Using Deep Packet Inspection for Monitoring and Security

Luca Deri <luca.deri@di.unipi.it>

Part I:
Past and Present

IDSs and ML [1/2]

- Traditional IDS (Intrusion Detection Systems), often based on signatures and rule-based approaches shown their limitations in detection capability, especially when attackers heavily rely on encryption to obfuscate communications.
- While we do believe that ML (machine learning) technologies are playing (and will play in the future) an important role in cybersecurity, we strongly believe that domain knowledge and feature engineering have tremendous value for any detection problem.

.

IDSs and ML [2/2]

- Increasing adoption of encryption technologies, DPI can be used to extract very strong signals from the raw traffic.
- While one could feed those signals to ML-based detectors, we highlight that when strong signals are available, one can greatly profit from them even with less sophisticated data processing technologies.
- This presentation shows how real-time, DPI-based cyber threat detection is feasible and effective using the concepts that will be explained later.

Signature-based IDSs (1998-Today)

alert tcp any any -> any [443,465] (msg:"Detected non-TLS on TLS port"; flow:to_server; app-layer-protocol:!tls; threshold: type limit, track by_src, seconds 90, count 1; sid:210003; rev:1;)

alert tcp any any <> any 443 (msg:"APT.Backdoor.MSIL.SUNBURST"; content:"|16 03|"; depth:2; content:"|55 04 03|"; distance:0; content:"digitalcollege.org"; within:50; sid:77600846; rev:1;)

- Techniques easy to circumvent.
- No application protocol visibility (packet header only, byte-based payload analysis).
- Outdated and error-prone format (“proto=TLS and SNI=digitalcollege.org”).

Cybersecurity and Network Edge [1/2]

- Today most traffic is encrypted (80%+) and many traditional clear-text protocols are moving to encryption (e.g. DNS vs DNS-over-HTTPS).
- As edge network speed is increasing, security threats on customer networks can propagate the issue to the core.
- Insecure devices (e.g. simple IoT devices) are placed in privileged network segments, thus requiring accurate supervision as they can cause severe troubles in case of breach.

Cybersecurity and Network Edge [2/2]

- Data centers with unhealthy customer traffic can affect neighbours and decrease the whole network reputation score.
- Limiting traffic observability to bandwidth usage is no longer wise: it is time to monitor customer traffic in an unobtrusive way in order to report users all threats they have not detected, mitigate issues and thus implement a healthier Internet.
- In essence we need to implement a lightweight (Raspberry an up, no GPU or GB of RAM) and scalable system able to model and analyse network traffic on a per-device basis, and being able to track device changes in behaviour.

Welcome to nDPI

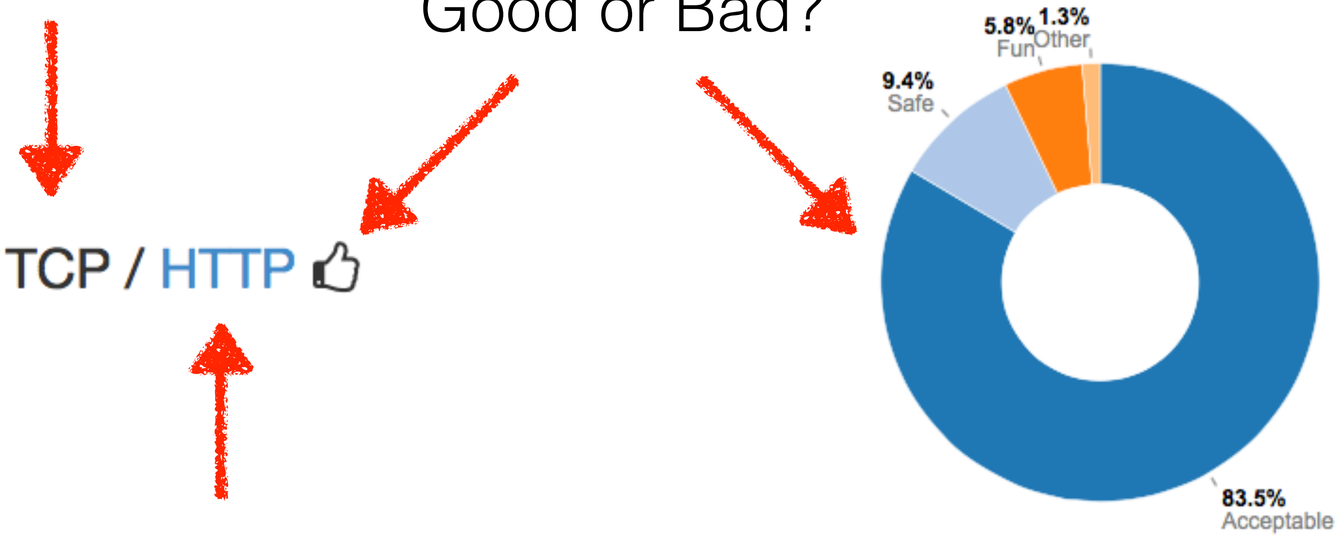
- In 2012 I decided to develop a new GNU LGPL DPI toolkit order to build an open source DPI layer.
- Protocols supported exceed 330 and include:
  - P2P (BitTorrent)
  - Messaging (Viber, Whatsapp, Telegram, Facebook)
  - Multimedia (YouTube, Last.gm, iTunes)
  - Conferencing (Skype, Webex, Teams, Meet, Zoom)
  - Streaming (Zattoo, Disney, Netflix)
  - Business (VNC, RDP, Citrix)
  - Gaming

GitHub https://github.com/ntop/nDPI

nDPI Traffic Analysis

Layer 4 Protocol

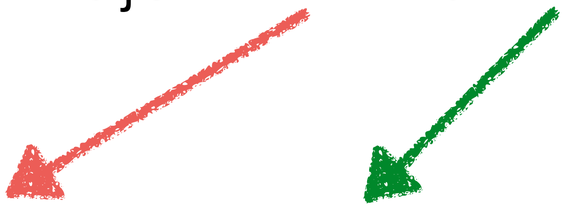
Good or Bad?



Layer 7 Protocol

What is a Protocol in nDPI? [1/2]

- Each protocol is identified as <major>.<minor> protocol. Example:
  - DNS.Facebook
  - QUIC.YouTube and QUIC.YouTubeUpload



- Caveat: Skype or Facebook are application protocols in the nDPI world but not for IETF.
- The first question people ask when they have to evaluate a DPI toolkit is: how many protocol do you support? This is not the right question.

What is a Protocol in nDPI? [2/2]

- Today most protocols are HTTP/TLS-based.
- nDPI includes support for string-based protocols detection:
  - DNS query name
  - HTTP Host/Server header fields
  - TLS/QUIC SNI (Server Name Indication)
- Example: NetFlix detection

```
{ "netflix.com", NULL,   "netflix" TLD,   “NetFlix”,  NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxext.com", NULL,   “nflxext" TLD,   "NetFlix",  NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflximg.com", NULL,   "nflximg" TLD,   "NetFlix",  NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflximg.net", NULL,   "nflximg" TLD,   "NetFlix",  NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxvideo.net", NULL, "nflxvideo" TLD, "NetFlix",  NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
{ "nflxso.net", NULL,    "nflxso" TLD,    "NetFlix",  NDPI_PROTOCOL_NETFLIX, NDPI_PROTOCOL_CATEGORY_STREAMING, NDPI_PROTOCOL_FUN },
```

Traffic Classification Lifecycle

- Based on traffic type (e.g. UDP traffic) dissectors are applied sequentially starting with the one that will most likely match the flow (e.g. for TCP/80 the HTTP dissector is tried first).
- Each flow maintains the state for non-matching dissectors in order to skip them in future iterations.
- Analysis lasts until a match is found or after too many attempts (8 packets is the upper-bound in our experience).

nDPI: Packet Processing Performance

```
nDPI Memory statistics:
      nDPI Memory (once):      203.62 KB
      Flow Memory (per flow):  2.01 KB
      Actual Memory:           95.60 MB
      Peak Memory:             95.60 MB
      Setup Time:              1001 msec
      Packet Processing Time:  813 msec

Traffic statistics:
      Ethernet bytes:         1090890957    (includes ethernet CRC/IFC/trailer)
      Discarded bytes:        247801
      IP packets:             1482145       of 1483237 packets total
      IP bytes:               1055319477    (avg pkt size 711 bytes)
      Unique flows:           36703
      TCP Packets:            1338624
      UDP Packets:            143521
      VLAN Packets:           0
      MPLS Packets:           0
      PPPoE Packets:          0
      Fragmented Packets:     1092
      Max Packet size:        1480
      Packet Len < 64:        590730
      Packet Len 64-128:      67824
      Packet Len 128-256:     66380
      Packet Len 256-1024:    157623
      Packet Len 1024-1500:   599588
      Packet Len > 1500:      0
      nDPI throughput:        1.82 M pps / 9.99 Gb/sec
      Analysis begin:         04/Aug/2010 04:15:23
      Analysis end:           04/Aug/2010 18:31:30
      Traffic throughput:     28.85 pps / 165.91 Kb/sec
      Traffic duration:       51367.223 sec
      Guessed flow protos:    0
```

Single Core (E3 1241v3)

nDPI Algorithms

- nDPI natively implements algorithms that power all this:
  - Substring Searching (Aho-Corasick).
  - IP Address Matching (Trie, Radix Tree).
  - Probabilistic Counting (HyperLogLog).
  - Anomaly Detection: Single/Double/Triple Exponential Smoothing.
  - Traffic Classification and Clustering: Data Binning.
  - Similarity Detection
  - Streaming Data Analysis: Variance, StdDev, Entropy, Jitter.
  - Data Serialisation.

nDPI in Cybersecurity

- Analyses encrypted traffic to detect issues un-inspectable due to encrypted payload content.
- Extracts metadata from selected protocols (e.g. DNS, HTTP, TLS..) and matches it against known algorithms for detecting selected threats (e.g. DGA hosts, Domain Generated Algorithm).
- Associates a “flow risk” with specific flows to identify communications that are affected by security issues.

nDPI: Flow Risks

- HTTP suspicious user-agent
- HTTP numeric IP host contacted
- HTTP suspicious URL
- HTTP suspicious protocol header
- TLS connections not carrying HTTPS (e.g. a VPN over TLS)
- Suspicious DGA domain contacted
- Malformed packet
- SSH/SMB obsolete protocol or application version
- TLS suspicious ESNI usage
- Unsafe Protocol used
- Suspicious DNS traffic
- TLS with no SNI
- XSS (Cross Site Scripting)
- SQL Injection
- Arbitrary Code Injection/Execution
- Binary/.exe application transfer (e.g. in HTTP)
- Known protocol on non standard port
- TLS self-signed certificate
- TLS obsolete version
- TLS weak cipher
- TLS certificate expired
- TLS certificate mismatch
- DNS suspicious traffic
- HTTP suspicious content
- Risky ASN
- Risky Domain Name
- Malicious JA3 Fingerprint
- Malicious SHA1 Certificate
- Desktop of File Sharing Session
- TLS Uncommon ALPN

Legenda: Clear Text Only, Encrypted/Plain Text, Encrypted Only

- TLS Certificate Validity Too Long
- Suspicious TLS Extension
- TLS Fatal Alert
- Suspicious Protocol traffic Entropy
- Clear-text Credentials Exchanged
- DNS Large Packet
- DNS Fragmented Traffic
- Invalid Characters Detected
- Possible Exploit Detected
- TLS Certificate Close to Expire
- Punycode/IDN Domain
- Error Code Detected
- Crawler/Bot Detected
- Anonymous Subscriber
- Unidirectional Traffic
- HTTP Obsolete Server

nDPI Encrypted Traffic Analysis

```
TCP 10.9.25.101:49184 <-> 187.58.56.26:449 [byte_dist_mean: 124.148883][byte_dist_std:
58.169660][entropy: 5.892724][total_entropy: 7124.302784][score: 0.9973][proto: 91/TLS]
[cat: Web/5][97 pkts/36053 bytes <-> 159 pkts/149429 bytes][Goodput ratio: 85/94][111.31
sec][bytes ratio: -0.611 (Download)][IAT c2s/s2c min/avg/max/stddev: 0/0 1129/662
19127/19233 2990/2294][Pkt Len c2s/s2c min/avg/max/stddev: 54/54 372/940 1514/1514
530/631][Risk: ** Self-signed Certificate **** Obsolete TLS version (< 1.1) **][TLSv1]
[JA3S: 623de93db17d313345d7ea481e7443cf][Issuer: C=AU, ST=Some-State, O=Internet Widgits
Pty Ltd][Subject: C=AU, ST=Some-State, O=Internet Widgits Pty Ltd][Certificate SHA-1:
DD:EB:4A:36:6A:2B:50:DA:5F:B5:DB:07:55:9A:92:B0:A3:52:5C:AD][Validity: 2019-07-23 10:32:39
- 2020-07-22 10:32:39][Cipher: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA]

TCP 10.9.25.101:49165 <-> 144.91.69.195:80 [byte_dist_mean: 95.694525][byte_dist_std:
25.418150][entropy: 0.000000][total_entropy: 0.000000][score: 0.9943][proto: 7/HTTP][cat:
Web/5][203 pkts/11127 bytes <-> 500 pkts/706336 bytes][Goodput ratio: 1/96][5.18 sec]
[Host: 144.91.69.195][bytes ratio: -0.969 (Download)][IAT c2s/s2c min/avg/max/stddev: 0/0
23/9 319/365 49/37][Pkt Len c2s/s2c min/avg/max/stddev: 54/54 55/1413 207/1514 11/134]
[URL: 144.91.69.195/solar.php][StatusCode: 200][ContentType: application/octet-stream]
[UserAgent: pwtyyEKzNtGatwnJjmCcBLbOveCVpc][Risk: ** Binary application transfer **][PLAIN
TEXT (GET /solar.php HTTP/1.1)]
```

