nccgroup
freedom from doubt

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

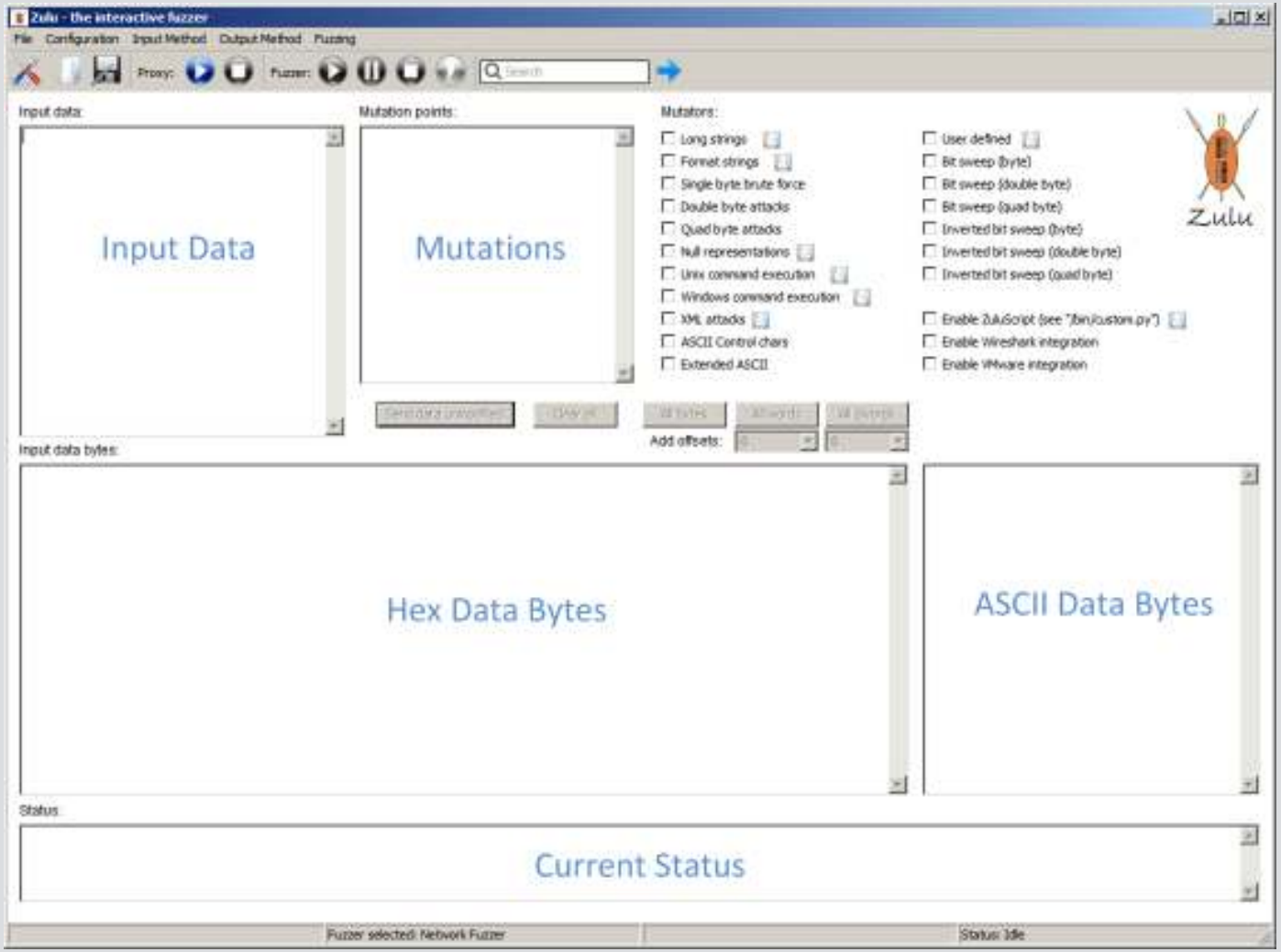
What is Zulu?

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

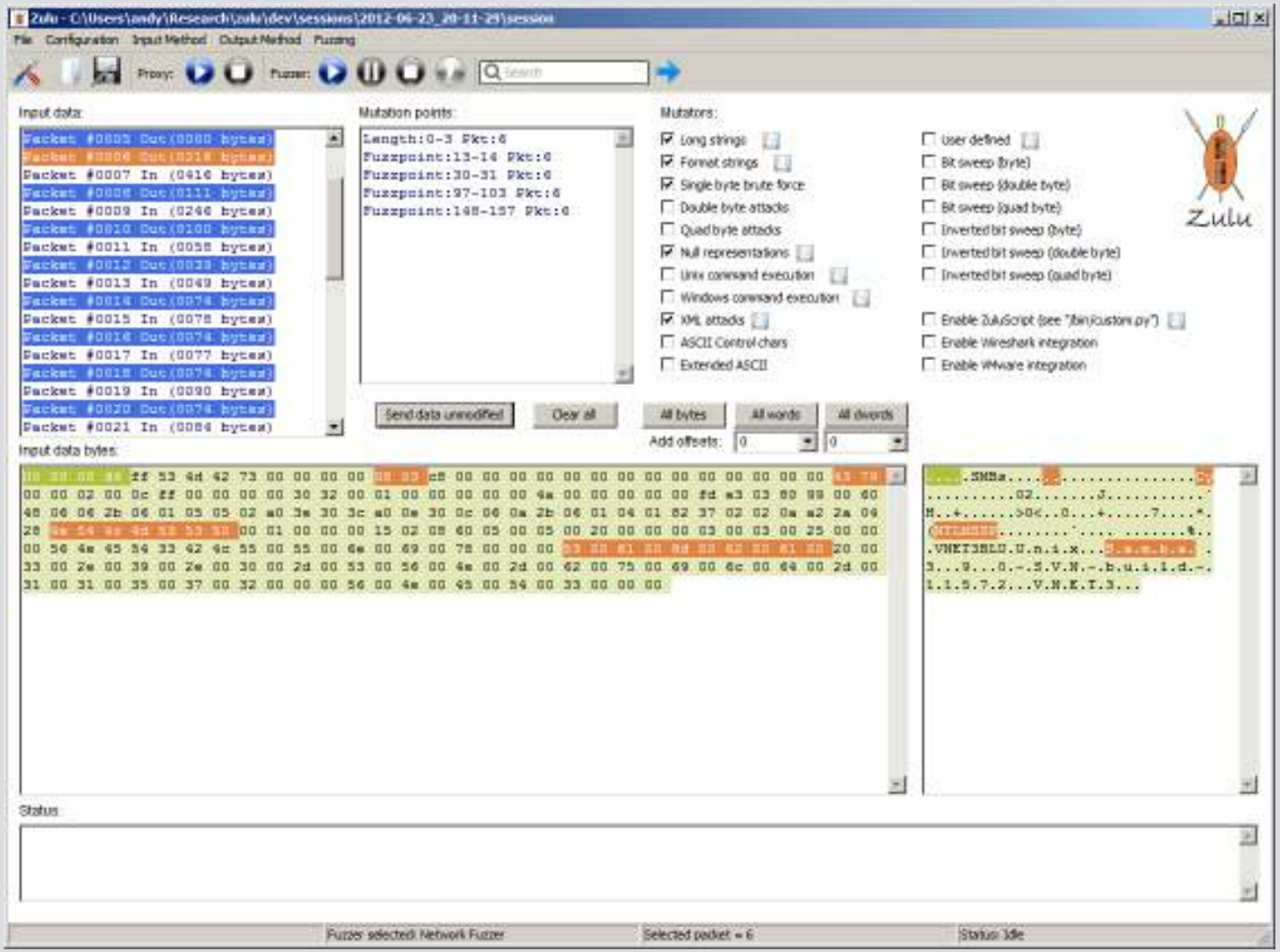
Motivations behind the tool

- 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



Zulu basics – the console

C:\Windows\system32\cmd.exe - zulu.py

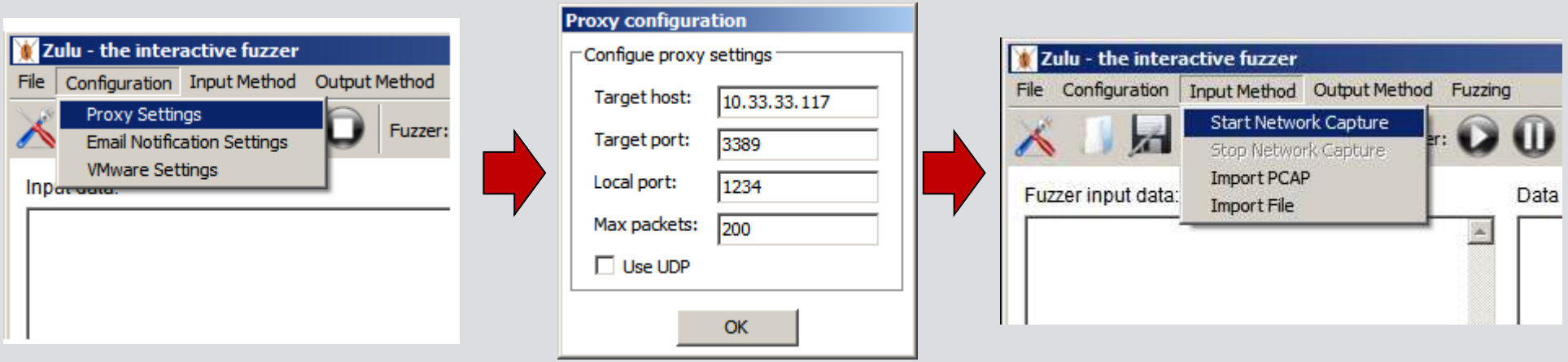
```
------------------------------------------------------
New fuzzing session
------------------------------------------------------

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\x0
0 \x00s\x00e\x00t\x00 \x00t\x00r\x00a\x00n\x00s\x00a\x00c\x00t\x00i\x00o\x00n\x00 \x00i\x00s\x00o\x00l\x00a\x00t\x00i\x0
0o\x00n\x00 \x00l\x00e\x00v\x00e\x00l\x00 \x00 \x00r\x00e\x00a\x00d\x00 \x00c\x00o\x00m\x00m\x00i\x00t\x00t\x00e\x00d\x0
0 \x00 \x00s\x00e\x00t\x00 \x00i\x00m\x00p\x00l\x00i\x00c\x00i\x00t\x00_\x00t\x00r\x00a\x00n\x00s\x00a\x00c\x00t\x00i\x0
0o\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\x
ff\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\x00
a\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\x00l\x00e\x00c\x00t\x00 \x00*\x00 \x00f\x00r\x00o\x00m\x00 \x00t\x00e\x00s\x00t\x00_\x00t\x00
a\x00b\x00l\x00e\x00_\x001\x00 \x00w\x00h\x00e\x00r\x00e\x00 \x00n\x00a\x00m\x00e\x00 \x00=\x00 \x00@\x00P\x000\x00 \x00
a\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\xe7@\x1f\t\x04\xd0\x004\x06\x00z\x00z\x00z\x00\x00\x00&\x04\x04\x02\x00\x00\x00'

receive
packet: 2

send
receive
```

