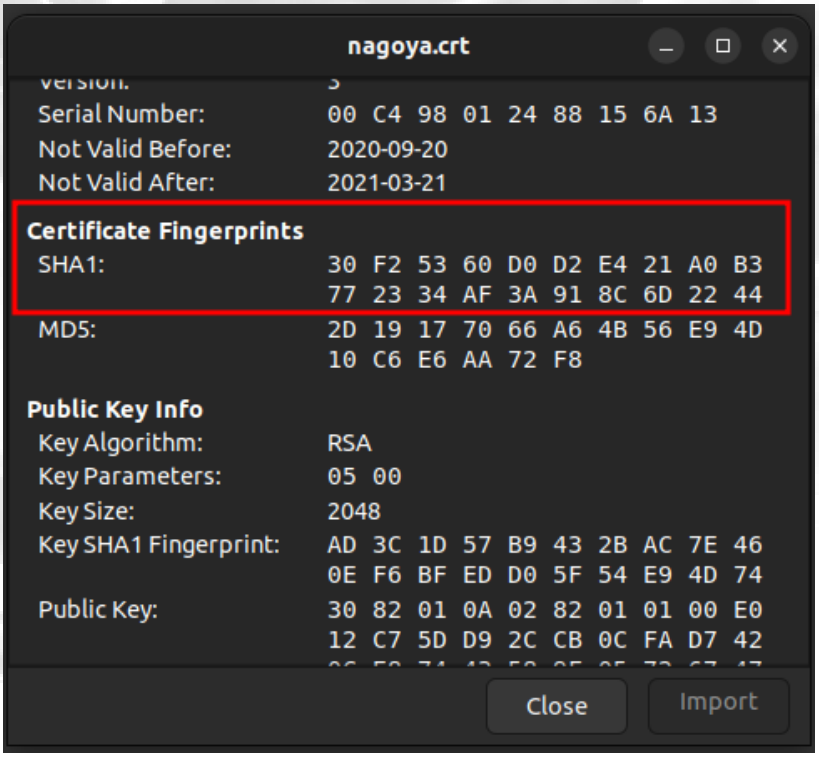
To develop the rule, we need to export the certificate and get the fingerprint, for it to click with the right button and select the option “Export Package Bytes”.



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.



@badposeidon1
https://github.com/brunopelegrini

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.

```
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:22:4
4"; sid:200023;)
elf@53e55de01d9a:~$
```

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

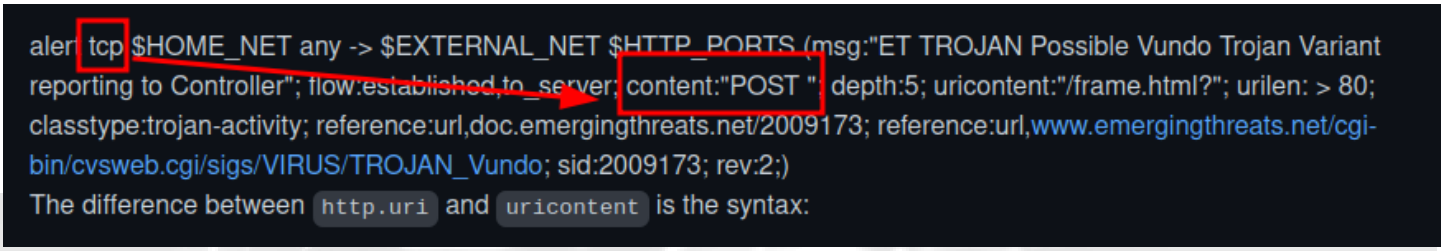
```
elf@cee7e209278c:~$ ./rule_checker
```

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 m
ode
13/1/2023 -- 11:24:58 - <Notice> - all 3 packet processing threads, 4 management threads in
itialized, 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 byt
es
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 Ja
vaScript 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.

