ACTIVE COUNTERMEASURES

Network Decoding C&C Channels - gcat

Brought to you by...

+ ACTIVE COUNTERMEASURES =

Red Team/Blue Team Awesomeness

This will be a series!

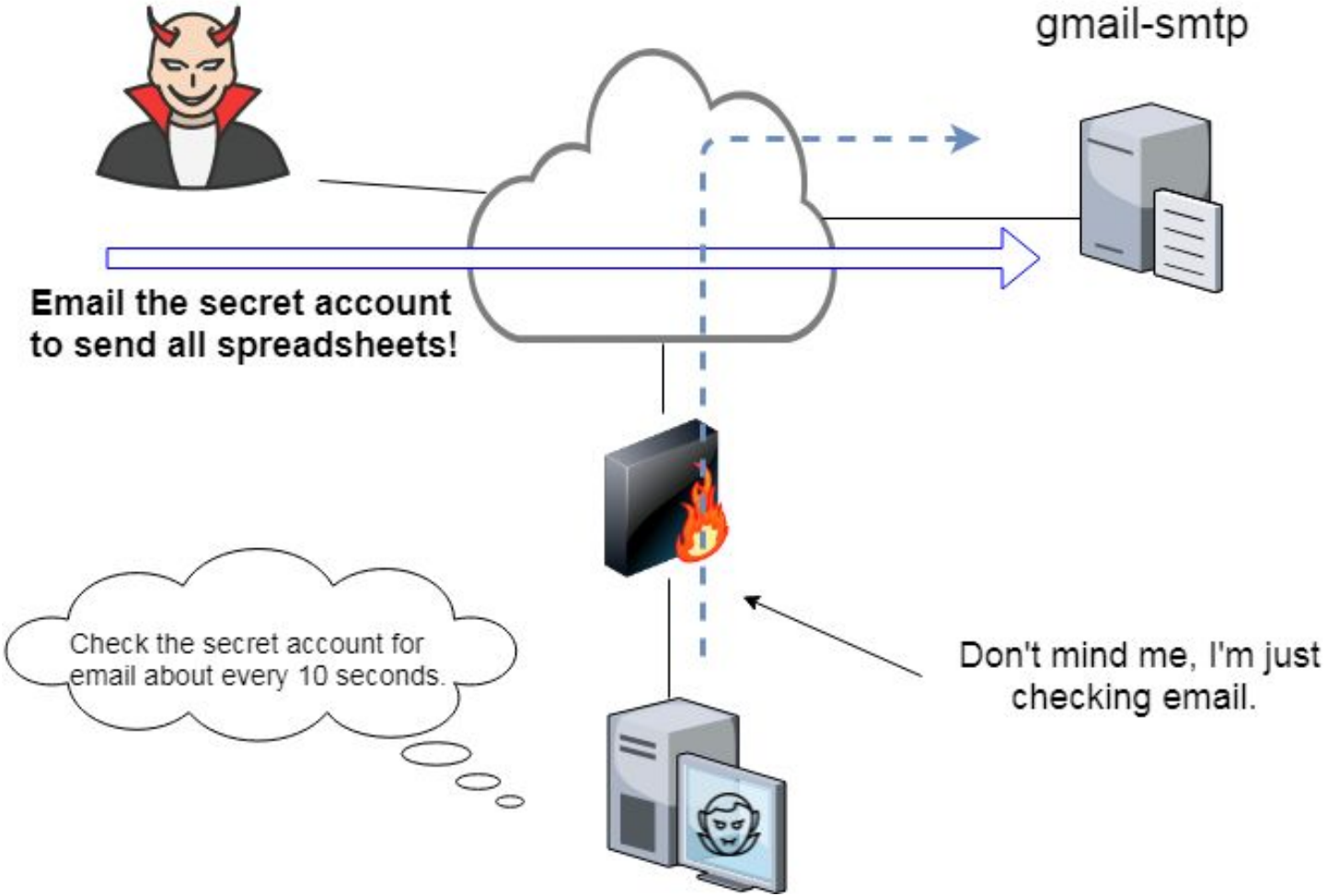
- Positive response to decoding dnscat2
- We've decided to make this a series
- Will dissect a C&C every few weeks
- Hit us up on Twitter if there is a C&C you want covered
  - @activecmeasures

