HELO Winnti: is that you?

Attack or Scan?

Stefano Ortolani, Jason Zhang

vb 2019 LONDON 2 – 4 October 2019

Speakers

Stefano Ortolani *@ostefano*

Director of Threat Research at Lastline, formerly GReAT Researcher at Kaspersky Lab

Jason Zhang *@jzhang88*

Senior Threat Researcher at Lastline, formerly Security Researcher at Sophos and Symantec

lastline

Threat Research @ Lastline

Lastline

- Network Detection and Response solution.
- Labs in Santa Barbara, Boston, and London.

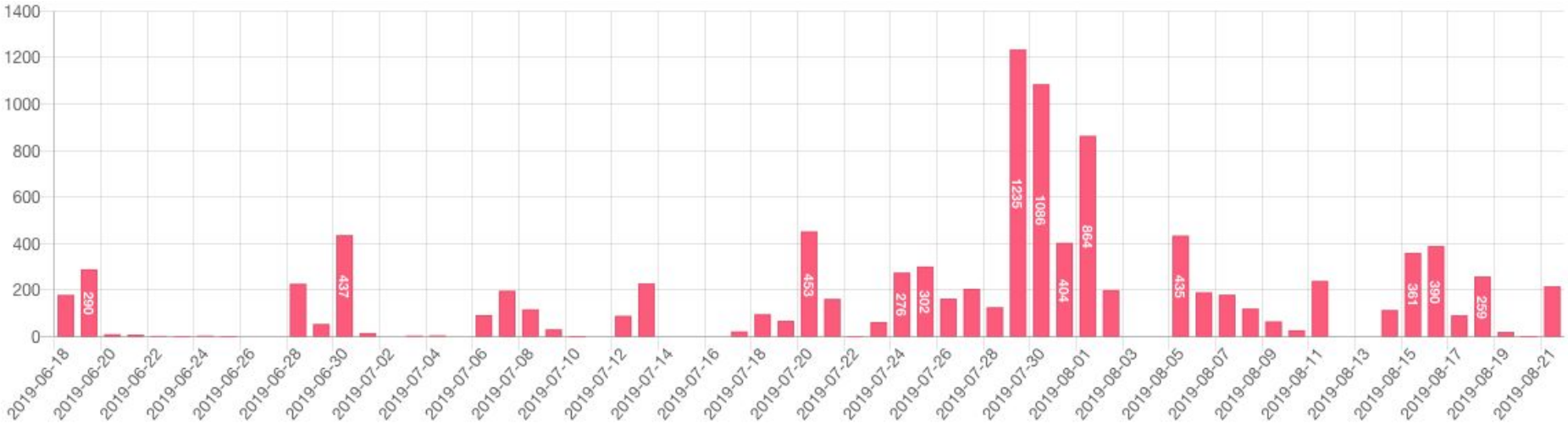
Threat Research Group

- Track and investigate threats to improve detection.
- Track and investigate anomalies:
  - Because there might be something wrong.
  - Because there might be something interesting.

lastline™

Something interesting...

- Two months of internal telemetry data since mid-June 2019.
- Silent signatures deployed on selected sensors.
- Winnti signature: https://github.com/TKCERT/winnti-suricata-lua



lastline™

Does it really matter?

- Winnti is one of the most complex and widely used toolkits.
- Amount of detections not fitting the profile of an advanced actor.
- Nor it was simply a background noise.
- We started digging...

Winnti

Evolution

Actors and Implants

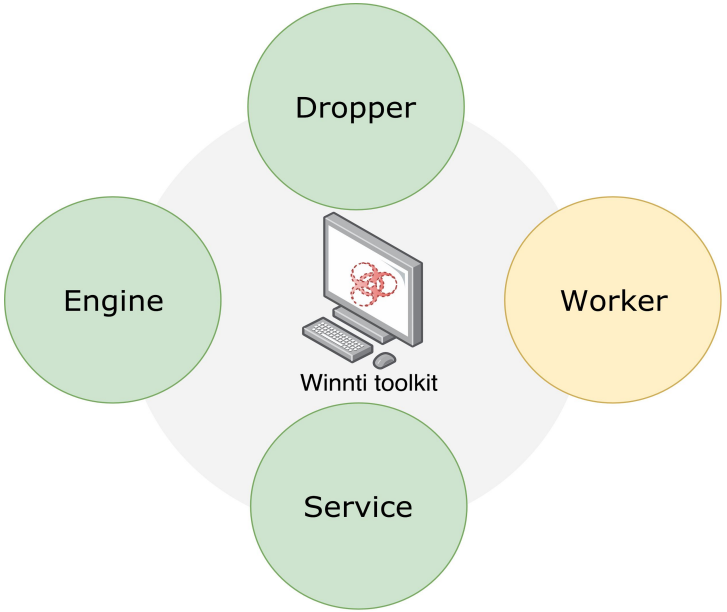
Implants and Targets

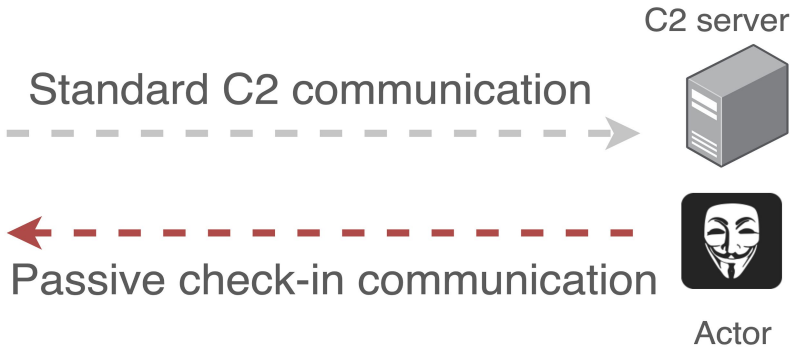
lastline™

Winnti and its components

Winnti represents a malware family with remote access trojan (RAT) functionalities:

- **Stealing** code-signing certificates.
- **Monetizing** stolen virtual funds.
- **Attacking** high-value organizations.

