Fuzzing - lab

Finding bugs by fuzzing

Petr Švenda svenda@fi.muni.cz
Faculty of Informatics, Masaryk University

CROCS
Centre for Research on Cryptography and Security

Centre for Research on Cryptography and Security

- Established 2006 at the Faculty of Informatics of the Masaryk University, Brno (formerly LaBaK)
- Security-related university courses
- Applied research with commercial partners
- Laboratory for students (http://crcs.cz/openlab)

CROCS
Centre for Research on Cryptography and Security

Pre-prepare

- Download and install WinDbg
  - https://msdn.microsoft.com/en-us/windows/hardware/hh852365
- Download zip with all binaries
  - http://crcs.cz/wiki/doku.php?id=public:research:secprog:fuzzing

Today main goals

- See few fuzzing tools in progress
- Take home working setup
- Motivate to fuzz your application

Approaches and tools to be covered

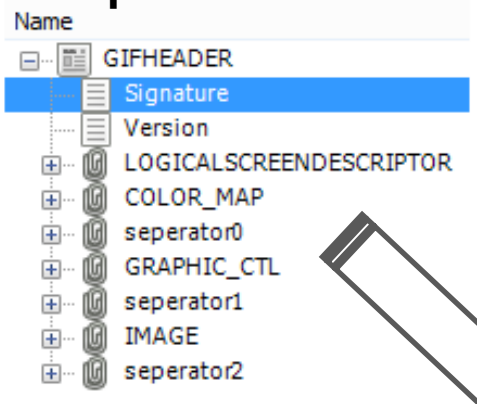
- Simple random file fuzzing (MS's SDL MiniFuzz)
- Simple content-aware file/net fuzzing (Radamsa)
- Test vectors against web applications (OWASP ZAP)
- Stateful, custom data format fuzzing (Peach)

Fuzzing - advantages/disadvantages

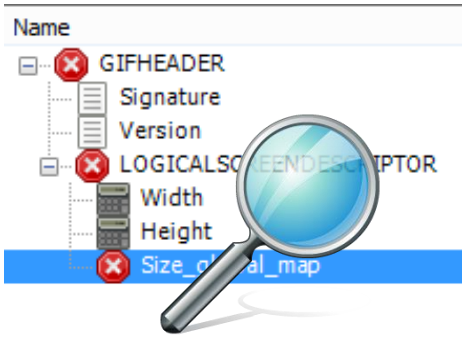
- Fuzzing advantages
  - Very simple design
  - Allow to find bugs missed by human eye
  - Sometimes the only way to test (closed system)
  - Repeatable (crash inputs stored)
- Fuzzing disadvantages
  - Usually simpler bugs found (low hanging fruit)
  - Increased difficulty to evaluate impact or dangerosity
  - Closed system is often evaluated, black box testing

