black hat USA 2021

August 4-5, 2021

BRIEFINGS

CnCHunter: An MITM-Approach to Identify Live CnC Servers

Ali Davanian, Ahmad Darki and Michalis Faloutsos

CnC·Hunter EXPLOITING MALWARE

IoT malware

IoT malware is on the rise!

57 | 2021 SonicWall Cyber Threat Report | *Cryptojacking Attempts by Industry*

IoT Malware Attacks Skyrocket

When the COVID-19 pandemic struck, work went home — and cybercriminals followed, propelling IoT malware attacks to new heights. While IoT malware attacks have been rising since SonicWall began tracking them in 2017, in 2020 they skyrocketed, based on a number of factors, including the use of compromised home IoT devices for personal gain.

In 2019, SonicWall Capture Labs threat researchers recorded 34.3 million IoT malware attacks. In 2020, that number rose to 56.9 million, a 66% increase.

The circumstances surrounding the pandemic did more

ET CIO.com
From The Economic Times

News | Industries | Tech-Talk | Data Center | CIO Life | Engage | ETCISO.in | Events | Brand Solutions

Security • Cloud Computing • Big Data • Internet of things • Business Analytics • Mobility • Research • Data Center • Next Gen Tech

IT News / Latest IT News / Security

New malware infects Android TVs, IoT devices in 84 nations

A new malware has infected roughly 13,500 Internet of Things (IoT) devices

ComputerWeekly.com

IT Management | Industry Sectors | Technology Topics | Search Computer Weekly

Orange Business Services taps Ericsson for enterprise IoT security

As billions of internet-of-things devices become connected and intelligent, security becomes more important, as comms tech provider delivers new security offering for CSPs

By Joe O'Halloran, Computer Weekly   Published: 04 Jun 2021 11:28

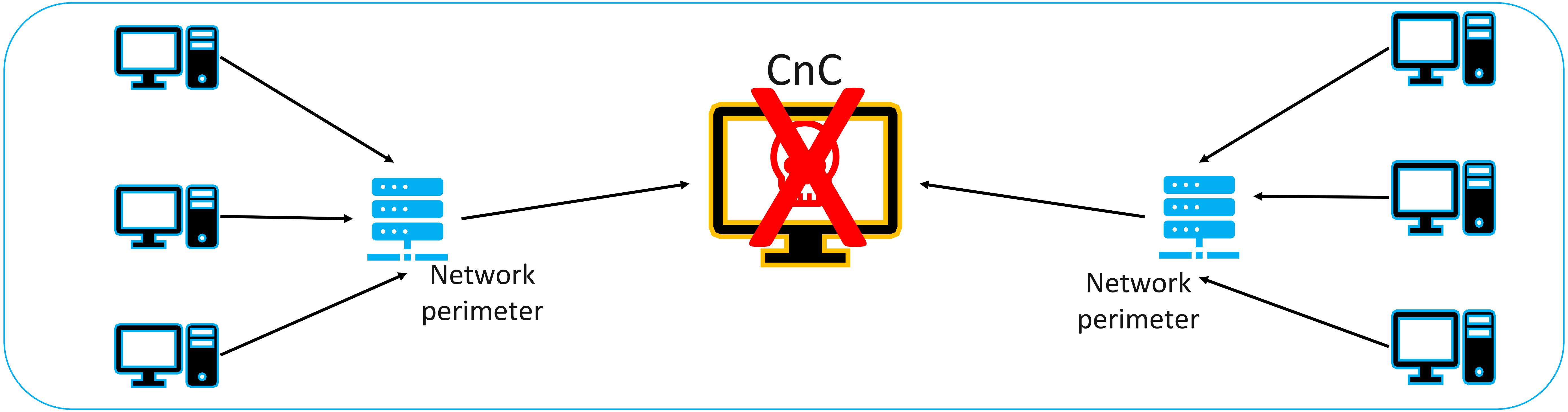
Ericsson has launched a new internet of things (IoT) security offering, Threat Monitoring and Mitigation (TMM), citing its own research that shows there will be nearly six billion cellular IoT devices in use by the end of 2026, and security will be a critical factor in their deployment by

Latest News

CnC-Hunter
EXPLOITING MALWARE

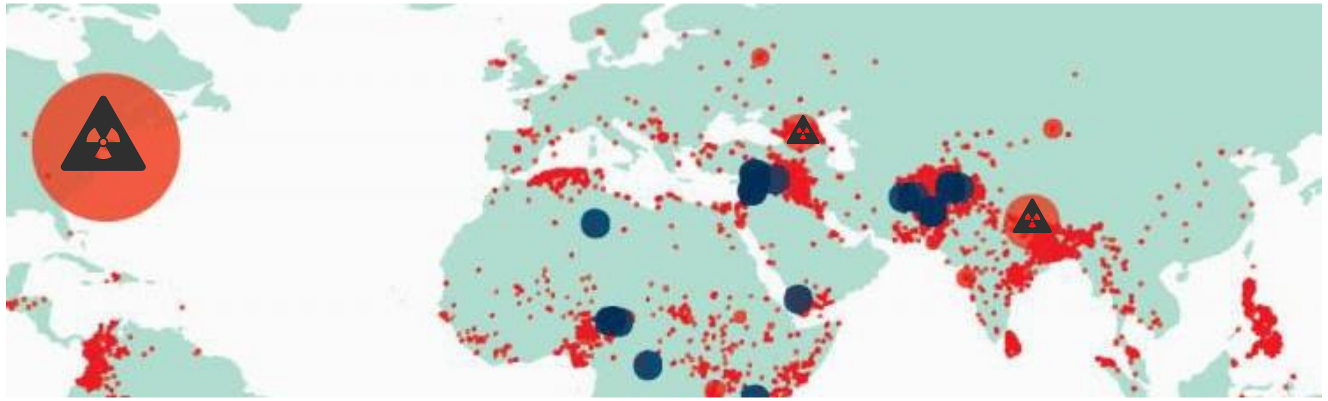
CnCs are the Achilles Heel of IoT botnets

- Understanding Command and Control (CnC) Servers help:
  - Detecting, monitoring, mitigating (e.g blacklisting), subverting
- IoT devices have limited computing resource and hence:
  - Defense at the network perimeter