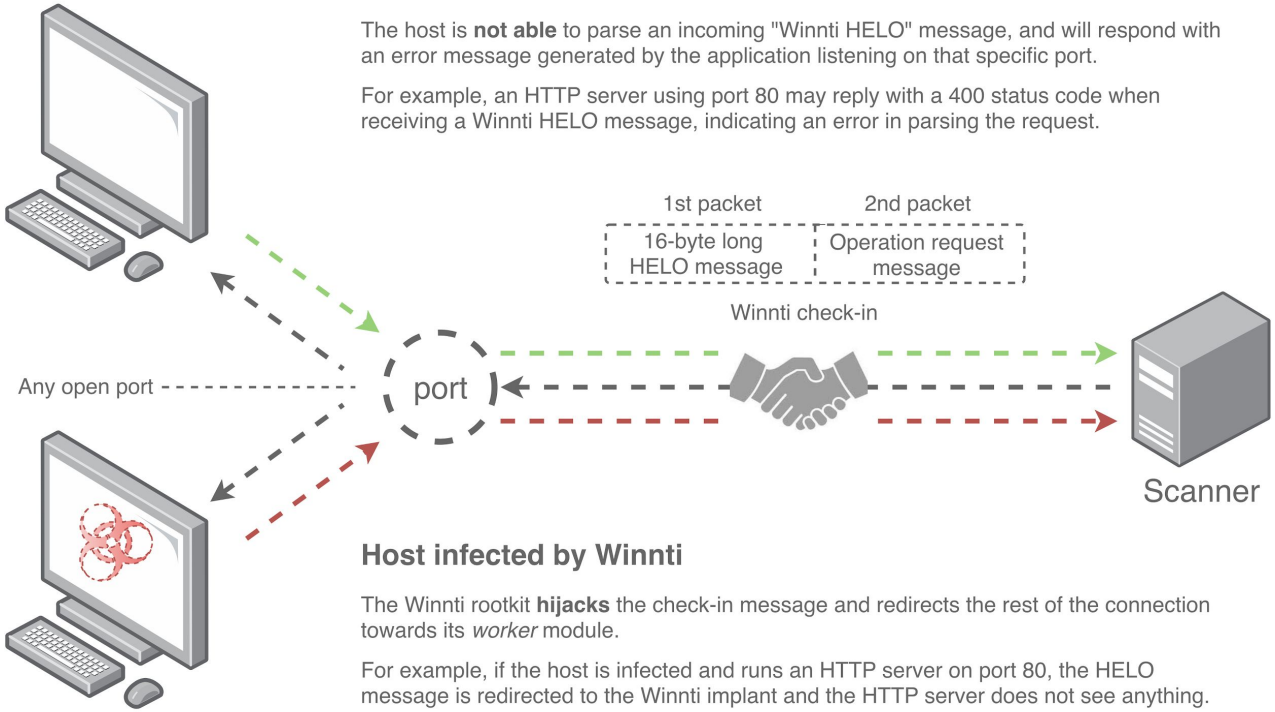




- **Dropper**: dropping the Winnti malware.
- **Worker**: communication and plugin management.
- **Service**: mainly for activating the engine component.
- **Engine**: fulfilling the malware installation process.

lastline™

Secondary communication channel



Host not infected by Winnti

The host is **not able** to parse an incoming "Winnti HELO" message, and will respond with an error message generated by the application listening on that specific port.

For example, an HTTP server using port 80 may reply with a 400 status code when receiving a Winnti HELO message, indicating an error in parsing the request.

Host infected by Winnti

The Winnti rootkit **hijacks** the check-in message and redirects the rest of the connection towards its *worker* module.

For example, if the host is infected and runs an HTTP server on port 80, the HELO message is redirected to the Winnti implant and the HTTP server does not see anything.

lastline™

… on the wire

```
0000   00 1c 7f 62 46 87 b0 c6 9a 15 50 7f 08 00 45 00   ...bF.....P...E.
0010   00 44 39 c9 40 00 32 06 4f e4 23 cc e8 25 83 70   .D9.@.2.O.#..%.p
0020   2f a5 bd 7e 00 50 53 85 6a 1f 6e 1e a8 87 80 18   /..~.PS.j.n.....
0030   00 de 1b 50 00 00 01 01 08 0a c4 21 18 0b 3e 54   ...P.......!..>T
0040   bf d3 1f a8 64 f8 29 38 c6 71 1d 06 bb 5d df a5   ....d.)8.q...]..
0050   02 ae
```

```
0000   00 1c 7f 62 46 87 b0 c6 9a 15 50 7f 08 00 45 00   ...bF.....P...E.
0010   00 96 39 ca 40 00 32 06 4f 91 23 cc e8 25 83 70   ..9.@.2.O.#..%.p
0020   2f a5 bd 7e 00 50 53 85 6a 2f 6e 1e a8 87 80 18   /..~.PS.j/n.....
0030   00 de ae 35 00 00 01 01 08 0a c4 21 18 0b 3e 54   ...5.......!..>T
0040   bf d3 62 66 00 42 42 42 42 42 f8 ce 83 e9 4e 1b   ..bf.BBBBB....N.
0050   31 4b 43 46 00 42 42 42 41 4a 0a a1 9d a0 21 74   1KCF.BBBAJ....!t
0060   cf 32 e8 42 42 42 42 42 69 69 42 42 42 42 6a 42   .2.BBBBBiiBBBBjB
0070   42 42 42 42 42 42 42 42 42 42 42 42 42 42 43 43   BBBBBBBBBBBBBBCC
0080   42 42 42 42 54 42 42 42 42 42 42 42 42 42 42 42   BBBBTBBBBBBBBBBB
0090   42 42 42 42 42 42 42 42 46 42 42 42 42 42 42 42   BBBBBBBBFBBBBBBB
00a0   42 42 04 00                                       BB..
```

"HELO" message:

- 16 bytes long.
- 4 dwords, 3 random.

"Operation Request" message:

- XOR encrypted.
- Key chosen by the client.

lastline™

Know thy enemy they say...