Trickbot Traffic

Behaviour and Fingerprinting

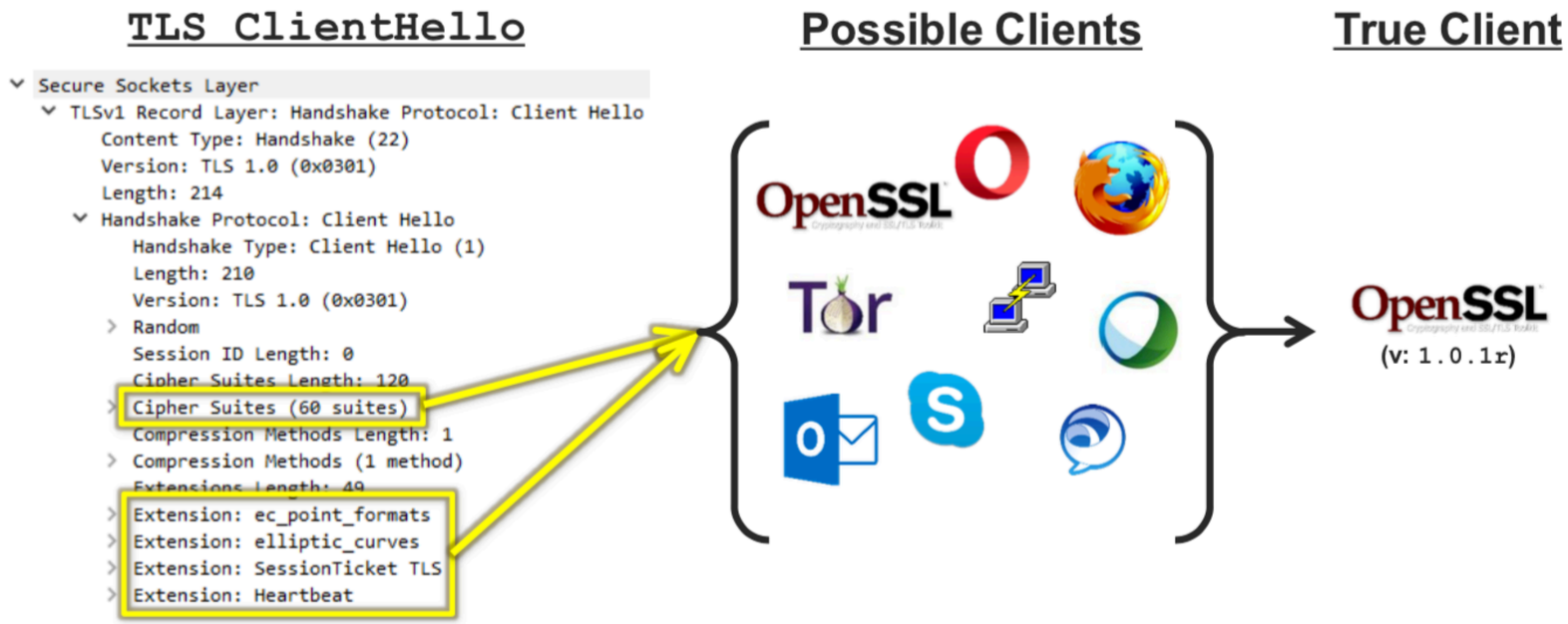
- nDPI is not only about application recognition but also:
  - Traffic classification: is this TLS connection a HTTPS connection, a VPN, or something else?
  - Malware recognition: traffic bins (time and packet size)
  - Content enforcement: bytes entropy (measure how bytes are distributed)

JA3: TLS Fingerprinting [1/2]

- Similar to HASSH (for SSH) but for TLS/SSL, it has been designed for malware detection.
- JA3 fingerprints the way that a client application communicates over TLS.
- JA3S fingerprints the server response.
- They essentially create a fingerprint of the cryptographic negotiation between client and server.

https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967

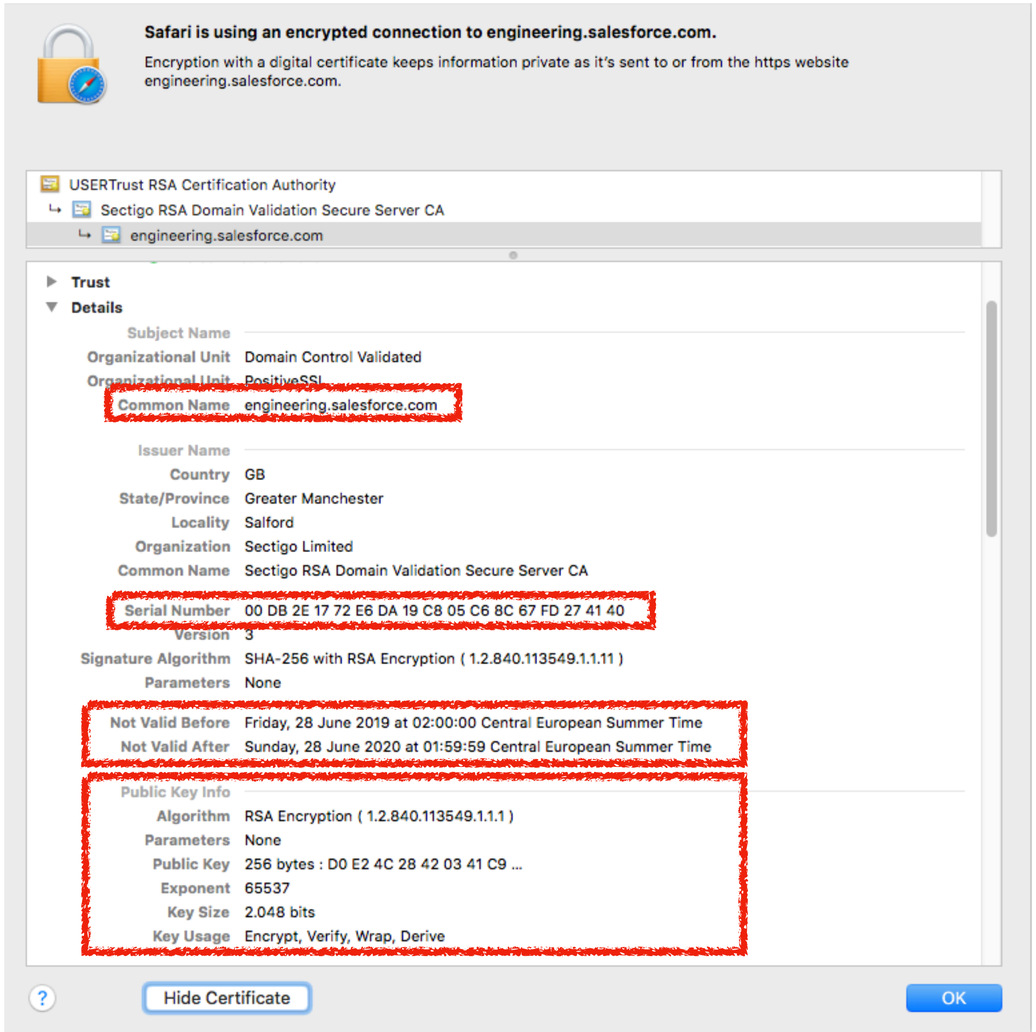
JA3: TLS Fingerprinting [2/2]



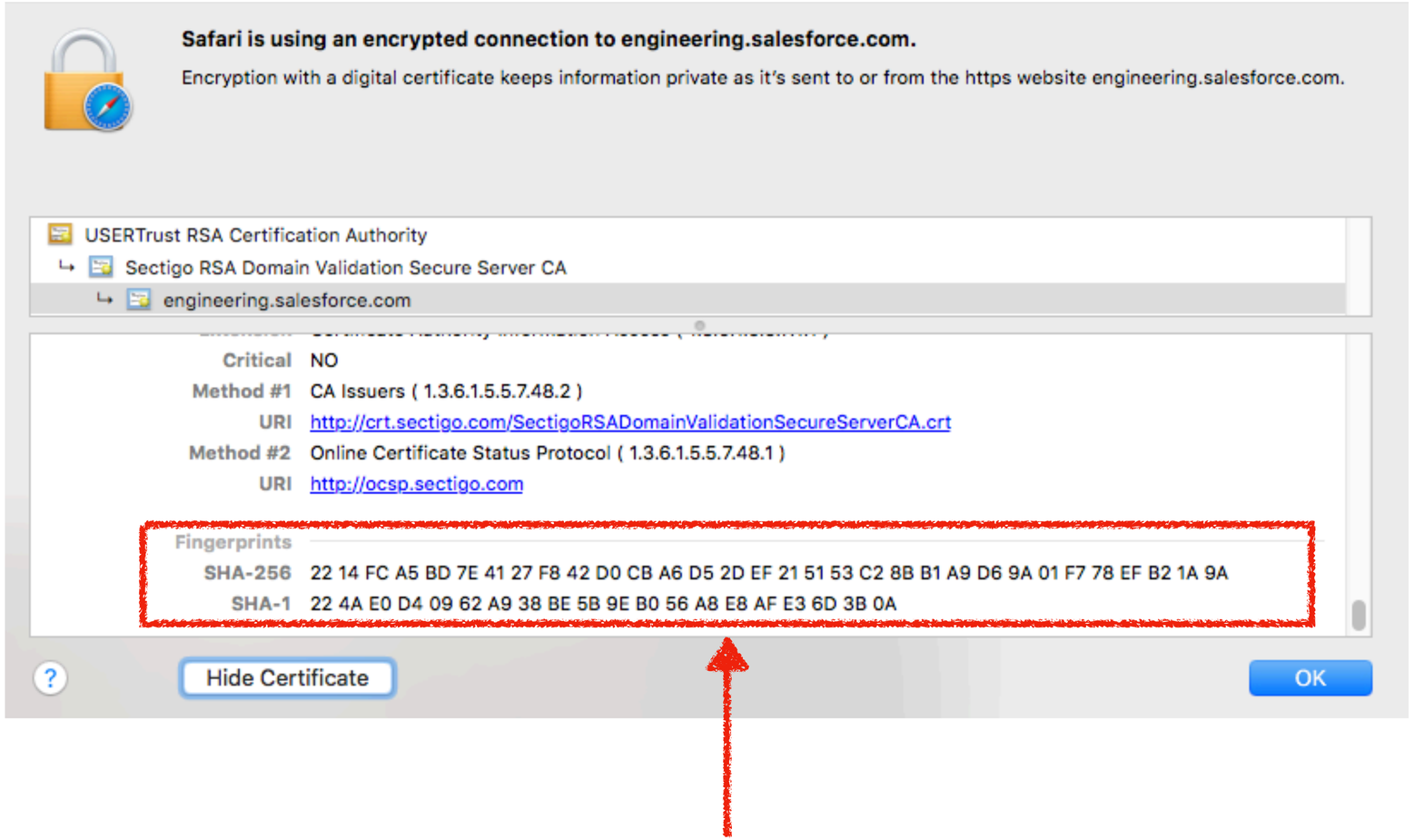
https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967

https://blogs.cisco.com/security/detecting-encrypted-malware-traffic-without-decryption

TLS Certificate Fingerprint [1/2]



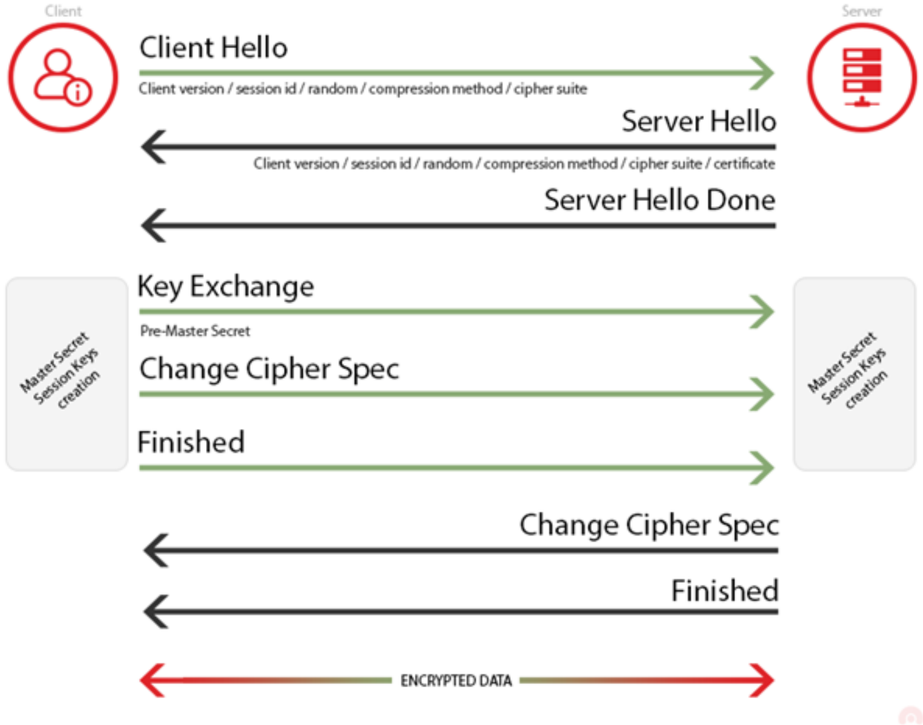
TLS Certificate Fingerprint [2/2]