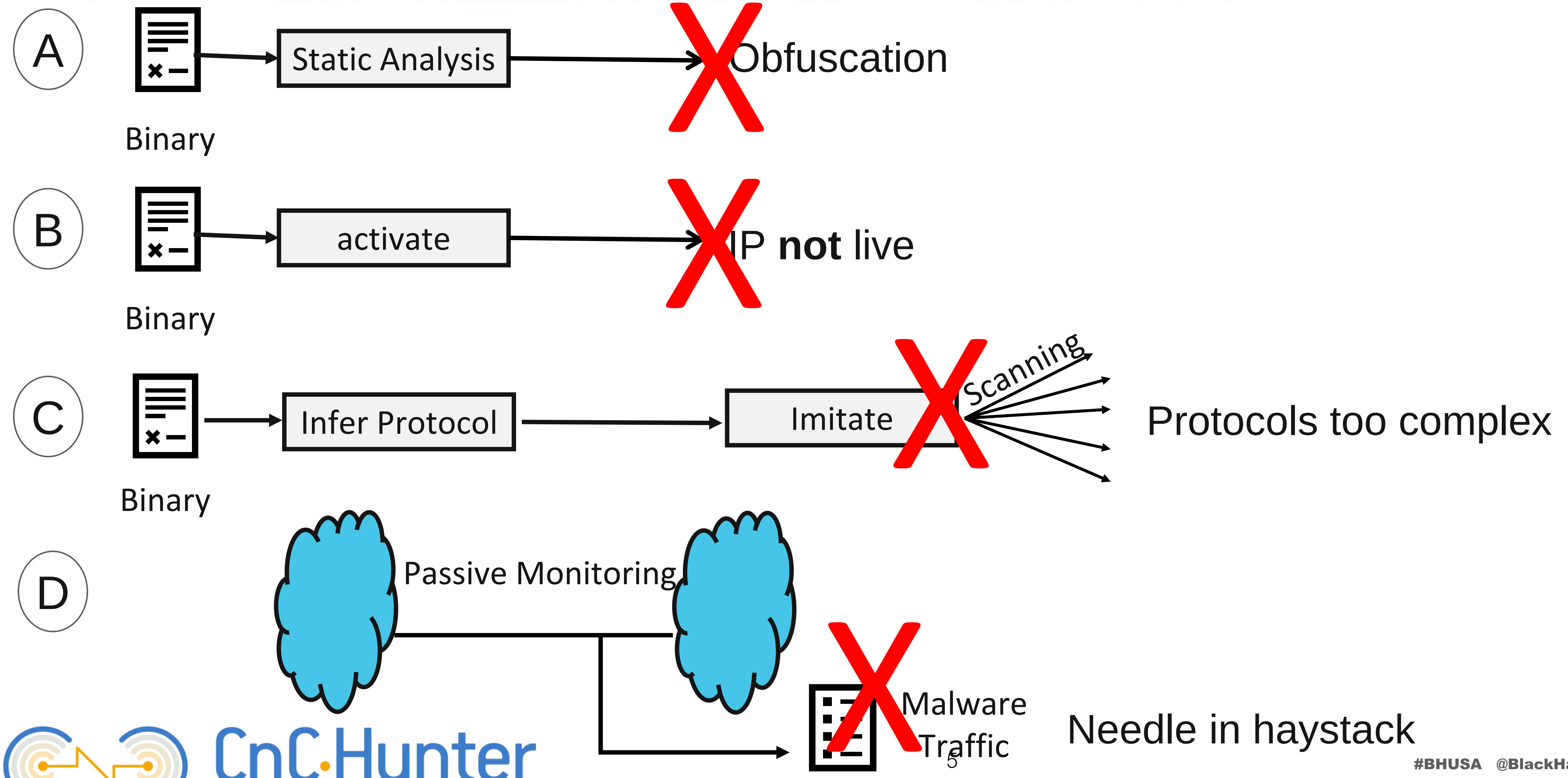
Botnet

Problem Definition



- **Goal**: Find all live CnC servers
- **Available public Information:**
  - Malware binaries
  - IP blacklists
  - Malware communication protocols (from threat reports)
- **Scope**
  - No access to network traffic
  - No access to AV companies' sensors
  - We are independent researchers

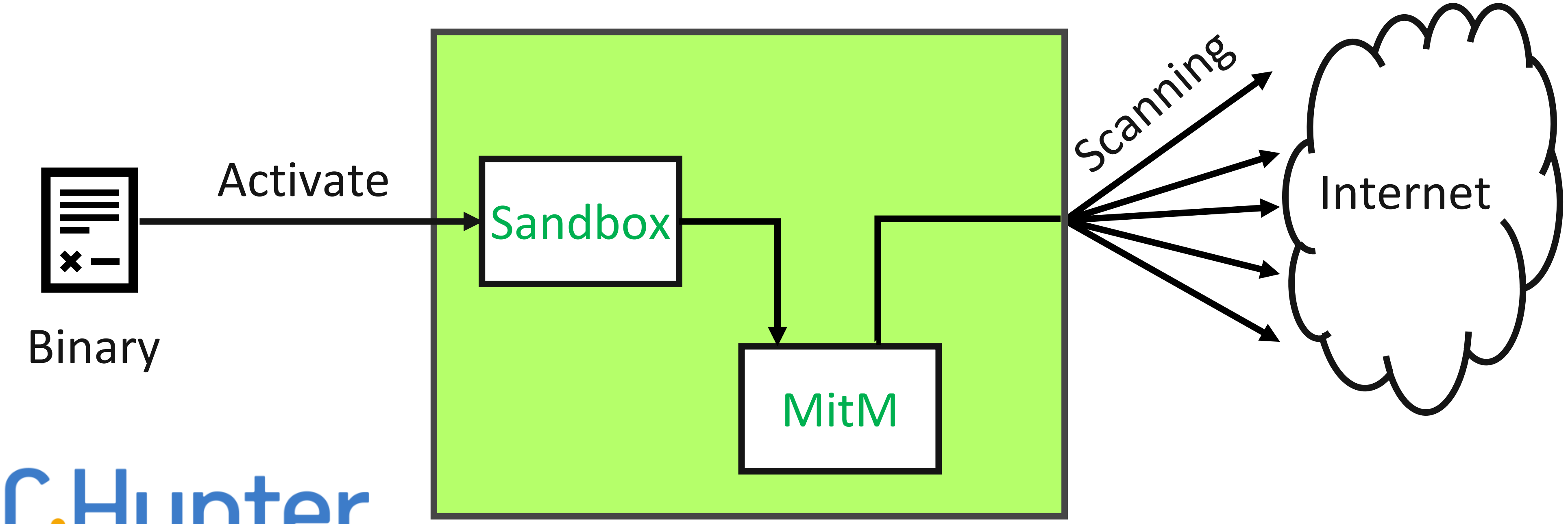
Previous approaches



Our Solution: CnC Hunter

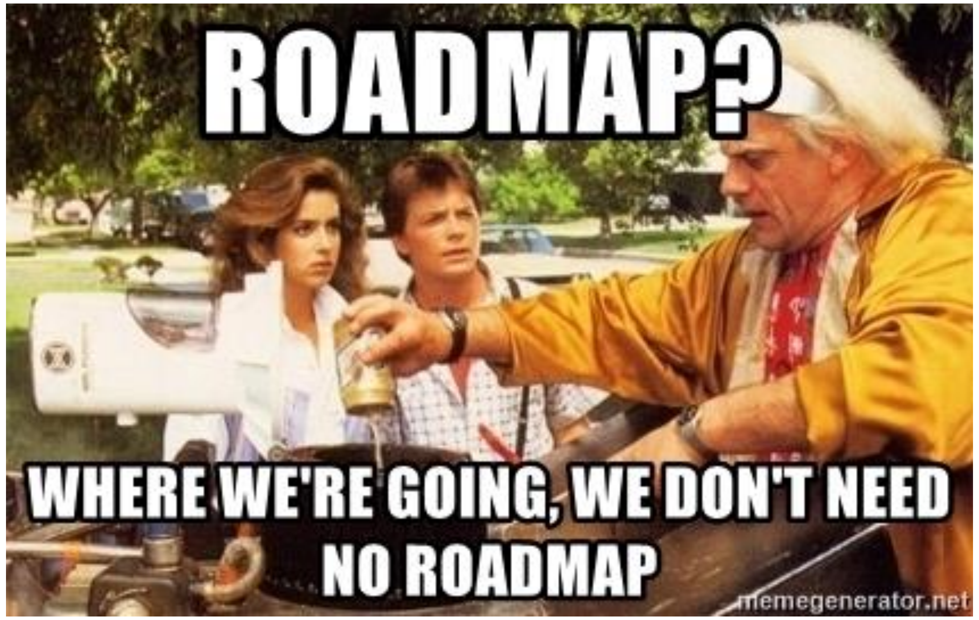
- The first open source tool designed for finding IoT malware CnCs
  - https://github.com/adava
- **Our novelty is a Man in the Middle (MitM) approach to CnC discovery:**
  - Activate the IoT malware
  - Channel the real CnC communication to potential candidates

CnC Hunter



Overview

- IoT malware network communication
- Previous Work
- CnC Hunter
  - Design
  - Implementation
  - Evaluation
- Demo
- Collaboration
- Closing Remarks