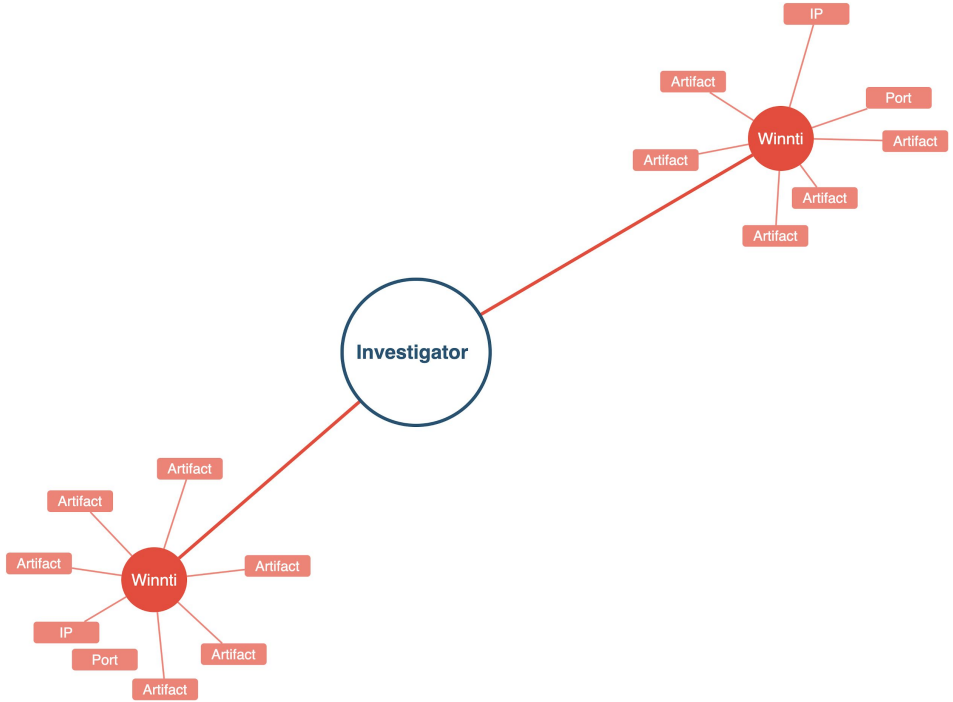
Is it an **attack** or a **scan**?

A simple question impacting:

- SOC analyst triaging network events.
- Researchers investigating and tracking Winnti activity.

lastline™

Impact for an investigator



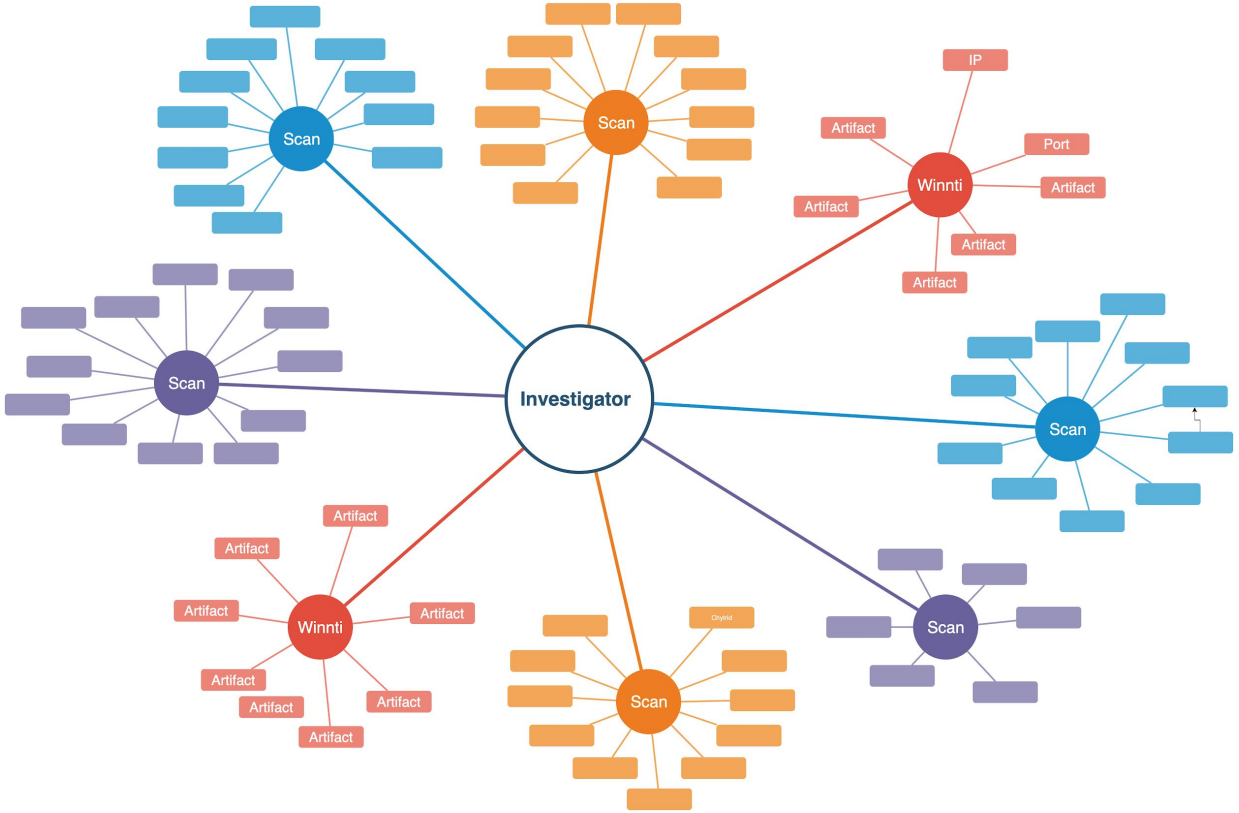
Before

A “Winnti HELO” message is the threat actor moving forward.

- SOC analysts could triage.
- Researchers could track.

lastline™

Impact for an investigator



Before

A “Winnti HELO” message is the threat actor moving forward.

- SOC analysts could triage.
- Researchers could track.

