



## Fuzzing the easy way: Using Zulu

**Andy Davis, Research Director NCC Group**



# Who am I?

- NCC Group Research Director
- >20 years in information security
- Still very hands-on
- Enjoy testing more unusual technologies
- Also developing tools to test them

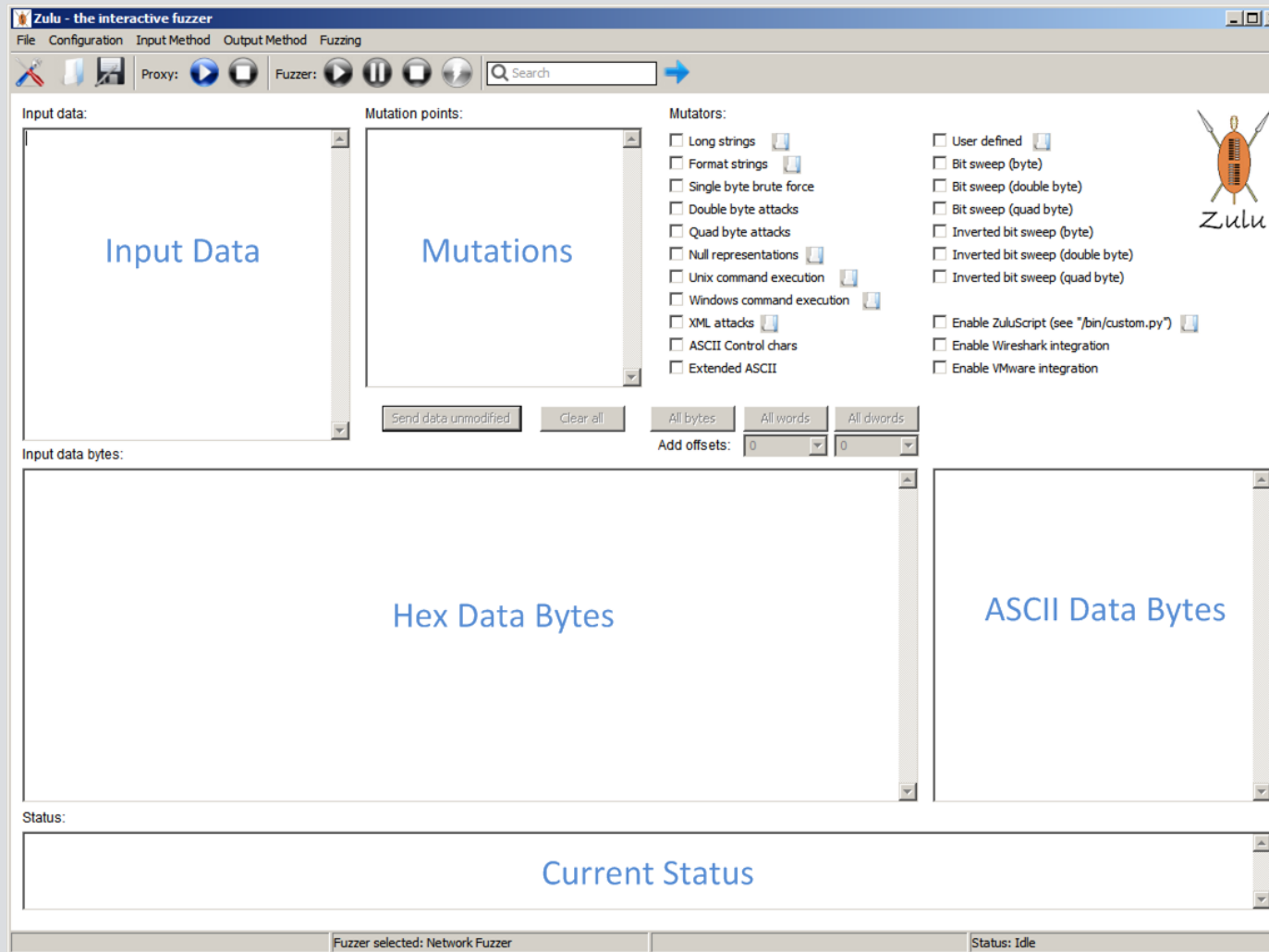
# What is Zulu?

- Zulu is an interactive GUI-based fuzzer
- Written in Python
- As much as possible, input and output-agnostic
- Multiple modules
- Extendible via ZuluScript

# Motivations behind the tool

- I had lots of unique “fuzzer scripts”
- Fuzzing frameworks have a steep learning curve
- Fuzzers should be quick and easy to setup
- Wanted a point-and-click solution
- Needed to be scriptable to add complexity where required

# Zulu basics – the GUI



# Zulu basics – typical data

The screenshot displays the Zulu fuzzer interface with the following sections:

- Input data:** A list of 17 packets, including Packet #0005 Out (0000 bytes) and Packet #0021 In (0084 bytes).
- Mutation points:** A list of mutation points such as Length:0-3 Pkt:6 and Fuzzpoint:13-14 Pkt:6.
- Mutators:** A list of mutators with checkboxes, including Long strings, Format strings, Single byte brute force, Null representations, XML attacks, and Extended ASCII.
- Input data bytes:** A hex dump of network traffic showing bytes like 00 00 00 d4 ff 53 4d 42 73 00 00 00 00 08 03 c8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 43 79.
- Network traffic:** A packet capture view showing SMB traffic with fields like .SMBs, .02, .J, .H, .>, .<, .0, .+, .7, .\*, .(NTLMSSP), .S.a.m.b.a., .VNET3BLU.U.n.i.x., .S.V.N., .b.u.i.l.d., and .1.1.S.7.2.V.N.E.T.3.

At the bottom, the status bar shows: Fuzzer selected: Network Fuzzer, Selected packet = 6, Status: Idle.

# Zulu basics – the console

```

C:\Windows\system32\cmd.exe - zulu.py
-----
New fuzzing session
-----

Total fuzzcases = 5

-----
Fuzzpoint 0/0, Testcase 0/4 Test type: user-defined, Test #0
-----

packet: 0

send
'\x01\x01\x00\xbe\x00\x00\x01\x00\x16\x00\x00\x00\x12\x00\x00\x00\x02\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00s\x00e\x00t\x00 \x00t\x00r\x00a\x00n\x00s\x00a\x00c\x00t\x00i\x00o\x00n\x00 \x00i\x00s\x00o\x001\x00a\x00t\x00i\x00o\x00n\x00 \x00t\x00e\x00v\x00e\x007\x00 \x00 \x00r\x00e\x00a\x00d\x00 \x00c\x00o\x00m\x00m\x00i\x00t\x00t\x00e\x00d\x00 \x00 \x00s\x00e\x00t\x00 \x00i\x00m\x00p\x007\x00i\x00c\x00i\x00t\x00_\x00t\x00r\x00a\x00n\x00s\x00a\x00c\x00t\x00i\x00o\x00n\x00s\x00 \x00o\x00f\x00f\x00 \x00'

receive
''

packet: 1

send
'\x03\x01\x01$\x00\x00\x01\x00\x16\x00\x00\x00\x12\x00\x00\x00\x02\x00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\xff\xff\r\x00\x00\x00\x00\x01&\x04\x04\x00\x00\x00\x00\x00\x00\xe7@\x1f\t\x04\xd0\x0044\x00@\x00P\x000\x00 \x00n\x00v\x00a\x00r\x00c\x00h\x00a\x00r\x00(\x004\x000\x000\x000\x00)\x00,\x00@\x00P\x001\x00 \x00i\x00n\x00t\x00\x00\x00\xe7@\x1f\t\x04\xd0\x004\x90\x00s\x00e\x007\x00e\x00c\x00t\x00 \x00*\x00 \x00f\x00r\x00o\x00m\x00 \x00t\x00e\x00s\x00t\x00_\x00t\x00a\x00b\x007\x00e\x00_\x001\x00 \x00w\x00h\x00e\x00r\x00e\x00 \x00n\x00a\x00m\x00e\x00 \x00=\x00 \x00@\x00P\x000\x00 \x00a\x00n\x00d\x00 \x00i\x00d\x00 \x00=\x00 \x00@\x00P\x001\x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00\xe7@\x1f\t\x04\xd0\x004\x06\x00z\x00z\x00z\x00\x00\x00&\x04\x04\x02\x00\x00\x00'

receive
packet: 2

send
receive