When this changes, the HTTP server configuration has been modified

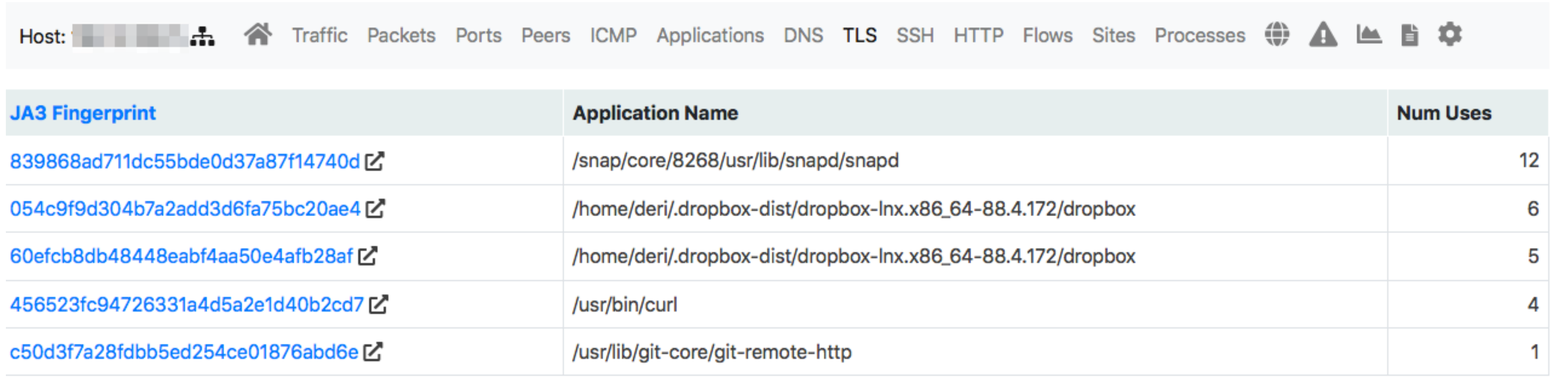
Catching Malware with Fingerprints [1/2]

- Some malware randomise the clientHello (and thus JA3C) trying to deceive security tools.
- Question: is this a good idea?



Catching Malware with Fingerprints [1/2]

- Answer: no it is not a good idea because a monitoring tool will easily detect cases where one IP address features many JA3C fingerprints.
- Question: how JA3C can be used to fingerprint application behaviour?

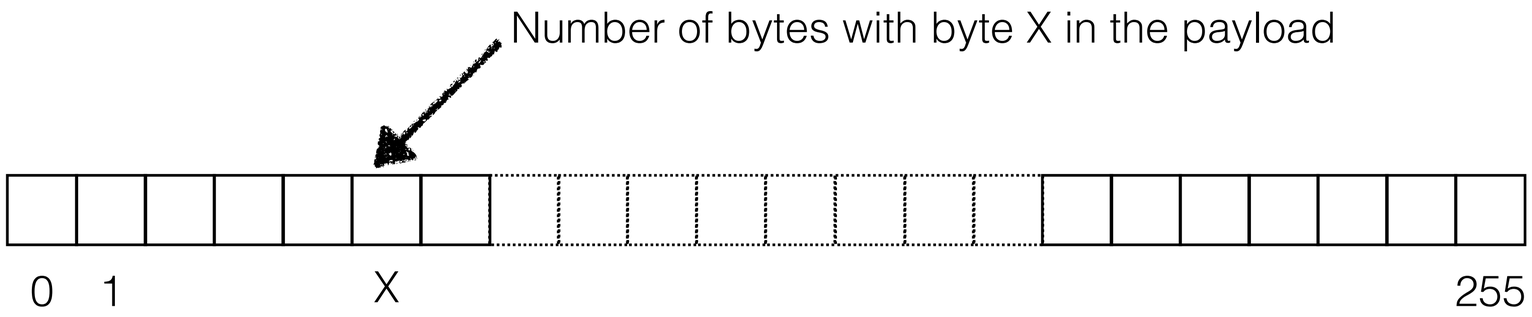


JA3 Fingerprint	Application Name	Num Uses
839868ad711dc55bde0d37a87f14740d	/snap/core/8268/usr/lib/snapd/snapd	12
054c9f9d304b7a2add3d6fa75bc20ae4	/home/deri/.dropbox-dist/dropbox-lnx.x86_64-88.4.172/dropbox	6
60efcb8db48448eabf4aa50e4afb28af	/home/deri/.dropbox-dist/dropbox-lnx.x86_64-88.4.172/dropbox	5
456523fc94726331a4d5a2e1d40b2cd7	/usr/bin/curl	4
c50d3f7a28fdbb5ed254ce01876abd6e	/usr/lib/git-core/git-remote-http	1

https://www.ntop.org/ntop/introducing-nprobe-agent-packetless-system-introspected-network-visibility/

Bytes Entropy [1/2]

- Metric used to measure how bytes are distributed: the larger the entropy, the greater the uncertainty in predicting the value of an observation.



https://csrc.nist.gov/csrc/media/publications/sp/800-90b/draft/documents/draft-sp800-90b.pdf

Bytes Entropy [2/2]

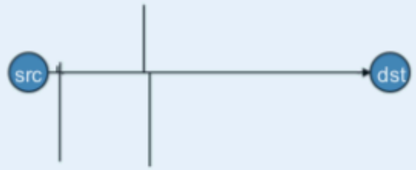
- Entropy of raw data before and after encryption (TLS) changes but is it within limited boundaries for homogeneous data.
- Useful to set boundaries on typical protocol entropy and “guess” (up to some extent) the nature of information being exchanged.

Payload Entropy Distribution

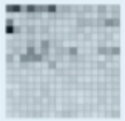
Byte Entropy	DNS	TLS	NetFlow
Average	4.285	7.789	4.079
Std Dev	0.272	0.231	0.533

Malware Traffic Analysis

- SPLT – Sequence of Packet Lengths and Arrival Times



- Byte Distribution
- Byte Entropy

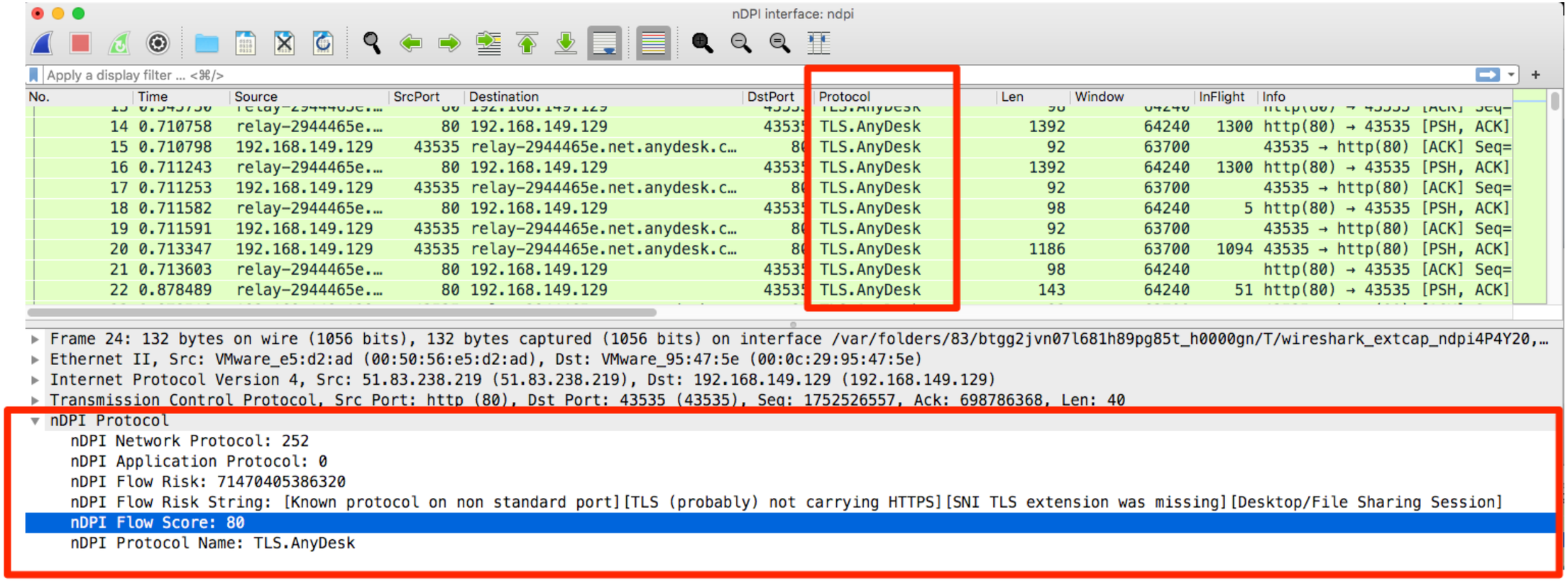


- TLS unencrypted header data
  - Certificates, SNI, Ciphersuites, Extensions

- DNS linked flows

- HTTP linked flows

nDPI in Wireshark



From Flow Risk To Score [1/2]

nDPI supported risks:

Id	Risk	Severity	Score	CliScore	SrvScore
1	XSS attack	Severe	250	225	25
2	SQL injection	Severe	250	225	25
3	RCE injection	Severe	250	225	25
4	Binary application transfer	Severe	250	125	125
5	Known protocol on non standard port	Medium	50	25	25
6	Self-signed Certificate	High	100	90	10
7	Obsolete TLS version (older than 1.2)	High	100	90	10
8	Weak TLS cipher	High	100	90	10
9	TLS Expired Certificate	High	100	50	50
10	TLS Certificate Mismatch	High	100	50	50
11	HTTP Suspicious User-Agent	High	100	90	10
12	HTTP Numeric IP Address	Low	10	5	5
13	HTTP Suspicious URL	High	100	90	10
14	HTTP Suspicious Header	High	100	90	10
15	TLS (probably) not carrying HTTPS	Low	10	5	5
16	Suspicious DGA domain name	High	100	90	10
17	Malformed packet	Low	10	5	5
18	SSH Obsolete Client Version/Cipher	High	100	90	10
19	SSH Obsolete Server Version/Cipher	Medium	50	5	45
20	SMB Insecure Version	High	100	90	10
21	TLS Suspicious ESNI Usage	Medium	50	25	25
22	Unsafe Protocol	Low	10	5	5
23	Suspicious DNS traffic	High	100	90	10
24	SNI TLS extension was missing	Medium	50	25	25
25	HTTP suspicious content	High	100	90	10
26	Risky ASN	Medium	50	25	25
27	Risky domain name	Medium	50	25	25
28	Possibly Malicious JA3 Fingerprint	Medium	50	25	25
29	Possibly Malicious SSL Cert. SHA1 Fingerprint	Medium	50	25	25
30	Desktop/File Sharing Session	Low	10	5	5
31	Uncommon TLS ALPN	Medium	50	25	25
32	TLS certificate validity longer than 13 months	Medium	50	25	25
33	TLS suspicious extension	High	100	90	10
34	TLS fatal alert	Low	10	5	5
35	Suspicious entropy	Medium	50	25	25
36	Clear-text credentials	High	100	90	10
37	DNS packet larger than 512 bytes	Medium	50	25	25
38	Fragmented DNS message	Medium	50	25	25
39	Text contains non-printable characters	High	100	90	10

From Flow Risk To Score [2/2]