What we will cover

- Deep dive on gcat
- Interesting in that many vendors ignore it
- We will show
  - What it looks like on the wire
  - Various methods of detection
    - Some scale easier than others
- Lab format so you can play along
  - Will make slides and Zeek logs available

gcat

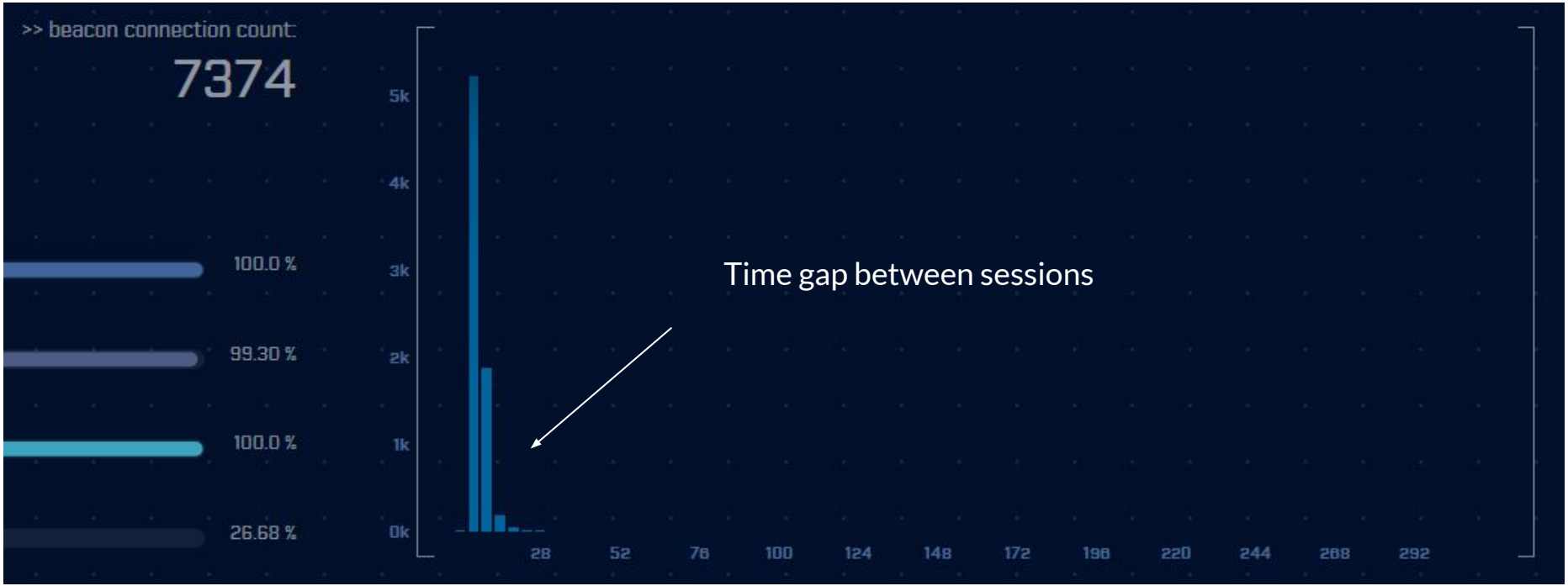
- Pretty simplistic C&C
  - But oh so hard to detect
- Basically, a Python based email client
- Communicates to GMail via IMAP4/TLS
  - Could easily be adapted to other mail services
  - Would not be that hard to adapt to other protocols
- Checks for email in an account you define
- Received email checked for commands


gmail-smtp
Email the secret account
to send all spreadsheets!
Check the secret account for
email about every 10 seconds.
Don't mind me, I'm just
checking email.

Some basic protections

- Uses IMAP4 over TLS
  - TCP/993 to check for commands
  - TCP/587 (SMTP/TLS) to send responses
  - Both can obviously be changed
- Can you lock this down?
  - Is there a business need for this traffic?
  - If not, close all remote email client traffic
  - Problematic if they switch to HTTPS
- The above applies to all public mail servers

Why is gcat hard to detect?



gcat uses the same signal timing as a regular email client

Let's work with Zeek (Bro)!