```

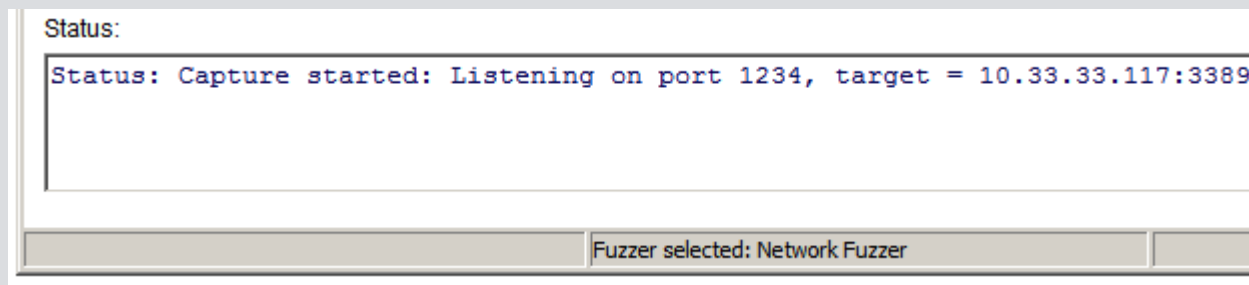
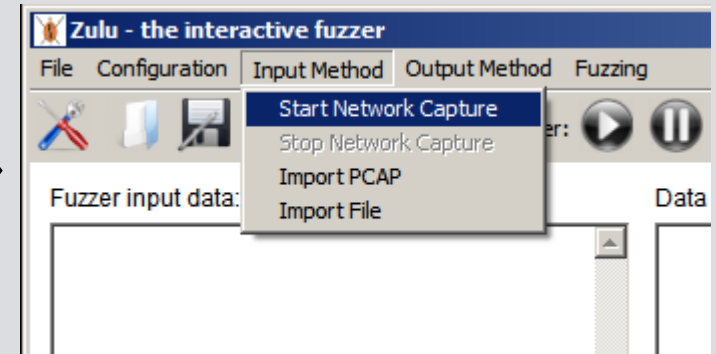
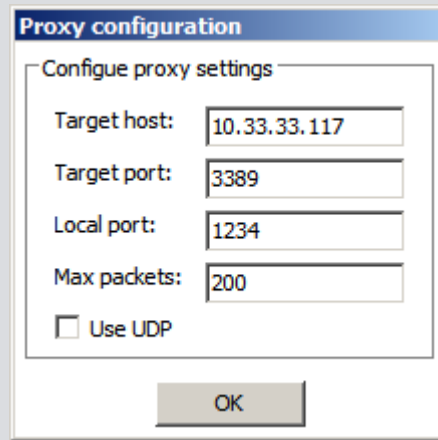
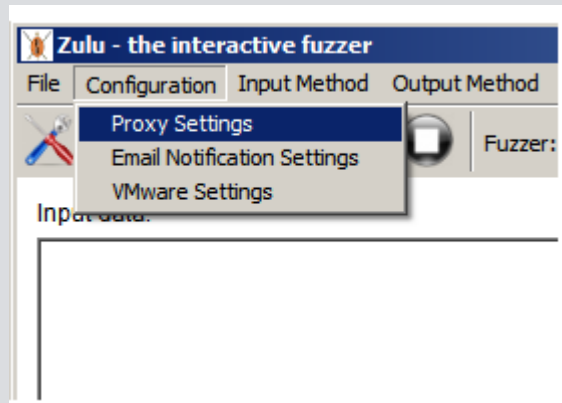
# File structure

- **/bin** - Zulu binaries and custom.py (ZuluScript Python)
- **/crashfiles** - When file fuzzing, files that have caused the target to crash
- **/fuzzdb** - the fuzzer testcase files
- **/images** - images used by the GUI
- **/logs** - log files
- **/pcap** - when Wireshark integration is enabled, auto-generated PCAP files
- **/PoC** - when a crash occurs a PoC is auto-generated
- **/sessions** - configuration options and captured packets
- **/tempfiles** - when file fuzzing, temp manipulated files are stored here
- **/templates** - the template used to generate the PoC files is in here

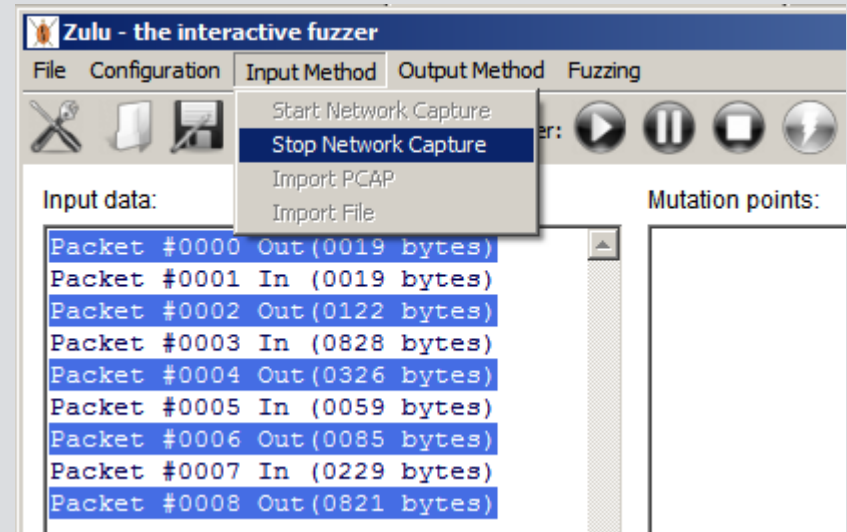


# Proxy-based network module

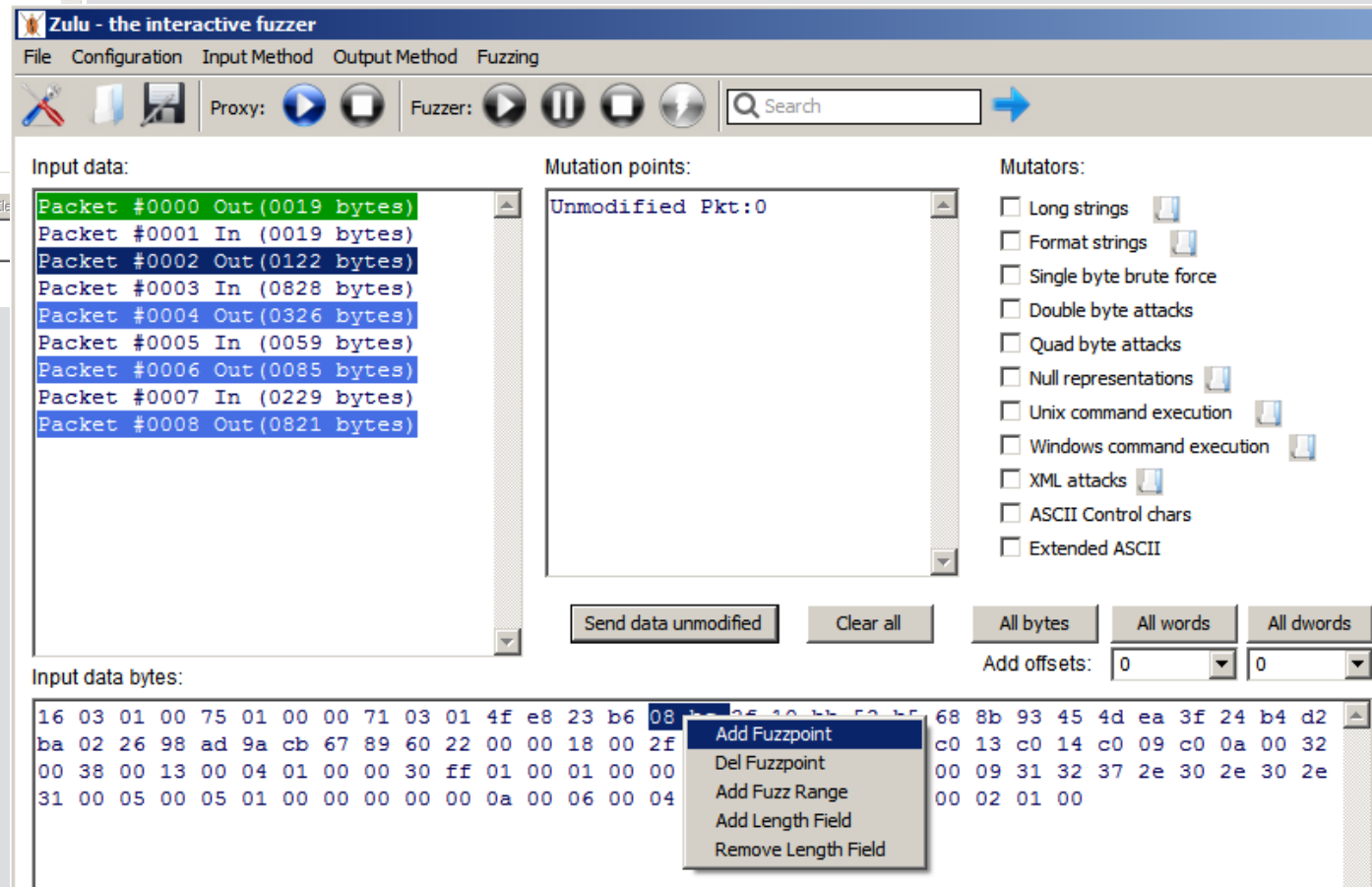
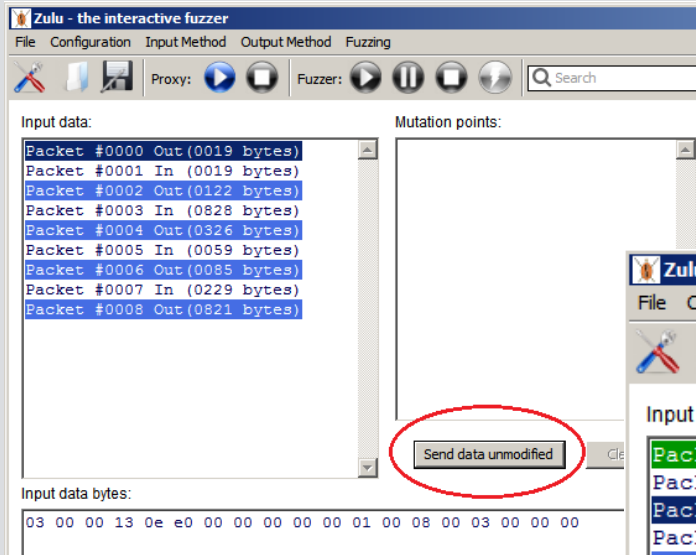
# Configure the proxy



# Use the standard network client










# Select some fuzz points



# Select mutators

Mutators:

<input checked="" type="checkbox"/> Long strings 	<input type="checkbox"/> User defined 
<input checked="" type="checkbox"/> Format strings 	<input type="checkbox"/> Bit sweep (byte)
<input type="checkbox"/> Single byte brute force	<input type="checkbox"/> Bit sweep (double byte)
<input type="checkbox"/> Double byte attacks	<input type="checkbox"/> Bit sweep (quad byte)
<input type="checkbox"/> Quad byte attacks	<input type="checkbox"/> Inverted bit sweep (byte)
<input type="checkbox"/> Null representations 	<input type="checkbox"/> Inverted bit sweep (double byte)
<input type="checkbox"/> Unix command execution 	<input type="checkbox"/> Inverted bit sweep (quad byte)
<input type="checkbox"/> Windows command execution 	
<input type="checkbox"/> XML attacks 	
<input type="checkbox"/> ASCII Control chars	
<input type="checkbox"/> Extended ASCII	

# Select output method

The screenshot shows the Zulu - the interactive fuzzer application. The main window has tabs for File, Configuration, Input Method, Output Method, and Fuzzing. The 'Output Method' tab is active, showing 'Network Fuzzer' and 'File Fuzzer' options. A search bar is present. The 'Input data' list shows packets from #0000 to #0008. The 'Mutation points' list shows 'Unmodified Pkt:0' and 'Fuzzpoint:15-16 Pkt:2', with the latter circled in red. The 'Mutators' list includes 'Long strings', 'Format strings', 'Single byte brute force', 'Double byte attacks', 'Quad byte attacks', 'Null representations', 'Unix command execution', 'Windows command execution', 'XML attacks', 'ASCII Control chars', and 'Extended ASCII'. A 'Send data unmodified' button is at the bottom. A 'Network Fuzzer configuration' dialog box is open, showing settings for Target host (10.33.33.117), Target port (3389), TCP Connect retries (2), Receive timeout (0.1), and Delay between fuzzcases (0). A red arrow points from the 'Fuzzpoint:15-16 Pkt:2' entry to the configuration dialog.