1. Investigate app in/out

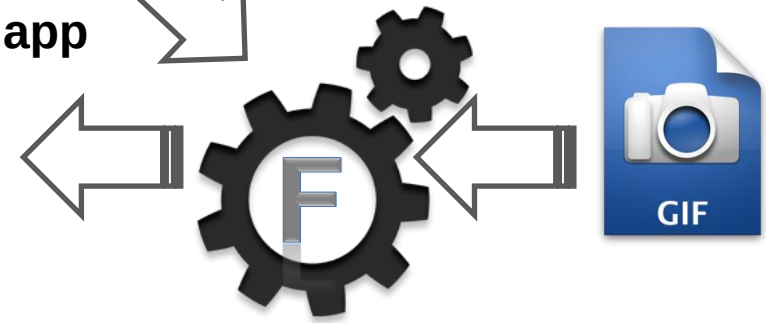
2. Prepare data model



3. Validate data model



4. Generate fuzzed inputs



5. Send fuzzed input to app

6. Monitor target app

7. Analyze logs

http://iconarchive.com,
http://awicons.com,
http://www.pelfusion.com

Note: Fuzzing and higher level languages

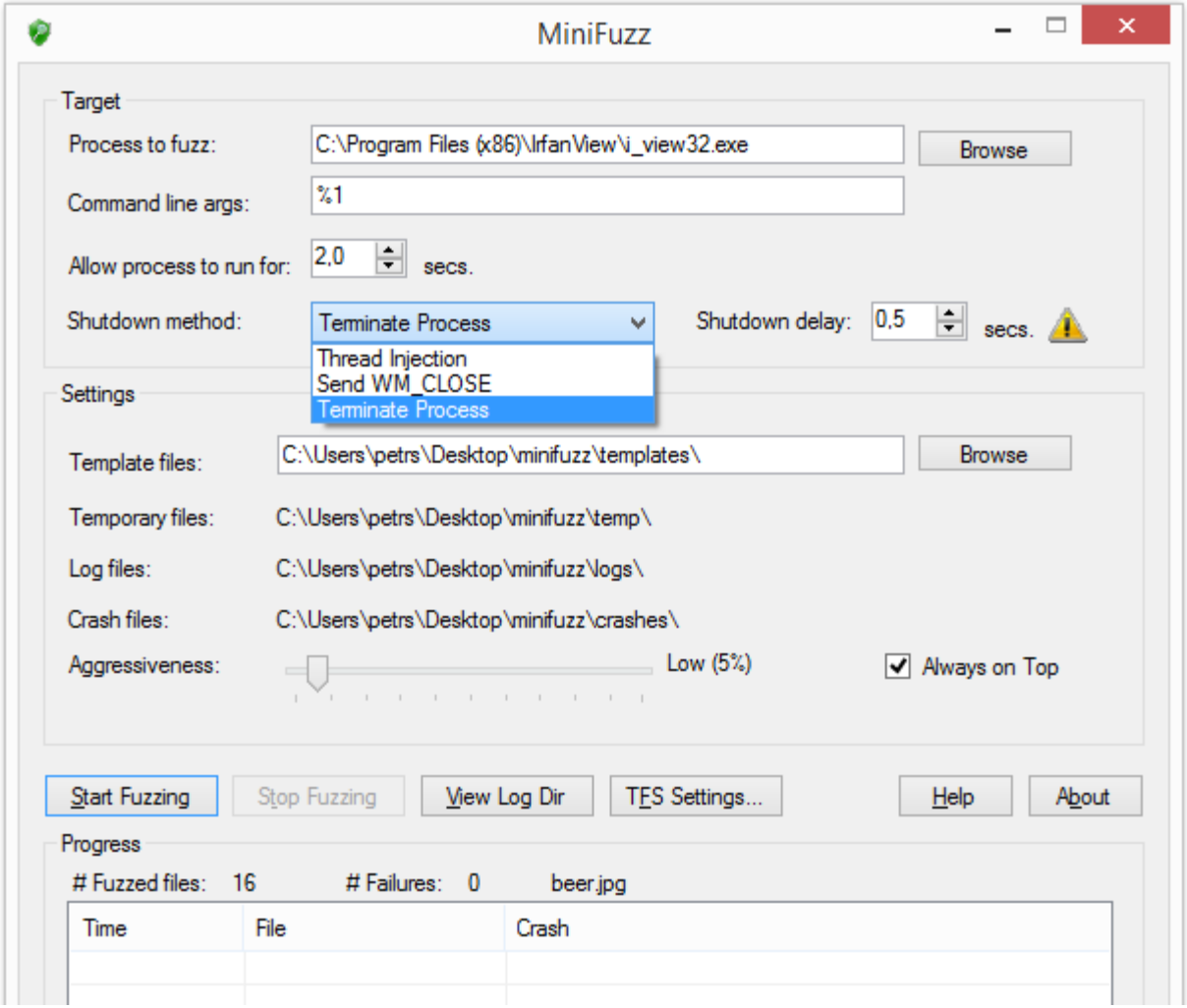
- Fuzzing tends to find simpler bugs
  - Memory issues for C/C++ (usually exploitable)
  - Insufficient range checking, invalid arguments… for Java (only sometimes exploitable - DOS or logical errors)
- Variants of known problematic inputs for logic error
- If fuzzing for logical errors
  - If you control source code, emit exception when unwanted state occur (assert, exception) – captured by fuzz tool
  - Or inspect results manually (e.g., OWAP ZAP)

SDL MINIFUZZ FILE FUZZER

Microsoft's SDL MiniFuzz File Fuzzer

- Application input files fuzzer
  - http://www.microsoft.com/en-us/download/details.aspx?id=21769
- Templates for valid input files (multiple)
- Modify valid input file (randomly, % aggressiveness)
- Run application with partially modified inputs
- Log resulting crash (if happen)
  - exception, CPU registers...
- Video overview
  - http://msdn.microsoft.com/en-us/security/gg675011.aspx