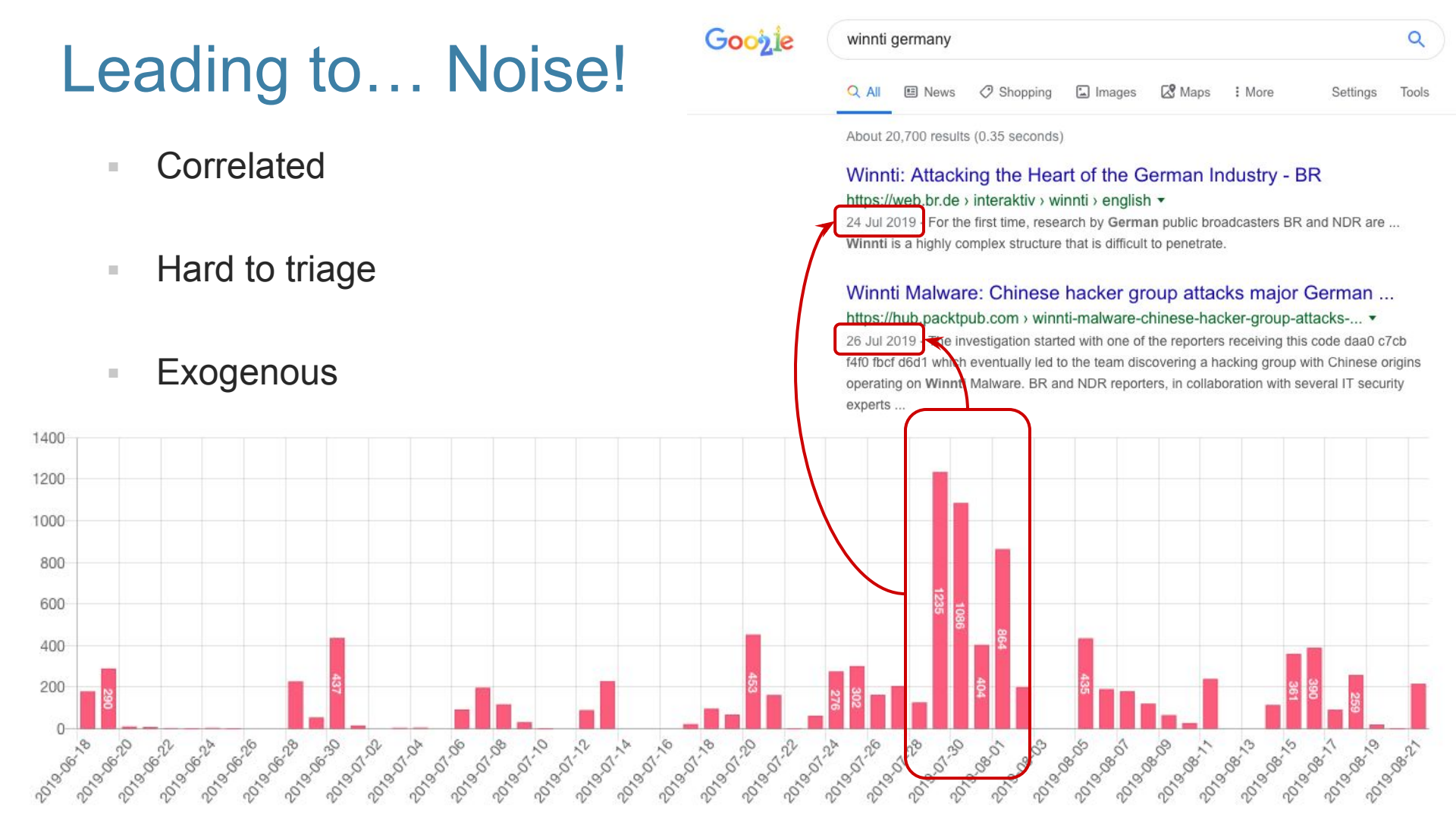
After

Polluted IOCs and benign connections hide threat actor actions in a sea of events.

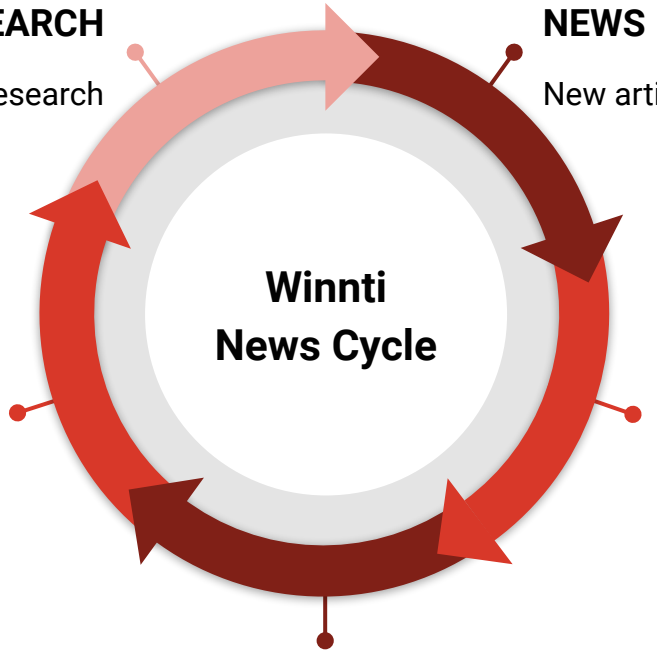
lastline™

Leading to… Noise!