Input data:

- Packet #0000 Out (0019 bytes)
- Packet #0001 In (0019 bytes)
- Packet #0002 Out (0122 bytes)
- Packet #0003 In (0828 bytes)
- Packet #0004 Out (0326 bytes)
- Packet #0005 In (0059 bytes)
- Packet #0006 Out (0085 bytes)
- Packet #0007 In (0229 bytes)
- Packet #0008 Out (0821 bytes)

Mutation points:

- Unmodified Pkt:0
- Fuzzpoint:15-16 Pkt:2

Mutators:

- Long strings
- Format strings
- Single byte brute force
- Double byte attacks
- Quad byte attacks
- Null representations
- Unix command execution
- Windows command execution
- XML attacks
- ASCII Control chars
- Extended ASCII

Send data unmodified

Input data bytes:

```
16 03 01 00 75 01 00 00 71 03 01 4f e8 23 06 08 bc 3f 10
ba 02 26 98 ad 9a cb 67 89 60 22 00 00 18 00 2f 00 35 00
00 38 00 13 00 04 01 00 00 30 ff 01 00 01 00 00 00 0e
31 00 05 00 05 01 00 00 00 00 0a 00 06 00 04 00 17 00
```

Network Fuzzer configuration

Configure network fuzzer settings

Target host: 10.33.33.117

Target port: 3389

TCP Connect retries: 2

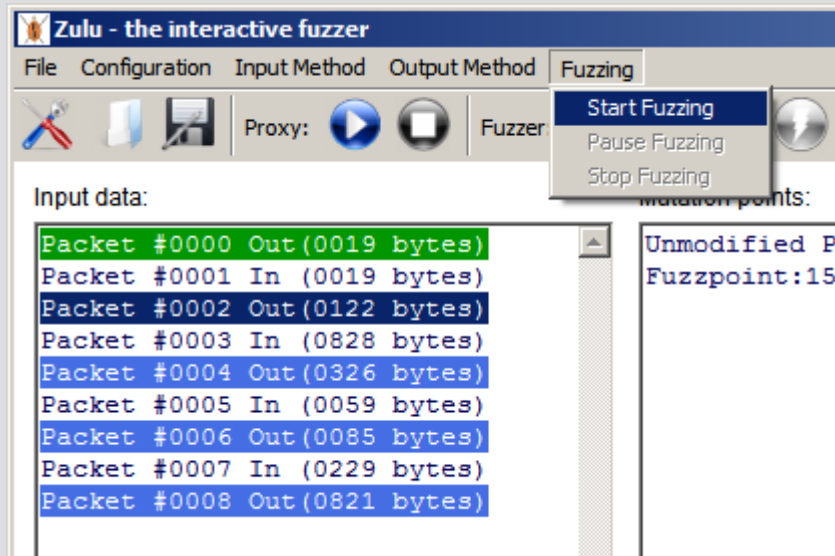
Receive timeout: 0.1

Delay between fuzzcases (seconds): 0

Use UDP

OK

# Start fuzzing



```
C:\Windows\system32\cmd.exe - zulu.py
-----
Fuzzpoint 0/0, Testcase 19/75 Test type: format-strings, Test #7
-----
packet: 0
send
'\x03\x00\x00\x13\x0e\xe0\x00\x00\x00\x00\x01\x00\x08\x00\x03\x00\x00\x00'
receive
'\x03\x00\x00\x13\x0e\xd0\x00\x00\x124\x00\x02\x01\x08\x00\x02\x00\x00\x00'
packet: 1
send
'\x16\x03\x01\x00u\x01\x00\x00q\x03\x010\xe8#\xb6%.1025d?\x10\xbbR\xb5h\x8b\x93
g\x89`"\x00\x00\x18\x00/\x005\x00\x05\x00\n\xc0\x13\xc0\x14\xc0\t\xc0\n\x002\x0
00\x01\x00\x00\x00\x00\xe\x00\x0c\x00\x00\t127.0.0.1\x00\x05\x00\x05\x01\x00\x
x00\x18\x00\x0b\x00\x02\x01\x00'
receive
-----
Fuzzpoint 0/0, Testcase 20/75 Test type: format-strings, Test #8
-----
Fuzzing: Connect error - attempt #1
-----
Fuzzing: Connect error - attempt #2
-----
Fuzzing: Connect error - check if target has crashed
-----
```

# Instrumentation and triage

**Zulu - the interactive fuzzer**

File Configuration Input Method Output Method Fuzzing

Proxy: [stop] [play] Fuzzer: [stop] [play] [lightning bolt] Search [input]

**Input data:**

- Packet #0000 Out (0019 bytes)
- Packet #0001 In (0019 bytes)
- Packet #0002 Out (0122 bytes)
- Packet #0003 In (0828 bytes)
- Packet #0004 Out (0326 bytes)
- Packet #0005 In (0059 bytes)
- Packet #0006 Out (0085 bytes)
- Packet #0007 In (0229 bytes)
- Packet #0008 Out (0821 bytes)

**Mutation points:**

Unmodified Pkt:0  
Fuzzpoint:15-16 Pkt:2

**Mutators:**

- Long strings
- Format strings
- Single byte brute force
- Double byte attacks
- Quad byte attacks
- Null representations
- Unix command execution
- Windows command execution
- XML attacks
- ASCII Control chars
- Extended ASCII
- User defined
- Bit sweep (byte)
- Bit sweep (double byte)
- Bit sweep (quad byte)
- Inverted bit sweep (byte)
- Inverted bit sweep (double byte)
- Inverted bit sweep (quad byte)
- Enable ZuluScript (see "/bin/custom.py")
- Enable Wireshark integration
- Enable VMware integration

**Input data bytes:**

```

16 03 01 00 75 01 00 00 71 03 01 4f e8 23 b6 08 bc 3f 10 bb 52 b5 68 8b 93 45 4d ea 3f 24 b4 d2
ba 02 26 98 ad 9a cb 67 89 60 22 00 00 18 00 2f 00 35 00 05 00 0a c0 13 c0 14 c0 09 c0 0a 00 32
00 38 00 13 00 04 01 00 00 30 ff 01 00 01 00 00 00 0e 00 0c 00 00 09 31 32 37 2e 30 2e 30 2e
31 00 05 00 05 01 00 00 00 00 0a 00 06 00 04 00 17 00 18 00 0b 00 02 01 00

```

**Status:**

Status: Fuzzing paused  
Status: Fuzzing paused  
Fuzzing: Press "Stop" to end this fuzzing session

Fuzzer selected: Network Fuzzer Selected packet = 2 Status: Fuzzing paused



Other inputs: PCAP files

# Wireshark captures

Intel(R) 82577LM Gigabit Network Connection

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
67	7.625915	10.33.33.104	88.151.219.102	TCP	44367 > 443
68	7.646893	88.151.219.102	10.33.33.104	SSL	Continuation
69	7.700724	DevoLo_d4:37:e0	Broadcast	HomePlug	Vendor Speci
70	7.721579	DevoLo_d4:37:e0	Broadcast	HomePlug	Network Stat
71	7.820034	Dell_2a:2c:98	Broadcast	ARP	who has 10.3
72	7.820262	QuantaCo_9a:a7:ae	Dell_2a:2c:98	ARP	10.33.33.117
73	7.820273	10.33.33.104	10.33.33.117	TCP	45427 > 3389
74	7.820814	QuantaCo_9a:a7:ae	Broadcast		who has 10.3
75	7.820852	Dell_2a:2c:98	QuantaCo_9a		10.33.33.104
76	7.820966	10.33.33.117	10.33.33.10		3389 > 45427
77	7.820994	10.33.33.104	10.33.33.11		45427 > 3389
78	7.821358	10.33.33.104	10.33.33.11		Connection R
79	7.833164	10.33.33.117	10.33.33.10		3389 > 45427
80	7.834035	10.33.33.117	10.33.33.10		Connection C
81	7.849880	10.33.33.104	88.151.219.		44367 > 443
82	7.870759	fe80::9504:d1e5:7e31:dced	ff02::c	SCMP	M-SEARCH * H
83	7.874661	88.151.219.102	10.33.33.10		Continuation
84	8.029959	10.33.33.104	10.33.33.11		45427 > 3389