Microsoft's SDL MiniFuzz File Fuzzer



Play with SDL MiniFuzz

- Goal: crash IrfanView v1.75 (1996)
  - Image file goes as first argument

1. Gather at least one input file into template folder
   - (try own, but Icon_ManBig_128.GIF should work)
2. Set proper shutdown method (experiment)
3. Run and observe crashes

Play with SDL MiniFuzz – bonus tasks

- Bonus: Can you increase the speed of testing?
- Bonus: What is the impact of aggressiveness?
- Bonus: What about your application?
- Note: MS SDL requires 100k runs without failure

RADAMSA FUZZER

Radamsa fuzzer

- *"…easy-to-set-up general purpose shotgun test to expose the easiest cracks…"*
  - https://code.google.com/p/ouspg/wiki/Radamsa
- Just provide input files, all other settings automatic
  - **cat** file **|** radamsa **>** **file.fuzzed**

```
>echo "1 + (2 + (3 + 4))" | radamsa --seed 12 -n 4
 1 + (2 + (2 + (3 + 4?)
 1 + (2 + (3 +?4))
 18446744073709551615 + 4)))
 1 + (2 + (3 + 170141183460469231731687303715884105727))
```

Radamsa as file fuzzer (XML example)

- **`radamsa -o fuzz_%n.xml -n 1000 *.xml`**
  - Takes file template from `*.xml` file(s)
  - Generates given number (1000) of fuzzed files (-n 1000)
- Testing you application
  1. Collect valid input file(s) for target app into *.xml file(s)
  2. Run Radamsa to create large number of fuzzed files
  3. Run your application with fuzzed input file and monitor
     - Custom code for monitoring (e.g., crash detected by success in acquire of named mutex)
     - WinDbg for monitoring, parse output log file

```
# powershell.exe -ExecutionPolicy Bypass ./ff_radamsa.ps1 beer.jpg irfan.exe 10
$fileTemplate = $args[0]
$fileTemplateResolved = Resolve-Path $args[0]
$targetApp = Resolve-Path $args[1]
$totalRuns = $args[2]
$radamsa= Resolve-Path "radamsa.exe"
$count=1
while ($count -le $totalRuns) {
  $fuzzFileName = "fuzz-" + $count + "_" + $fileTemplate
  $fuzzFileWindbRes = $fuzzFileName + ".wdbg.log"
  # run Radamsa to generate single fuzzed file
  & $radamsa -o $fuzzFileName $fileTemplate
  Write-Host "New file $fuzzFileName generated"
  # run target application with fuzzed file as argument under WinDbg monitoring
  & windbg -logo $fuzzFileWindbRes $targetApp $fuzzFileName
  # wait some time
  Start-Sleep -s 2
  # terminate target program inside windbg
  $a = Get-WmiObject win32_process -Filter "name = 'windbg.exe'"
  $a | % {Invoke-WmiMethod -Name terminate -InputObject $_ | out-null}
  # TODO: parse output log files *.wdbg.log
  $count++
}
```

Radamsa as fuzzing client – test server

- `radamsa -o ip:80 -n inf samples/*.http-req`
  - Connects as client to server at ip:80, runs infinitelly (-n inf)
  - Takes template inputs from `*.http-req` file(s)
  - Send fuzzed input to server
- Testing you server
  1. Capture valid request for your client to server (e.g., GET request) and store into *.http-req file(s)
  2. Run (repeatedly) Radamsa as TCP client
  3. Monitor behaviour of your server under Radamsa requests

Radamsa as fuzzing server – test client

- `radamsa -o :8888 -n inf samples/*.http-resp`
  - Starts as server on port 8888, runs infinitelly (-n inf)
  - Takes template inputs from `*.http-resp` files
  - Return fuzzed input to connecting client
- Testing you client
  1. Capture valid responses from your server (e.g., HTML page) and store into *.http-resp file(s)
  2. Run Radamsa as server
  3. Run your client (repeatedly) and monitor its behaviour

Questions for Radamsa

- In what is SDL MiniFuzz better then Radamsa?
- Why is Radamsa better in fuzzing text files?
- Can you fuzz vulnserver.exe?
  - 127.0.0.1:9999
- How to test server/client in stateful protocol?