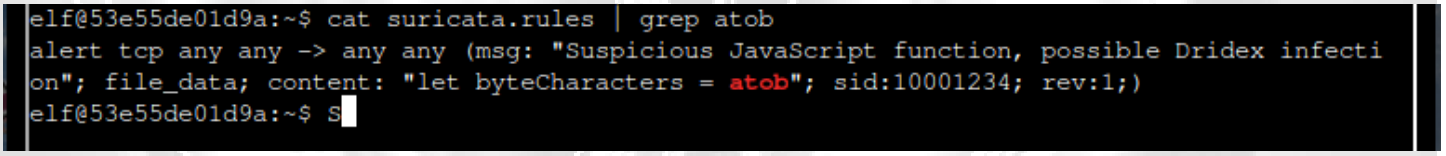
In this example we have an alert for tcp with content inspection by string “POST”, then I have an idea to develop the rule using it!



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

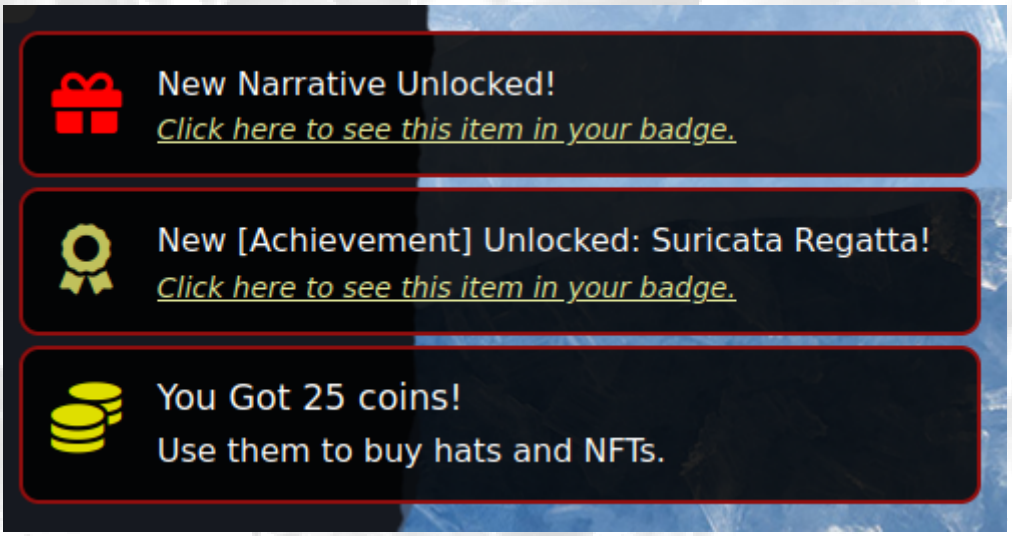
alert tcp any any -> any any (msg: "Suspicious JavaScript function, possible Dridex infection"; file_data; content: "let byteCharacters = atob"; sid:10001234; rev:1;)

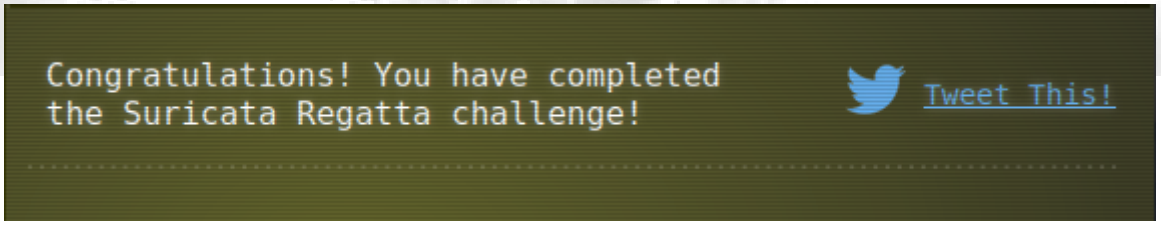
In this example, red is the action, green is the header and blue are the options.



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

We finished this challenge.





@badposeidon1
https://github.com/brunopelegrini