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
proto   service duration        orig_bytes      resp_bytes      conn_state      local_or
ig      local_resp      missed_bytes    history orig_pkts       orig_ip_bytes   resp_pkt
s       resp_ip_bytes   tunnel_parents
#types  time    string  addr    port    addr    port    enum    string  interval
count   count   string  bool    bool    count   string  count   count   count   count
set[string]
1518764388.106897       CUxfDy1yAfC0uE9x9i      192.168.88.2    13324   84.53.139.129
53      udp     dns     0.156880        73      91      SF      T       F       0
Dd      1       101     1       119     -
1518764388.264079       CERle52HPi1iLJ4wjh      192.168.88.2    23818   2.22.230.130
53      udp     dns     0.155248        69      87      SF      T       F       0
Dd      1       97      1       115     -
1518764388.419608       CBiJjv1w7hS6QIWJw5      192.168.88.2    52939   84.53.139.129
53      udp     dns     0.149188        69      85      SF      T       F       0
Dd      1       97      1       113     -
1518764383.094336       Cdgu4i16mvjFvFJKc9      10.55.100.111   62788   108.177.112.108
993     tcp     ssl     11.271044       991     4193    SF      T       F       0
ShADadfF        13      1523    17      4885    -
1518764333.507371       CTutuG4NoEQXFn6CD6      192.168.88.2    123     45.33.48.4
123     udp     -       0.081533        48      48      SF      T       F       0
Dd      1       76      1       76      -
:_
```

Absolute time only

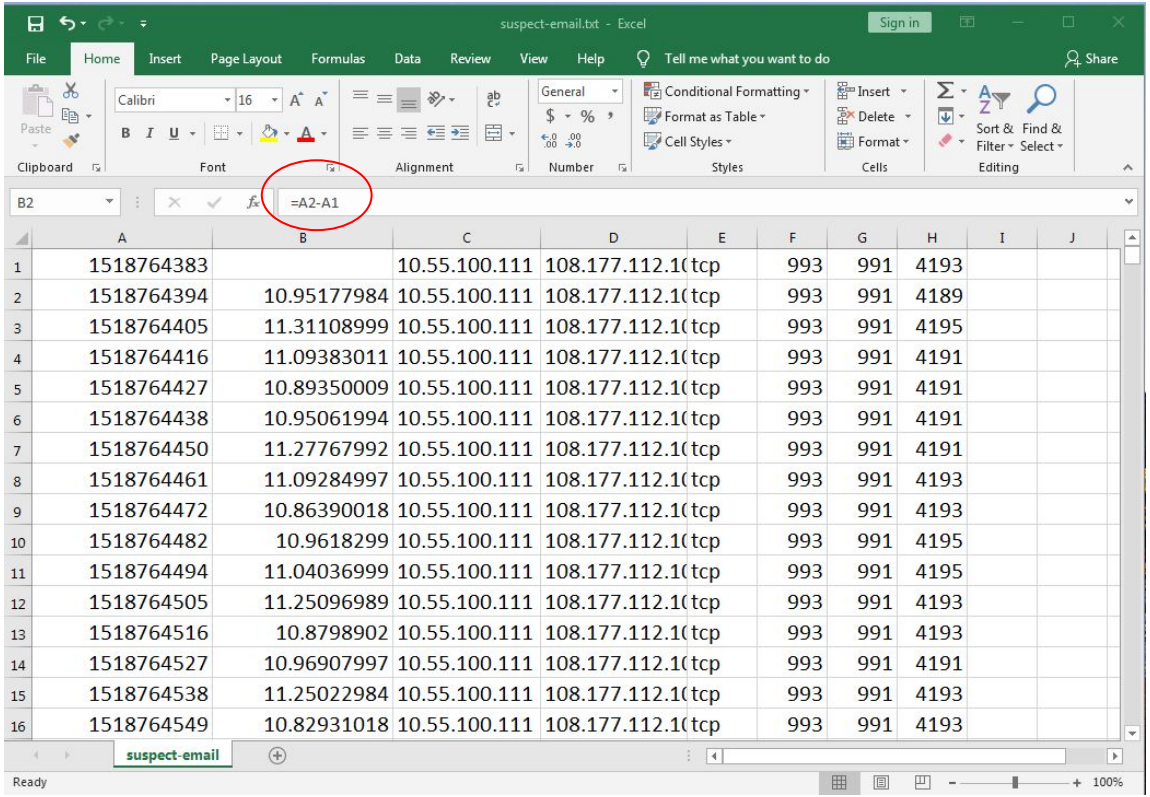
24-hours of data