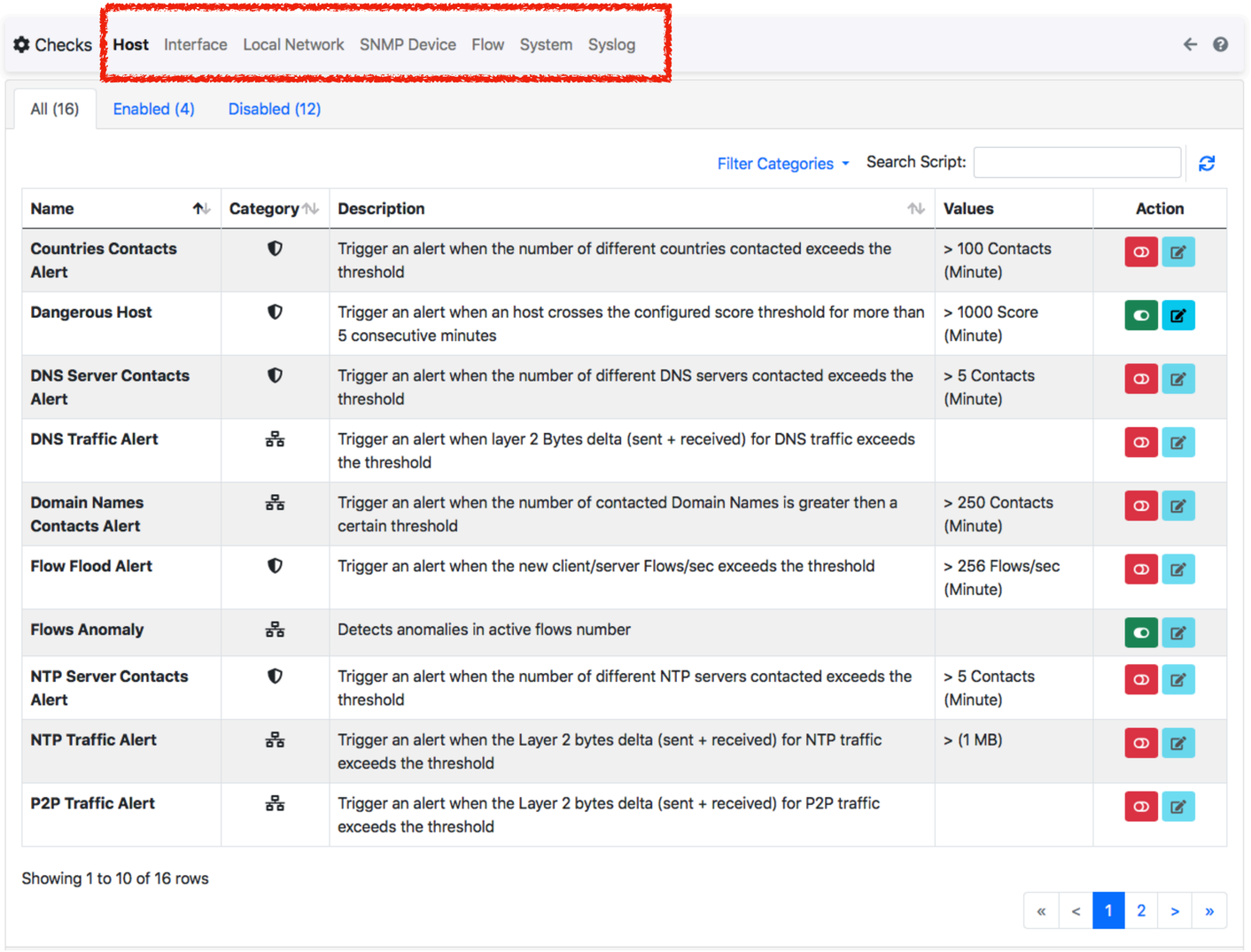
TCP 192.168.149.129:43535 <-> 51.83.238.219:80 [proto: 91.252/**TLS.AnyDesk**][cat: RemoteAccess/12][2942 pkts/175103 bytes <-> 4001 pkts/2618640 bytes][Goodput ratio: 9/92][55.97 sec][bytes ratio: -0.875 (Download)][IAT c2s/s2c min/avg/max/stddev: 0/0 19/14 7028/7028 153/126][Pkt Len c2s/s2c min/avg/max/stddev: 54/60 60/654 1514/1514 50/618][Risk: ** Known protocol on non standard port **** TLS (probably) not carrying HTTPS **** SNI TLS extension was missing **** Desktop/File Sharing Session **][**Risk Score: 80**][TLSv1.2][JA3C: 201999283915cc31cee6b15472ef3332][JA3S: 107030a763c7224285717ff1569a17f3][Issuer: CN=AnyNet Root CA, O=philandro Software GmbH, C=DE][Subject: C=DE, O=philandro Software GmbH, CN=AnyNet Relay][Certificate SHA-1: 9E:08:D2:58:A9:02:CD:4F:E2:4A:26:B8:48:5C:43:0B:81:29:99:E3][Firefox][Validity: 2018-11-18 02:14:23 – 2028-11-15 02:14:23][Cipher: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384][Plen Bins: 0,7,17,2,1,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,3,0,0,33,0,0,0,0,29,0,0]

Detected Risk	Risk Score Value
Known protocol on non standard port	10
TLS (probably) not carrying HTTPS	10
SNI TLS extension was missing	50
Desktop/File Sharing Session	10
Flow Score Total	80

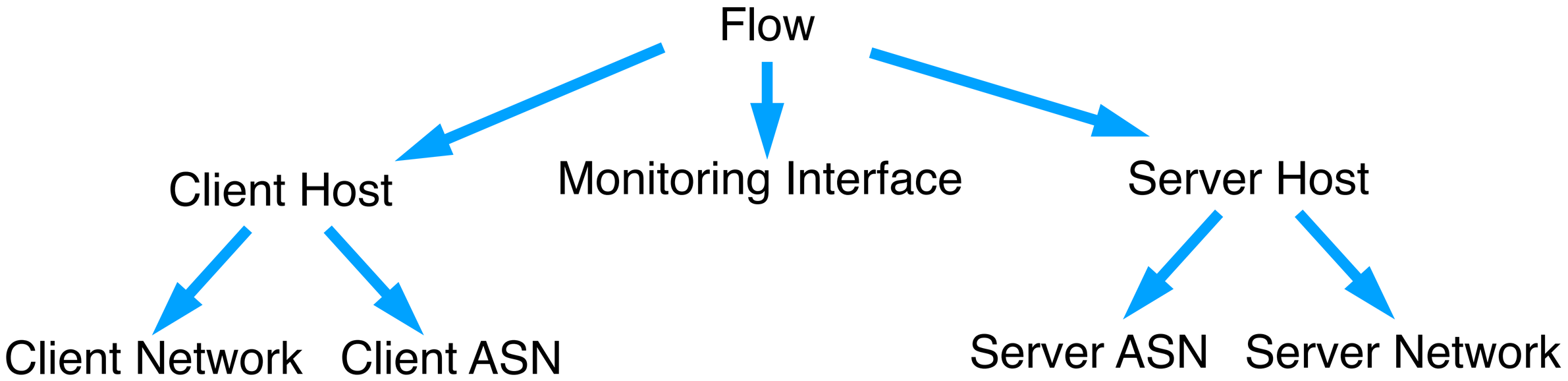
Consolidating Score [1/3]

- Flow traffic analysis is too granular and it needs to be consolidated into:
  - Network Interface
  - Host/Network/Customer.
  - ASN/Country
- In essence that is the pillar for creating a (client/server) numerical score that can be quickly used to spot issues (network, security…).

Consolidating Score [2/3]



Consolidating Score [3/3]



- Flow score is computed in realtime (flow lifetime)
- (Host/Interface/….) Checks are performed every minute

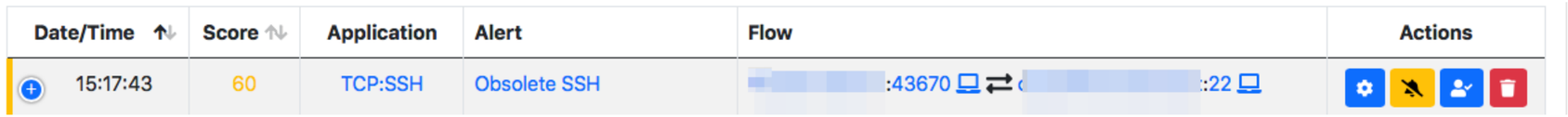
What about Risk Exceptions ? [1/3]

- Many cybersecurity products are very strict with policies and they divide the world in good and bad.
- Unfortunately reality is a bit more complicated (indeed grey exists), and “modern” needs to coexist with “ancient” that in computing terms can be just a few years old.
- The score principle is effective only if there are no false positives as otherwise they can deceive detection algorithms by generating false alerts.

What about Risk Exceptions ? [2/3]

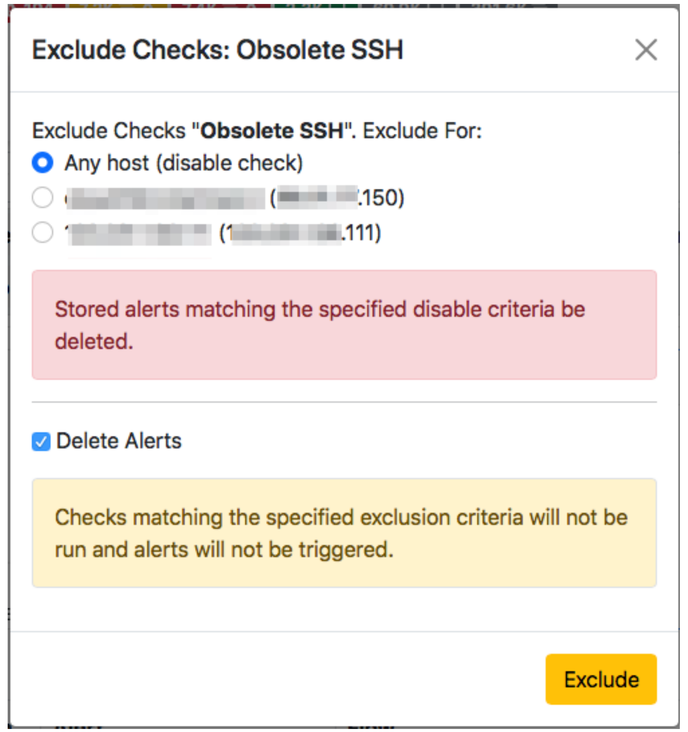
- A few typical exception examples:
  - Private IPs with self-signed TLS certificates.
  - Insecure protocols/hosts that cannot be upgraded but that provide a specific service to a few clients.
  - Applications running on non standard ports (e.g. SSH server on port 2222).
  - TLS towards numeric IP address (no symbolic hostname).

What about Risk Exceptions ? [3/3]

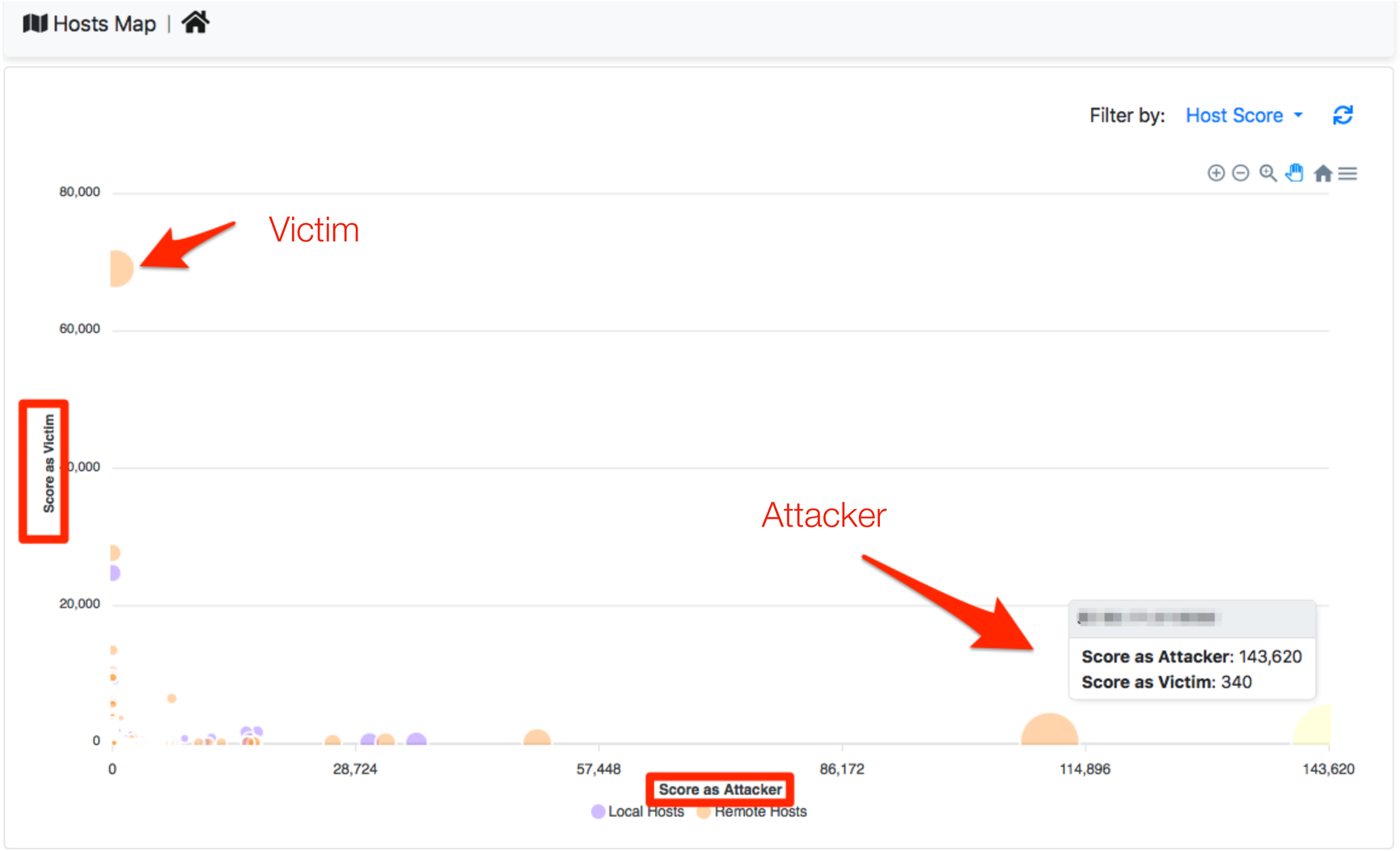


Available options:

- Disable Check (for everybody).
- Exclude the check for a specific host.
- Acknowledge the alert