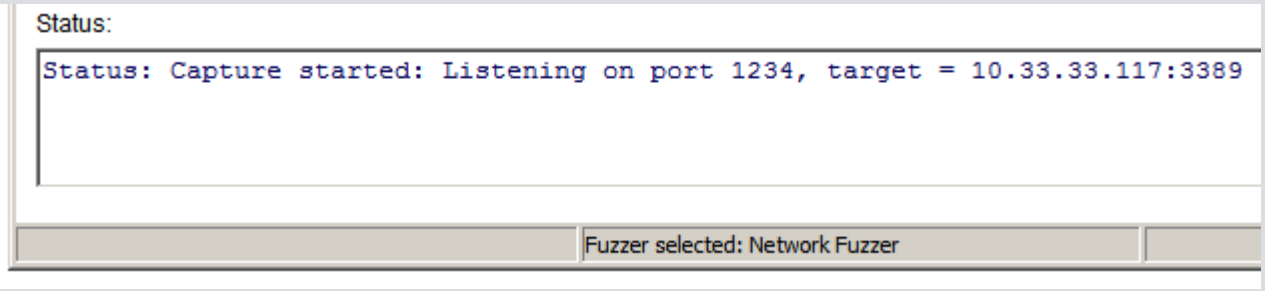
File structure

- /**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

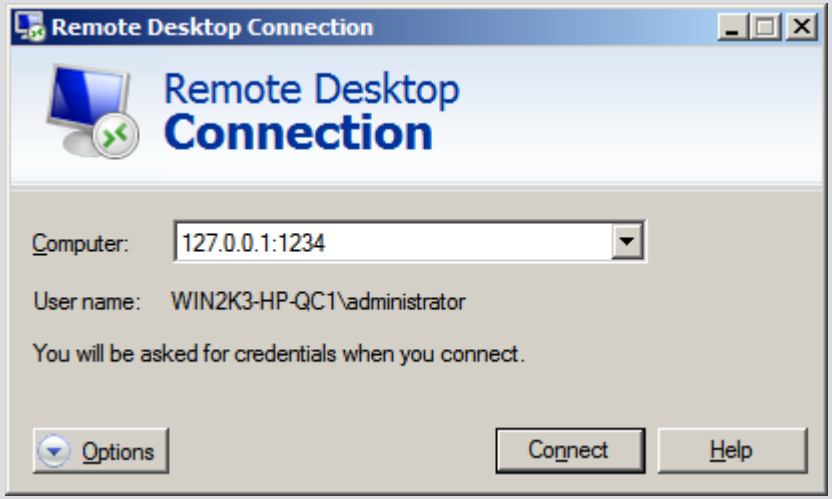
Proxy-based network module

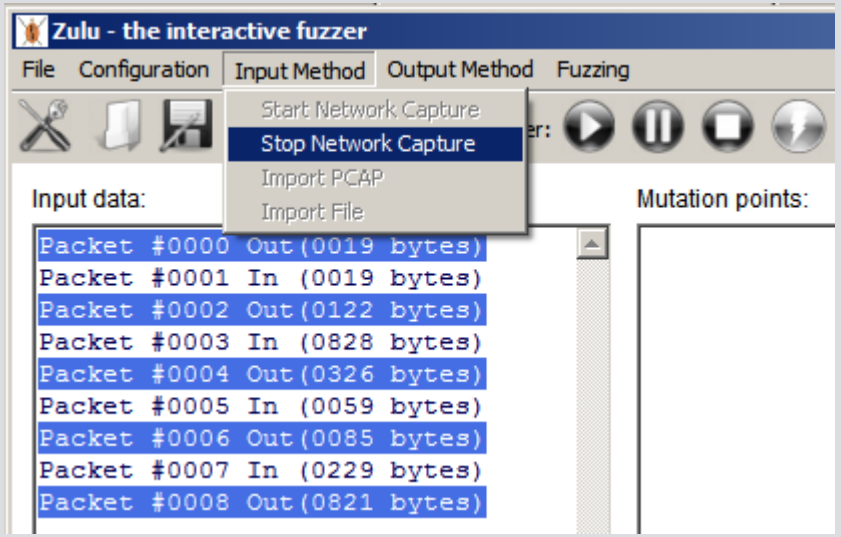
Configure the proxy



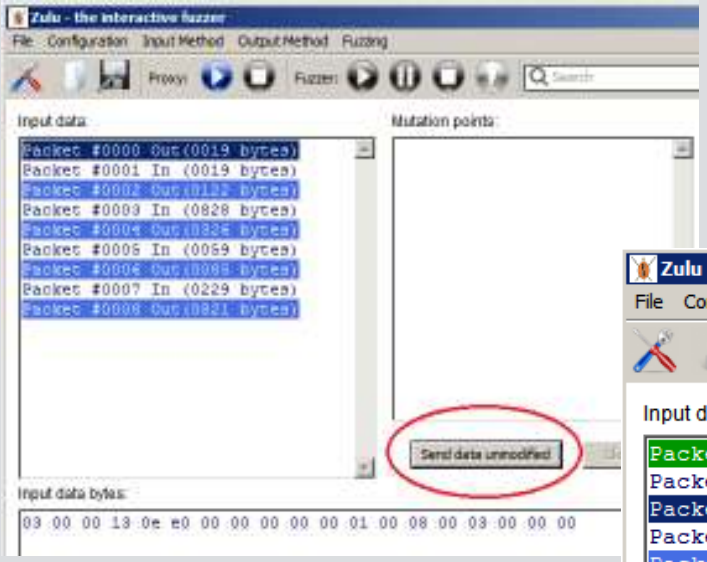


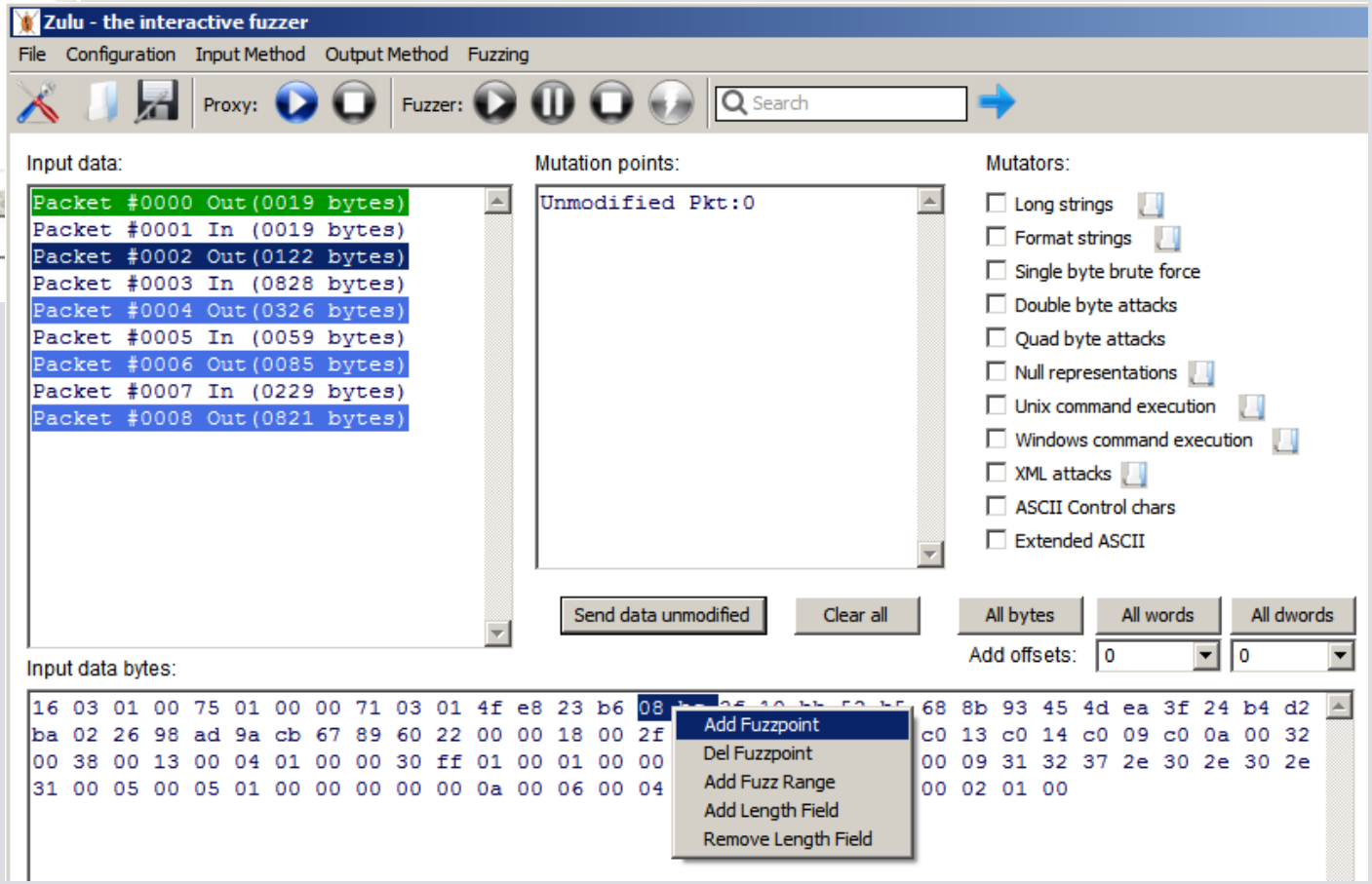
Use the standard network client





Select some fuzz points



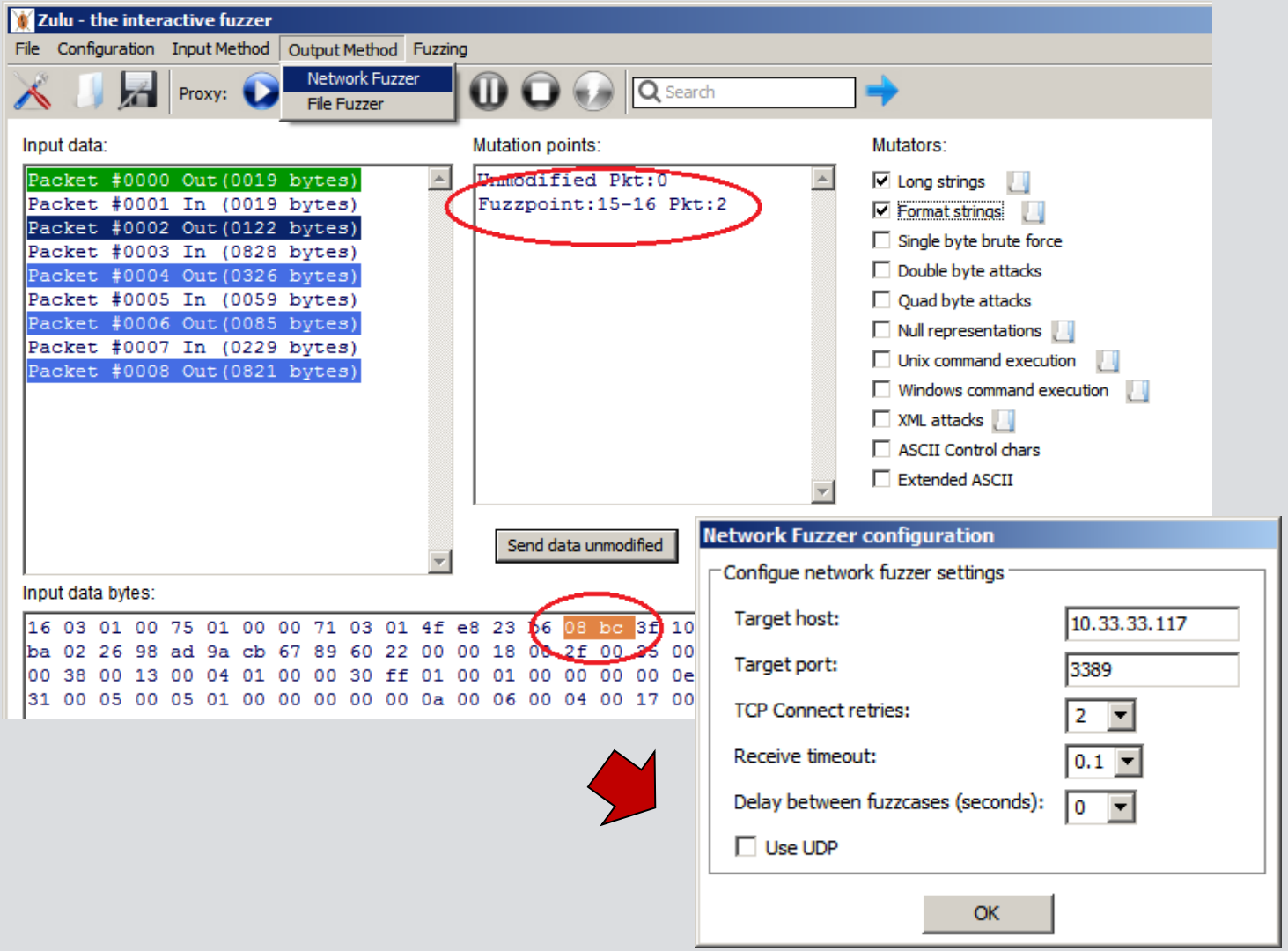


Select mutators

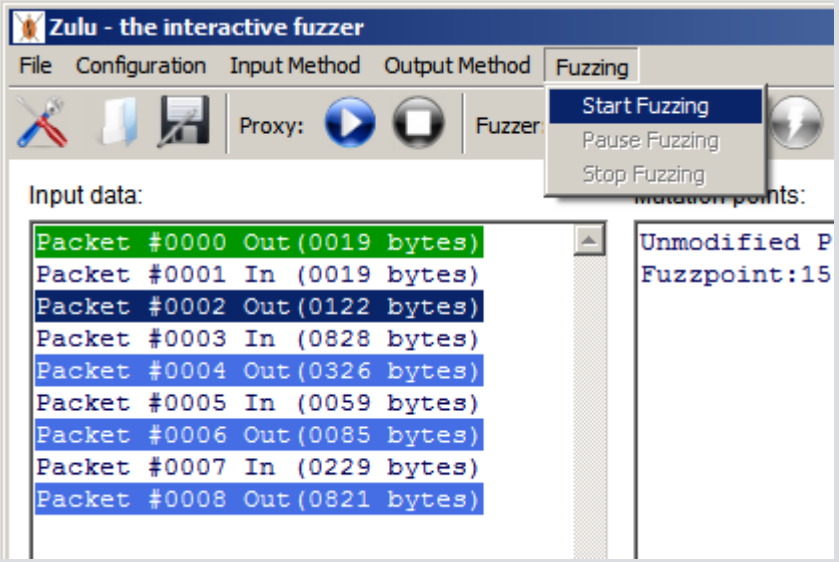
Mutators:

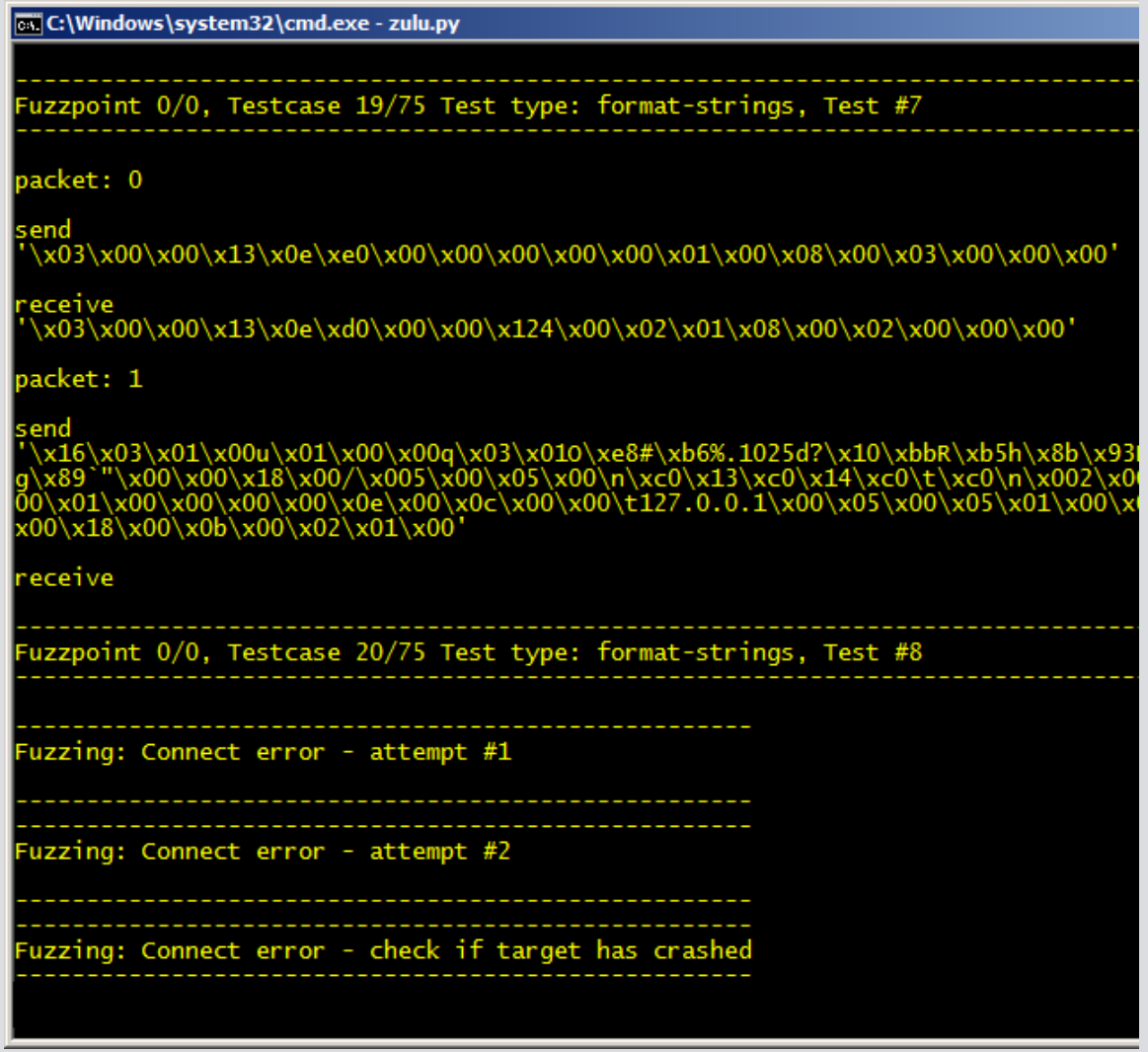
- [x] Long strings
- [x] Format strings
- [ ] Single byte brute force
- [ ] Double byte attacks
- [ ] Quad byte attacks
- [ ] Null representations
- [ ] Unix command execution
- [ ] Windows command execution
- [ ] XML attacks
- [ ] ASCII Control chars
- [ ] Extended ASCII
- [ ] User defined
- [ ] Bit sweep (byte)
- [ ] Bit sweep (double byte)
- [ ] Bit sweep (quad byte)
- [ ] Inverted bit sweep (byte)
- [ ] Inverted bit sweep (double byte)
- [ ] Inverted bit sweep (quad byte)

