



Fuzzing

Information Security Inc.

Contents

- What is Fuzzing
- Why Fuzzing
- User testing vs Fuzzing
- Fuzzing techniques
- Fuzzing example
- Fuzzing tools
- Conclusions
- References

What is Fuzzing

- Black Box software testing technique, which basically consists in finding implementation bugs using malformed/semi-malformed data injection in an automated fashion
- A form of vulnerability analysis

Standard HTTP GET request

- § GET /index.html HTTP/1.1

Anomalous requests

- § AAAAAA...AAAA /index.html HTTP/1.1
- § GET //////////index.html HTTP/1.1
- § GET %n%n%n%n%n%n.html HTTP/1.1
- § GET /AAAAAAAAAAAAAAAA.html HTTP/1.1
- § GET /index.html HTTTTTTTTTTTTTTTP/1.1
- § GET /index.html HTTP/1.1.1.1.1.1.1.1
- § etc...

Why fuzzing

- Fuzzing is an area that has gained a lot of attention in the past few years and several more advanced approaches have emerged in both academic circles and industry
- The purpose of fuzzing relies on the assumption that there are bugs within every program, which are waiting to be discovered. Therefore, a systematical approach should find them sooner or later

User testing vs Fuzzing

- User testing: run program on many normal inputs, look for bad things to happen
 - Goal: Prevent normal users from encountering errors
- Fuzzing: Run program on many abnormal inputs, look for bad things to happen
 - Goal: Prevent attackers from encountering exploitable errors

Fuzzing Techniques

- Mutation Base – “Dumb Fuzzing”
- Generation Based – “Smart Fuzzing”
- Evolutionary

Fuzzing Techniques

© Dumb fuzzing (mutational) details and demo

- Take a good input and add anomalies to it.
- Inputs are generally either files (.pdf, .png, etc) , network based (http,tftp,SNMP etc).

Fuzzing Example

Simple Example: /* TFTP Protocol fuzzing */

© Fuzzing Setup

- Debian Linux (python script) >>> Windows 8.1 (TFTP Server)
- TFTP packet format (RFC 1350 <http://pentan.info/doc/rfc/j1350.html>)


2 bytes	string	1 byte	string	1 byte
Opcode	Filename	0	Mode	0

Fuzzing Example

© Attempting a crash

- Fuzzing TFTP's "Mode" field (only eight characters or less) with a Python script and hopefully the program will crash.
- The script receives no response from the server when a string length of 600 "I"s => transport mode of 500 "I"s crashed the server.

```
root@root:~# ./Fuzz.py
Fuzzing with length 100
('^x00\x05\x00\x04Unknown or unsupported transfer mode : 
IT\x00', ('192.168.10.93', 60817))
Fuzzing with length 100
('^x00\x05\x00\x04Unknown or unsupported transfer mode : 
IT\x00', ('192.168.10.93', 60818))
Fuzzing with length 200
('^x00\x05\x00\x04Unknown or unsupported transfer mode : 
IT\x00', ('192.168.10.93', 60819))
Fuzzing with length 300
('^x00\x05\x00\x04Unknown or unsupported transfer mode : 
IT\x00', ('192.168.10.93', 60820))
Fuzzing with length 400
('^x00\x05\x00\x04Unknown or unsupported transfer mode : 
IT\x00', ('192.168.10.93', 60821))
Fuzzing with length 500
('^x00\x05\x00\x04Unknown or unsupported transfer mode : 
IT\x00', ('192.168.10.93', 60822))
Fuzzing with length 600
```



Fuzzing Example

Python script used for this simple fuzzing example

```
#!/usr/bin/env python
import socket

""" first entry in the array; string of 100 "I"s """
buffarray = ["I"*100]
add = 100

""" Progressively append longer strings until it is 50 elements long """
while len(buffarray) <=50:
    buffarray.append("I"*add)    #new element added to the array
    add += 100

""" Grab each element of the array in turn and send it within the Mode
field of a legitimate TFTp packet """

for value in buffarray:
    tfttpak = "\x00\x02" + "Adi" + "\x00" + value + "\x00"
    print "Fuzzing with length " + str(len(value))
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) #set up UDP socket
    s.sendto(tfttpak, ("192.168.10.93", 69))
    response = s.recvfrom(4096)
    print response
```

Fuzzing Tools

- **American fuzzy lop** (<http://lcamtuf.coredump.cx/afl/>)
- **Zzuf** (<https://github.com/samhocevar/zzuf>)
- **Bunny the Fuzzer** (<http://code.google.com/p/bunny-the-fuzzer/>)
- **Peach** (<http://peachfuzzer.com/>)
- **Sulley** (<http://code.google.com/p/sulley/>)
- **Jbro Fuzz** (<https://www.owasp.org/index.php/JBroFuzz>)
- **Wfuzz** (<https://github.com/xmendez/wfuzz>)

Fuzzing Tools

Radamsa (<https://github.com/aoh/radamsa>)

```
root@LUCKY64:~# for i in {1..8};do echo $i;done | radamsa
1P
1
257
4
0
6
7
8
```

Conclusions

Even in its simplest form, fuzzing can be a very useful tool for uncovering vulnerabilities and should be in the repertoire of every information security engineer.

References

- Introduction to Fuzzing

<http://www.cs.ucr.edu/~heng/teaching/cs260-winter2017/fuzzing.pdf>

- OWASP Org Fuzzing

<https://www.owasp.org/index.php/Fuzzing>

- Fuzz vectors OWASP

https://www.owasp.org/index.php/OWASP_Testing_Guide_Appendix_C:_Fuzz_Vectors