- Correlated
- Hard to triage
- Exogenous



Self-sustaining Noise



RESEARCH

Triggering new research

NEWS

New article on prominent news source

INTEREST

More concern about current infections

SCANNERS

New scanners are deployed

DETECTION

Resulting in more detections

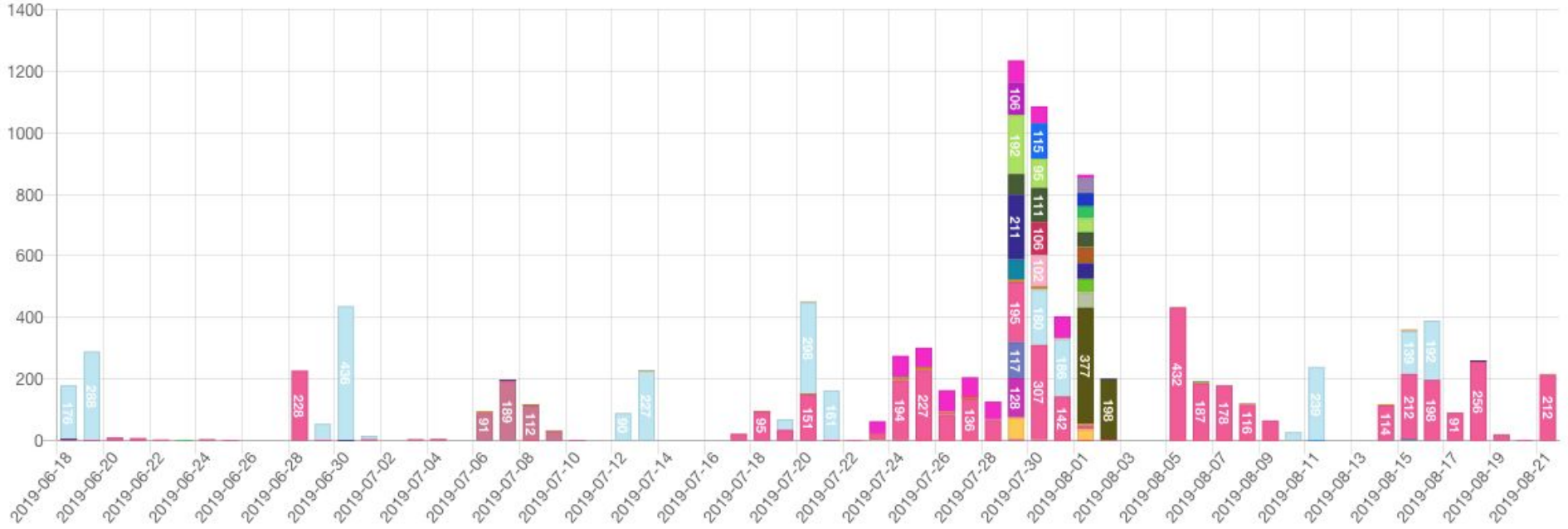
lastline™

Triaging a check-in

Rely on the source IP address?

More than 20 different IPs sending Winnti HELO messages over 2 months time.

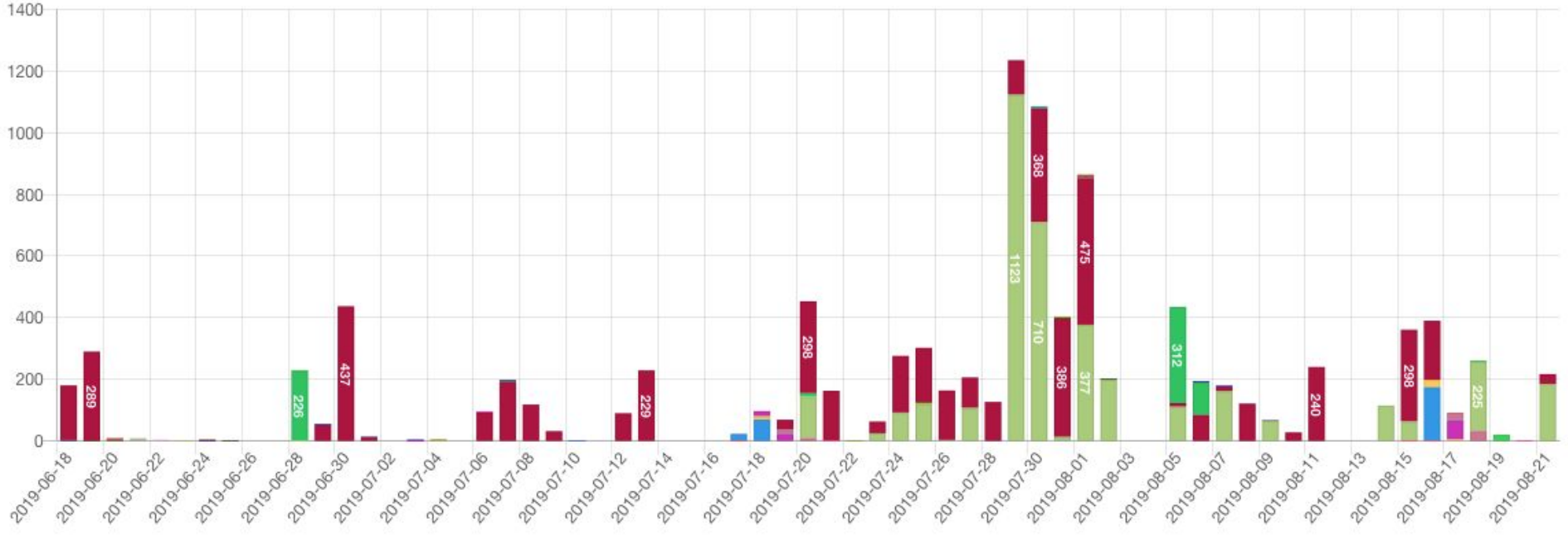
- Some explicitly marked as scanners, e.g., threatsinkhole.com.
- Some just a random virtual machines in the cloud, mainly linode and GCP.
- Even private IP addresses in enterprise settings, due to SNAT and port forwarding.



Rely on the destination port?

Winnti does not listen on a specific port, it can be 80 or 5648.

- **37%** of the traffic HTTP, **51%** HTTPS, then POP3, Telnet, etc…
- For a scan to be successful, the host must be exposed to the public Internet.
- Inherent bias: HTTP and HTTPs the most common exposed services.



Rely on connection status?

Only **25.8%** connection attempts were answered with an RST segment.

- Otherwise, if a port was open (or forwarded) there was a service listening on it.
- A successful connection and exchanged data just indicate that an application replied.
- Is it a Winnti implant or just web server?