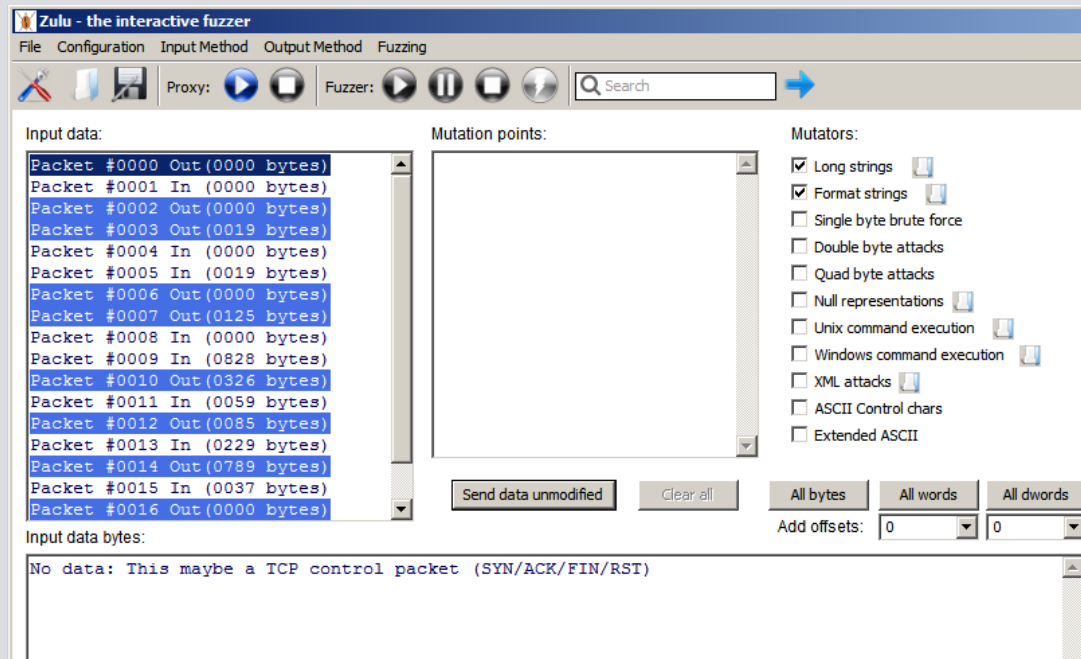
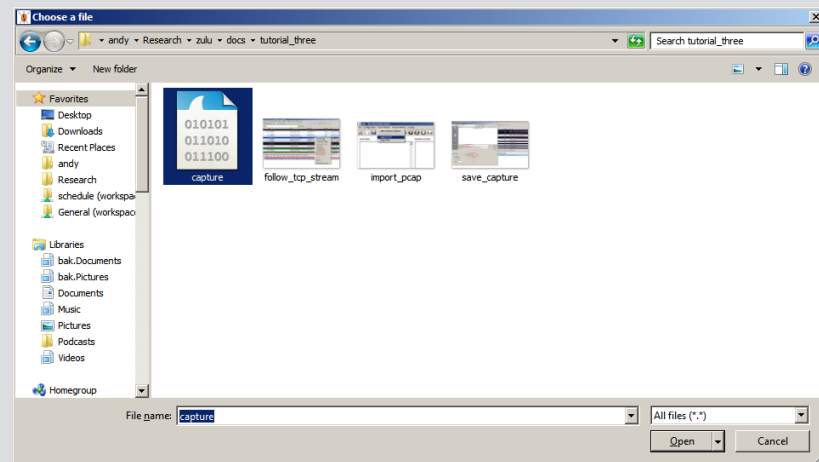
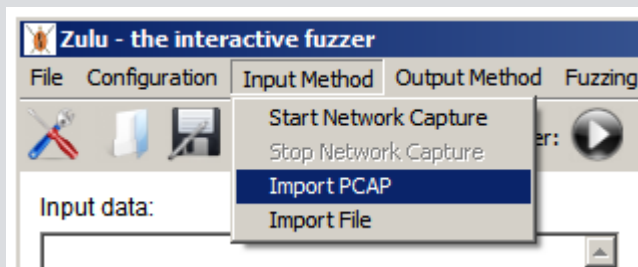
Context menu for packet 73:

- Mark Packet (toggle)
- Ignore Packet (toggle)
- Set Time Reference (toggle)
- Manually Resolve Address
- Apply as Filter
- Prepare a Filter
- Conversation Filter
- Colorize Conversation
- Follow TCP Stream
- Follow UDP Stream
- Follow SSL Stream
- Copy
- Decode As...
- Print...
- Show Packet in New Window

Packet details for packet 73:

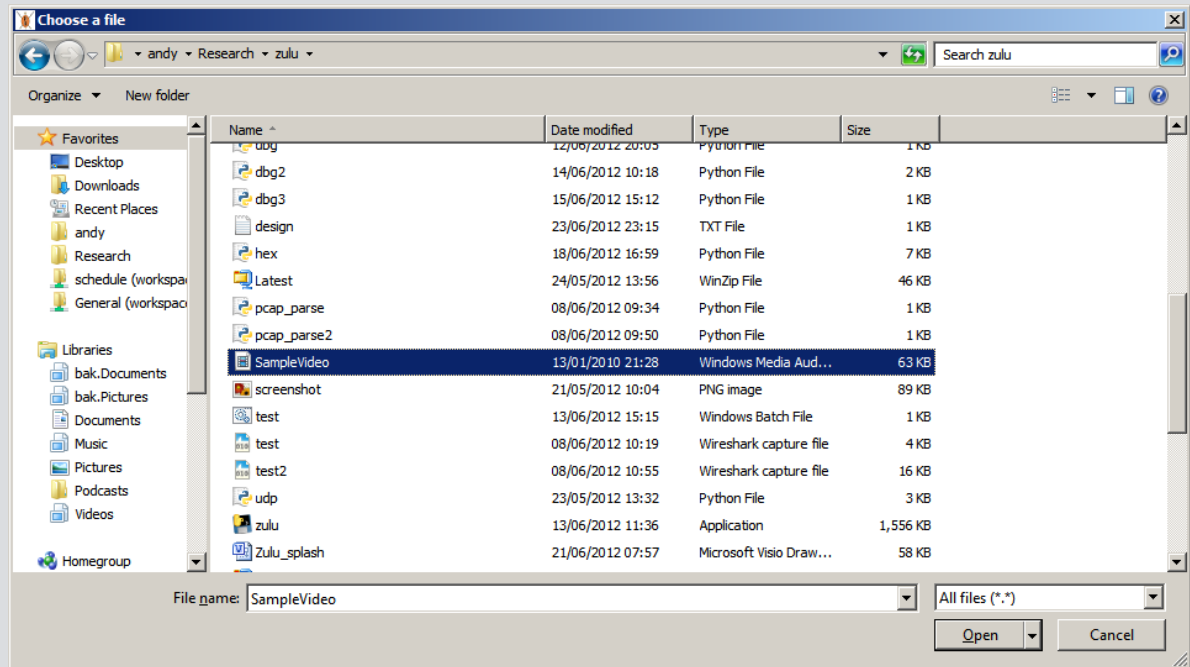
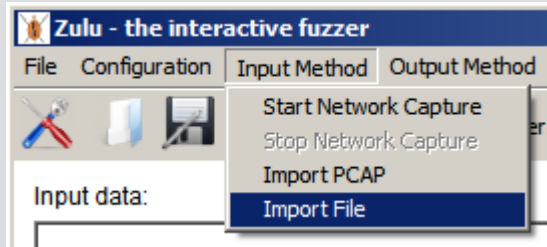
- Frame 73: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
- Ethernet II, Src: Dell\_2a:2c:98 (5c:26:0a:2a:2c:98), Dst: QuantaCo\_9a:a7:ae
- Internet Protocol Version 4, Src: 10.33.33.104 (10.33.33.104), Dst: 10.33.33.117
- Transmission Control Protocol, Src Port: 45427 (45427), Dst Port: 3389 (3389)

# Importing a PCAP

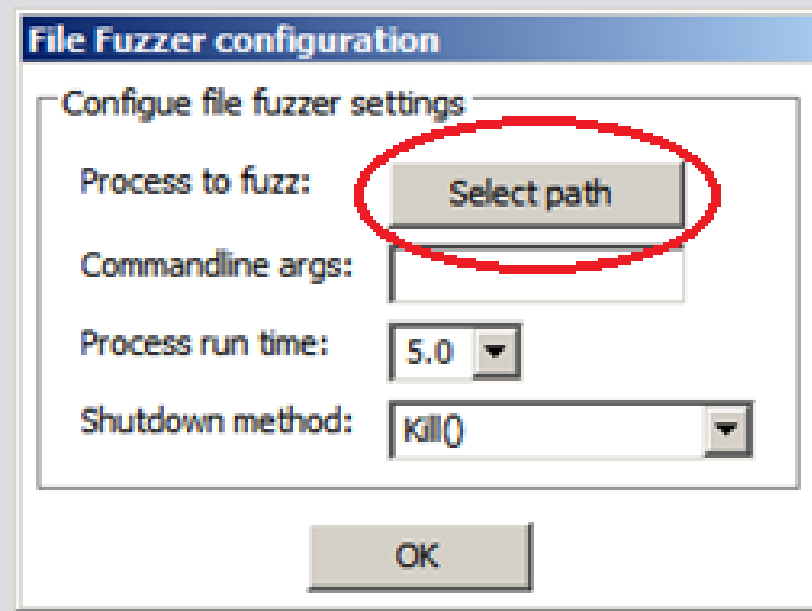
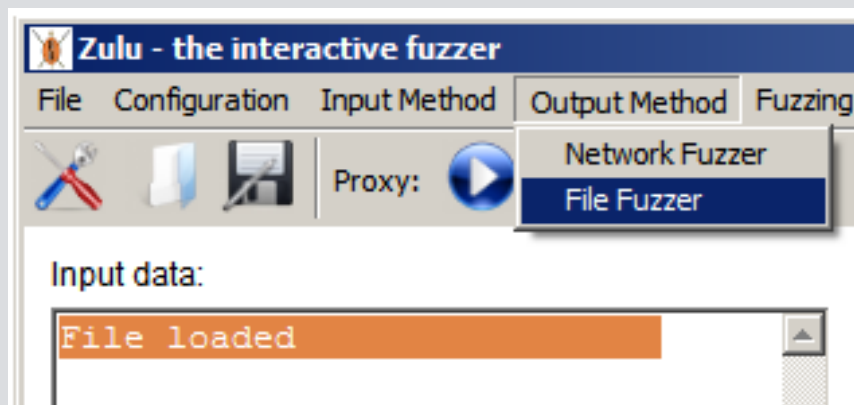