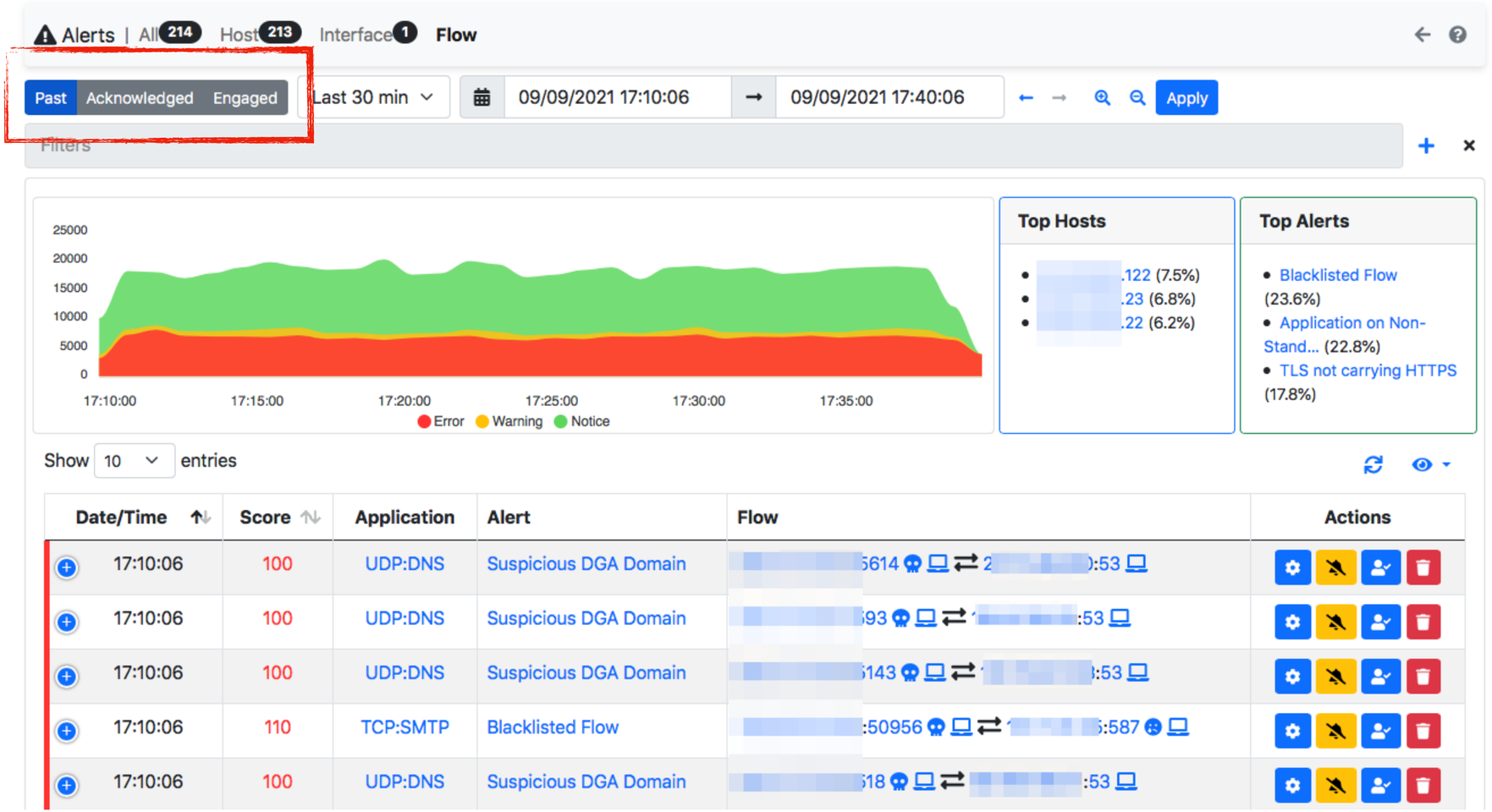


Visualising Cybersecurity: Bubbles



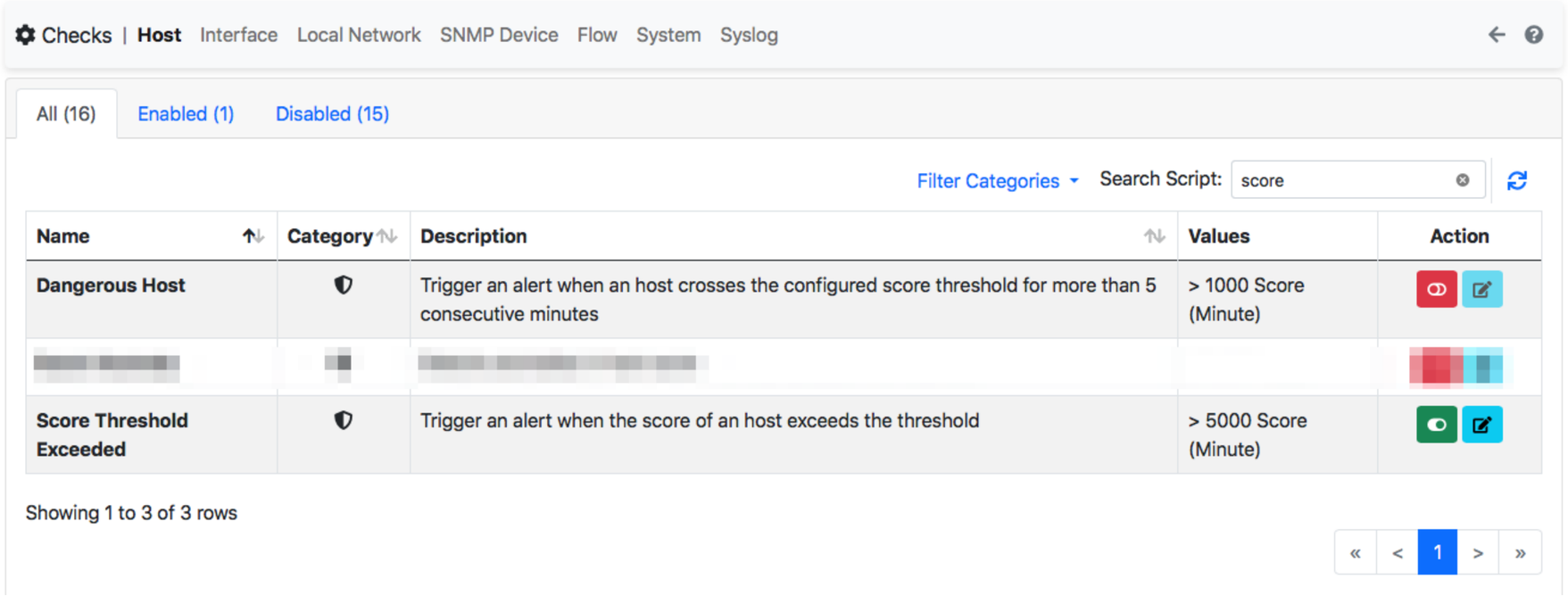
Score-based Alerts [1/2]

Alarm Lifecycle



Score-based Alerts [2/2]

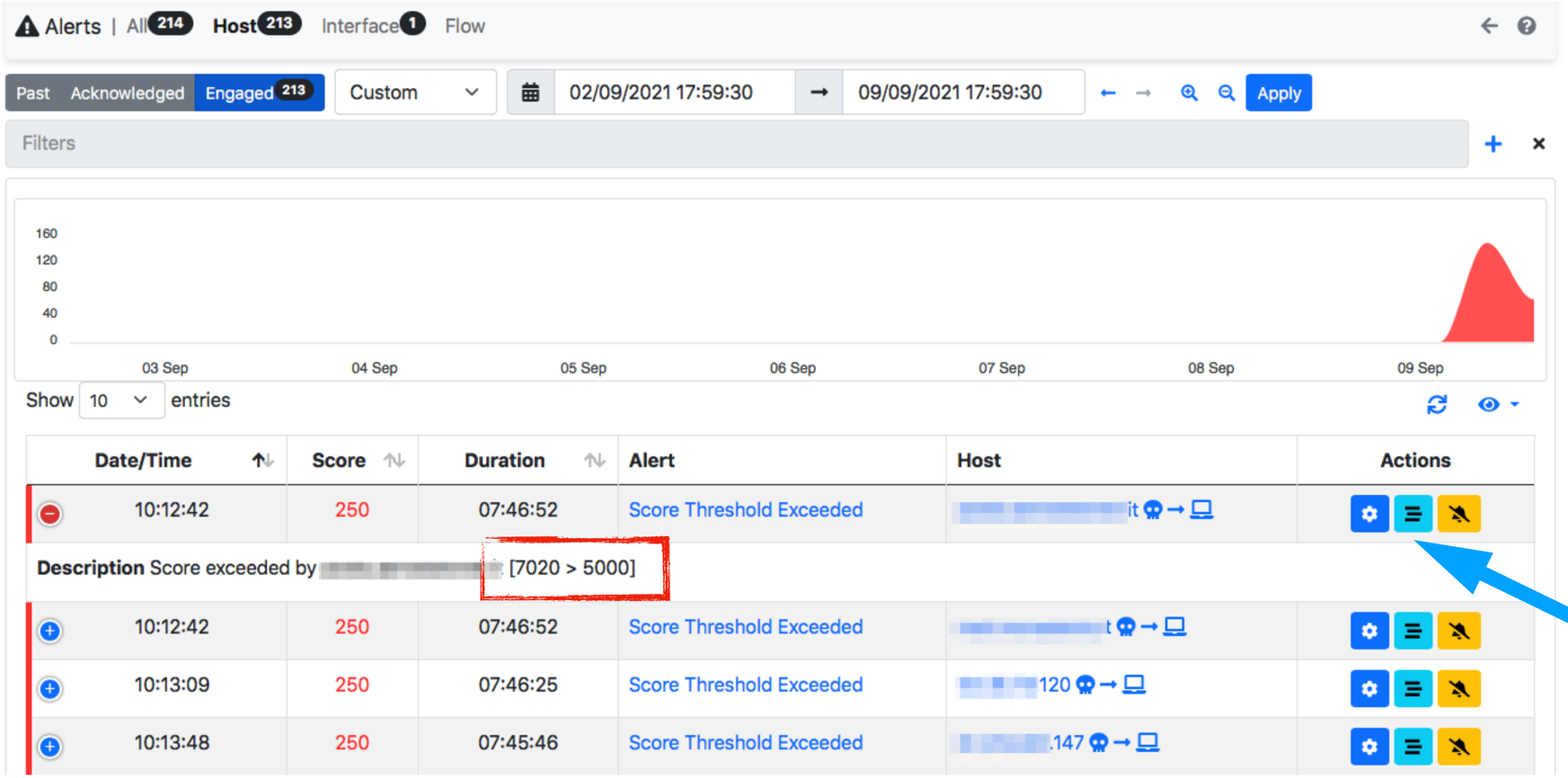
Threshold-based Score Alerts [1/2]



Simple to use for detecting hosts with high score:

- Continuously
- Score spikes

Threshold-based Score Alerts [2/2]



Flows
(Drill Down)

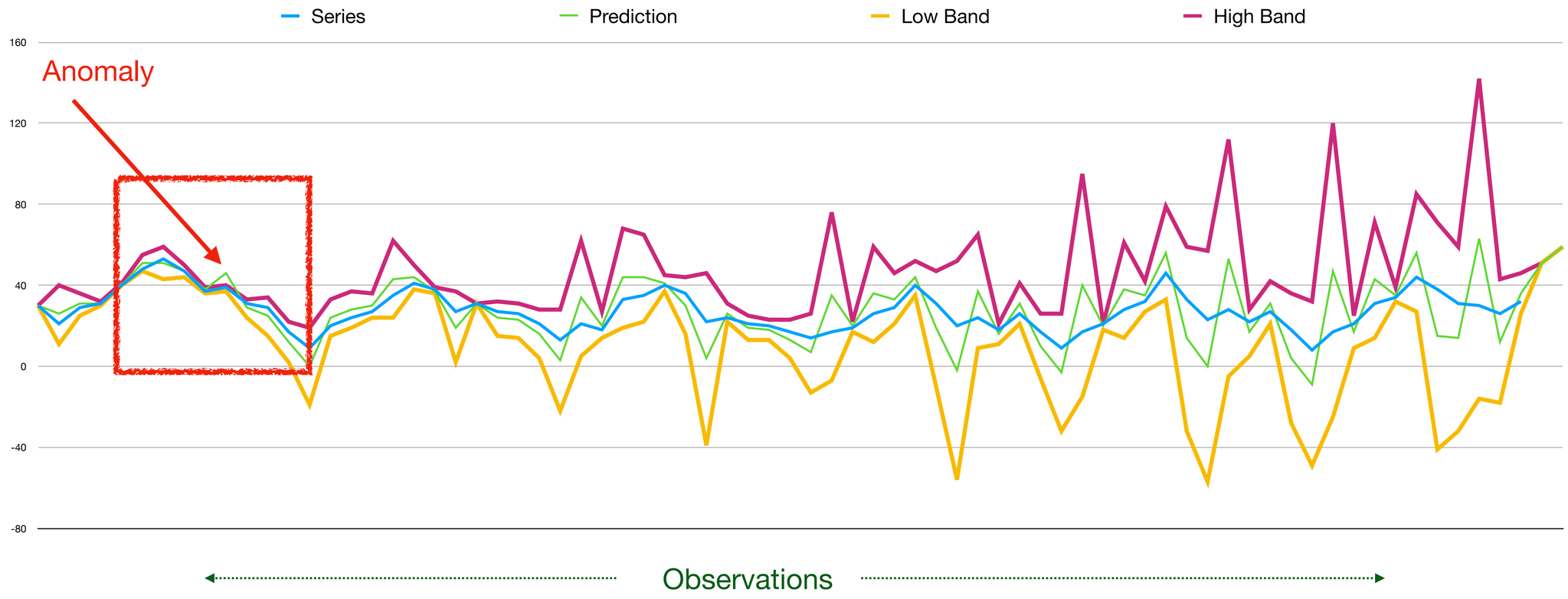
Score-based Behaviour Analysis [1/5]

- Thresholds are useful to spot issues that can be identified with boundaries.
- However
  - How do you define a typical host threshold? Not all hosts behave the same way.
  - How can I detect changes in behaviour? A host can double its score and still be unalarmed, but the network operator needs to be informed that something has changed.