Triaging means content inspection

```
 1  0.000000 35.203.53.10 → 10.112.4.65 TCP 74 8430 → 80 [SYN, ECN, CWR] Seq=0 Win=28400 Len=0 MSS=1420 SACK_PERM=1 TSval=3890810652 TSecr=0 WS=128
 2  0.001265 10.112.4.65 → 35.203.53.10 TCP 74 80 → 8430 [SYN, ACK, ECN] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=974335536 TSecr=[...]
 3  0.221930 35.203.53.10 → 10.112.4.65 TCP 66 8430 → 80 [ACK] Seq=1 Ack=1 Win=28416 Len=0 TSval=3890810875 TSecr=974335536
 4  0.259534 35.203.53.10 → 10.112.4.65 TCP 82 8430 → 80 [PSH, ACK] Seq=1 Ack=1 Win=28416 Len=16 TSval=3890810913 TSecr=974335536
 5  0.259536 35.203.53.10 → 10.112.4.65 TCP 164 8430 → 80 [PSH, ACK] Seq=17 Ack=1 Win=28416 Len=98 TSval=3890810913 TSecr=974335536
 6  0.260028 10.112.4.65 → 35.203.53.10 TCP 66 80 → 8430 [ACK] Seq=1 Ack=17 Win=14592 Len=0 TSval=974335795 TSecr=3890810913
 7  0.260335 10.112.4.65 → 35.203.53.10 TCP 66 80 → 8430 [ACK] Seq=1 Ack=115 Win=14592 Len=0 TSval=974335795 TSecr=3890810913
 8  0.481150 10.112.4.65 → 35.203.53.10 TCP 1474 80 → 8430 [ACK] Seq=1 Ack=115 Win=14592 Len=1408 TSval=974336016 TSecr=3890810913
 9  0.481154 10.112.4.65 → 35.203.53.10 TCP 1474 80 → 8430 [ACK] Seq=1409 Ack=115 Win=14592 Len=1408 TSval=974336016 TSecr=3890810913
10  0.481157 10.112.4.65 → 35.203.53.10 TCP 1474 80 → 8430 [ACK] Seq=2817 Ack=115 Win=14592 Len=1408 TSval=974336016 TSecr=3890810913
11  0.481159 10.112.4.65 → 35.203.53.10 TCP 1474 80 → 8430 [ACK] Seq=4225 Ack=115 Win=14592 Len=1408 TSval=974336016 TSecr=3890810913
12  0.481162 10.112.4.65 → 35.203.53.10 TCP 1474 80 → 8430 [ACK] Seq=5633 Ack=115 Win=14592 Len=1408 TSval=974336016 TSecr=3890810913
13  0.481164 10.112.4.65 → 35.203.53.10 TCP 1339 80 → 8430 [PSH, ACK] Seq=7041 Ack=115 Win=14592 Len=1273 TSval=974336016 TSecr=3890810913
14  0.518434 10.112.4.65 → 35.203.53.10 TCP 1474 80 → 8430 [ACK] Seq=8314 Ack=115 Win=14592 Len=1408 TSval=974336053 TSecr=3890810913
15  0.518438 10.112.4.65 → 35.203.53.10 TCP 1474 80 → 8430 [ACK] Seq=9722 Ack=115 Win=14592 Len=1408 TSval=974336053 TSecr=3890810913
16  0.518440 10.112.4.65 → 35.203.53.10 TCP 1474 80 → 8430 [ACK] Seq=11130 Ack=115 Win=14592 Len=1408 TSval=974336053 TSecr=3890810913
```

HELO + OPERATION REQUEST

DATA

lastline™

Content Inspection, port 80

Easy, it's plaintext

LINKS	TIMESTAMP	BYTES SENT	BYTES RECEIVED	PROTOCOL	INFO
	2019-07-31 00:11:46	↑ 450	↓ 591	HTTP	WINNTI

RAW IP

↑ 9\xcd\xc3V\x8c&\x12Dz/\xb7\xc0t\x96C\xe2

HELO

↓ HTTP/1.1 400 Bad Request
Server: nginx
Date: Wed, 31 Jul 2019 00:11:46 GMT
Content-Type: text/html
Content-Length: 166
Connection: close

<html>
<head><title>400 Bad Request</title></head>
<body bgcolor="white">
<h1>400 Bad Request</h1>
<hr> nginx
</body>
</html>

DATA

Content Inspection, port 25

Easy, it's plaintext

LINKS	TIMESTAMP	BYTES SENT	BYTES RECEIVED	PROTOCOL	INFO
	2019-08-17 02:31:31	↑ 394	↓ 262		WINNTI

RAW IP

HELO + OPERATION REQUEST

↑ 9\xcd\xc3V\x8c&\x12Dz/\xb7\xc0t\x96C\xe2 \x00F\x00\x00\x00\x00\x00\xba\x8c\xc1\xab\x00\x00\x00\x00\x01\x04F\x00\x00\x00\x03\x08\xbb=)\xb2\x17\xc4$\xed\xaa\x00\x00\x00\x00\x00++\x00\x00\x00\x00(\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x01\x00\x00\x00\x00\x16\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00

↓ 220 .jp ESMTP

DATA

Analyst Info

Threat description

Mitigation

Content Inspection, port 443

Ehr...

LINKS	TIMESTAMP	BYTES SENT	BYTES RECEIVED	URLS	PROTOCOL	INFO
	2019-07-31 07:59:47	↑ 584	↓ 391			WINNTI

RAW IP

↑ F\x9fu\x94\xb4\xdd\xe1\xf8\x17\x06%\xc8P\\Q\x99 **HELO**

↓ \x15\x03\x03\x00\x02\x02 **DATA**

↑ \x10\xca~\x980\xee<\x98\x8ab\xfd3<\xee<\x981\xea~\x980\xee?\x900\xee<\x980\xee<\x98\x9a\xee<\x980\xee\x17\xb30\xee<\x98\x18\xee<\x980\xee<\x980\xee<\x980\xee<\x981\xef<\x980\xee*\x980\xee<\x980\xee<\x980\xee<\x980\xee<\x980\xee8\x980\xee<\x980\xee<\x98\x04\x00

OPERATION REQUEST

Analyst Info

Threat description

Content Inspection, port 443

Can not rely on decryption, not even in enterprise settings.