```
cbrenton@cbrenton-3:~/test/2018-02-16$ zcat conn.* | bro-cut ts id.orig_h id.
resp_h proto id.resp_p orig_bytes resp_bytes | grep 108.177.112.108 | tr "\\t
" "," > suspect-email.txt
cbrenton@cbrenton-3:~/test/2018-02-16$ head suspect-email.txt
1518764383.094336,10.55.100.111,108.177.112.108,tcp,993,991,4193
1518764394.046118,10.55.100.111,108.177.112.108,tcp,993,991,4189
1518764405.357205,10.55.100.111,108.177.112.108,tcp,993,991,4195
1518764416.451039,10.55.100.111,108.177.112.108,tcp,993,991,4191
1518764427.344538,10.55.100.111,108.177.112.108,tcp,993,991,4191
1518764438.295155,10.55.100.111,108.177.112.108,tcp,993,991,4191
1518764449.572839,10.55.100.111,108.177.112.108,tcp,993,991,4191
1518764460.665683,10.55.100.111,108.177.112.108,tcp,993,991,4193
1518764471.529585,10.55.100.111,108.177.112.108,tcp,993,991,4193
1518764482.491416,10.55.100.111,108.177.112.108,tcp,993,991,4195
cbrenton@cbrenton-3:~/test/2018-02-16$
```

Other options

- tshark will print time deltas
- Time deltas let us analyze beacon timing
  - Need to look at the time gap between signals
- Zeek will only give us absolute time
  - In conn.log, other log formats support ts_delta
  - Doesn't matter - C&C and email use same timing
- Other options
  - What if we wanted to work with time deltas?
  - What other data can be analyzed for beacons?

Works but does not scale



	A	B	C	D	E	F	G	H	I	J
1	1518764383		10.55.100.111	108.177.112.1(	tcp	993	991	4193		
2	1518764394	10.95177984	10.55.100.111	108.177.112.1(	tcp	993	991	4189		
3	1518764405	11.31108999	10.55.100.111	108.177.112.1(	tcp	993	991	4195		
4	1518764416	11.09383011	10.55.100.111	108.177.112.1(	tcp	993	991	4191		
5	1518764427	10.89350009	10.55.100.111	108.177.112.1(	tcp	993	991	4191		
6	1518764438	10.95061994	10.55.100.111	108.177.112.1(	tcp	993	991	4191		
7	1518764450	11.27767992	10.55.100.111	108.177.112.1(	tcp	993	991	4191		
8	1518764461	11.09284997	10.55.100.111	108.177.112.1(	tcp	993	991	4193		
9	1518764472	10.86390018	10.55.100.111	108.177.112.1(	tcp	993	991	4193		
10	1518764482	10.9618299	10.55.100.111	108.177.112.1(	tcp	993	991	4195		
11	1518764494	11.04036999	10.55.100.111	108.177.112.1(	tcp	993	991	4195		
12	1518764505	11.25096989	10.55.100.111	108.177.112.1(	tcp	993	991	4193		
13	1518764516	10.8798902	10.55.100.111	108.177.112.1(	tcp	993	991	4193		
14	1518764527	10.96907997	10.55.100.111	108.177.112.1(	tcp	993	991	4191		
15	1518764538	11.25022984	10.55.100.111	108.177.112.1(	tcp	993	991	4193		
16	1518764549	10.82931018	10.55.100.111	108.177.112.1(	tcp	993	991	4193		

gcat - Focus on packets and bytes

```
cbrenton@cbrenton-3:~/test/2018-02-16$ zcat conn.* | bro-cut id.orig_h id.res
p_h proto id.resp_p orig_pkts resp_pkts orig_bytes resp_bytes | grep 108.177.
112.108 | tr "\\t" "," > analyze-email.txt
cbrenton@cbrenton-3:~/test/2018-02-16$ head analyze-email.txt
10.55.100.111,108.177.112.108,tcp,993,13,17,991,4193
10.55.100.111,108.177.112.108,tcp,993,13,17,991,4189
10.55.100.111,108.177.112.108,tcp,993,13,17,991,4195
10.55.100.111,108.177.112.108,tcp,993,13,17,991,4191
10.55.100.111,108.177.112.108,tcp,993,13,17,991,4191
10.55.100.111,108.177.112.108,tcp,993,13,17,991,4191
10.55.100.111,108.177.112.108,tcp,993,13,17,991,4191
10.55.100.111,108.177.112.108,tcp,993,13,18,991,4193
10.55.100.111,108.177.112.108,tcp,993,13,17,991,4193
10.55.100.111,108.177.112.108,tcp,993,14,17,991,4195
cbrenton@cbrenton-3:~/test/2018-02-16$ _
```

Consistency in packet quantity