OWASP ZAP

OWASP's ZAP – fuzz strategy settings

Quick Start | Request | Response | Break | Script Console

Welcome to the OWASP Zed Attack Proxy (ZAP)

ZAP is an easy to use integrated penetration te

Please be aware that you should only attack ap

To quickly test an application, enter its URL bel

URL to attack: http://ysoft.com

Attack

Progress: Not started

For a more in depth test you should explore you

If you are using Firefox 24.0 or later you can us

Configure your browser: Plug-n-Hack

Or point your browser at: http://localhost:8080

Select...

Options

- Options
  - Active Scan
  - Active Scan Input V
  - AJAX Spider
  - Anti CSRF Tokens
  - API
  - Applications
  - Authentication (De
  - Breakpoints
  - Certificate
  - Check For Updates
  - Connection
  - Database
  - Display
  - Dynamic SSL Certi
  - Encode/Decode
  - Extensions
  - Forced Browse
  - Fuzzer
  - Global Exclude UR

Fuzzer

Default category: jbrofuzz / XSS

Concurrent scanning threa

0 5 10 15 2

Add custom Fuzz file:

- jbrofuzz / SQL Injection
- jbrofuzz / URI Exploits
- jbrofuzz / User Agents
- jbrofuzz / Web Server
- jbrofuzz / XML Injection
- jbrofuzz / XPath Injection
- jbrofuzz / XSS
- jbrofuzz / Zero Fuzzers

OK | Cancel

ced Browse | Fuzzer | Params | Http Sessions | Zest Results | Clients | WebSockets | AJAX Spider | Output

Fuzzing server with OWASP's ZAP

- Predefined list of known problematic test vectors

1. Map structure of target server (Spider)
   – Quick start → URL to attack → *target.com*
2. Select site/request you like to fuzz
   – Sites → *target.com* (→ GET:*index.html*)
3. Select field you like to fuzz and fuzzing test set
   – Request → *RClick* → Fuzz…
4. Inspect results in Fuzzer tab

Sites | Scripts | Quick Start | Request | Response | Break

- Sites
  - http://astrolight.cz
    - GET:Canon400D_IRmod.html
    - GET:stuff.html
    - GET:style.css
    - GET:index.html
    - GET:frameset.htm
    - GET:Svenda_Astrophotography_Europen2013_final.pdf
    - images
    - GET:kalibrace.gif

Header: Text | Body: Text

```
GET http://astrolight.cz/index.html HTTP/1.1
User-Agent: Mo
Pragma: no-cac
Cache-control:
Content-Length
Host: astrolig
```

- Find...
- Encode/Decode/Hash...
- Resend...
- Open URL in Browser
- Fuzz

Quick Start | Request | Response | Break | Script Console

Header: Text | Body: Text

```
GET http://astrolight.cz/index.html HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;)
Pragma: no-cache
Cache-control: no
Content-Length: 6
Host: astrolight.
```

Fuzz

String to Fuzz:	Mozilla/4.0 (compatible; MSIE 6.0; Windo...
Follow Redirects:	☑
URL Encode:	☑
Fuzz Category:	jbrofuzz / HTTP

- All User Agents
- HTTP Methods

- jbrofuzz / HTTP
- jbrofuzz / Alphabets
- jbrofuzz / Base
- jbrofuzz / Biology
- jbrofuzz / Buffer Overflows
- jbrofuzz / Exploits
- jbrofuzz / Format String Errors
- jbrofuzz / HTTP
- jbrofuzz / Headers
- jbrofuzz / Injection
- jbrofuzz / Integer Overflows
- jbrofuzz / LDAP Injection
- jbrofuzz / Number Systems
- jbrofuzz / O/S Variables
- jbrofuzz / Recursive Fuzzers
- jbrofuzz / Replacive Fuzzers
- jbrofuzz / SQL Injection
- jbrofuzz / URI Exploits
- jbrofuzz / User Agents
- jbrofuzz / Web Server
- jbrofuzz / XML Injection
- jbrofuzz / XPath Injection
- jbrofuzz / XSS
- jbrofuzz / Zero Fuzzers
- Custom fuzzers

Questions for OWASP ZAP

- Why should you test only “your” server?
- Why is ZAP said to be “manual” pen test tool?
- Can ZAP handle SSL/TLS sessions?
- What Reflected status means?
  - False positive? Example of attack?
- How you can verify the attack?
- How can you use ZAP to feed inputs for Radamsa?
- Can ZAP perform other then fuzzing test?