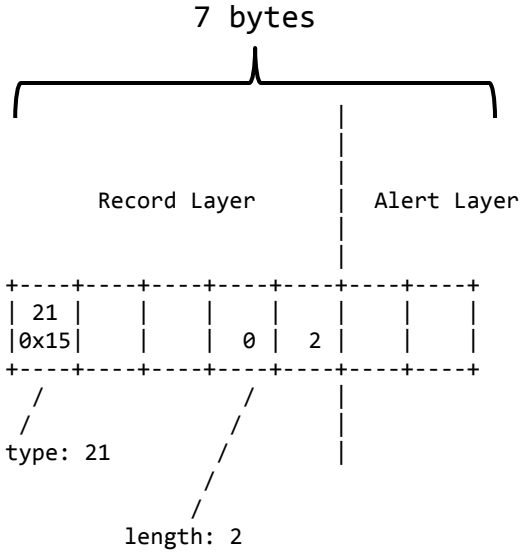
TLS (as per RFC5246) protocol replies with an alert when unexpected data is received.

- Alert record is **5 + 2** bytes long (record layer + alert layer).
- Winnti HELO message is at least **16 + X** bytes long (key exchange + data).

Any message smaller than **16** bytes (more than **50%** of our dataset) **is not Winnti**.

```
    [Window size scaling factor: 2]
    Checksum: 0x248c [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
    TCP payload (7 bytes)
▼ Transport Layer Security
  ▼ TLSv1.2 Record Layer: Alert (Level: Fatal, Description: Unexpected Message)
      Content Type: Alert (21)
      Version: TLS 1.2 (0x0303)
      Length: 2
    ▼ Alert Message
        Level: Fatal (2)
        Description: Unexpected Message (10)
```

```
0000  b0 c6 9a bf 47 c5 b0 c6  9a 15 50 7f 08 00 45 00   ····G··· ··P···E·
0010  00 3b 42 f9 40 00 7d 06  98 ba 83 70 04 ab 9b 5e   ·;B·@·}· ···p···^
0020  fe 8f 01 bb b5 84 25 14  c0 d9 48 9d 8b f9 80 18   ····%· ··H·····
0030  7f f8 24 8c 00 00 01 01  08 0a c9 9f aa 52 03 d0   ··$····· ·····R··
0040  a2 94 15 03 03 00 02 02  0a                      ···· ····· ·
```



Triaging a check-in, deterministically

Get the PCAP,

1. If the connection is RST, discard it.
2. If the first response is smaller than 16 bytes, discard it.
3. Inspect and filter all known application banners (SSH, Apache, etc…).
4. Escalate remaining network detections.

… 0 detection of 14972 alerts was a malicious Winnti HELO!

lastline

Content inspection is expensive

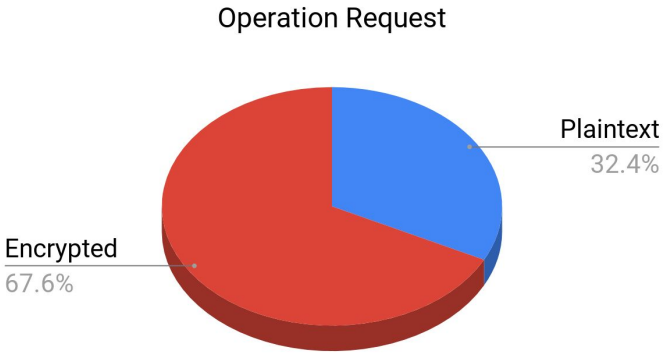
Are scanners predictable?

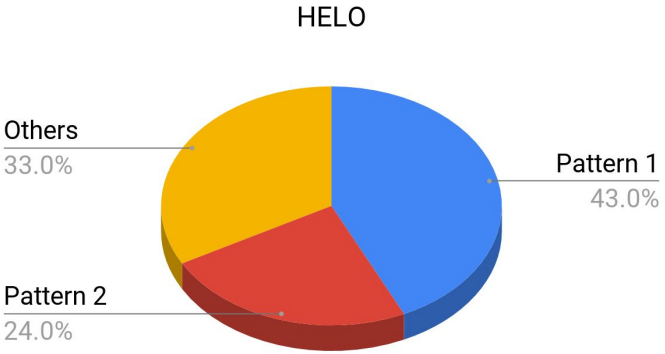
Anomaly 1

32.4% of all “Operation Request” messages were not encrypted.

Cause

Just a broken scanner.





Anomaly 2

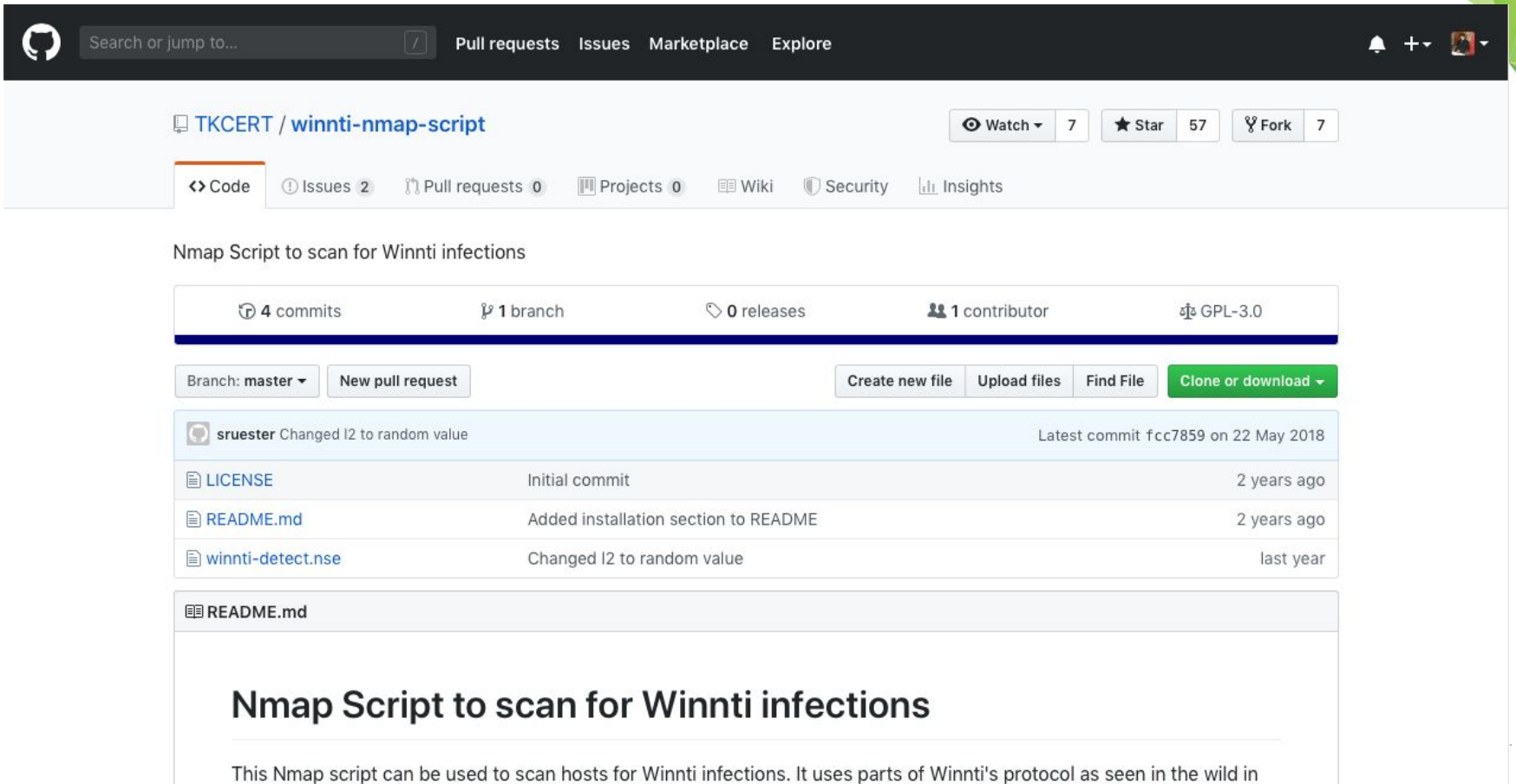
67% of all “HELO” messages used only two patterns, 43% and 24% respectively.