Select output method

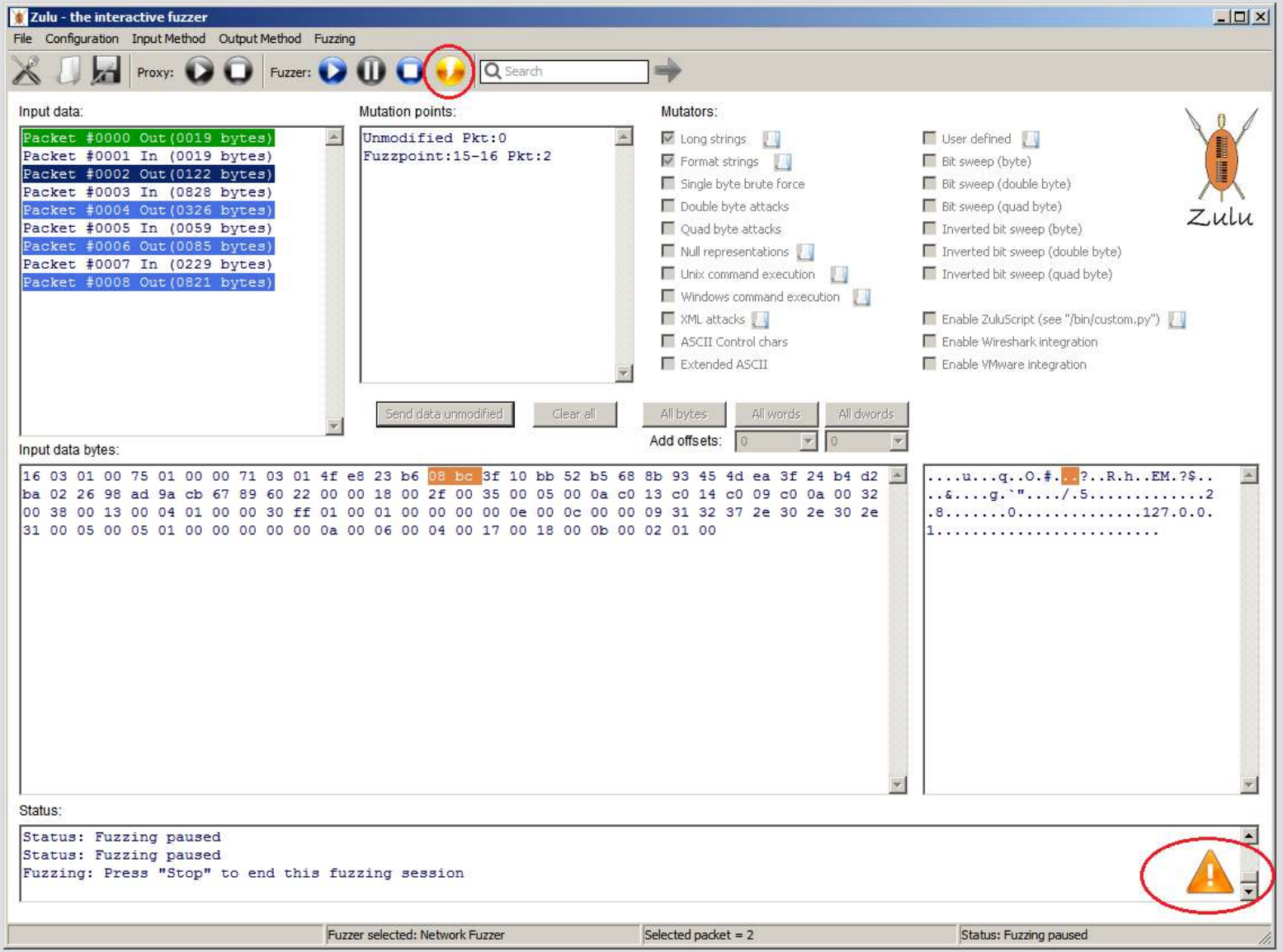


Start fuzzing



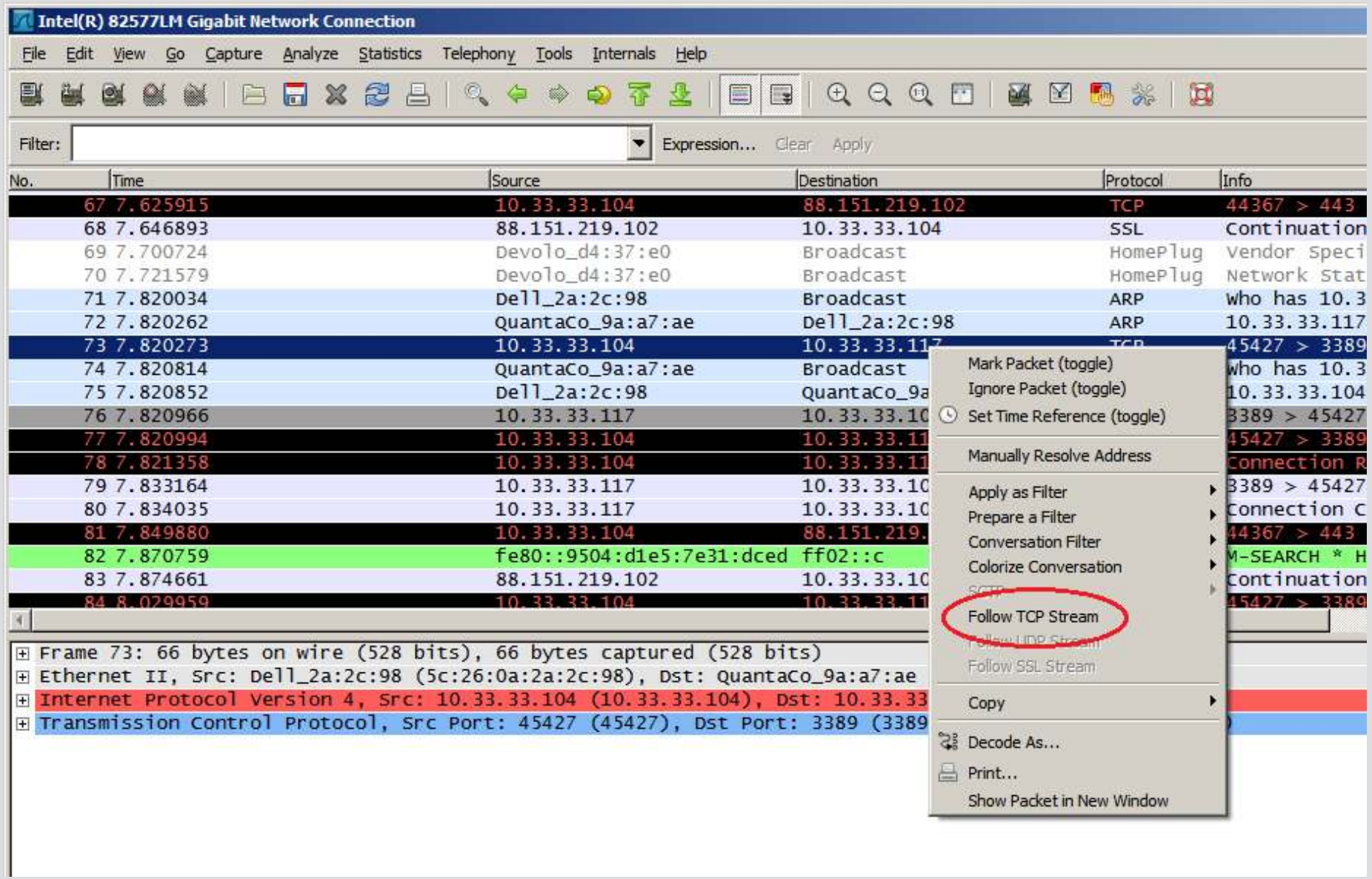


Instrumentation and triage

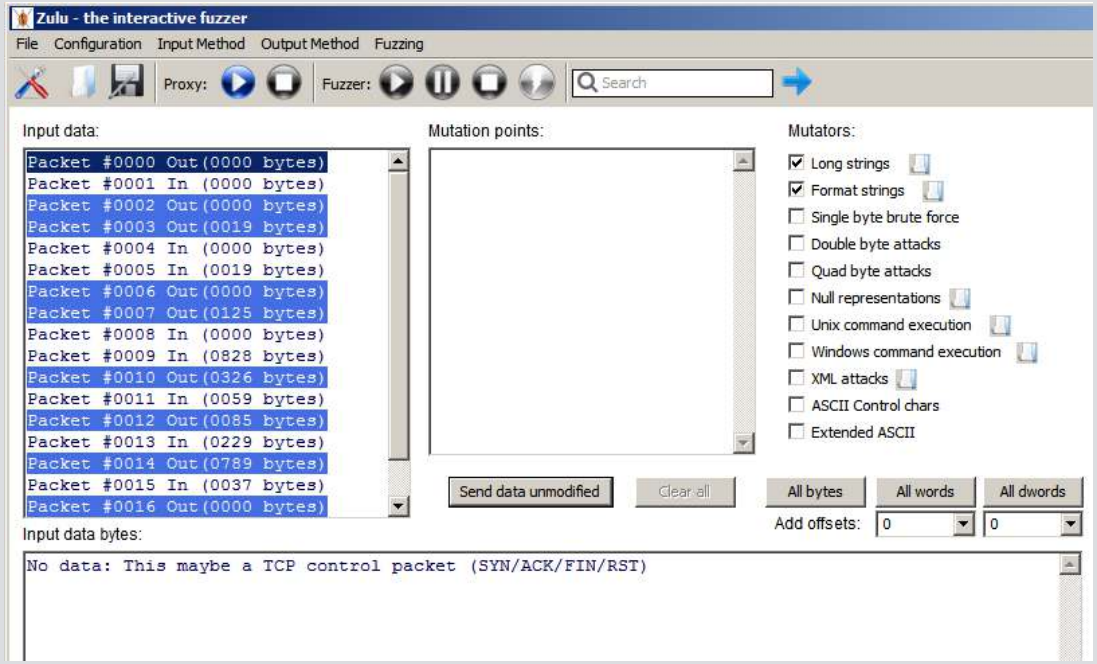


Other inputs: PCAP files

Wireshark captures

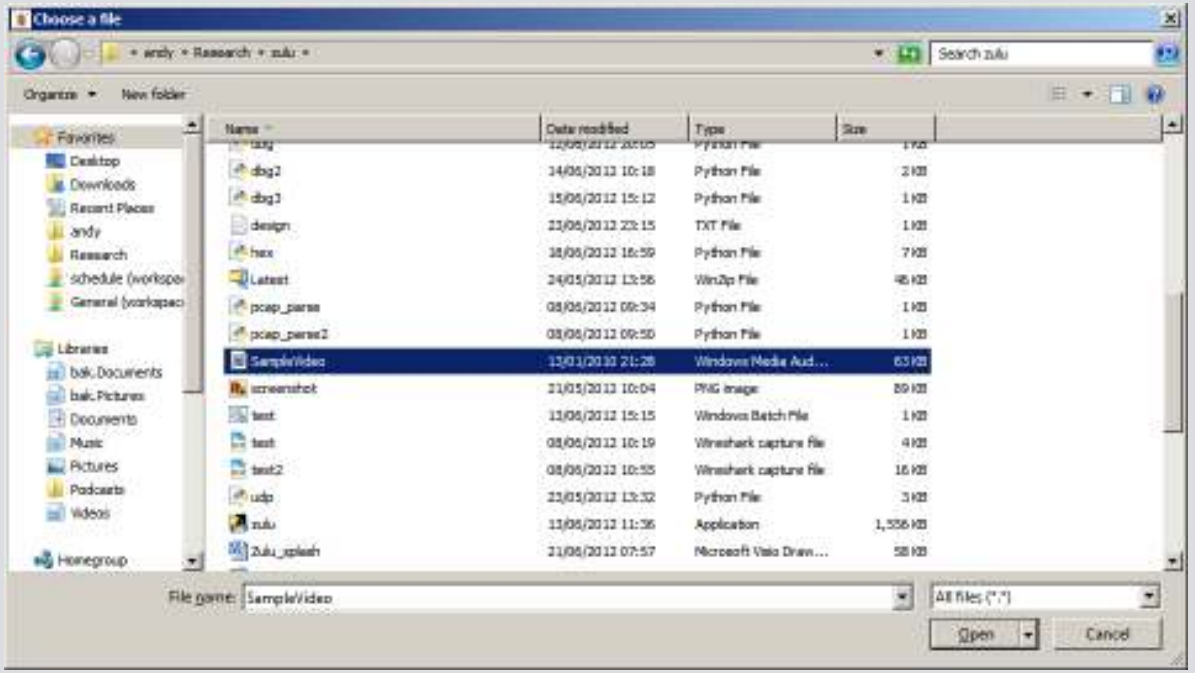


Importing a PCAP

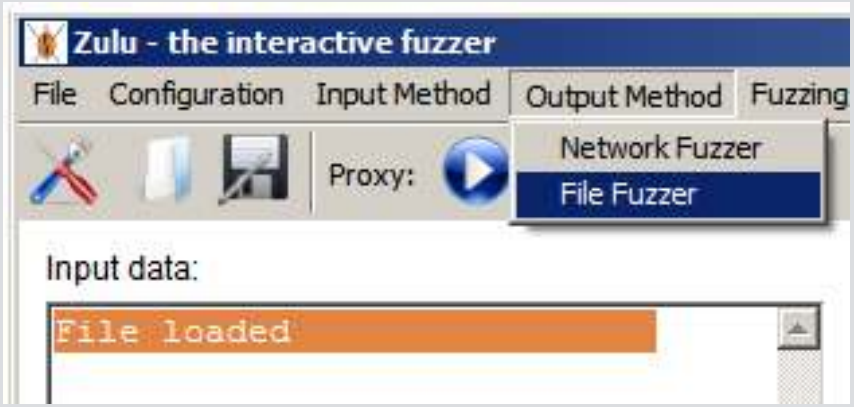


File module

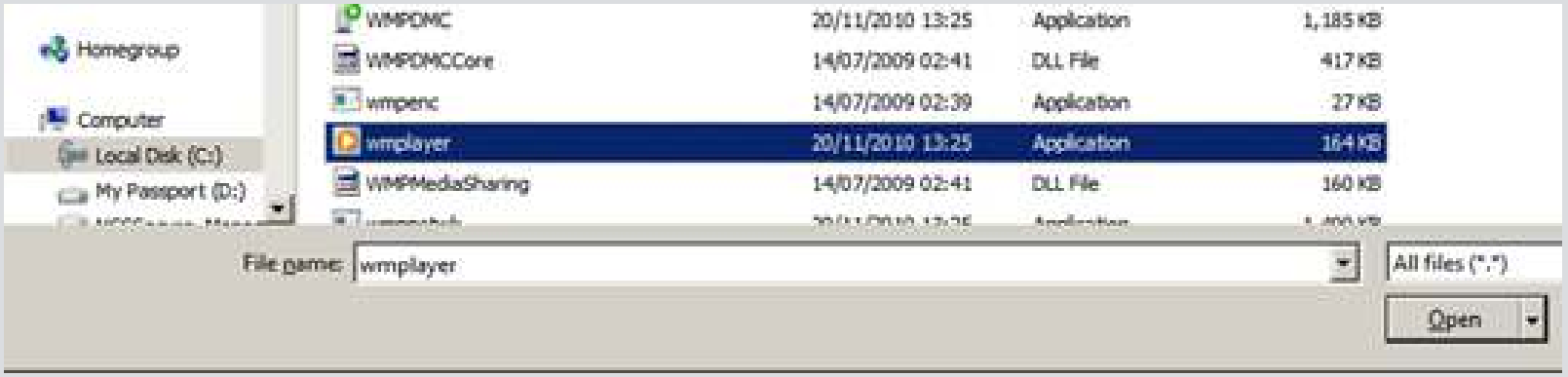
Select input file

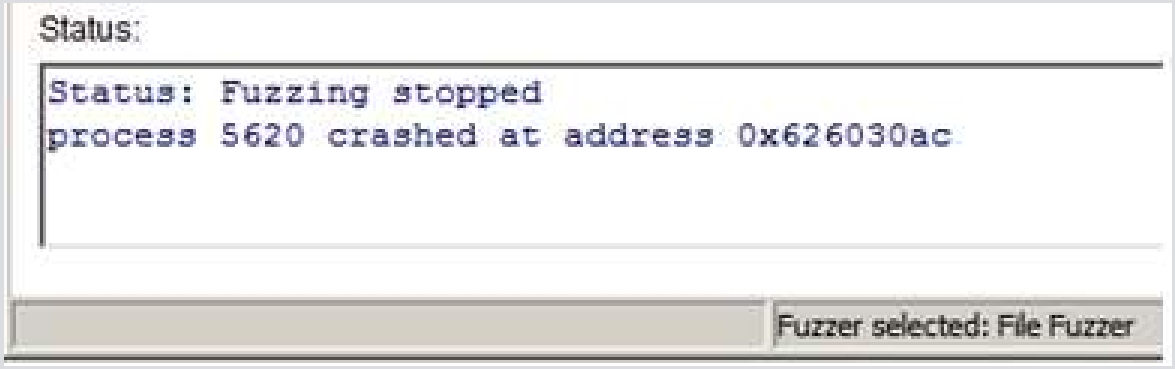


Select file fuzzer + fuzz process



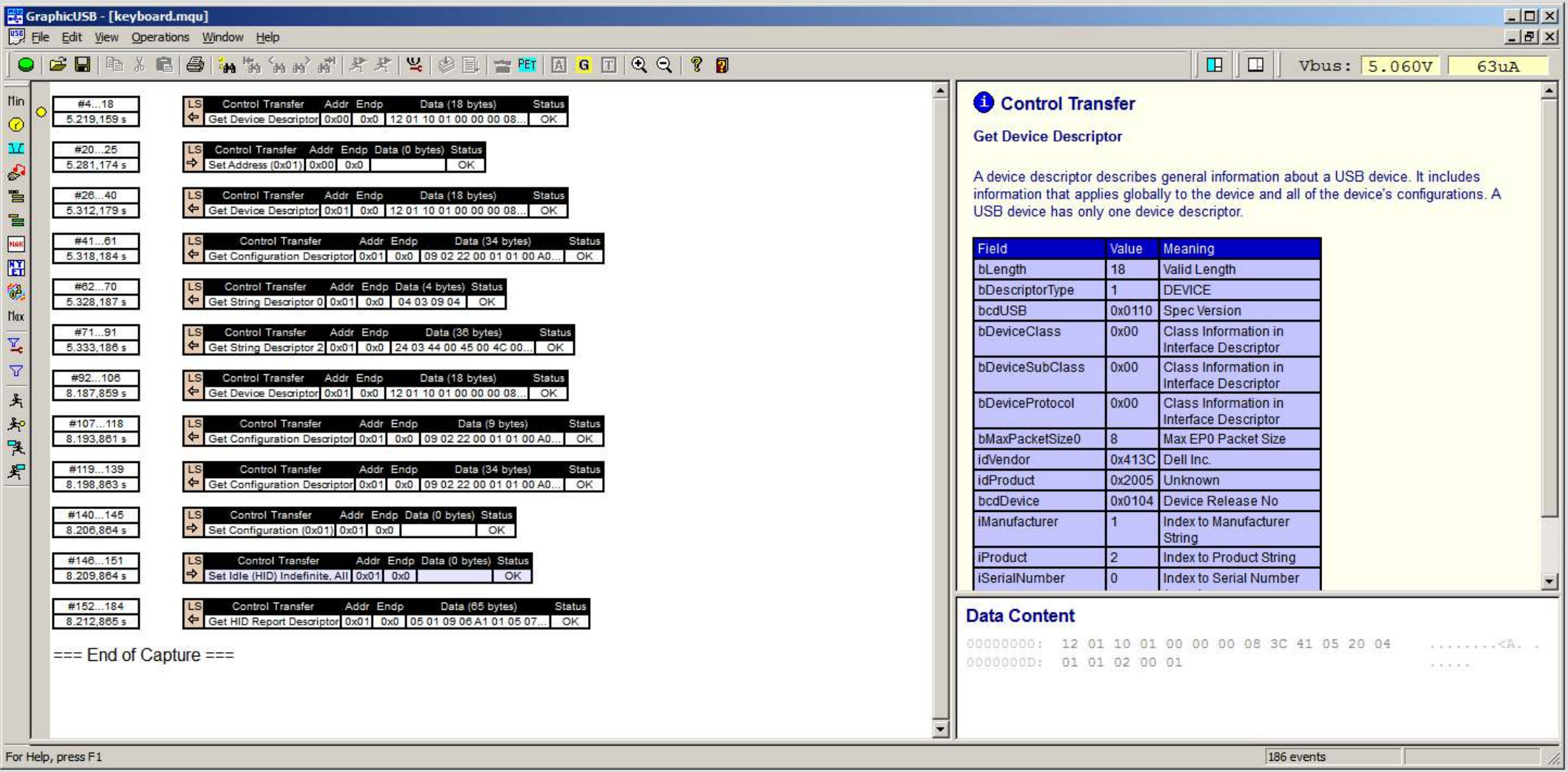
Fuzz process + debugging



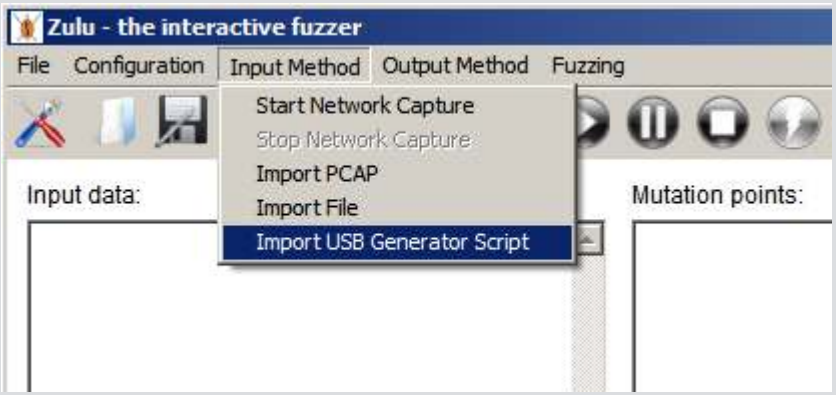


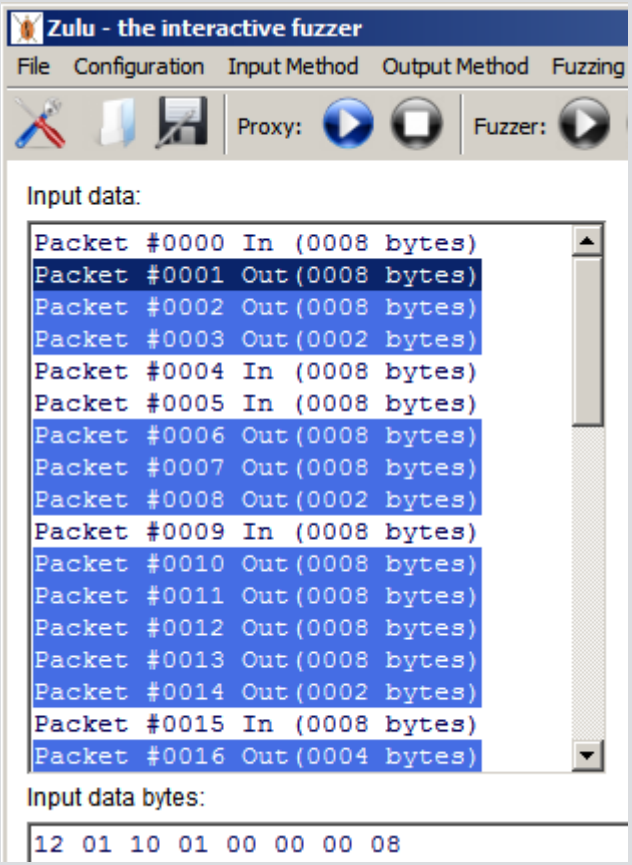
USB module

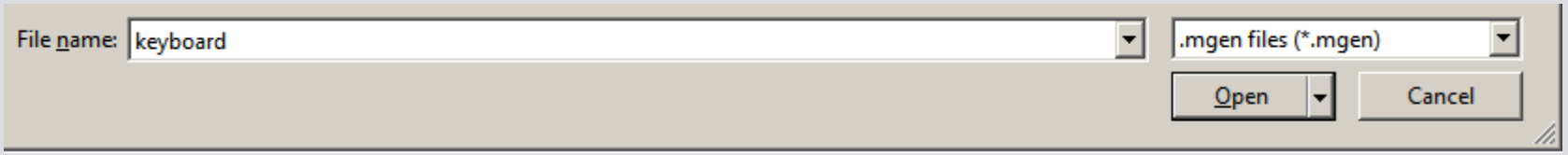
Graphic USB


GraphicUSB - [keyboard.mqu]
File Edit View Operations Window Help
Vbus: 5.060V 63uA
#4...18 5.219,159 s | Control Transfer | Addr 0x00 | Endp 0x0 | Data (18 bytes) 12 01 10 01 00 00 00 08... | Get Device Descriptor | Status OK
#20...25 5.281,174 s | Control Transfer | Addr 0x00 | Endp 0x0 | Data (0 bytes) | Set Address (0x01) | Status OK
#26...40 5.312,179 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (18 bytes) 12 01 10 01 00 00 00 08... | Get Device Descriptor | Status OK
#41...61 5.318,184 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (34 bytes) 09 02 22 00 01 01 00 A0... | Get Configuration Descriptor | Status OK
#62...70 5.328,187 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (4 bytes) 04 03 09 04 | Get String Descriptor 0 | Status OK
#71...91 5.333,186 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (36 bytes) 24 03 44 00 45 00 4C 00... | Get String Descriptor 2 | Status OK
#92...106 8.187,859 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (18 bytes) 12 01 10 01 00 00 00 08... | Get Device Descriptor | Status OK
#107...118 8.193,861 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (9 bytes) 09 02 22 00 01 01 00 A0... | Get Configuration Descriptor | Status OK
#119...139 8.198,863 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (34 bytes) 09 02 22 00 01 01 00 A0... | Get Configuration Descriptor | Status OK
#140...145 8.206,864 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (0 bytes) | Set Configuration (0x01) | Status OK
#146...151 8.209,864 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (0 bytes) | Set Idle (HID) Indefinite, All | Status OK
#152...184 8.212,865 s | Control Transfer | Addr 0x01 | Endp 0x0 | Data (65 bytes) 05 01 09 06 A1 01 05 07... | Get HID Report Descriptor | Status 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