IoT CnC protocols are diverse

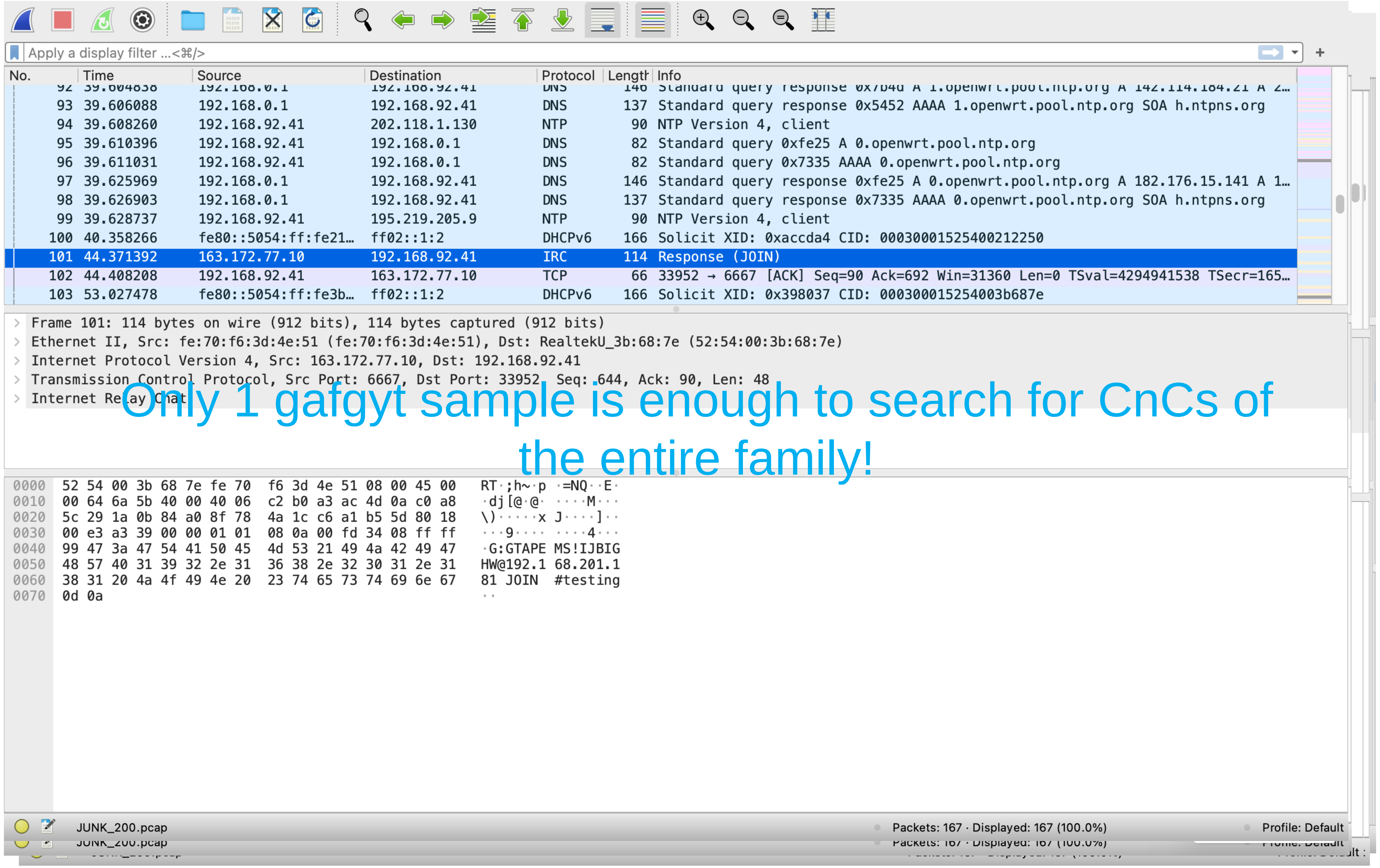
- Communication protocols Complexity:
  - Custom binary protocols
  - Encryption
- Diversity makes generic probing hard

Malware	Communication	Details
Gafgyt	Custom application layer protocol	One IRC command (PONG), other text commands
Mirai	Custom application layer protocol	Binary commands
Lightaidra	IRC protocol	Wraps C2 commands inside PRIVMSG (private) messages
Linux.wifatch	Custom application layer protocol	Binary commands
Remaiten	IRC protocol	Wraps inside PRIVMSG (private) messages
Lizkebab	Custom application layer protocol	One IRC command (PONG), other text commands
LuaBot	Encrypted payload	MatrixSSL library for encryption
Torlus	Custom application layer protocol	One IRC command (PONG), other text commands
Tsunami	IRC protocol	Wraps C2 commands inside NOTICEmessages
BASHLIFE	Custom application layer protocol	One IRC command (PONG), other text commands

Communication protocol barely changes within a family

- We used a generic IRC server to imitate Gafgyt malware CnC

- Gafgyt family protocol
  - Text based
  - Similar to IRC



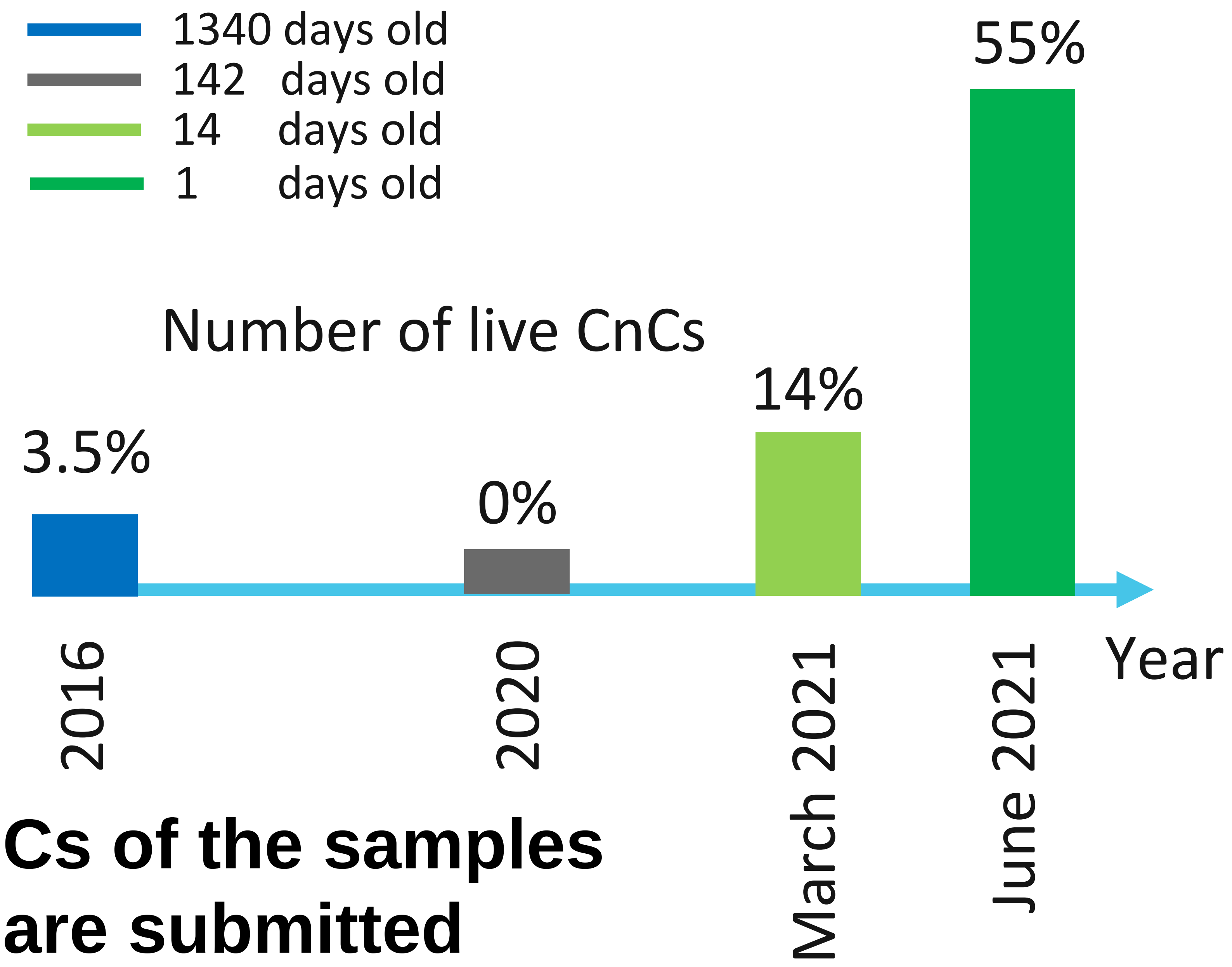
Only 1 gafgyt sample is enough to search for CnCs of the entire family!

99.5% of our Gafgyt samples successfully communicated!