```
cbrenton@cbrenton-3:~/test/2018-02-16$ cut -d ',' -f 5 analyz
e-email.txt | Rscript -e 'y <-scan("stdin", quiet=TRUE)' -e '
cat(min(y), max(y), mean(y), sd(y), sep="\n")'
1
18
13.30978
0.4911679
cbrenton@cbrenton-3:~/test/2018-02-16$ cut -d ',' -f 6 analyz
e-email.txt | Rscript -e 'y <-scan("stdin", quiet=TRUE)' -e '
cat(min(y), max(y), mean(y), sd(y), sep="\n")'
1
22
17.30693
0.5159443
cbrenton@cbrenton-3:~/test/2018-02-16$ _
```

Consistency in data transferred

```
cbrenton@cbrenton-3:~/test/2018-02-16$ cut -d ',' -f 7 analyz
e-email.txt | Rscript -e 'y <-scan("stdin", quiet=TRUE)' -e '
cat(min(y), max(y), mean(y), sd(y), sep="\n")'
0
1049
990.8463
11.79306
cbrenton@cbrenton-3:~/test/2018-02-16$ cut -d ',' -f 8 analyz
e-email.txt | Rscript -e 'y <-scan("stdin", quiet=TRUE)' -e '
cat(min(y), max(y), mean(y), sd(y), sep="\n")'
0
5451
4191.595
51.74911
cbrenton@cbrenton-3:~/test/2018-02-16$ _
```

Let's look at it with RITA

- Open source tool supported by ACM
- Designed to identify C&C channels
- Command line based, but powerful
- Will identify
  - Beacons
  - Long connections
  - Suspect DNS
  - Blacklist communications
  - Plus a whole lot more

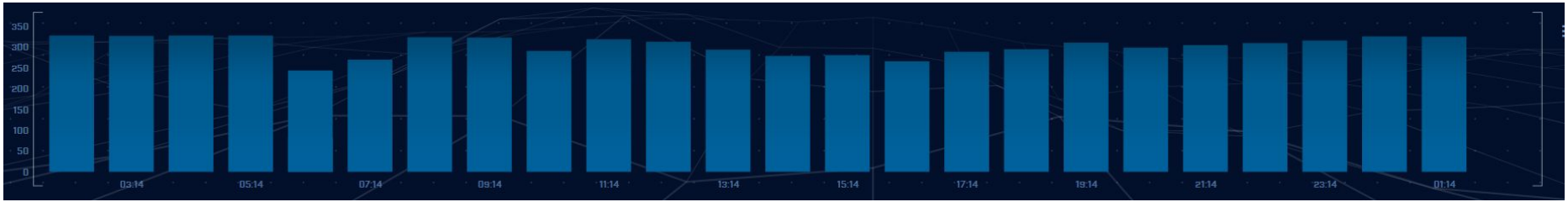
What RITA detected