# File module

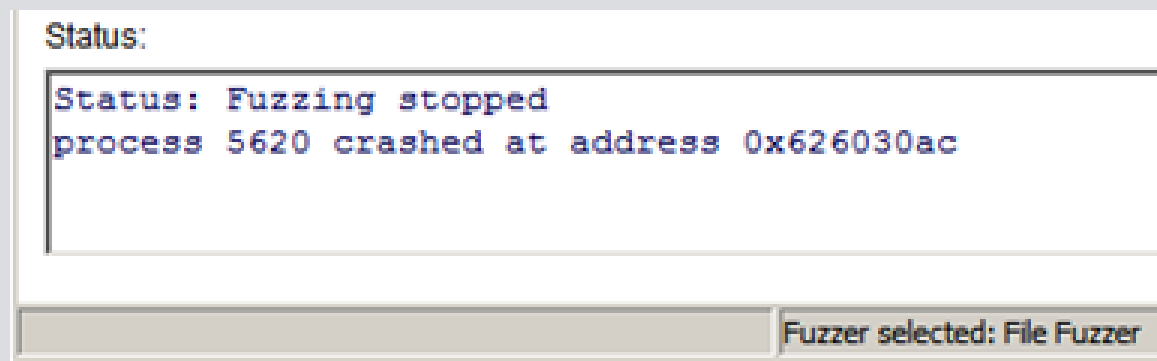
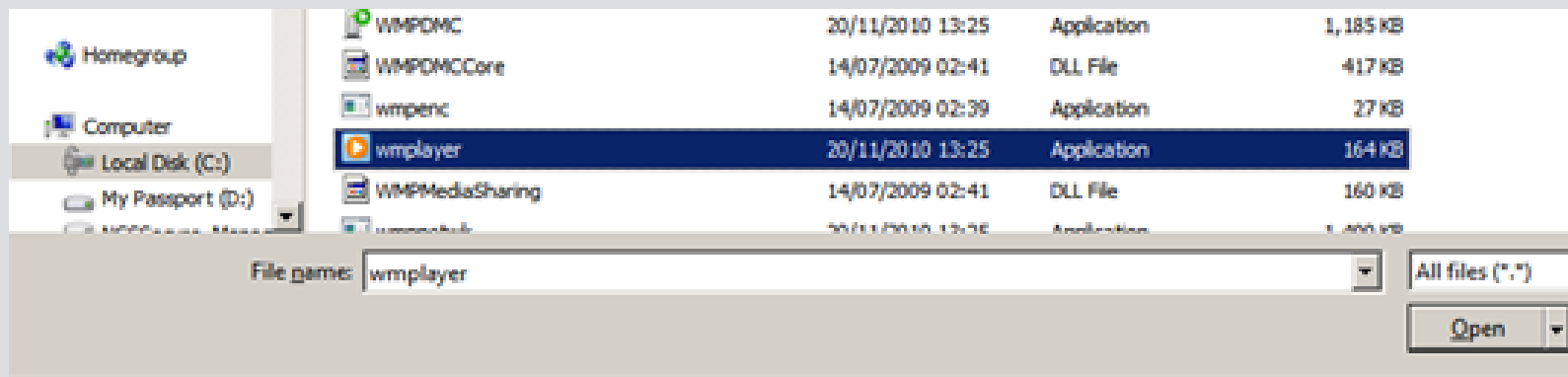
# Select input file



# Select file fuzzer + fuzz process



# Fuzz process + debugging



# USB module



# Graphic USB

GraphicUSB - [keyboard.mqu]

File Edit View Operations Window Help

Vbus: 5.060V 63uA

#	Time	LS	Control Transfer	Addr	Endp	Data	Status
#41...81	5.318,184 s	←	Get Device Descriptor	0x00	0x0	12 01 10 01 00 00 00 08...	OK
#41...81	5.318,184 s	→	SetAddress (0x01)	0x00	0x0		OK
#41...81	5.318,184 s	←	Get Configuration Descriptor	0x01	0x0	09 02 22 00 01 01 00 A0...	OK
#41...81	5.318,184 s	←	Get String Descriptor 0	0x01	0x0	04 03 09 04	OK
#41...81	5.318,184 s	←	Get String Descriptor 2	0x01	0x0	24 03 44 00 45 00 4C 00...	OK
#41...81	5.318,184 s	←	Get Device Descriptor	0x01	0x0	12 01 10 01 00 00 00 08...	OK
#41...81	5.318,184 s	←	Get Configuration Descriptor	0x01	0x0	09 02 22 00 01 01 00 A0...	OK
#41...81	5.318,184 s	→	Set Configuration (0x01)	0x01	0x0		OK
#41...81	5.318,184 s	→	Set Idle (HID) Indefinite, All	0x01	0x0		OK
#41...81	5.318,184 s	←	Get HID Report Descriptor	0x01	0x0	05 01 09 06 A1 01 05 07...	OK

=== End of Capture ===

### Control Transfer

#### Get Device Descriptor

A device descriptor describes general information about a USB device. It includes information that applies globally to the device and all of the device's configurations. A USB device has only one device descriptor.

Field	Value	Meaning
bLength	18	Valid Length
bDescriptorType	1	DEVICE
bcdUSB	0x0110	Spec Version
bDeviceClass	0x00	Class Information in Interface Descriptor
bDeviceSubClass	0x00	Class Information in Interface Descriptor
bDeviceProtocol	0x00	Class Information in Interface Descriptor
bMaxPacketSize0	8	Max EP0 Packet Size
idVendor	0x413C	Dell Inc.
idProduct	0x2005	Unknown
bcdDevice	0x0104	Device Release No
iManufacturer	1	Index to Manufacturer String
iProduct	2	Index to Product String
iSerialNumber	0	Index to Serial Number

#### Data Content

```

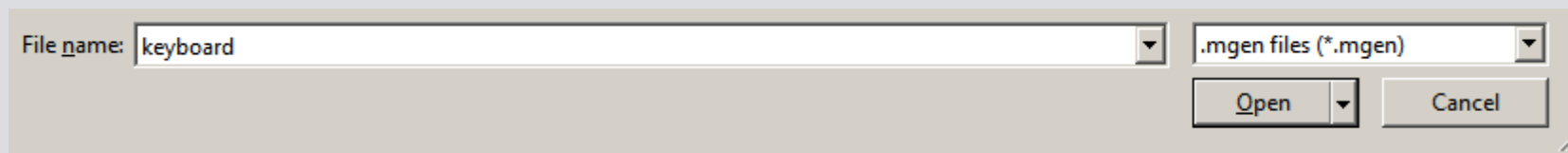
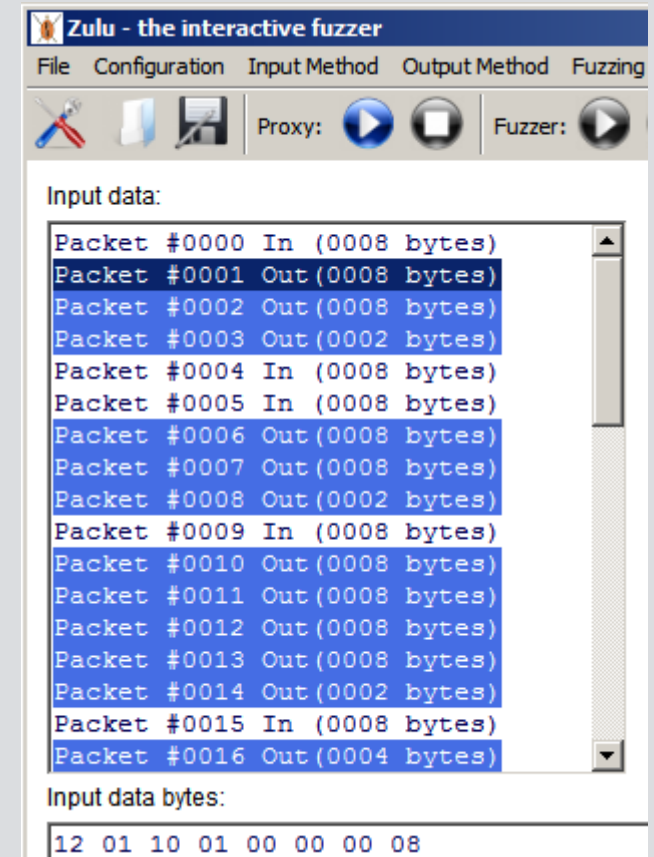
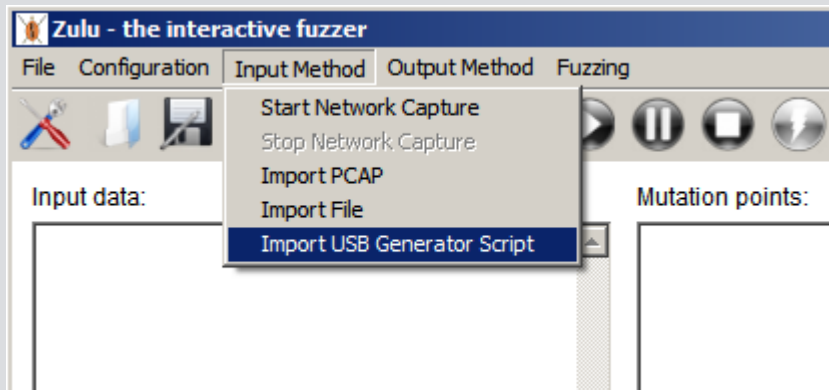
00000000: 12 01 10 01 00 00 00 08 3C 41 05 20 04  .....<A. .
0000000D: 01 01 02 00 01  .....

