Hiding in Complexity

Marc “van Hauser” Heuse
GSEC Singapore 2015
Hello, my name is ...
I want to talk about:

1. The power of /64
2. IDS bypasses
/64
/56
/48
18.446.744.073.709.551.616
4.722.366.482.869.645.213.696
1.208.925.819.614.629.174.706.176
Be millions
Be millions

Diagram showing network traffic from sources A, B, C, and D to a destination with ports 1 to 4.
Scan as millions

# parasite6 eth0 &
# alive6 -I 2001:db8::/64
   -i targets.txt eth0
# alive6 -I 2001:db8::/64
   -s portscan eth0 target
DOS as millions

# thcsyn6 -r eth0 TARGET PORT
# ndpexhaust26 -r eth0 TARGET/64
while : ; do
    IP=`printf 2001:db8::%x:%x \n        $RANDOM $RANDOM`
    ip -6 addr add $IP/64 dev eth0
    curl -6 --interface $IP \n        http://target/vote?choice=3
    ip -6 addr del $IP/64 dev eth0
done
How to protect?

- Always block a full /64
  - Attackers from DSL lines will have 256 tries
  - Attackers from companies/tunnels 65536 tries

- Voting: tie to an account
Split up connections!

New tool: connsplit6
IDS, right?
IPv6 Protocol Background
IPv6 encapsulation with extension headers

1. IPv6 Header
   Next Header = 6
   TCP Header
   Application Data

2. IPv6 Header
   Next Header = 0
   Hop-by-Hop Header
   Next Header = 6
   TCP Header
   Application Data

3. IPv6 Header
   Next Header = 60
   Destination Header
   Next Header = 44
   Fragment Header
   Next Header = 17
   UDP Header
   Data
Hop-by-Hop / Destination Header

Next Header | Length | Option Number | Length | Value Value | Padding

8 byte length
Fragmentation Header

- Next Header
- Length = 0
- Fragmentation Offset (in Octets)
- Fragmentation ID
- Reserved Bit
- More Fragments Bit

4 byte length
How to find IDS bypasses?

1. Test target OS: what packet weirdness is accepted?
2. Create test cases: how could accepted packet weirdness used for IDS bypasses?
3. Try on a an IDS ranch setup
The Disruptor Packets
Simple disruption against Snort

| Data
| TCP Header
| Destination Extension Header
| Destination Extension Header
| Destination Extension Header
| Destination Extension Header
| Destination Extension Header
| Destination Extension Header
| Destination Extension Header
| Destination Extension Header
| Destination Extension Header
| IPv6 Header
Snort is helplessly crying

Snort snort: [116:456:1] (snort_decoder)
WARNING: too many IP6 extension headers
[Classification: Misc activity]
[Priority: 3] {IPV6-OPTS}
2001:db8:b42:0:3e97:eff:fee8:57df ->
2001:db8:a42:0:de4:7af8:f11e:29ad

config max_ip6_extensions: 8
The Ninja Packets
<will show you several examples 😊>
Test Step 1: what packet weirdness is accepted?
firewall6 eth0 target

Windows & Linux
Linux 3.18

- Unlimited destination headers
- Only one of each other extension header type
- One fragmentation header only
- Extension headers may not be fragmented
- No change of next header type in fragmentation chains (ID + proto is hashed)
- No overlapping fragments
Windows 7

• Unlimited headers of any kind
• Unlimited fragmentation headers
• Extension headers may be fragmented
• No change of next header type in fragmentation chains
• No overlapping fragments
• Resending fragments with different data: last received is used
Test Step 2+3: create & test IDS bypasses based on accepted packet weirdness used for IDS bypasses
The IDS Test Bench

Thanks to ERNW for the support!
All configured to highest settings

- Newest update of engine and rules (27\textsuperscript{th} August 2015)
- Snort & Suricata: *all* rules enabled
- Tipping Point: Hyper Aggressive
# Bypasses