```
cbrenton@cbrenton-3:~/testing/gcat-testing$ sudo rita show-beacons suspect-email | head -10
Score,Source IP,Destination IP,Connections,Avg Bytes,Intvl Range,Size Range,Top Intvl,Top Size
,Top Intvl Count,Top Size Count,Intvl Skew,Size Skew,Intvl Dispersion,Size Dispersion
0.905,10.55.200.10,172.16.200.11,71588,75,227,262,1,68,5223,5739,0,0.384615,0,6
0.874,10.55.100.111,108.177.112.108,7374,6437,309,46697,11,1523,4868,5078,0,0,0,0
0.836,10.55.200.11,205.251.193.184,324,308,349,4,300,70,173,313,0,0,0,0
0.834,10.55.254.100,91.189.89.199,42,152,1,0,2048,76,31,42,0,0,0,0
0.834,10.55.254.103,52.165.231.192,51,490,4562,2227,1680,153,35,30,0,0,0,0
0.834,10.55.254.107,91.189.91.157,42,152,1,0,2048,76,31,42,0,0,0,0
0.833,10.55.100.106,23.52.161.212,29,5489,1800,52,1800,505,24,24,0,0,0,0
0.833,10.55.100.103,23.52.161.212,30,5484,1,92,1800,505,24,27,0,0,0,0
0.833,10.55.100.100,23.52.161.212,31,5493,67,92,1800,505,25,23,0,0,0,0
cbrenton@cbrenton-3:~/testing/gcat-testing$ _
```

87.4% certain this is a beacon
Usually > 90% is actionable

Reminder of why this is hard

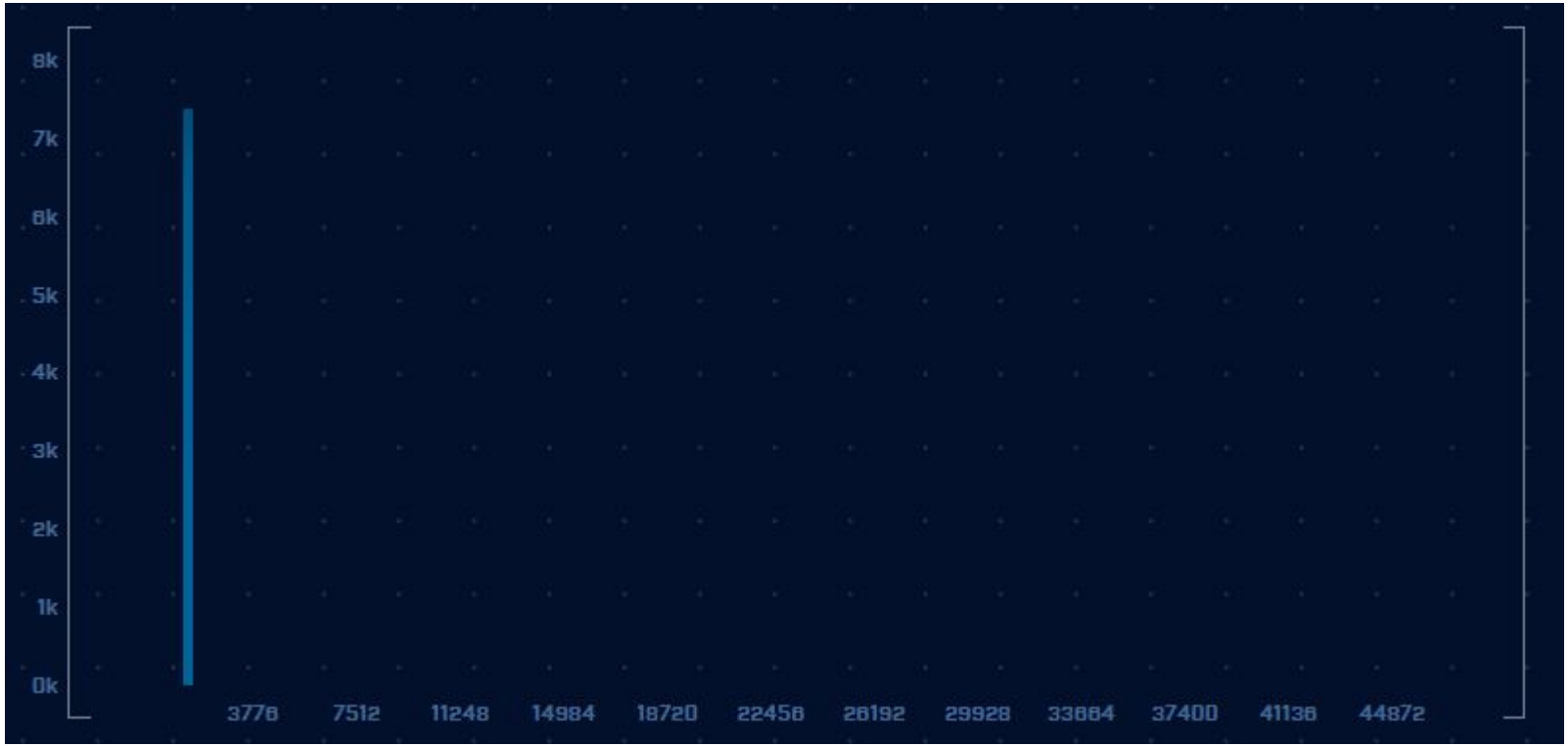
Plot of session activity over 24 hours