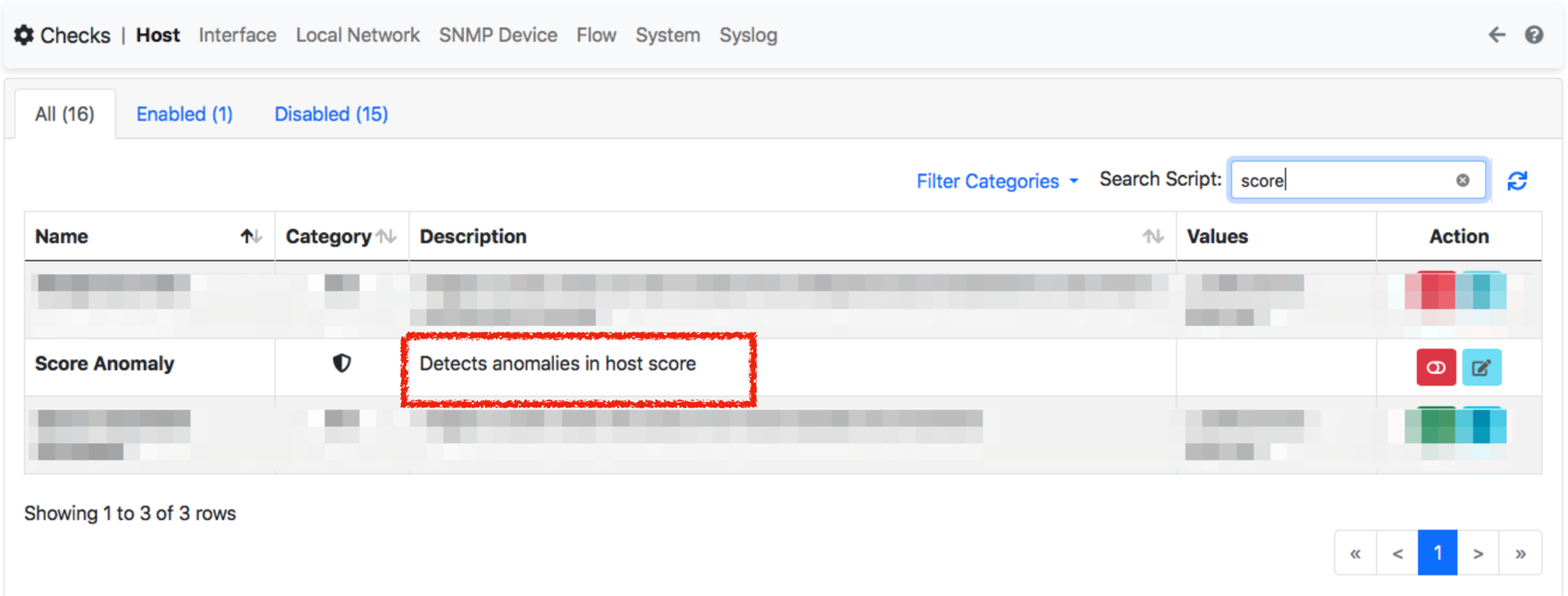
Score-based Behaviour Analysis [2/5]

- Without having to disturb ML that can be heavy for many users, we have decided to use (mature) statistical methods for spotting these changes.
- The advantage of statistical methods is that we can create a lightweight model per metric (hosts have tent of metrics) that uses little memory and CPU.
- For the record, we have used DES (Double Exponential Smoothing) that implements data forecasting and high/lower band for detecting changes in behaviour.

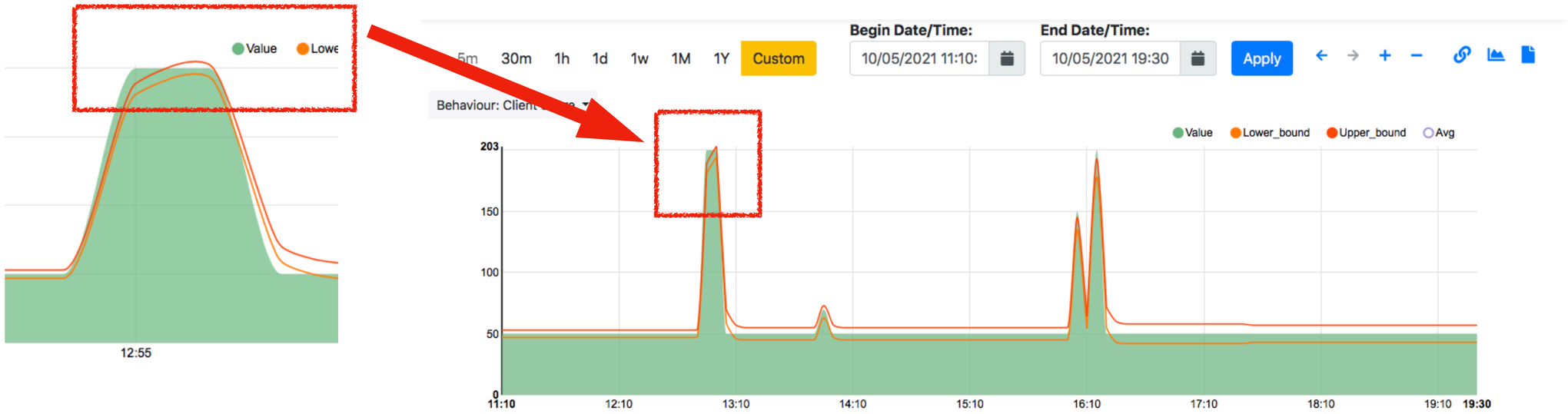
Score-based Behaviour Analysis [3/5]



Score-based Behaviour Analysis [4/5]



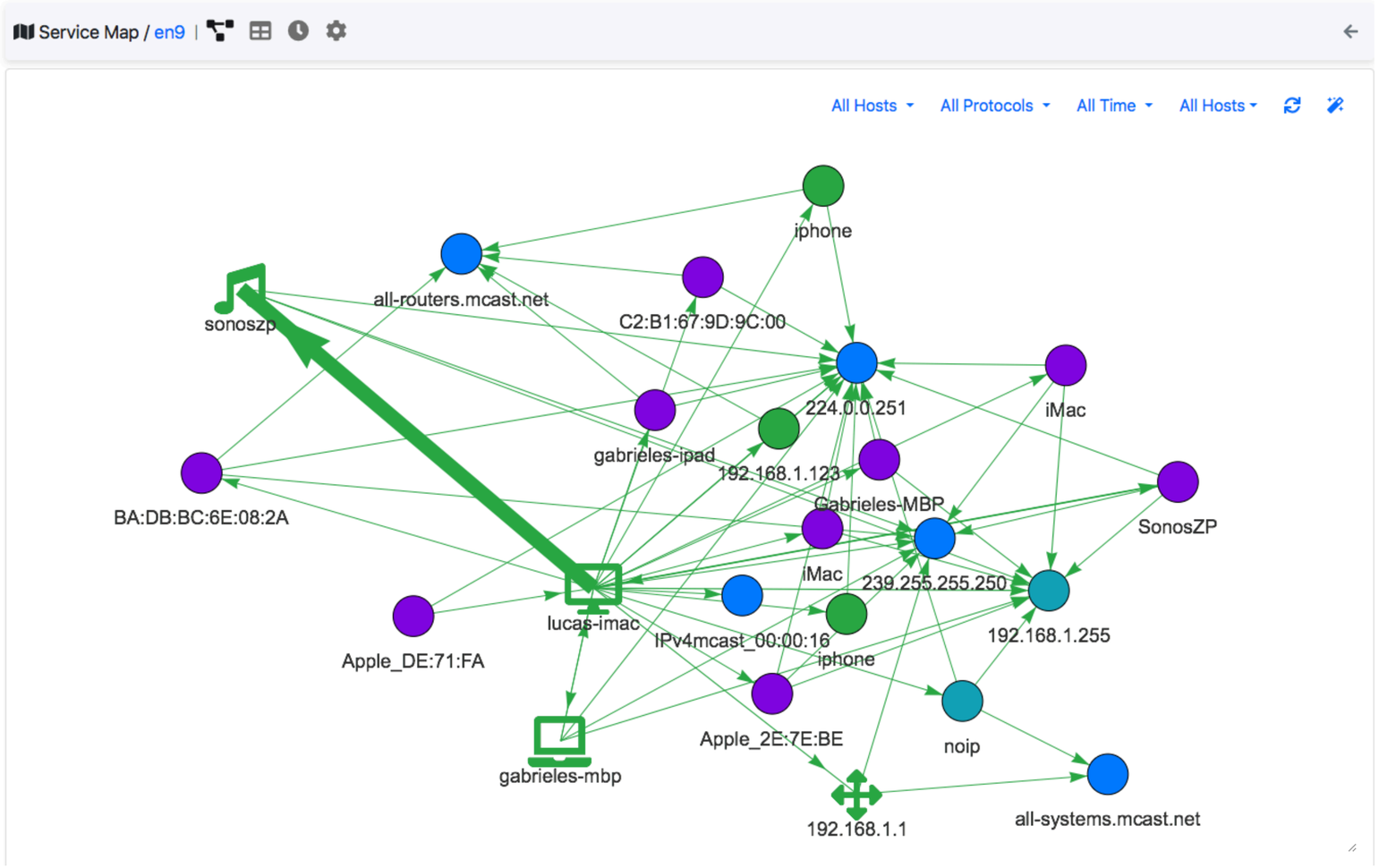
Score-based Behaviour Analysis [5/5]



Lateral Movement [1/4]

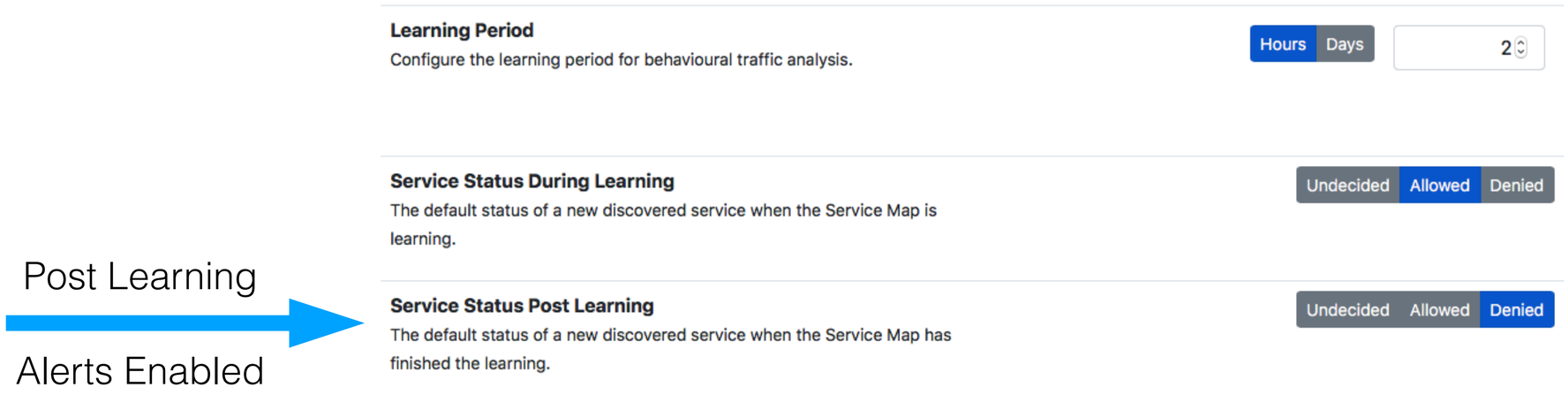
- What happens if a malware is roaming in our network? How can we spot it?
- In addition to the checks just presented, it can help to create a model of the network traffic and to continuously match it against live communications.
- Communications not matching the model are probably an indication of mistakes or new traffic patterns worth to be analysed.