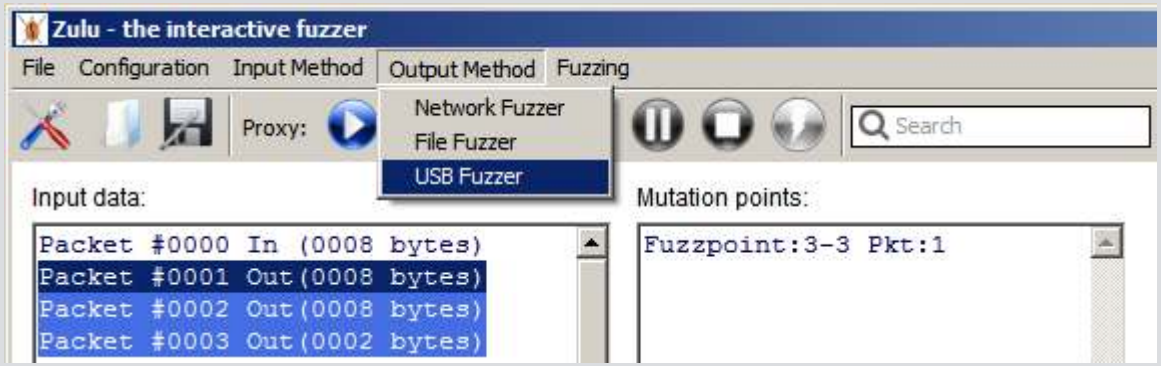
Import generator script

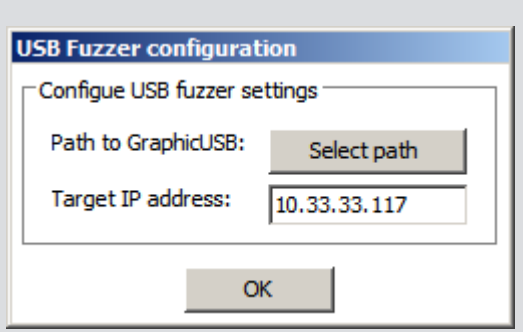




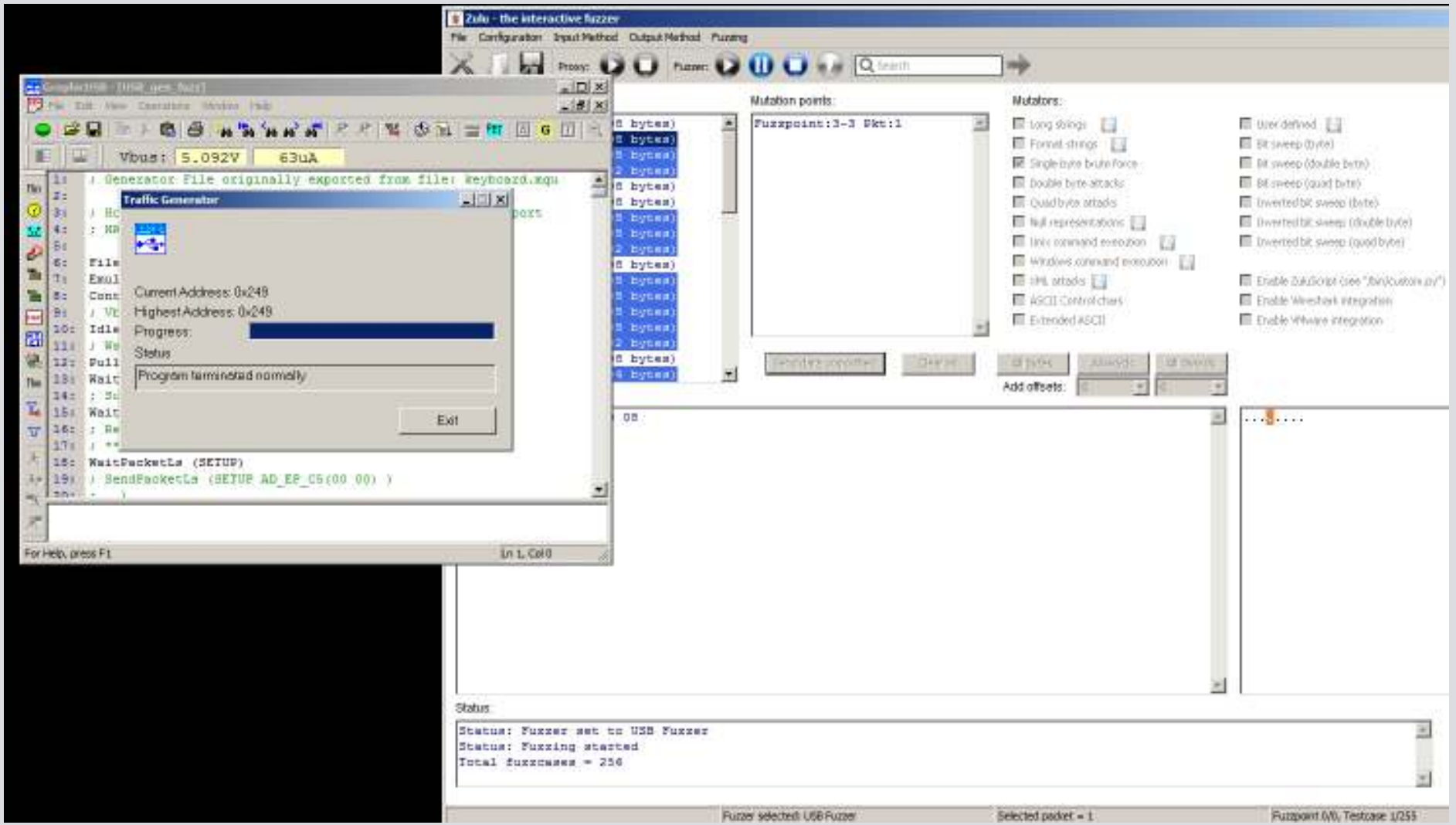


Select USB fuzzer



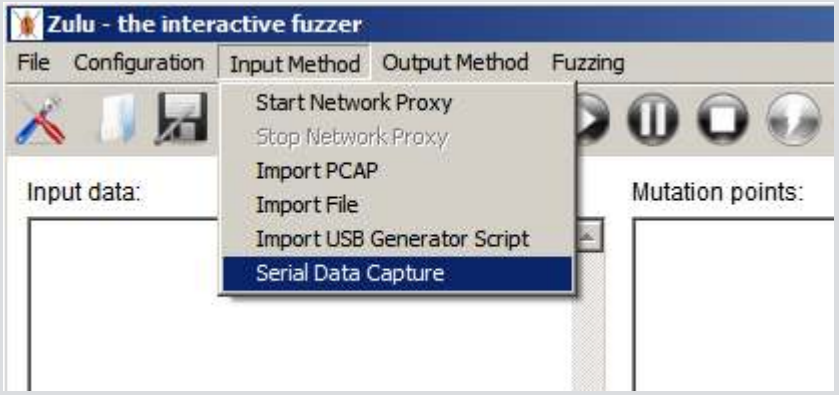


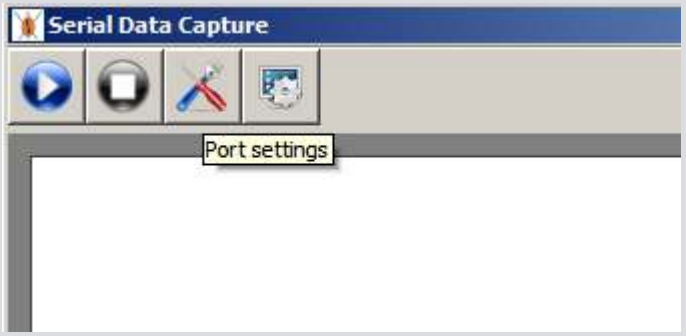
Fuzzer running

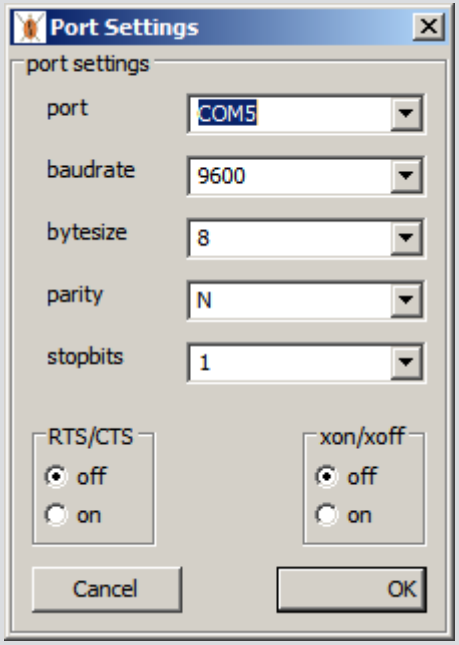


Serial module

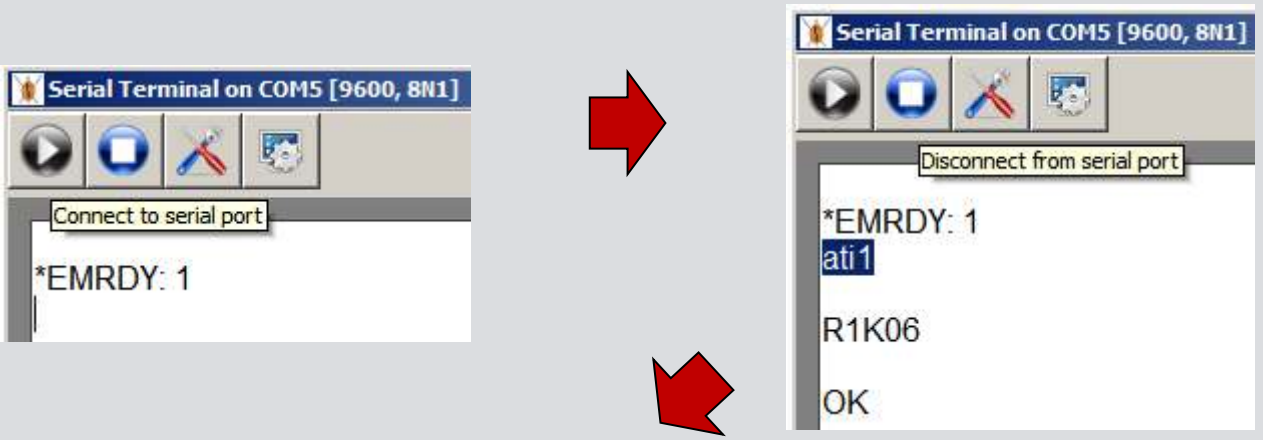
Serial settings

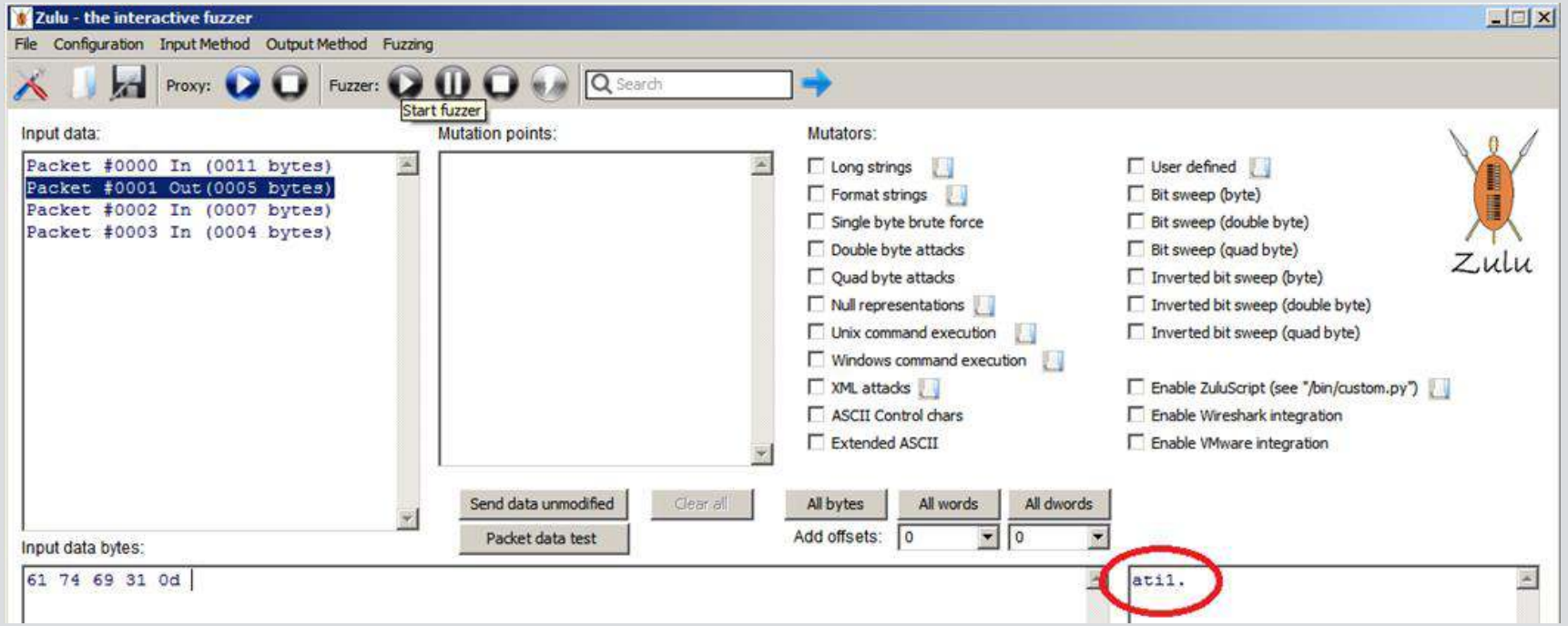




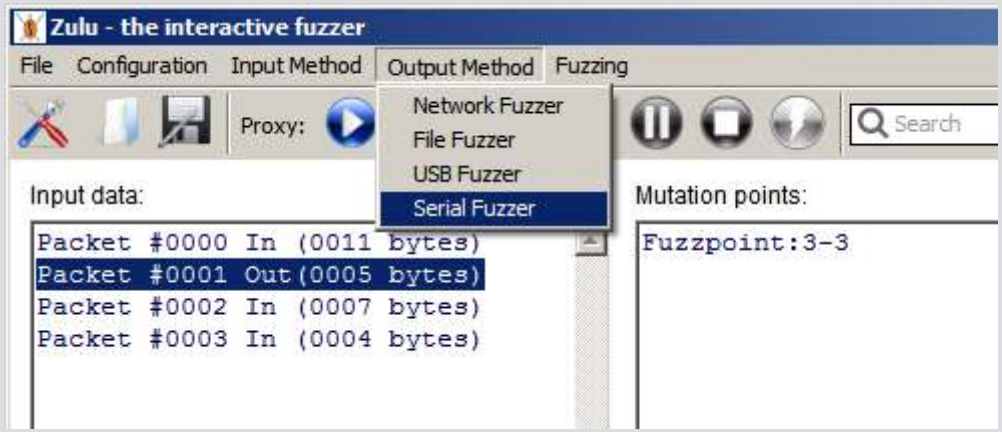


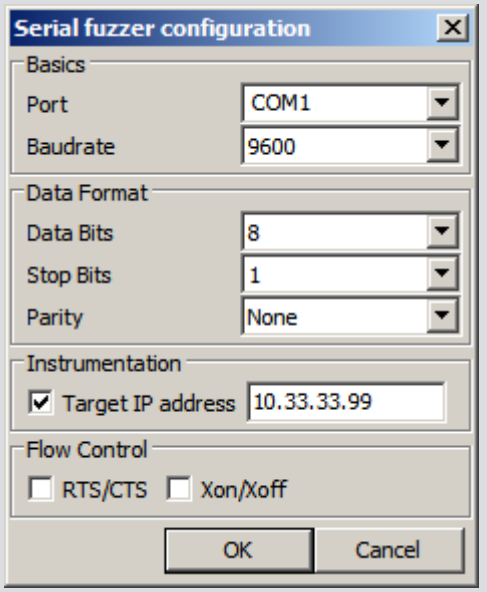