CnC·Hunter
EXPLOITING MALWARE

CnC servers are short lived

- We manually analyzed 100 IoT malware and found their CnC server



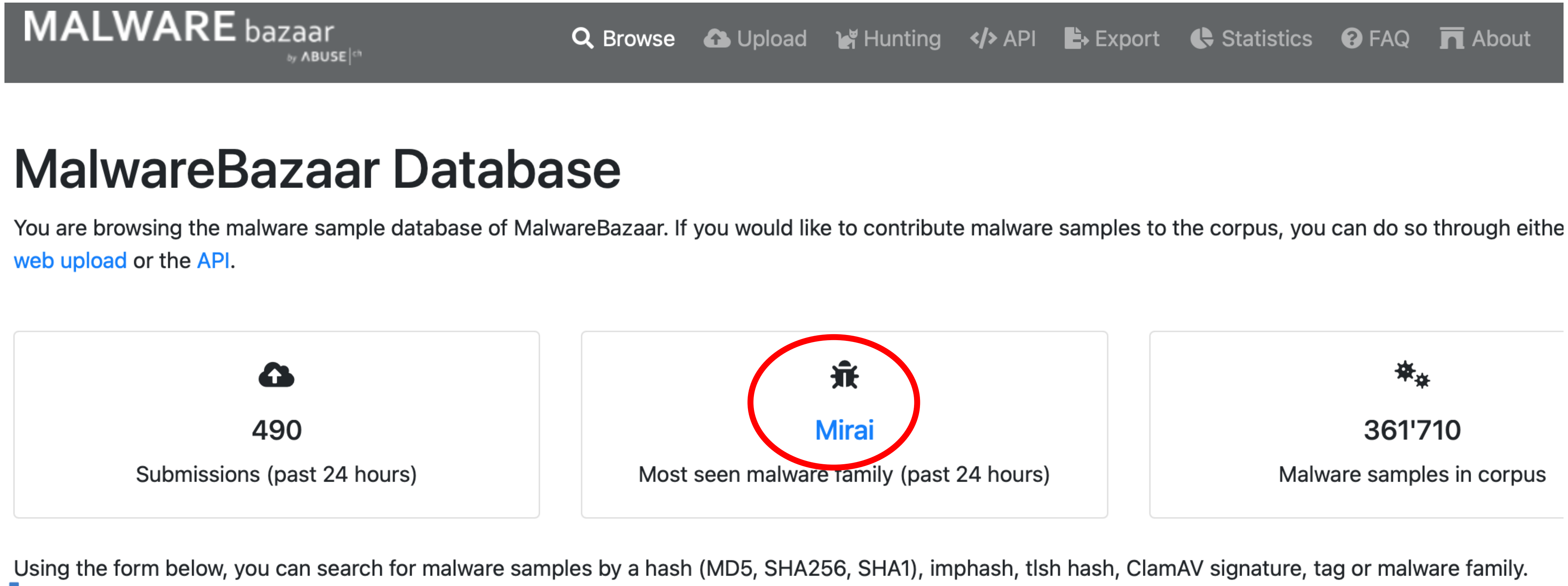
- **Roughly, <u>only half</u> of the CnCs of the samples are <u>live</u> by the day they are submitted**

CnC discovery using active probing

Related work is mainly focused on actively probing the Internet in search of CnC servers

Mirai Malware

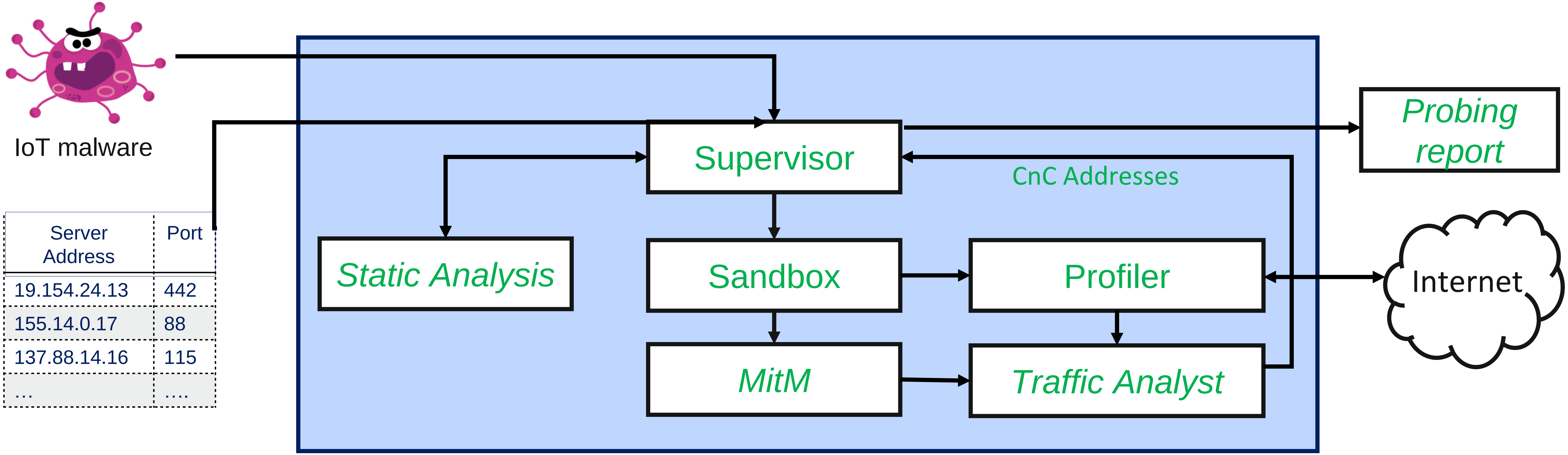
- First appeared in August 2016
- Responsible for disrupting several high-profile websites: including Github, Twitter, Reddit, Netflix, Airbnb
- Still very active!



Active Probing for Mirai

- After a successful infection, the bot starts communicating with its CnC

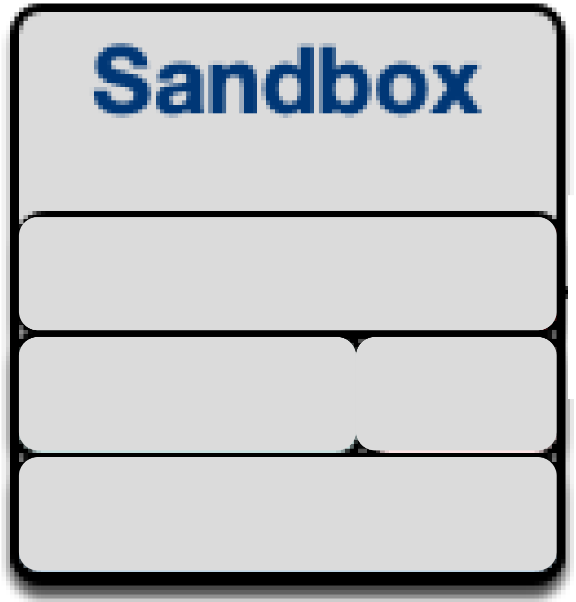
CnC Hunter Design



Server Address	Port
19.154.24.13	442
155.14.0.17	88
137.88.14.16	115
…	….