Lateral Movement [2/4]

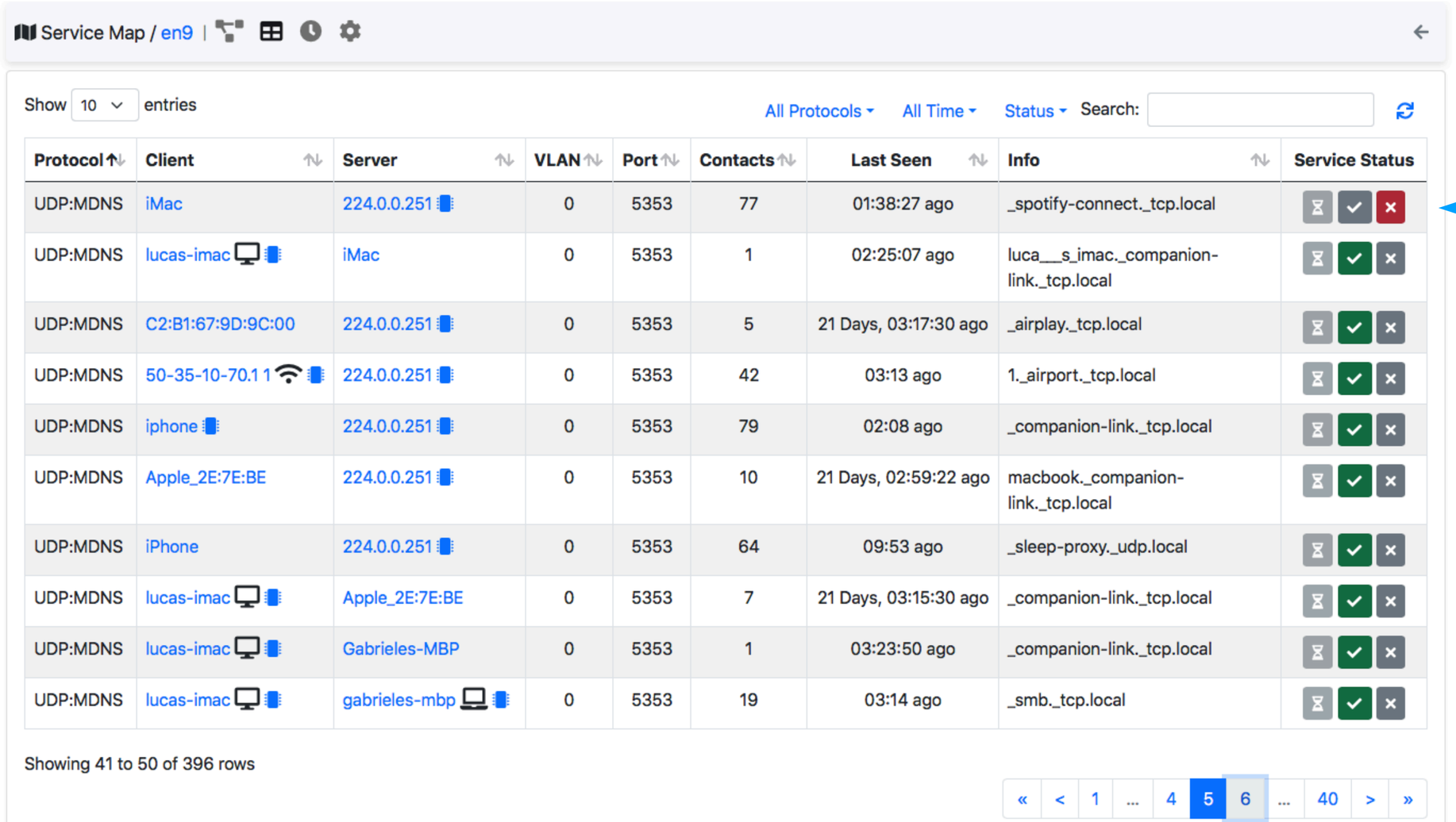


Lateral Movement [3/4]

- Learning Period
  - Discover new services and assign a default policy to them.
  - No alert is generated during learning.



Lateral Movement [4/4]



Forbidden

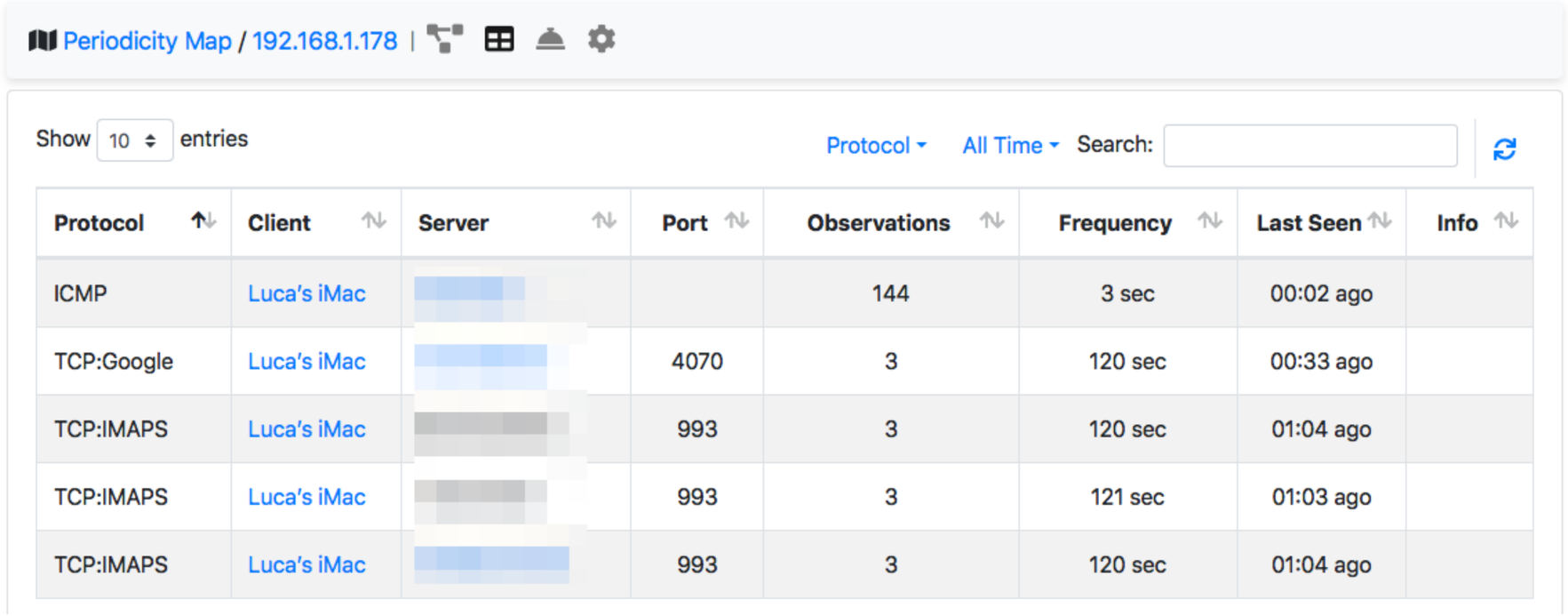
Beaconing Detection [1/3]

- Beacons are periodic low-volume communications that can be easily hidden inside the overall traffic.
- They are:
  - Often used by malware to talk back with the master.
  - An indication of failures (e.g. periodic connection to a service that is unavailable).
  - Used to identify monitoring activities (e.g. scans etc) or periodic checks (e.g. email download).
- In essence beaconing is not just for cybersecurity but also for spotting activities worth to be analysed.

Beaconing Detection [2/3]

- Instead of using AI or complex algorithms for beaconing detection we use a simple method:
  - Keep track of quadruplets <source/destination IP, destination port, layer 4 protocol>.
  - As soon as a new flow is detected a quadruplet is created (if not already present) or updated (if already created).
  - Idle quadruplets or quadruplets whose periodicity isn't too constant (of course we take into account time drifts) are discarded.

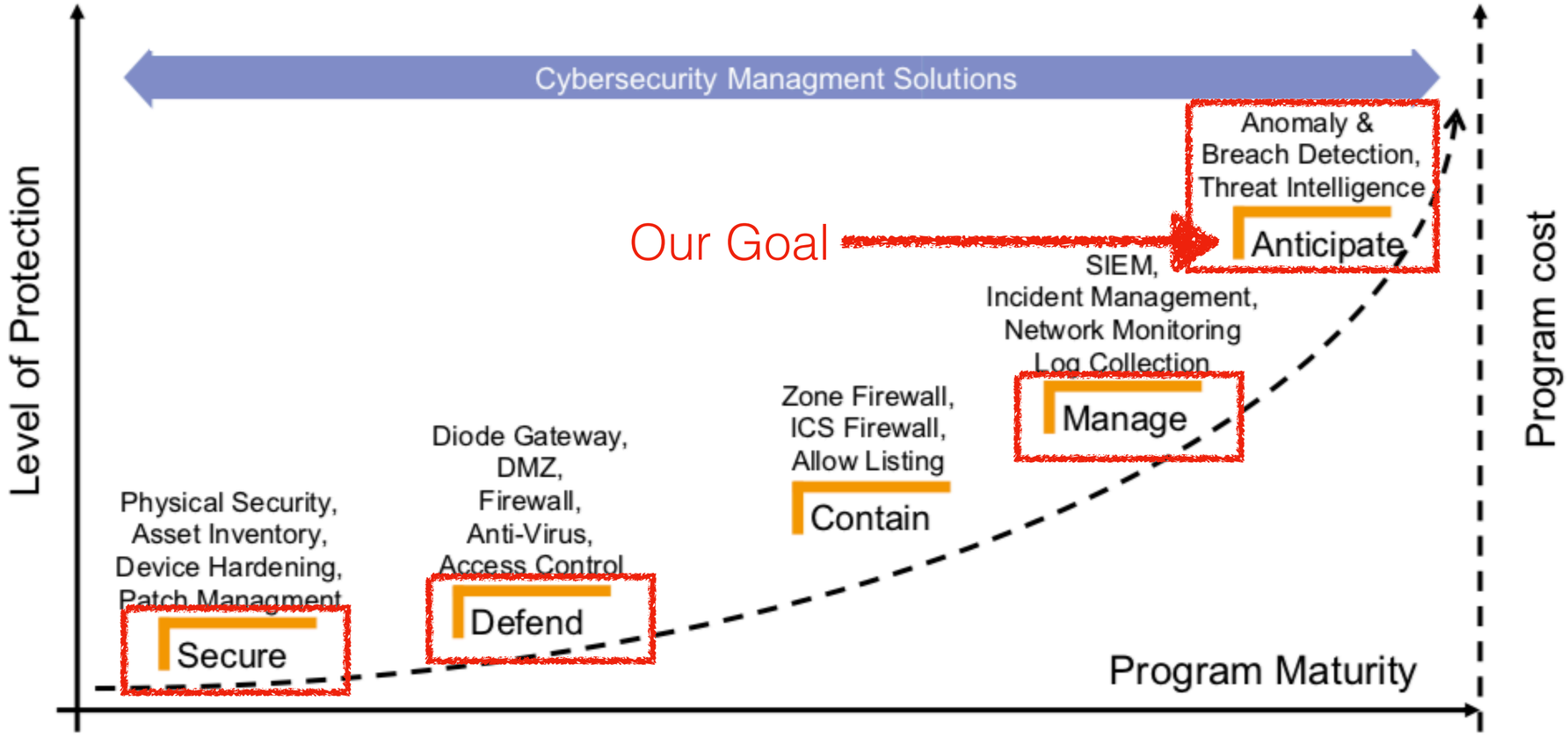
Beaconing Detection [3/3]



- Beaconing with Unknown or "unpleasant" (e.g. IRC) protocols are an indicator of suspicious communications.
- Beaconing begin/end is reported as informative alert.

Part II:
Ongoing Developments

2022 Monitoring Goals



Picture courtesy of switch.ch

How Can we Anticipate a Problem?

- Monitoring can show you when a problem is happening or (better) what are suspicious flows that can be an indication of a future problem.
- Can we do anything better than this? What if I could detect the user and application that generated a traffic flow?
- Goal: extend current monitoring capabilities with system analysis in order to report richer information and build new, more powerful checks.

Cybersecurity and Networking

- In a way, cybersecurity would not be that important without the Internet as networks propagate threats.
- Using DPI and traffic analysis techniques so far presented it is possible to have a great level of visibility and protection but…
- East-west traffic monitoring is not so simple and available techniques (e.g. sFlow) are sampled.
- Threats do their best to hide themselves: volumetric attacks are “nice” as they can be easily spotted.
- More packets, more ML and more checks are the only viable solution to this problem ?

Merging Network and System Visibility

- Advantages
  - Map traffic to processes/users: finally we know “who is doing what”.
  - Detect unexpected processes making traffic.
  - Simplified troubleshooting and incident analysis with contextual data.
- Limitations
  - Still a passive tool: the collector has the knowledge.
  - It is unable to detect “changes” but only “facts” (i.e. annotated flows with limited system metadata).

Towards a Host-based EDR

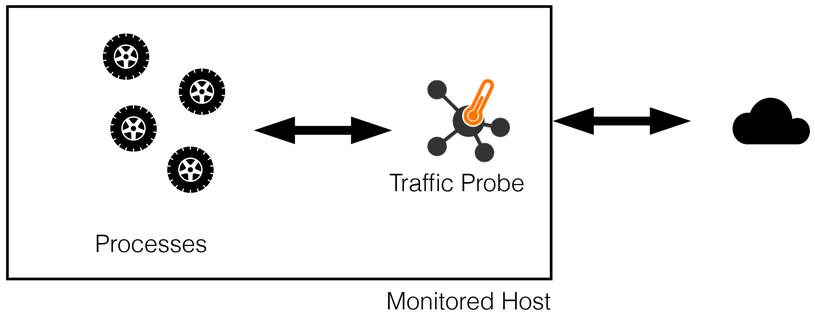
- What if we could:
  - Detect changes in configuration invisible to the network.
  - Use process and user information to properly evaluate risks in communications.
  - Use contextual information (e.g. process) not just for enriching flow data but also for preventing threats from spreading in the network?
- What about a host-based EDR (Endpoint Detection and Response) ?