Serial data capture





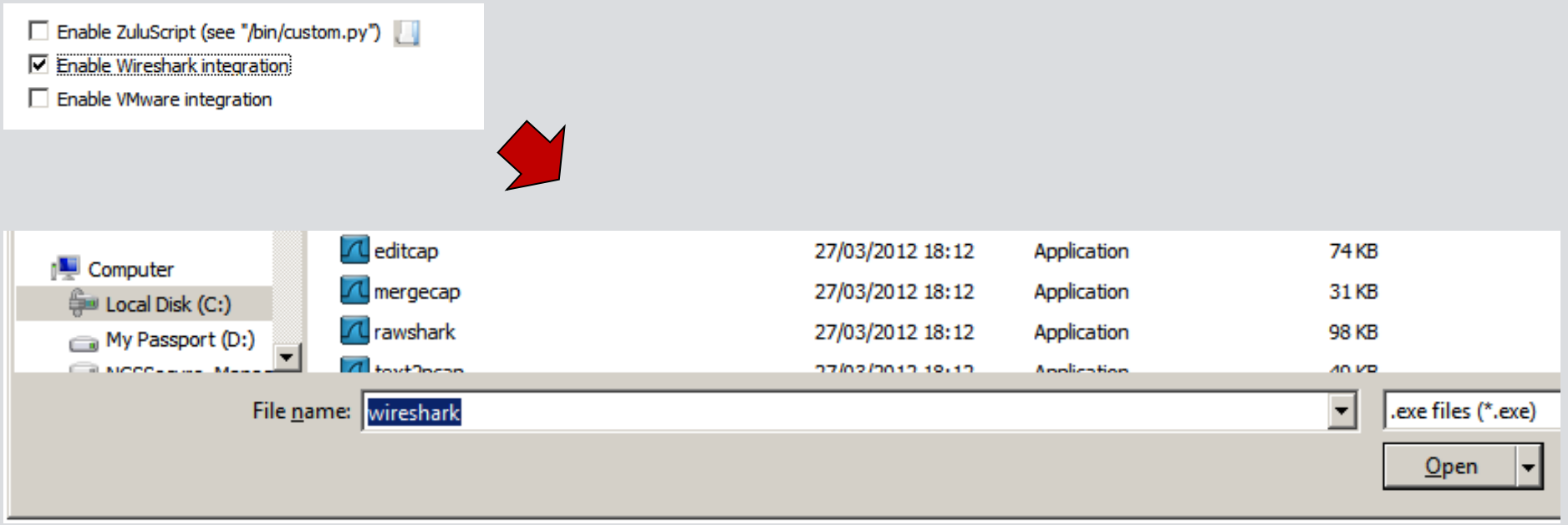
Serial fuzzing



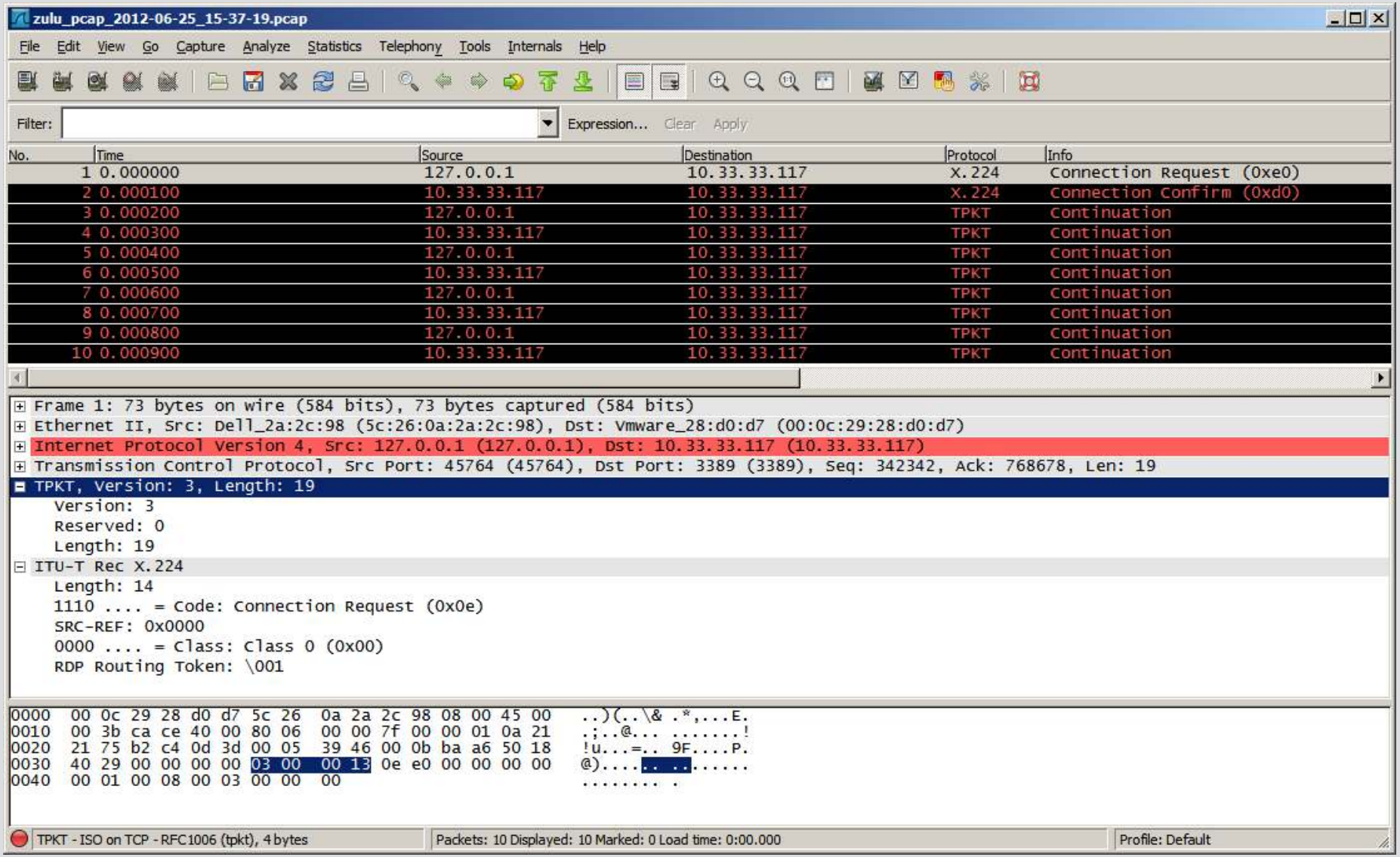


Wireshark integration

Point to Wireshark binary

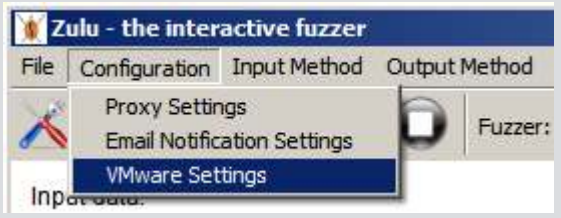


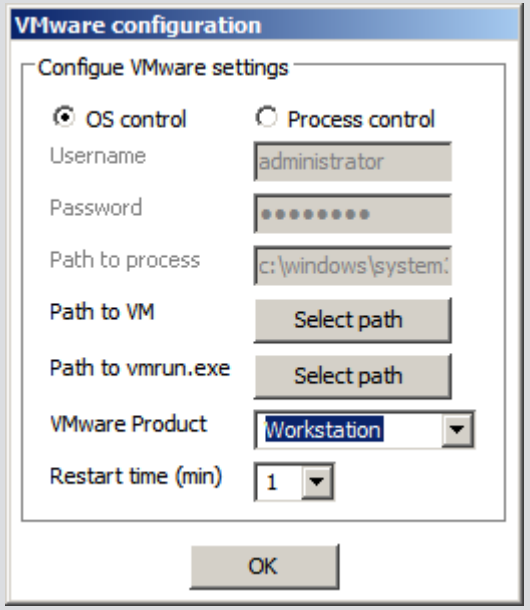
Auto-load Wireshark

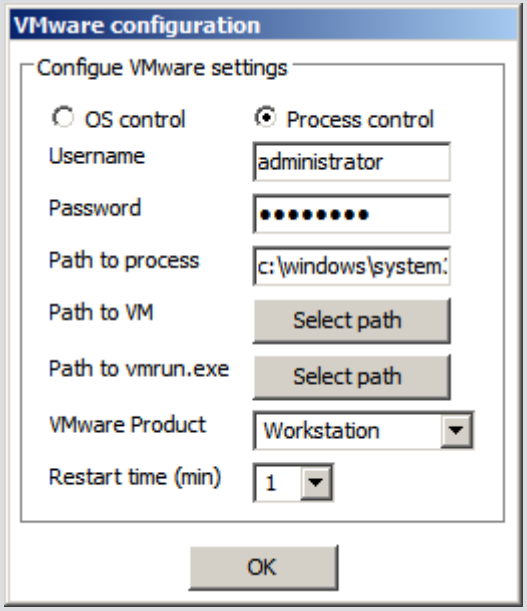


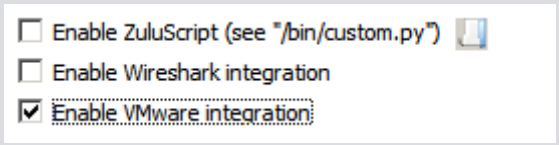
VMware integration

Select file fuzzer + fuzz process



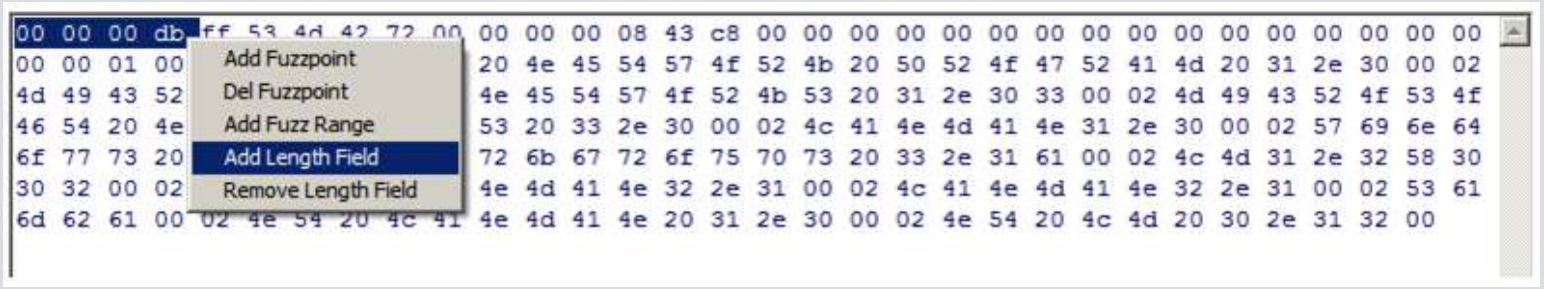