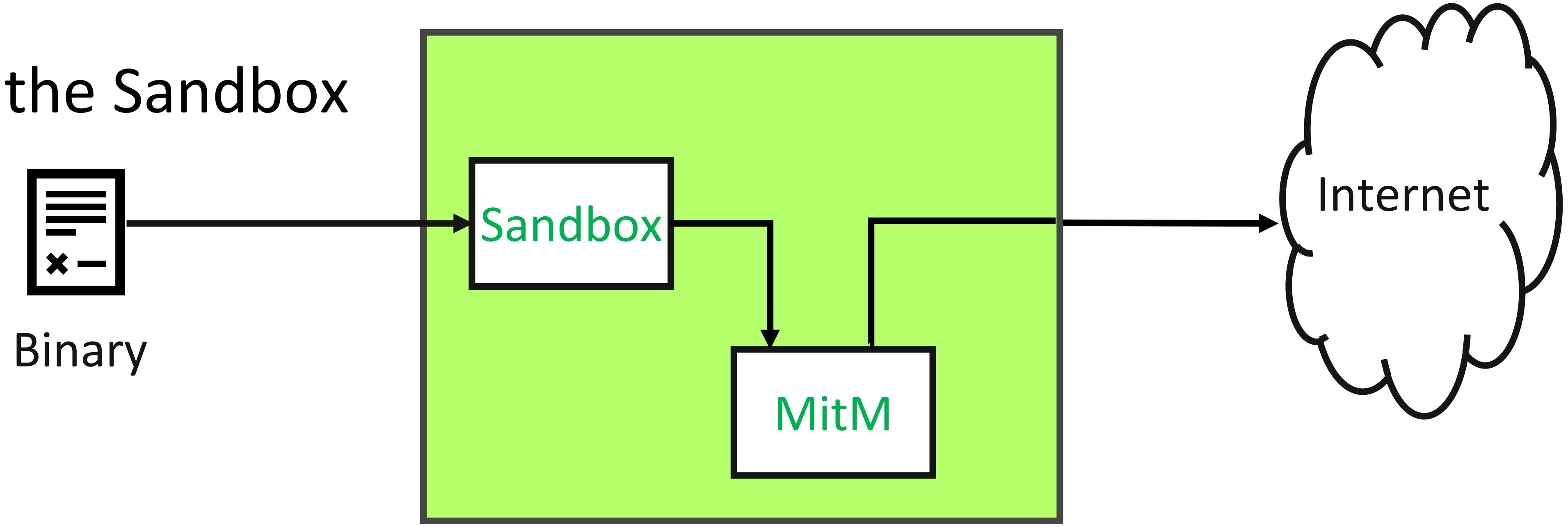
The architecture of **CnC Hunter**

Sandbox and Profile Modules



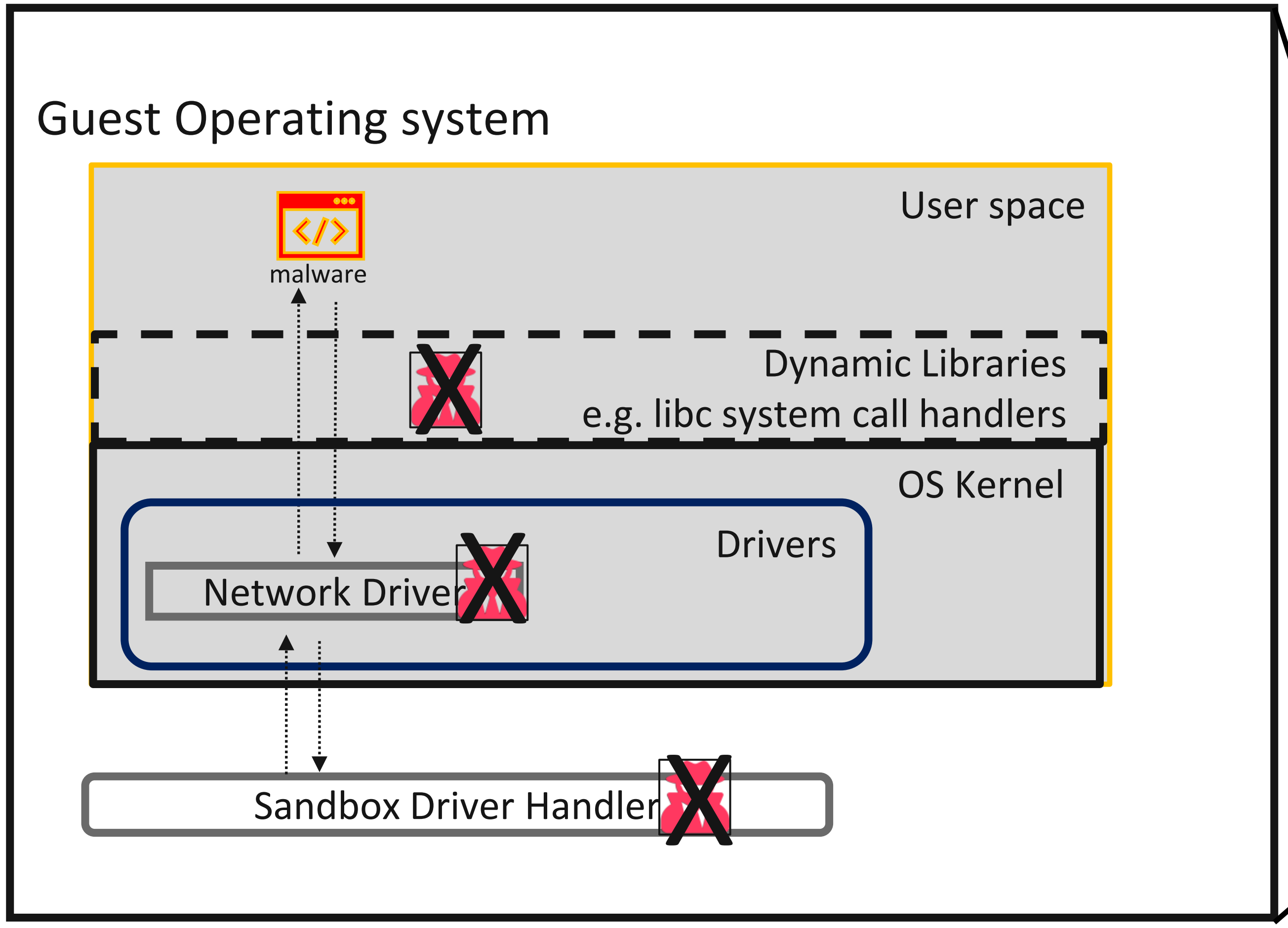
MitM & Network Proxy modules

- MitM
  - Redirect CnC traffic to candidate addresses
  - IP based
- Network Proxy
  - Tap malware traffic
  - Provide Internet for the Sandbox