```

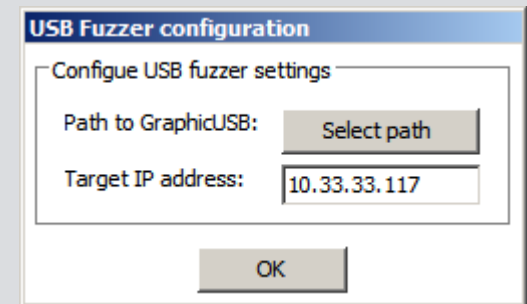
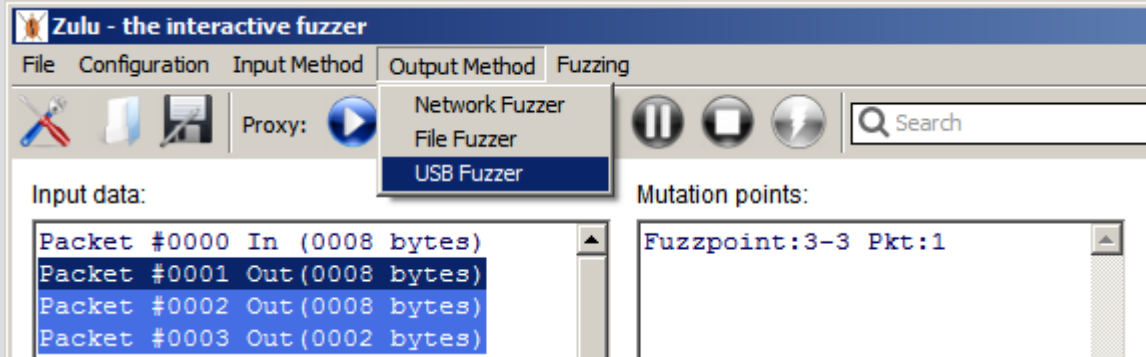
For Help, press F1

186 events

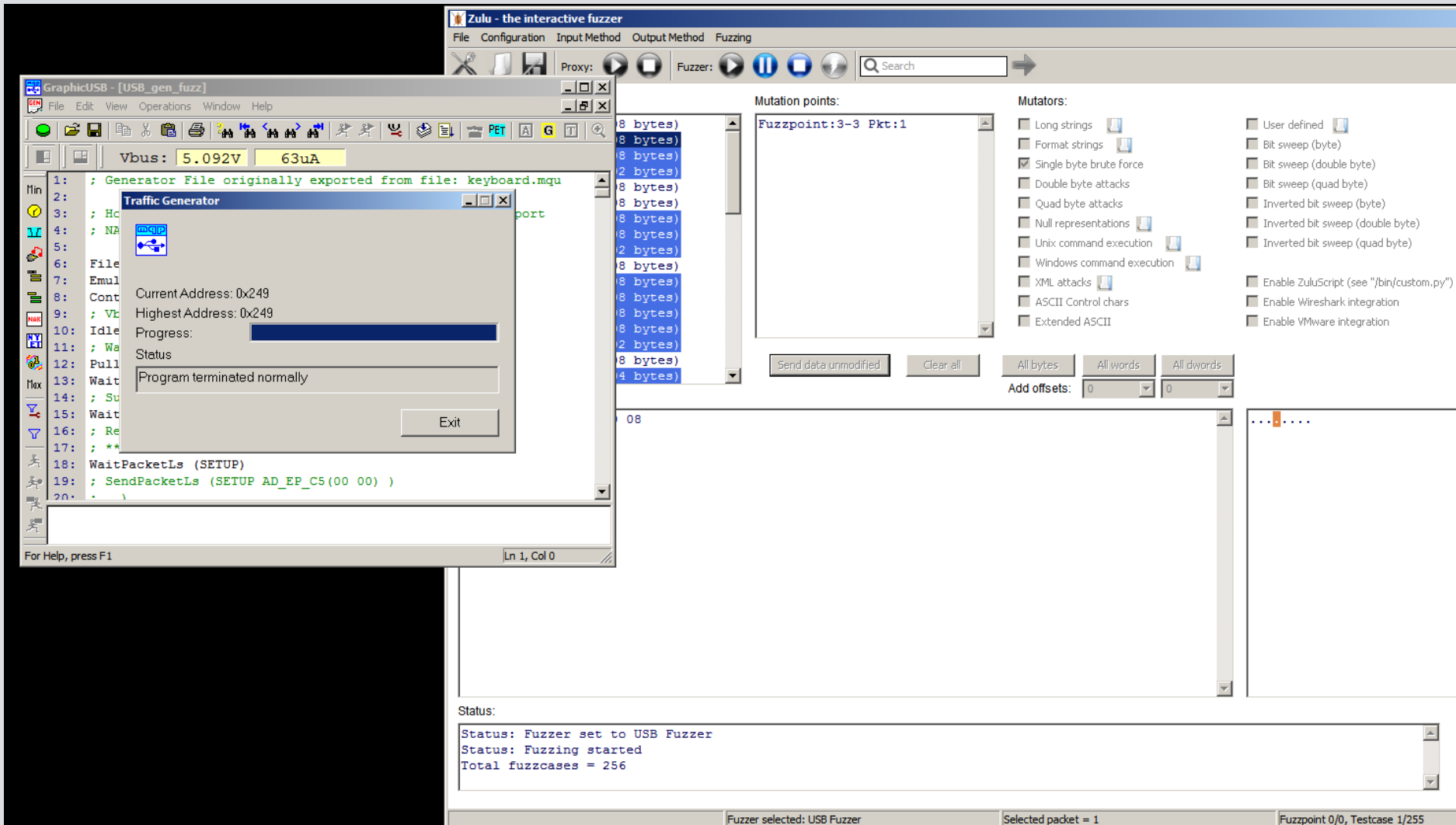
# Import generator script



# Select USB fuzzer

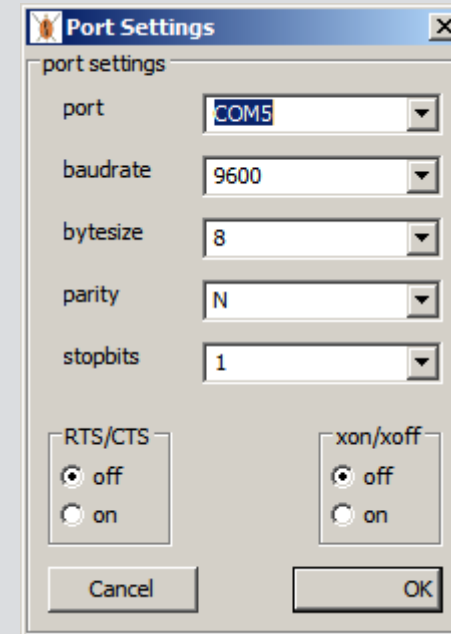
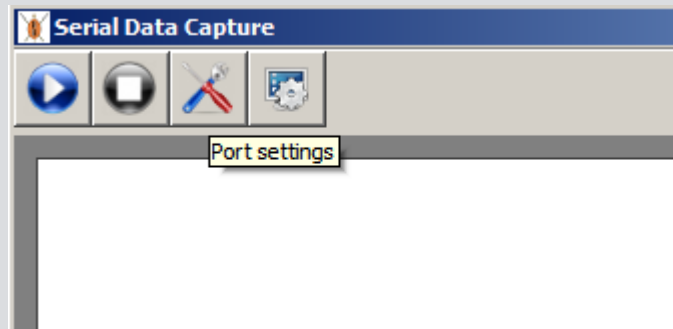
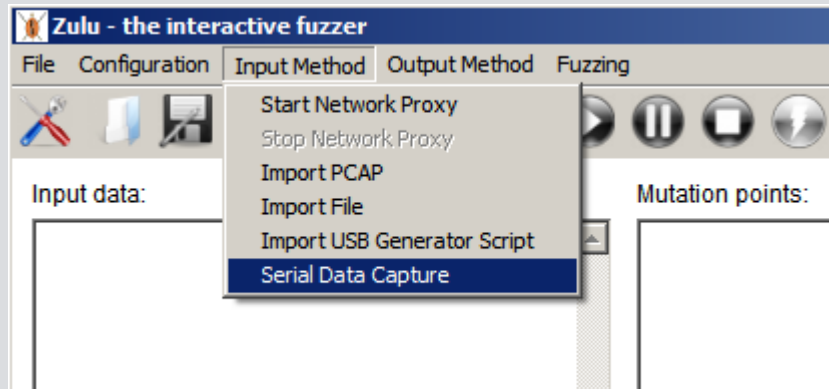