Cause

Unknown… so we investigated further

lastline™

The original Winnti scanner



```
                        "\x01\x04\x42\x00\x00\x00\x03\x08" ..
                        "\x48\xE3\xDF\xE2\x63\x36\x8D\x70" ..
                        "\xAA\x00\x00\x00\x00\x00\x2B\x2B" ..
                        "\x00\x00\x00\x00\x28\x00\x00\x00" ..
                        "\x00\x00\x00\x00\x00\x00\x00\x00" ..
                        "\x00\x00\x00\x00\x01\x01\x00\x00" ..
                        "\x00\x00\x16\x00\x00\x00\x00\x00" ..
                        "\x00\x00\x00\x00\x00\x00\x00\x00" ..
                        "\x00\x00\x00\x00\x00\x00\x04\x00" ..
                        "\x00\x00\x00\x00\x00\x00\x00\x00"
  -- The last two bytes 0400 define a message handler (maybe ^^)
  local enc_pkt = wnti_encrypt(pkt_queryhostinfo) .. "\x04\x00"

  stdnse.debug("Constructed QueryHostInfo packet: %s", tohex(enc_pkt))
  return enc_pkt
end


-- Return a WINNTI HELO packet
function wnti_get_helo_pkt()
  local l1 = math.random(1, 0xffffffff);
  local l2 = math.random(1, 0xffffffff);
  local l3 = math.random(1, 0xffffffff);

  local t3 = ( ( (l3 & 0xffff) << 16) | ((l3 & 0xffff0000) >> 16) )
  local l0 = t3 ~ l2
```

Winnti HELO message is 4 DWORDs:

- 3 randomly generated.
- 1 computed from 2 of those.

Resulting HELO is fully randomized:

- for each scanned port.
- for each execution.

… or is it?

```
$ nmap -sT 127.0.0.1 -p 80 --script ./winnti-detect.nse -d | grep "Constructed HELO" | xargs -IL date +"%Y%m%d_%H%M%S:L"
20190927_162159:NSE: [winnti-detect 127.0.0.1] Constructed HELO packet: F58F2454CE8A16D78C47F664D23079C8

$ nmap -sT 127.0.0.1 -p 80 --script ./winnti-detect.nse -d | grep "Constructed HELO" | xargs -IL date +"%Y%m%d_%H%M%S:L"
20190927_162208:NSE: [winnti-detect 127.0.0.1] Constructed HELO packet: F58F2454CE8A16D78C47F664D23079C8
```

Why?

Random numbers are (not) random

Nmap-based Winnti scanner relies on NSE, which is based on a Lua interpreter.
Lua has a known issue with `math.random()` in MacOS and FreeBSD:

- The difference of the seeds generated is very small.
- Seed often remains the same each time `math.random()` is called.
- Result: pseudo-random numbers not really random.

So much that `os.time()` is a better PRNG!

2 winnti-detect.nse

```
@@ -262,7 +262,7 @@ end
262 262  ----------------------------------------------------------------------------
263 263  function wnti_get_helo_pkt()
264 264    local l1 = math.random(1, 0xffffffff);
265     -  local l2 = os.time()
    265 +  local l2 = math.random(1, 0xffffffff);
266 266    local l3 = math.random(1, 0xffffffff);
267 267
268 268    local t3 = ( ( (l3 & 0xffff) << 16) | ((l3 & 0xffff0000) >> 16) )
```


Conclusions

- Network scanners are the **main culprit** when it comes to Winnti detections.
  - The more we talk about Winnti, the more network scanners we deploy, the more detections, the more we talk about Winnti…
  - Noise quickly impacts analysts' ability to triage detection, and researchers' efforts to track the threat actor.

- Triaging requires **content inspection**.
  - It can be expedited by relying on some properties of the Winnti protocol.
  - Some scanners can be fingerprinted and filtered out.

- Scripts using NSE are more **deterministic** than we might expect.
- All collected Winnti HELO messages **originated from scanners**.

lastline

Full report: https://www.lastline.com/labsblog/helo-winnti-attack-scan/

HELO Winnti: Attack or Scan?

POSTED BY **JASON ZHANG** AND **STEFANO ORTOLANI** ON SEP 30, 2019

LATEST LABS TWEETS

September 30, 2019

A massive increase of

Thank you