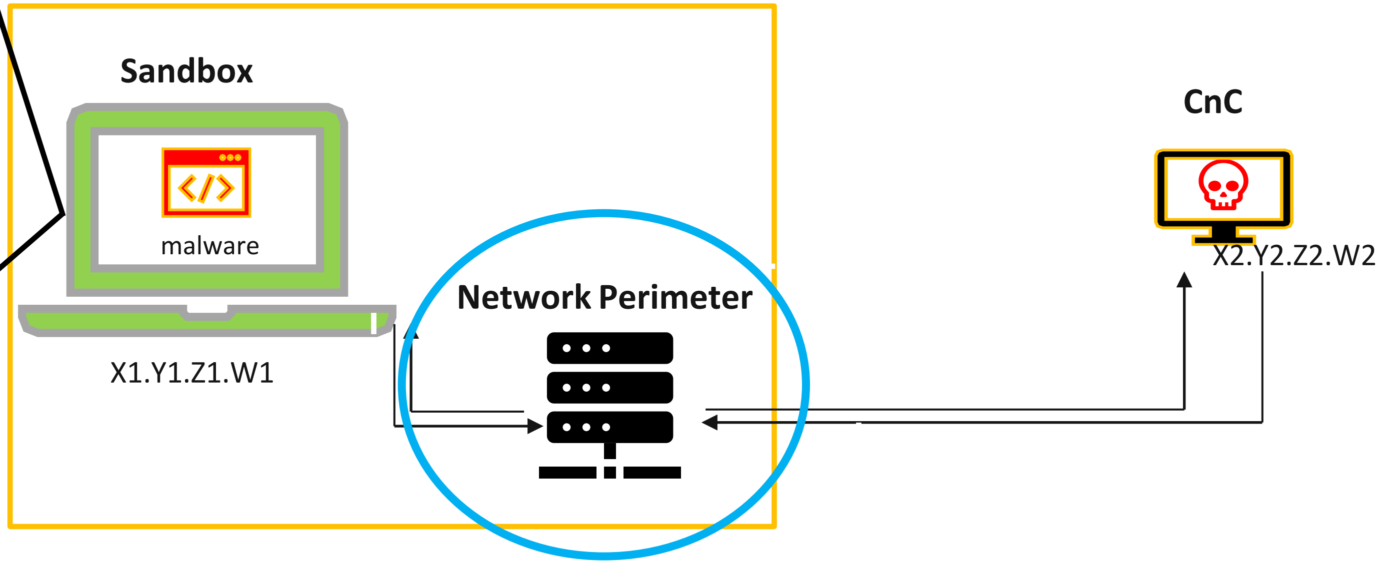


MitM module - Solution design

Sandbox: Qemu

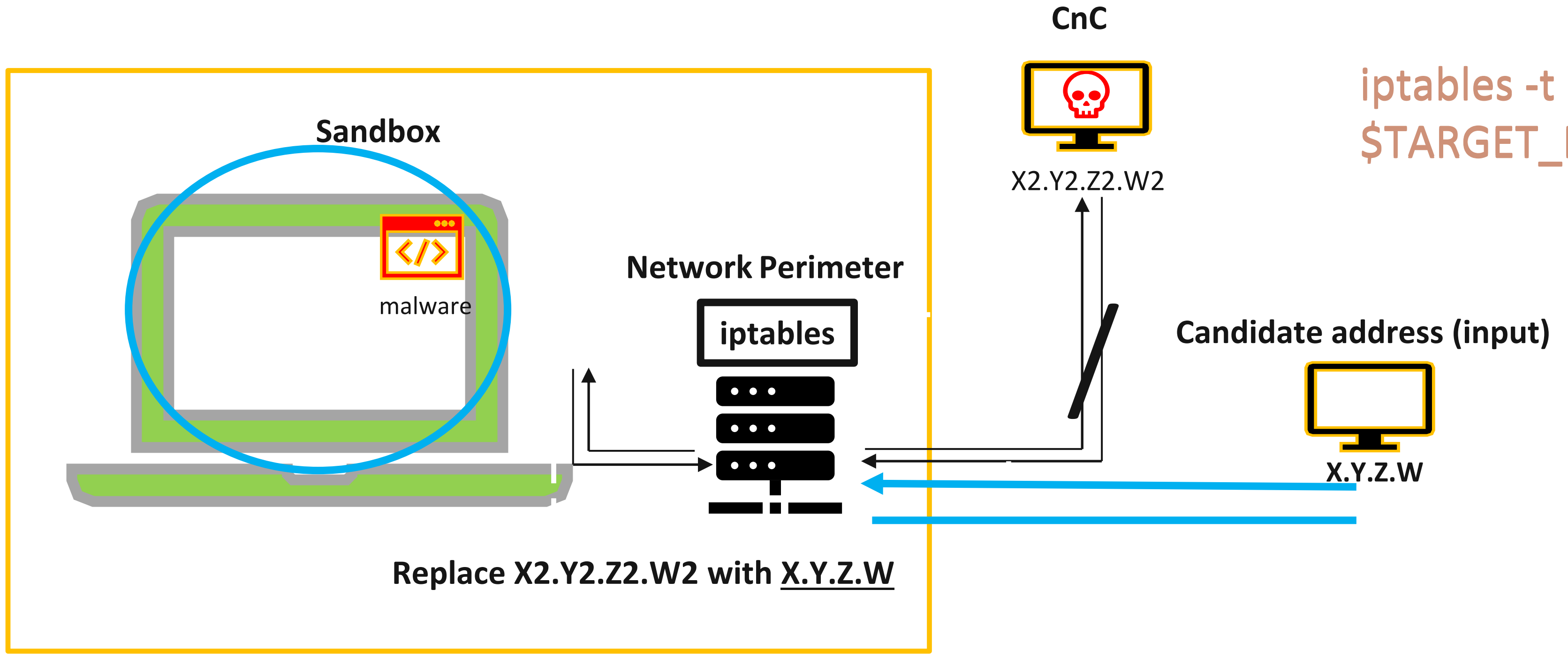


- ~~Alternative 1: Hooking system calls (libc)~~
- ~~Alternative 2: Network Driver Instrumentation~~
- ~~Alternative 3: Emulator Instrumentation~~
- **Alternative 4: Proxy Redirection**



MitM must happen on the guest (implementation details)

- We use iptables for traffic proxying



That POSTROUTING rule wouldn’t work

iptables -t nat -A POSTROUTING -p tcp -d X2.Y2.Z2.W2 --dport $TARGET_PORT -j DNAT --to-destination **X.Y.Z.W**

echo "$CnC_DNS_ADDR $CANDIDATE_IP" >> /etc/hosts

How can we support DNS based CnC addresses?

- Manipulate local DNS resolution
- Resolve CnC DNS address to the candidate address

What traffic should be redirected?

Only the traffic to the CnC

Finding CnC Traffic

Finding which traffic is to the CnC is non trivial!

Apply a display filter …<⌘/>

No.	Time	Source	Destination	Protocol	Length	Info
222	83.695642	192.168.203.3	68.212.180.4	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
223	83.700152	192.168.203.3	51.106.233.125	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
224	83.719374	192.168.203.3	84.178.150.184	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
225	83.739451	192.168.203.3	46.147.188.26	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
226	83.747724	192.168.203.3	141.104.2.31	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
227	83.767839	192.168.203.3	32.52.20.18	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
228	83.791153	95.70.179.145	192.168.203.3	TCP	54	23 → 58214 [RST, ACK] Seq=1 Ack=1 Win=0
229	83.799526	192.168.203.3	195.247.54.62	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
230	83.819499	192.168.203.3	80.177.4.96	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
231	83.832206	192.168.203.3	190.53.144.107	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
232	83.884096	192.168.203.3	163.115.13.172	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
233	83.899219	84.178.150.184	192.168.203.3	ICMP	70	Destination unreachable (Communication a
234	83.915564	192.168.203.3	192.31.78.180	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
235	83.920135	192.168.203.3	196.97.165.90	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
236	83.947883	192.168.203.3	67.162.107.12	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
237	83.976170	192.168.203.3	91.239.7.116	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
238	83.999656	192.168.203.3	192.125.164.21	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
239	84.087457	192.168.203.3	71.241.137.179	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
240	84.092093	192.168.203.3	135.13.237.164	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
241	84.119024	192.168.203.3	194.29.137.93	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
242	84.167565	192.168.203.3	9.16.13.138	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0
243	84.204145	192.168.203.3	152.198.5.136	TCP	54	58214 → 23 [SYN] Seq=0 Win=60271 Len=0

Other Traffic:

1. Proliferation (Scanning)
2. Background
3. Random

Traffic Analysis Module

- Find a sample's original CnC server
- **Tool**: Pyshark

code — Python — 103×34

```
Layer TCP:
        Source Port: 30726
        Destination Port: 23
        Stream index: 181
        TCP Segment Len: 0
        Sequence Number: 0    (relative sequence number)
        Sequence Number (raw): 3399321086
        Next Sequence Number: 1    (relative sequence number)
        Acknowledgment Number: 0
        Acknowledgment number (raw): 0
        0101 .... = Header Length: 20 bytes (5)
        Flags: 0x002 (SYN)
        000. .... .... = Reserved: Not set
        ...0 .... .... = Nonce: Not set
        .... 0... .... = Congestion Window Reduced (CWR): Not set
        .... .0.. .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...0 .... = Acknowledgment: Not set
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..1. = Syn: Set
        Expert Info (Chat/Sequence): Connection establish request (SYN): server port 23
        Connection establish request (SYN): server port 23
        Severity level: Chat
        Group: Sequence
        .... .... ...0 = Fin: Not set
        TCP Flags: ··········S·
        Window: 20523
        Calculated window size: 20523
        Checksum: 0xbefe [unverified]
        Checksum Status: Unverified
        Urgent Pointer: 0
        Timestamps
        Time since first frame in this TCP stream: 0.000000000 seconds
```

Find_CnC algorithm

- Assign a ***score*** to each IP address (in a malware traffic)
  - $Score \propto Connection\_frequency$
  - $Score \propto \frac{1}{port\_frequency}$
  - $Score = coefficient * \frac{connection\_frequency}{port\_frequency}$
- CnC address has the highest ***Score***