# Fuzzer running

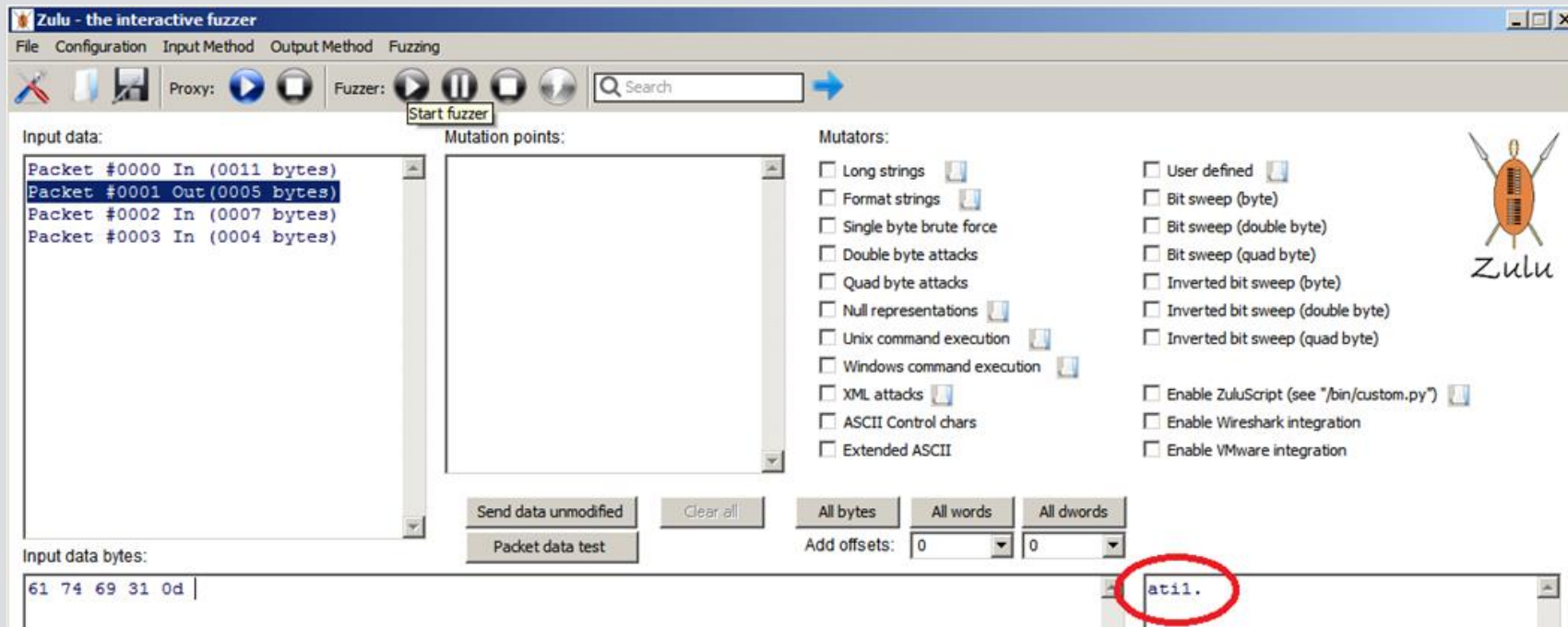
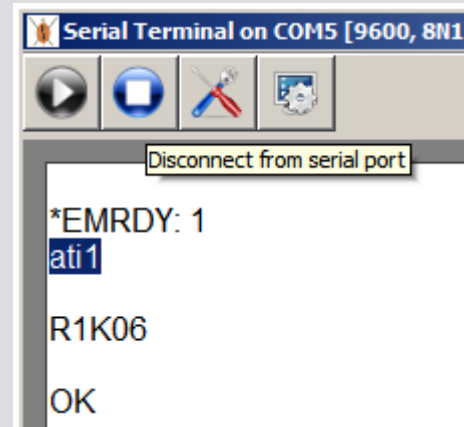
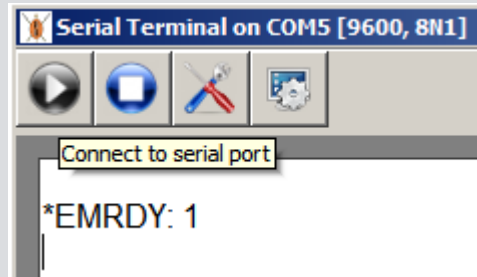


# Serial module

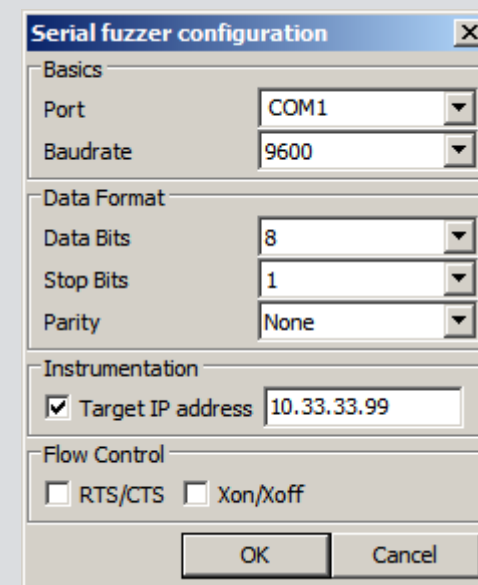
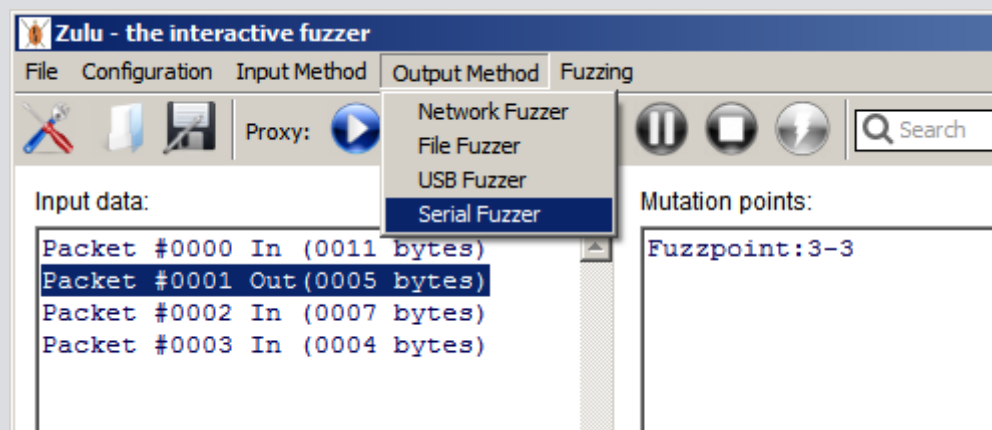
# Serial settings



# Serial data capture




# Serial fuzzing





# Wireshark integration

# Point to Wireshark binary

Enable ZuluScript (see "/bin/custom.py") 

Enable Wireshark integration

Enable VMware integration



File Name	Date Modified	Type	Size
editcap	27/03/2012 18:12	Application	74 KB
mergecap	27/03/2012 18:12	Application	31 KB
rawshark	27/03/2012 18:12	Application	98 KB
text2pcap	27/03/2012 18:12	Application	40 KB

File name:  .exe files (\*.exe)

# Auto-load Wireshark

The screenshot shows the Wireshark interface with a packet capture of an RDP connection request. The packet list pane shows 10 packets, with the first packet being a Connection Request (0xe0) from 127.0.0.1 to 10.33.33.117. The packet details pane shows the structure of the TPKT (Transmission Protocol Key) and ITU-T Rec X.224 (RDP) fields. The hex dump pane shows the raw bytes of the packet.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	127.0.0.1	10.33.33.117	X.224	Connection Request (0xe0)
2	0.000100	10.33.33.117	10.33.33.117	X.224	Connection Confirm (0xd0)
3	0.000200	127.0.0.1	10.33.33.117	TPKT	Continuation
4	0.000300	10.33.33.117	10.33.33.117	TPKT	Continuation
5	0.000400	127.0.0.1	10.33.33.117	TPKT	Continuation
6	0.000500	10.33.33.117	10.33.33.117	TPKT	Continuation
7	0.000600	127.0.0.1	10.33.33.117	TPKT	Continuation
8	0.000700	10.33.33.117	10.33.33.117	TPKT	Continuation
9	0.000800	127.0.0.1	10.33.33.117	TPKT	Continuation
10	0.000900	10.33.33.117	10.33.33.117	TPKT	Continuation

Frame 1: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)  
 Ethernet II, Src: Dell\_2a:2c:98 (5c:26:0a:2a:2c:98), Dst: Vmware\_28:d0:d7 (00:0c:29:28:d0:d7)  
 Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 10.33.33.117 (10.33.33.117)  
 Transmission Control Protocol, Src Port: 45764 (45764), Dst Port: 3389 (3389), Seq: 342342, Ack: 768678, Len: 19  
 TPKT, Version: 3, Length: 19  
 Version: 3  
 Reserved: 0  
 Length: 19  
 ITU-T Rec X.224  
 Length: 14  
 1110 .... = Code: Connection Request (0x0e)  
 SRC-REF: 0x0000  
 0000 .... = Class: Class 0 (0x00)  
 RDP Routing Token: \001