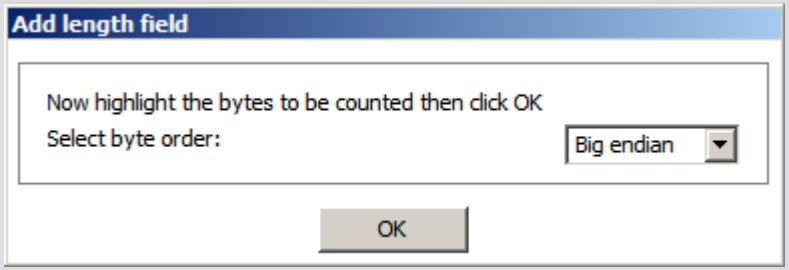


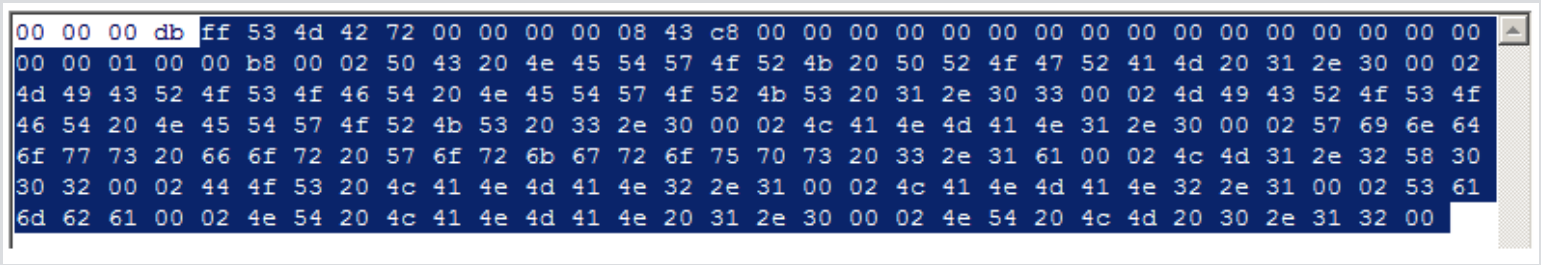


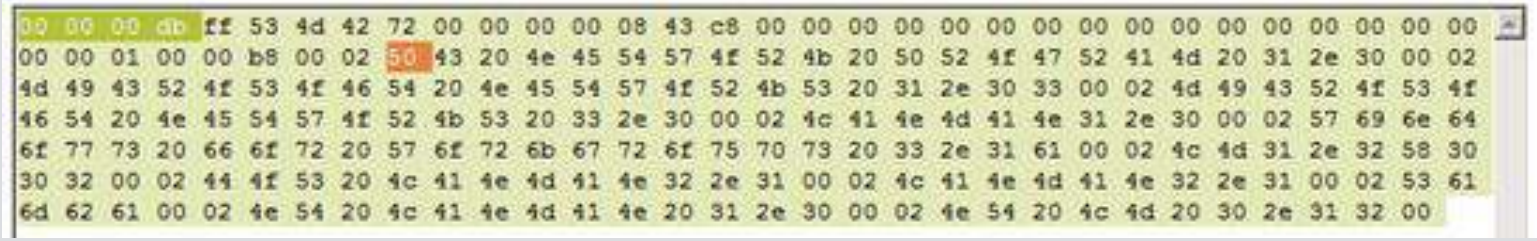
GUI-power

Adding a length field



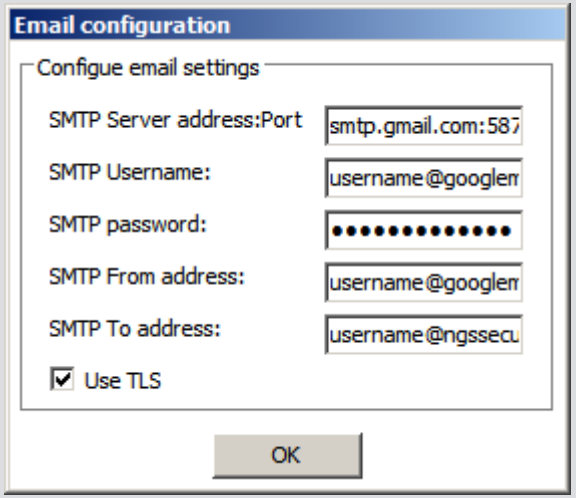






No need to watch! Email alerts

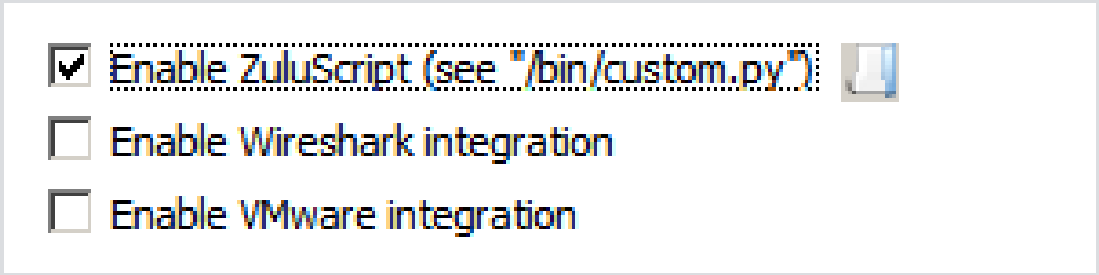
Select email settings



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



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

nccgroup
freedom from doubt

Questions?

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