Find_CnC algorithm data structures

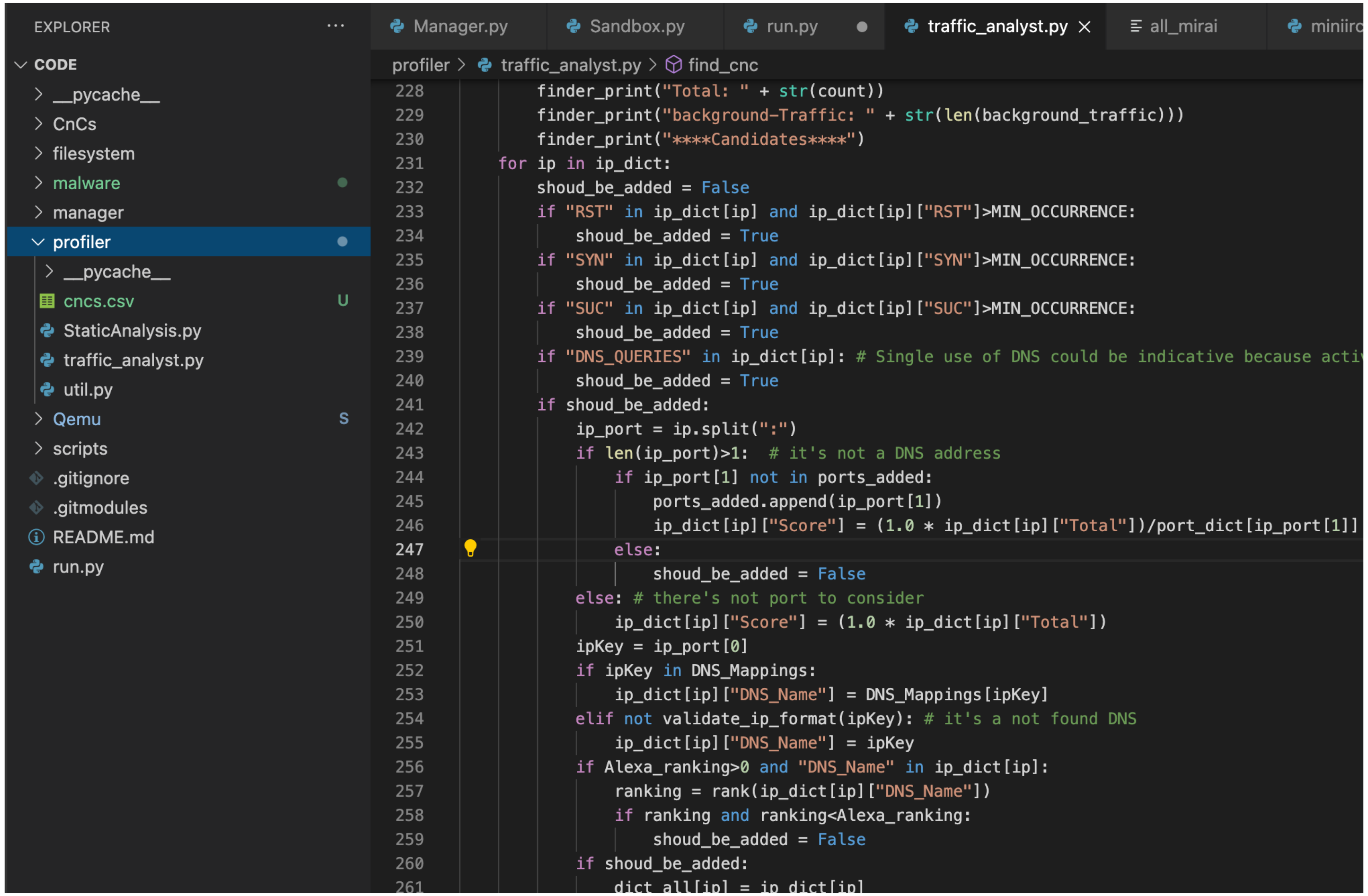
- How can we calculate the score for each IP address?

Address Hash Table	RST flag count	SYN flag count	ACK flag count	DNS not found
155.10.1.4:32134	2	9	0	0
evil.domain.com	0	0	0	8
19.1.143.12:80	0	1	11	0
…				

port Hash Table	
80	123
23	234
32134	1
443	85

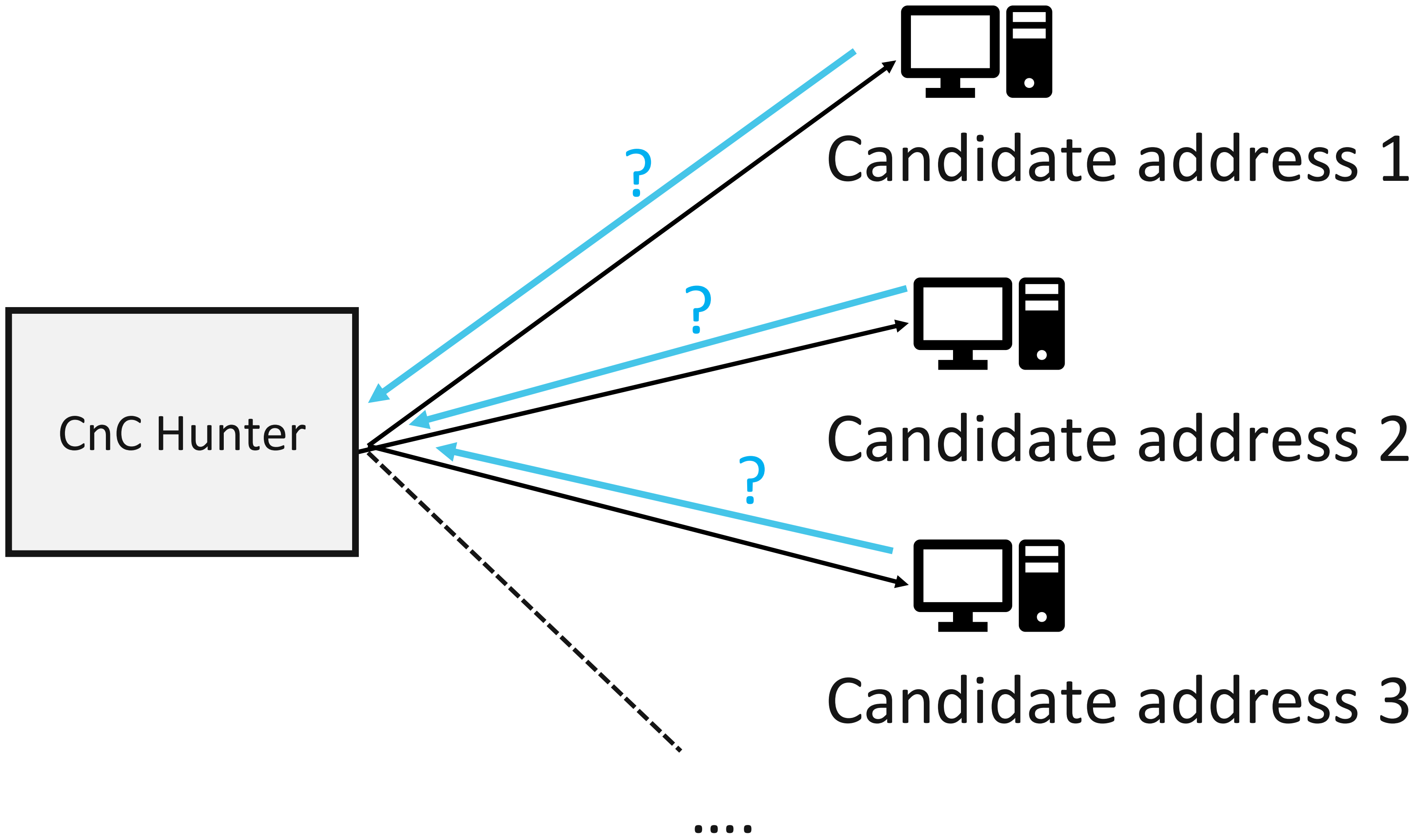
Find_CnC algorithm

- Other functionalities of Find_CnC are:
  - Port filtering
  - Reputation filtering



CnC·Hunter
EXPLOITING MALWARE

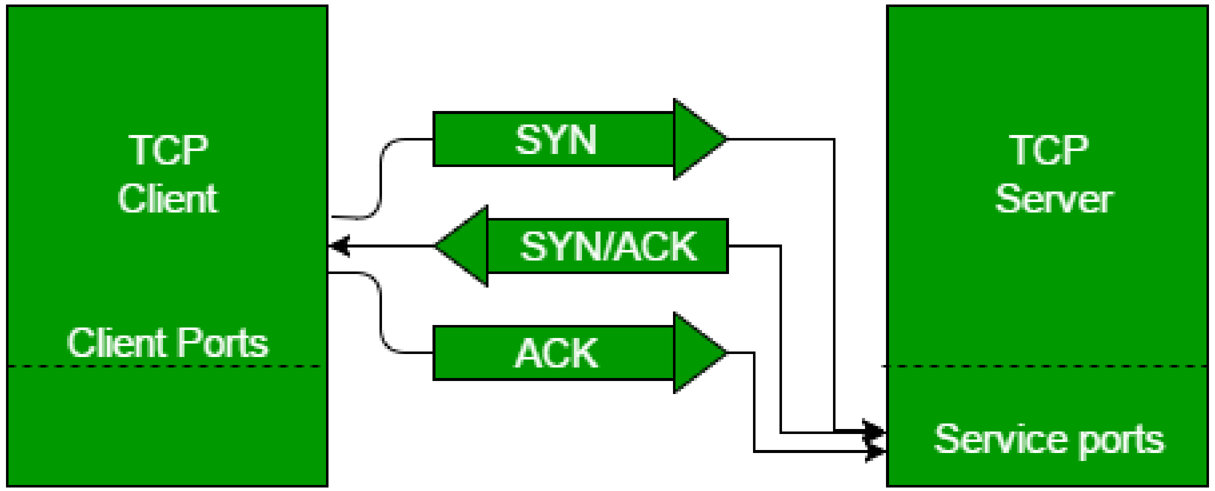
Which candidate address is a CnC?