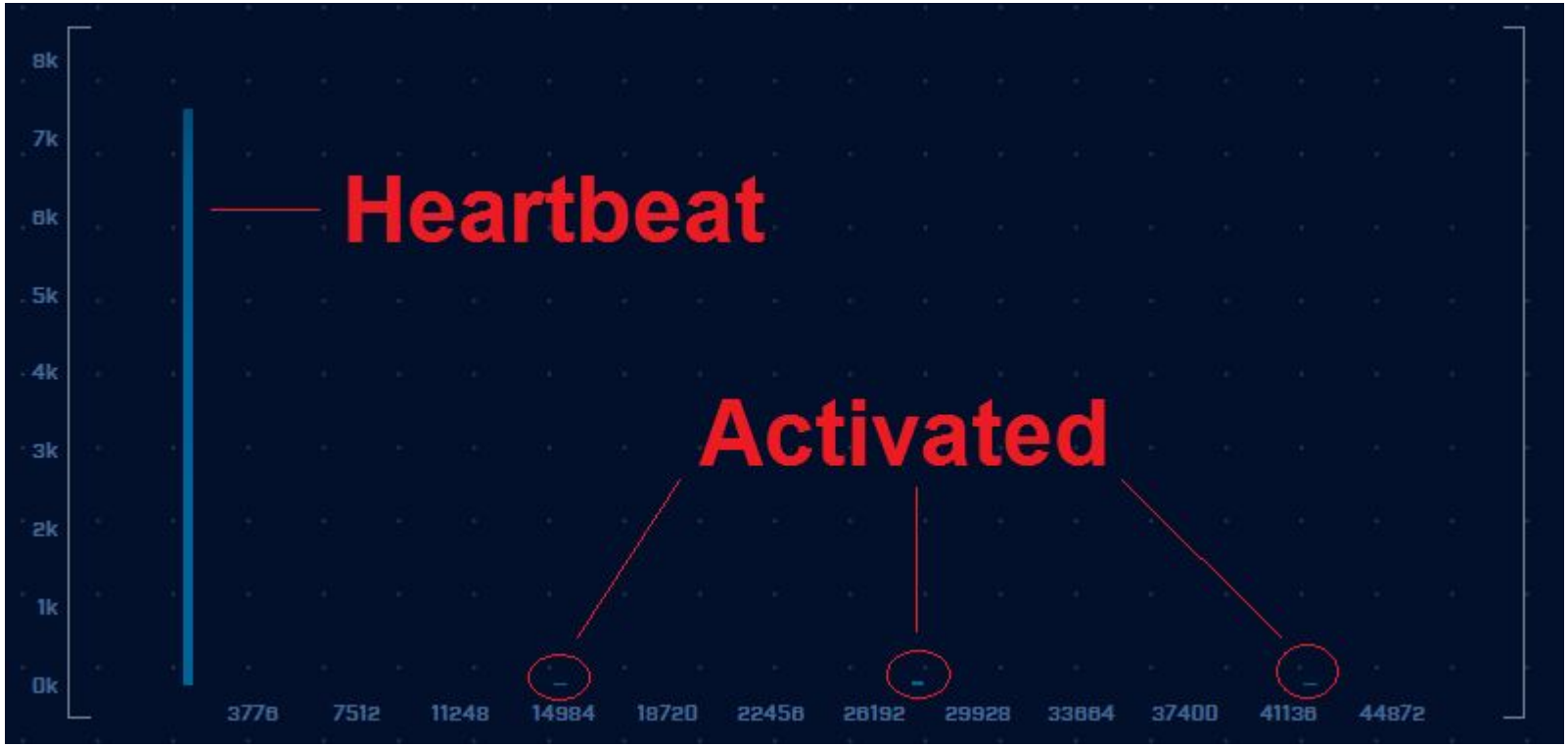
Could be an email client or gcat, both use the same timing.

Session size analysis of user email

Well this looks odd...



gcat once it's activated



User email versus gcat

- Similar session timing used for both
- User email
  - Expect to see lots of unique session sizes
  - 130 emails per day is the industry average
- gcat
  - One very strong signal for heartbeat
  - Some small number of other sizes
  - Once each time gcat is activated

What have we learned?

- gcat cannot be detected based on timing
  - Mimics normal email clients too closely
  - This is why many tools ignore this channel
- gcat can be detected through other means
  - Packet quantity
  - Session size comparison
- Tag by understanding "normal" and identifying deviations

Wrap up / Q&A

- Drop a tweet to @activecmeasures and tell us what C&C channel to cover next
  - https://twitter.com/ActiveCmeasures
- Type “demo” in the chat if you would like a demo of AI-Hunter
- To grab RITA:

  `http://acm.re/free-tools/rita/`

- To grab the pcaps from this webcast:

  `http://acm.re/webcast-file-downloads/`