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Suricata</th>
<th>Snort</th>
<th>TippingPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>1 fragmentation EH</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>2 fragmentation EH</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>9+ fragmentation EH</td>
<td>✔️</td>
<td>✔️</td>
<td>STOP</td>
</tr>
<tr>
<td>Large dst EH that fragments</td>
<td>✔️</td>
<td>✔️</td>
<td>STOP</td>
</tr>
<tr>
<td>Mini fragments</td>
<td>✔️</td>
<td>✔️</td>
<td>STOP</td>
</tr>
<tr>
<td>Fake TCP data (HC-1)</td>
<td>STOP</td>
<td>✔️</td>
<td>STOP</td>
</tr>
<tr>
<td>Fake RST (HC-1)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Fake fragmented TCP data (HC-1)</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Fake 9+ fragmented TCP data (HC-1)</td>
<td>STOP</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
fraggerouter6
fragrouter6

• Linux ip6tables NF queues
• WIP
• Use any existing tool (nmap, OpenVAS, ...): bypass modifications are done transparently! 😊
fragrouter6

- Send any number of fragmentation and destination headers
- Fragment packets to any size
- Fragment over large destination header
- Hop Count minus 1 attacks:
  - TCP RST
  - TCP fake data
- ... more to come!
How to protect?

• Filter any EH with the exception of one fragmentation header

• Needs a new RFC for specific extension header definitions
  – Order of EHs
  – # of occurrence of Ehs

• Good start but incomplete:
  – “Implications of Oversized IPv6 Header Chains”
    (draft-ietf-6man-oversized-header-chain-09)
And finally ...
flood_router26 –s eth0

*** Panic Report ***
panic (cpu 7 caller 0xffffff8002a16df2): Kernel trap at 0xffffff8002cf80eb, type 14=
page fault, registers:
CR0: 0x000000008001003b, CR2: 0x0000000000000060, CR3: 0x0000000059e9b01c, CR4: 0x00000000001627e0
RAX: 0x0000000000000000, RBX: 0x0000000000000000, RCX: 0x0000000009000000, RDX: 0xffffff802c96ef10
RSP: 0xffffff81e1b9bab0, RBP: 0xffffff81e1b9bbc0, RSI: 0x0000000000000000, RDI: 0x0000000000000000
R8: 0x0000000000000000, R9: 0xffffff8006d12268, R10: 0x0000000000000378, R11: 0xffffff803645e914
R12: 0x00000000c1206950, R13: 0xffffff8029043008, R14: 0x000000000000000c, R15: 0x0000000000000000
RFL: 0x0000000000000000, RIP: 0xffffff8002cf80eb, CS: 0x0000000000000008, SS: 0x0000000000000010
Fault CR2: 0x0000000000000060, Error code: 0x0000000000000000, Fault CPU: 0x7
Backtrace (CPU 7), Frame : Return
0xffffff81e1b9b760 : 0xffffff800292ad21
0xffffff81e1b9b7e0 : 0xffffff8002a16df2
0xffffff81e1b9b9a0 : 0xffffff8002a33ca3
0xffffff81e1b9b9c0 : 0xffffff8002cf80eb
0xffffff81e1b9bbc0 : 0xffffff8002cc9899
0xffffff81e1b9bcc0 : 0xffffff8002b99aac
0xffffff81e1b9bd90 : 0xffffff8002b99cbf
0xffffff81e1b9bdc0 : 0xffffff8002df1f0a
0xffffff81e1b9be00 : 0xffffff8002dadddb
0xffffff81e1b9be30 : 0xffffff8002decd64
0xffffff81e1b9bf50 : 0xffffff8002e4b376
0xffffff81e1b9bfb0 : 0xffffff8002a344a6
BSD process name corresponding to current thread: configd
Mac OS version: 10.10.4

Your PC ran into a problem that it couldn’t handle, and now it needs to restart.
You can search for the error online: HAL INITIALIZATION FAILED

OS X Yosemite (configd)   Windows 10   Ubuntu (NetworkManager)
Questions?
Contact

Marc Heuse

+49 (0)177 961 15 60
+49 (0)30 37 30 97 26
mh@mh-sec.de
www.mh-sec.de
winsstrasse 68
d-10405 berlin
End