Whoever is live, listens to our target port and is not filtered will respond!

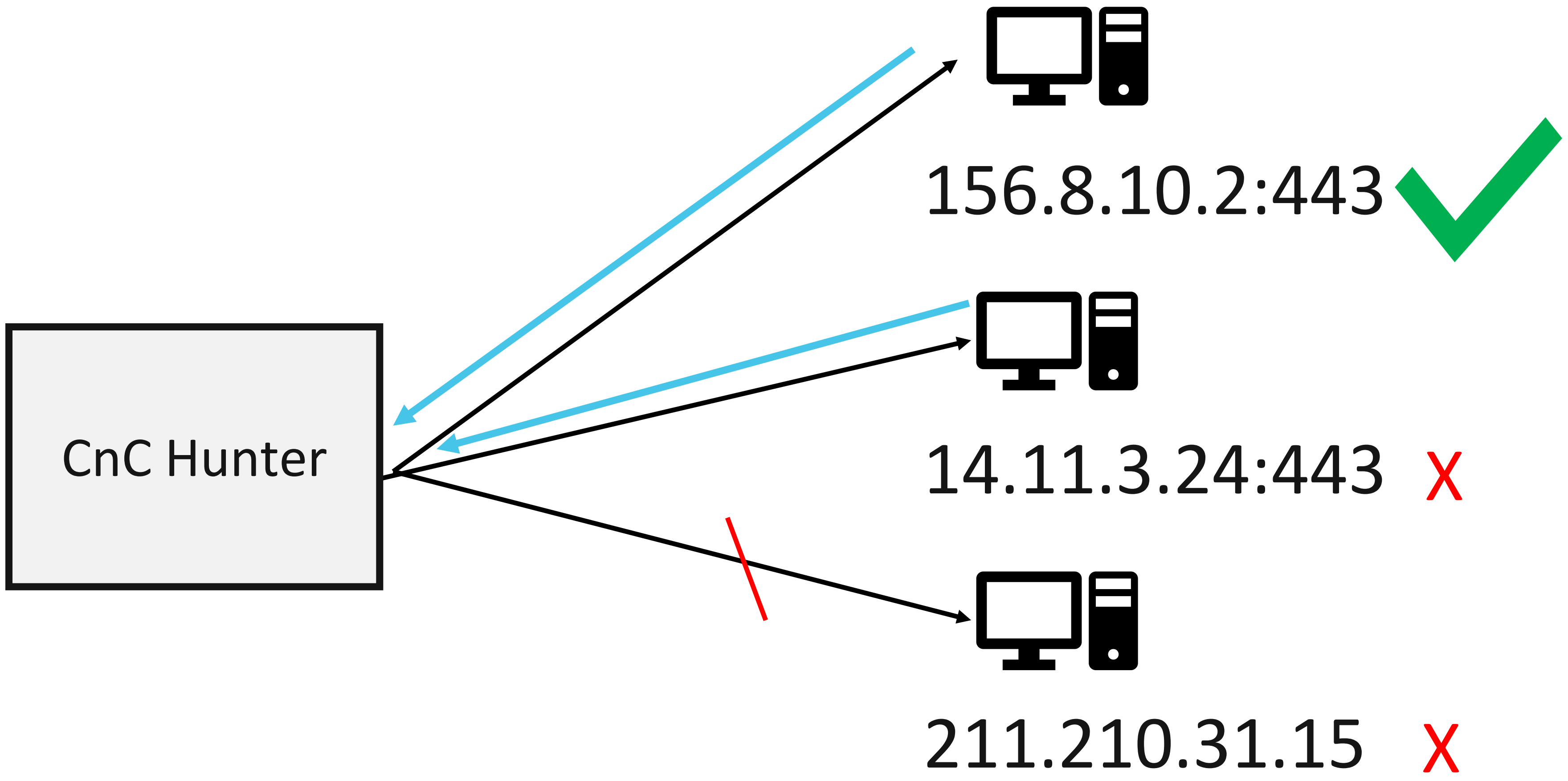
Evaluating candidates' responses

- Live => there is a response
- Listens => successful TCP handshake
- Not filtered => no RST flag



Are these enough?

Which candidate address is a CnC?



The second address listens and responds to requests on port 443 but is not CnC

Which port listener is a CnC?

- We observed that CnCs respond with *Significantly* ***lower*** *number of SYN flag*
- We use simple SYN frequency *outlier detection based on* *standard deviation*

CnC Hunter is accurate!

- We evaluated CnC finding functionality of CnC Hunter
- **Dataset:** A set of 100 samples collected between 2016 to 2021
  - Mirai, Gafgyt, Tsunami, Remaiten, LightAidra and VPNFilter
  - Could activate 90% of samples

Precision of CnC finding: 92%

Exclusively found by CnC Hunter: 18%

Demo

- Demo 1: Given an unknown IoT malware binary, find its CnC server
  - Malware: Mirai sample
  - Challenge: Identify the CnC address among all traffic (scanning, infiltration etc.)

- Demo 2: Given the malware and IP addresses, find a live CnC server
  - Malware: Gafgyt
  - The target address: CnC of Gafgyt/Mirai/BashLite (according to VT) CnC

File Edit View Search Terminal Help

```
osboxes@osboxes:~$ cd CnC_Hunter/
osboxes@osboxes:~/CnC_Hunter$ su
Password:
root@osboxes:/home/osboxes/CnC_Hunter# ls
analysis  filesystem  killAll.sh  manager   Qemu         README.md  run.py   start_network.sh  z_stop.sh
CnCs      kernels     malware     profiler  qemu_helper  report     scripts  stop_network.sh
root@osboxes:/home/osboxes/CnC_Hunter# ls malware/malware/
7bf2d60dcbba36b48647684728a525d378f99ace9f9146902abbb210b762a302.elf
root@osboxes:/home/osboxes/CnC_Hunter# python3 run.py -p 23 2323
```

File Edit View Search Terminal Help

```
osboxes@osboxes:~$ cd CnC_Hunter/
osboxes@osboxes:~/CnC_Hunter$ su
Password:
root@osboxes:/home/osboxes/CnC_Hunter# ls
analysis    kernels    profiler     report                              scripts
CnCs        killAll.sh Qemu         riotman_06-17-2021-00_35_27.log     start_network.sh
cncs.csv    malware    qemu_helper  riotman_06-17-2021-00_56_46.log     stop_network.sh
filesystem  manager    README.md    run.py                              z_stop.sh
root@osboxes:/home/osboxes/CnC_Hunter# ls malware/malware/
4a0ec48aac4097b1484ce731ac6ab97ae6b105345809beb6668f250be2fcc3e4
87e9b4c47d8fe3fd651aa222826d2a6e47e071b02e573c33e302057375121cea
root@osboxes:/home/osboxes/CnC_Hunter# python3 run.py -t 198.46.188.140:23
```

We're open to collaboration

Please talk to us if

- You have an active honeypot
- You have reliable IoT filesystems
- You have Intelligence on IoT malware CnCs
- You used our tool
- You have insight on IoT malware AV evasion

CnC·Hunter
EXPLOITING MALWARE

Acknowledgement

- A Shout out to Martina Lindorfer and VirusTotal

I (Ali Davanian) will be in job market in 6 months!

Takeaway messages

- We need to proactively scan the Internet and find CnC servers because:
  - CnC servers are very short lived
  - IoT malware communication protocols are diverse and complex; hence, real malware is needed for probing
- CnC Hunter provides a CnC discovery solution via Man-In-The-Middling malware
  - CnC Hunter is fully automated
  - CnC Hunter is open source

Q&A

- CnC Hunter repository:
  - https://github.com/adava

- Twitter Handle:
  - @sinaDavanian
- Email:
  - adava003@ucr.edu

- Twitter Handle:
  - @adarkione
- Email:
  - adark001@ucr.edu