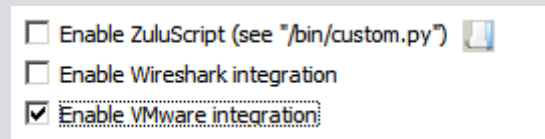
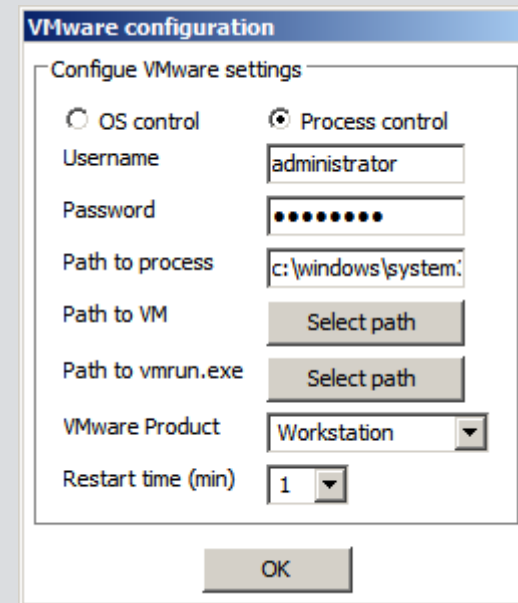
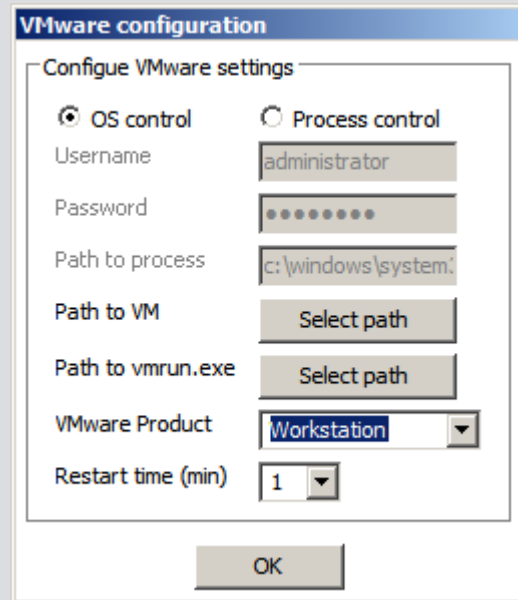
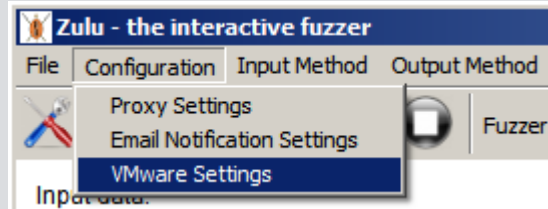
```

0000 00 0c 29 28 d0 d7 5c 26 0a 2a 2c 98 08 00 45 00  ..)(..\&.*,...E.
0010 00 3b ca ce 40 00 80 06 00 00 7f 00 00 01 0a 21  :;..@... ..!
0020 21 75 b2 c4 0d 3d 00 05 39 46 00 0b ba a6 50 18  !u...=. 9F...P.
0030 40 29 00 00 00 00 03 00 00 13 0e e0 00 00 00 00  @).... ..
0040 00 01 00 08 00 03 00 00 00  .....
```

TPKT - ISO on TCP - RFC1006 (tpkt), 4 bytes | Packets: 10 Displayed: 10 Marked: 0 Load time: 0:00.000 | Profile: Default

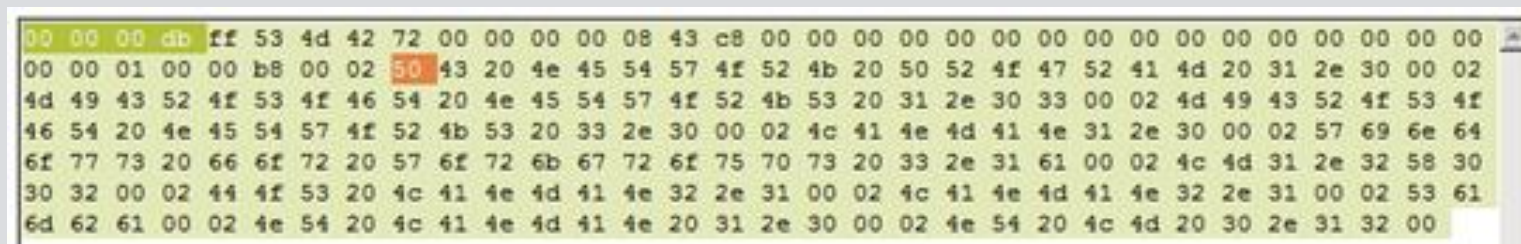
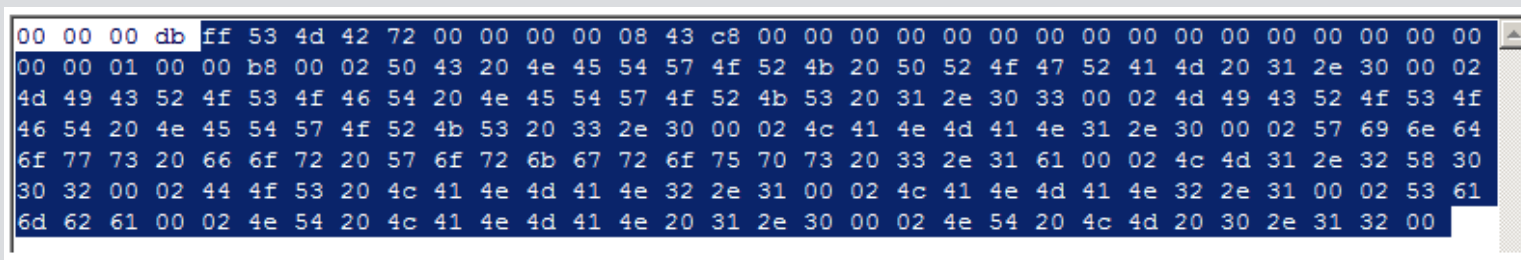
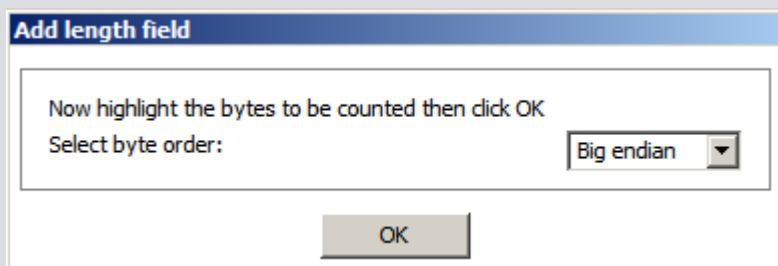
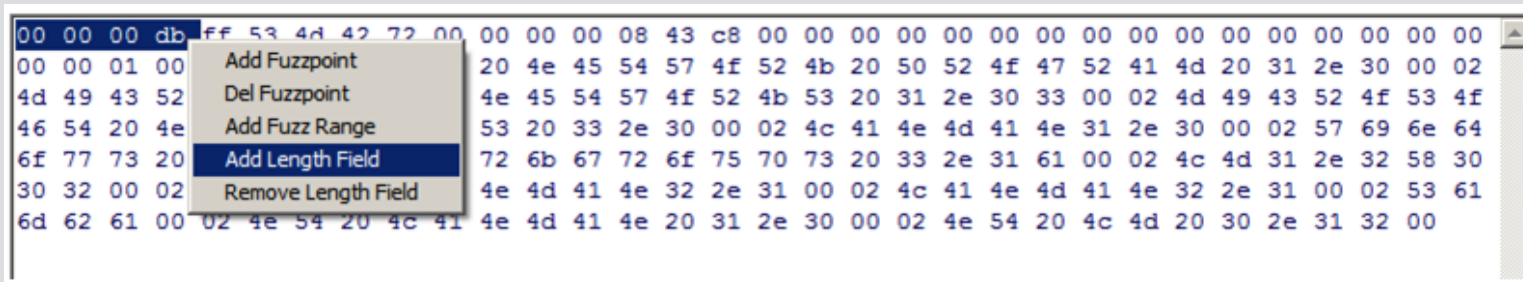
# VMware integration

# Select file fuzzer + fuzz process



# GUI-power

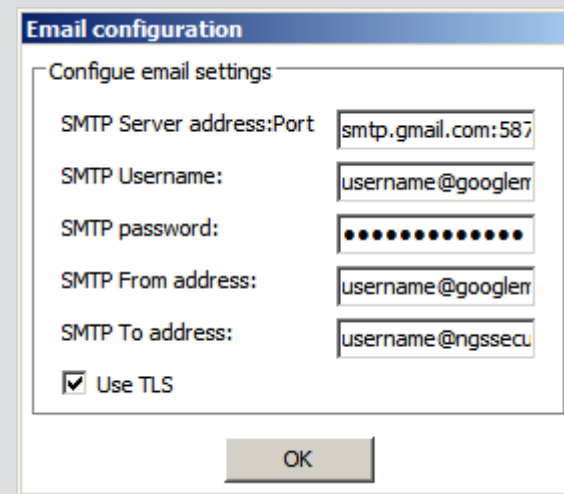
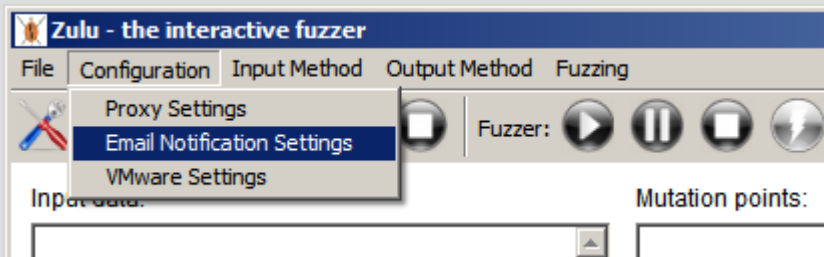
# Adding a length field



No need to watch! Email alerts




# Select email settings



# Advanced features - ZuluScript

# Using ZuluScript

- How do you modify a packet after the mutator but before being processed by the target?
- The answer is by using ZuluScript
- Python script stored in a special file (/bin/custom.py)
- Includes a sample ***UpdateContentLengthField()*** function

- Enable ZuluScript (see "/bin/custom.py") 
- Enable Wireshark integration
- Enable VMware integration

# Access to data

- ***self.packets\_selected\_to\_send*** = list of packets selected to send [[packet number, data],[packet number, data]...]
- ***self.all\_packets\_captured*** = list of all packets captured [[[source IP,source port],data], [[source IP,source port],data]...]
- ***self.modified\_data*** = list of all the data in the current packet (after any modification with fuzzpoint data) [byte1, byte2, byte3...]
- ***self.current\_packet\_number*** = the number of the current packet being processed (packet 0 is the first packet)

# Bugs that Zulu has found

- Samba 'AndX' request remote heap overflow (CVE-2012-0870)
- Oracle 11g TNS listener remote null pointer dereference
- Apple OS X USB Hub Descriptor bNbrPorts Field Handling Memory Corruption
- ...and many others that haven't been fixed yet

# Zulu is available on Github

Zulu can be downloaded today at:

<https://github.com/nccgroup/zulu>



## Questions?

**Andy Davis, Research Director NCC Group**  
andy.davis 'at' nccgroup 'dot' com