Cybersecurity Simplified [1/2]

- Challenge: can we allow administrators to block threats before the problem shows up?
- Options: block traffic of applications that
  - Are not installed as package or that are started from non-standard locations (e.g. /tmp).
  - Have not been running previously.
  - Communicate with blacklisted IPs.
  - Have a periodicity and are not monitoring tools.
  - …(cont).

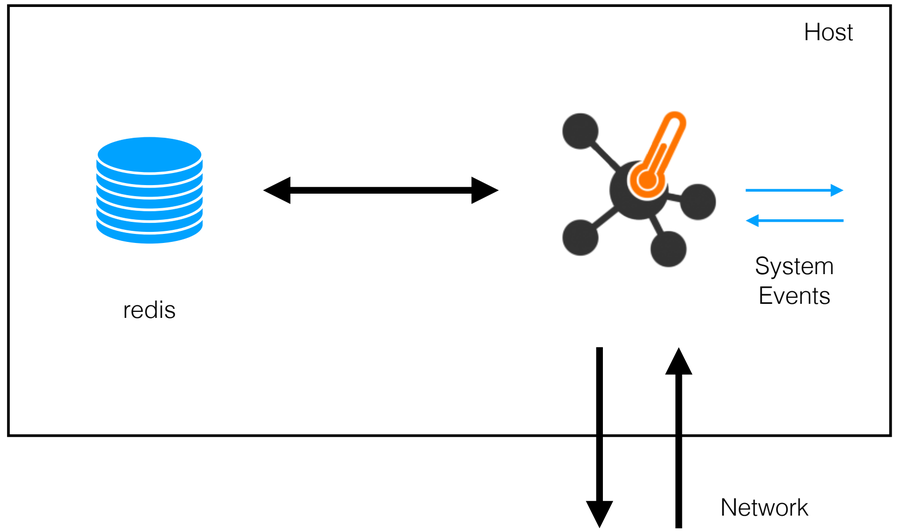
Cybersecurity Simplified [2/2]

- Combining system visibility with network monitoring, enabled us to create an active probe able to block specific application traffic and that can very well fit with the zero-trust principle that is becoming increasingly popular.



Introducing System Visibility [1/5]

- Sit on top of the network stack (including containers) in order to receive traffic and inspect/block it.
- Listen to system events in order to bind local traffic to processes and users.



Introducing System Visibility [2/5]

- We use redis as local policy cache for storing learnt information and as inter-process communication in case of high traffic rates that need to be handled by multiple processes.
- During the learning period, we store on redis observed <user>:<process> associations.
- Past learning, redis is used to retrieve known policies to be used for enforcement.

Introducing System Visibility [3/5]

- It is possible to query redis for users who sent data out, and for each process (that transmitted/received data) run by each user.

```
$ redis-cli keys "process.*"
 1) "process.root"
 2) "process.www-data"
 3) "process.influxdb"
 4) "process._apt"
 5) "process.postgres"
 6) "process.avahi"
 7) "process.clickhouse"
 8) "process.chronograf"
 9) "process.deri"
10) "process.grafana"
```

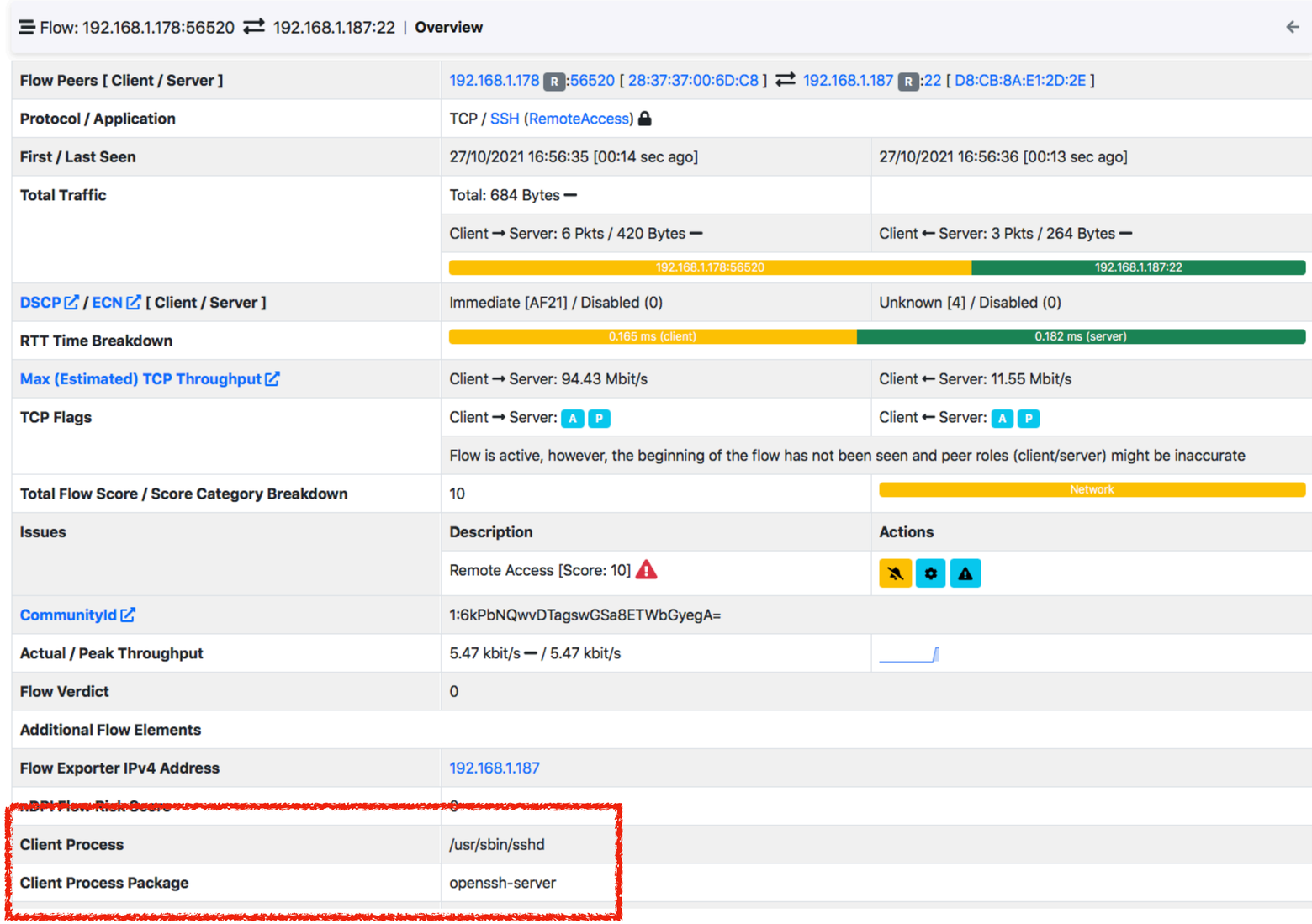
```
$ redis-cli hkeys "process.root"
1) "/usr/sbin/NetworkManager"
2) "/usr/lib/sm.bin/sendmail"
3) "/usr/sbin/ntpdate"
4) "/sbin/dhclient"
5) "/usr/sbin/cups-browsed"
6) "/snap/core/11606/usr/lib/snapd/snapd"
7) "/home/deri/nprobe"
8) "sendmail-mta"
```

- Is an unknown process allowed to do networking ? Probably not.

Introducing System Visibility [4/5]

- Unless you are developing software, applications need to be installed with packages.
- Malware applications are (usually) not packaged, so this can be a good indicator of compromise.
- Currently we support Linux packaging: both .deb and .rpm families are supported.
- Windows packaging is planned albeit not yet supported.

Introducing System Visibility [5/5]



Further Visibility: Server Side [1/3]

- As said before, a good strategy for detecting issues/ reconfigurations/malware is to track changes.
- When a malware speaks with remote peers, nProbe can detect the flow and report contextual information (process and package name).
- What if the malware isn't making any traffic (so it's in essence invisible to flows) but it's ready to accept connections from applications? Or if the traffic is so little that hides itself in background noise?

Further Visibility: Server Side [2/3]

- The probe has been enhanced with local host port monitoring for:
  - Binding a port with an application and a package.
  - Detecting changes in port allocation: a new port is open, an existing port is closed, or a different process is listening to an existing open port.
  - Reporting this information to flow collectors for increased visibility.

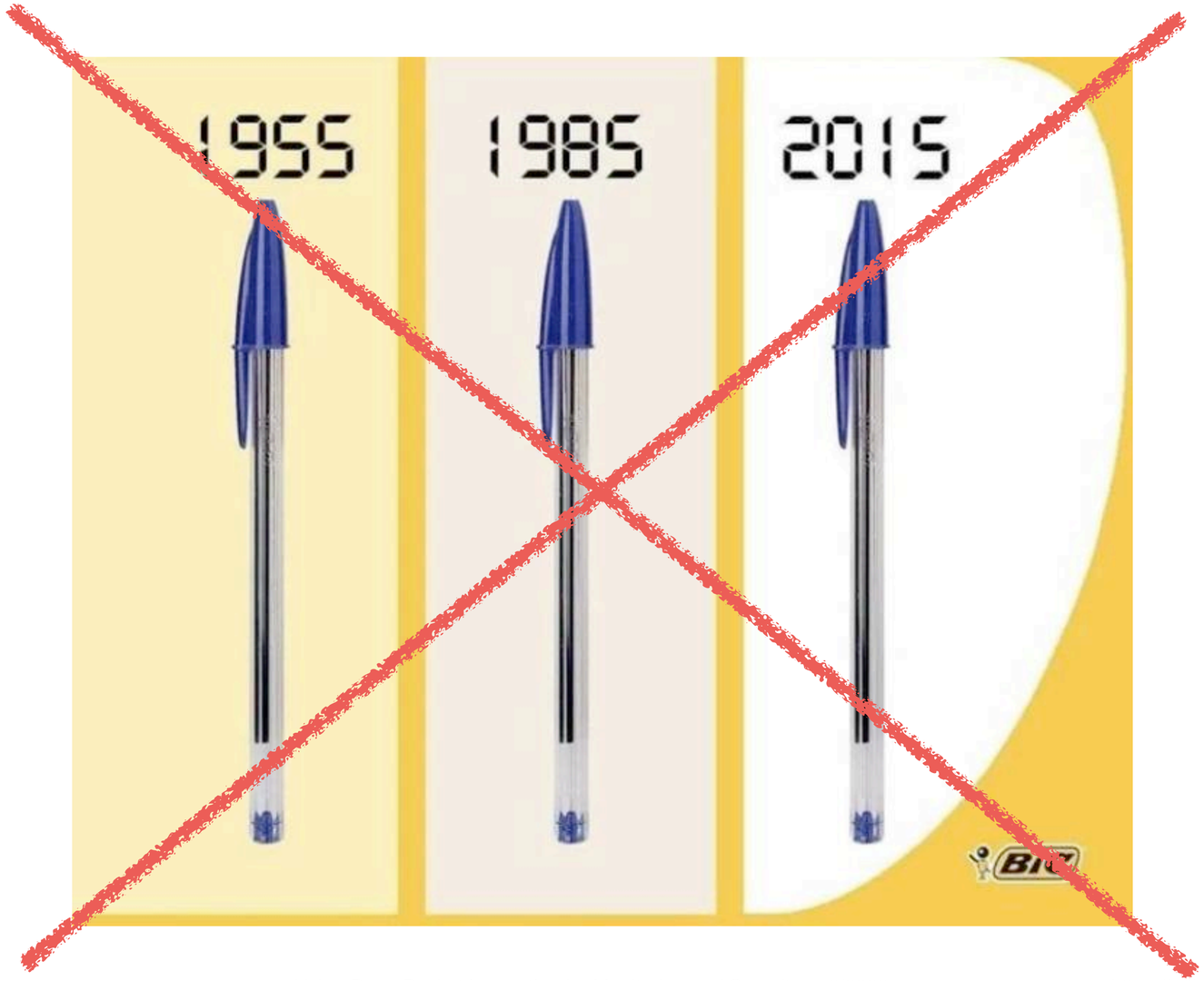
Further Visibility: Server Side [3/3]


```
{
    "ip-addresses": ["10.3.240.28", "192.168.1.187"],
    "listening-ports": {
        "tcp4": [{
            "port": 22,
            "proc": "/usr/sbin/sshd",
            "pkg": "openssh-server"
        }, {
            "port": 53,
            "proc": "/usr/sbin/dnsmasq",
            "pkg": "dnsmasq-base"
        }, {
            "port": 1234,
            "proc": "/home/deri/nProbe/nprobe",
            "pkg": ""
        }],
        "tcp6": [{
            "port": 9000,
            "proc": "/usr/bin/clickhouse",
            "pkg": “clickhouse-common-static"
...
```


No Package !

Part III:
Final Remarks

This Isn’t Cybersecurity



Cybersecurity in Pisa

HACK THE TOWER

https://hackthetower.it

- https://github.com/ntop/
- deri@ntop.org luca.deri@di.unipi.it