Bonus: ZAP@Mutillidae – vulnerable app

- Install on Win machine with XAMPP
  - https://www.youtube.com/watch?v=1hF0Q6ihvjc&feature=youtu.be
  - Download and install XAMPP
  - Download and unpack Mutillidae into xampp_install\htdocs
  - Start XAMPP control console, run Apache and MySQL
  - Visit localhost/mutillidae
- Or download Samurai Web Testing Framework http://samurai.inguardians.com/
- List of vulnerabilities:
  - http://sourceforge.net/projects/mutillidae/files/documentation/listing-of-vulnerabilities-in-mutillidae.txt/download

PEACH FUZZER

Peach – fuzzing vulnerable network server

1. Prepare Peach Pit file (example hter_pit.xml)
   - data model, state model, agent…
2. Run Peach Agent (first terminal)
   - `peach -a tcp`
3. Run Peach fuzzing (second terminal)
   - `Peach hter_pit.xml TestHTER`
   - Wait for detected crash (fault)
4. Inspect directory with crash logs
   - *Logs\hter_pit.xml_TestHTER_???\Faults\EXPLOITABLE_???*
5. Debug crash using fuzzed data from crash log
   - E.g., *1.Initial.Action.bin, 2.Initial.Action_1.bin…*

```
<DataModel name="DataHTER">
 <String value="HTER " mutable="false" token="true"/>
 <String value=""/>
 <String value="\r\n" mutable="false" token="true"/>
</DataModel>

<StateModel name="StateHTER" initialState="Initial">
 <State name="Initial">
  <Action type="input" ><DataModel ref="DataResponse"/></Action>
  <Action type="output"><DataModel ref="DataHTER"/></Action>
  <Action type="input" ><DataModel ref="DataResponse"/></Action>
 </State>
</StateModel>

<DataModel name="DataResponse">
 <String value=""/>
</DataModel>

<Agent name="RemoteAgent" location="tcp://127.0.0.1:9001">
 <!-- Run and attach windbg to a vulnerable server. -->
 <Monitor class="WindowsDebugger">
    <Param name="CommandLine" value="c:\Install\Fuzzers\Peach tutorial\VulnServer\vulnserver.exe"/>
    <Param name="WinDbgPath" value="c:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\" />
 </Monitor>
</Agent>

<Test name="TestHTER">
 <Agent ref="RemoteAgent"/>
 <StateModel ref="StateHTER"/>
 <Publisher class="TcpClient">
  <Param name="Host" value="127.0.0.1"/>
  <Param name="Port" value="9999"/>
 </Publisher>

 <Logger class="File">
  <Param name="Path" value="Logs"/>
 </Logger>
```

Model of input data
'HTER *anything* \r\n'

1. Read any string
2. Send fuzzed input
3. Read any string

Agent responsible for starting target application with debugger connected

Test scenario with specified settings

How to communicate with target application

How to store results

```
<DataModel name="DataHTER">
  <String value="HTER " mutable="false" token="true"/>
  <String value=""/>
  <String value="\r\n" mutable="false" token="true"/>
</DataModel>
```

C:\Windows\System32\cmd.exe

```
[297.--] Performing iteration
ing: DataHTER.DataElement_1
tor: DataElementDuplicateMutator

] Performing iteration
ing: DataHTER.DataElement_1
tor: StringMutator

ht fault at iteration 298, trying to reproduce --

] Performing iteration
ing: DataHTER.DataElement_1
tor: StringMutator

oduced fault at iteration 298 --

] Performing iteration
ing: DataHTER.DataElement_1
tor: UnicodeBadUtf8Mutator

] Performing iteration
ing: DataHTER.DataElement_1
```

C:\Windows\System32\cmd.exe - nc localhost 9999

```
AAAA
```

Questions for Peach

- Is Peach able to fuzz stateful protocols?
- Is Peach able to specify custom data format?
- Does Peach monitor only application crash?
- What are advantages/disadvantages wrt ZAP?

Peach fuzzer: DataModel (gif)

```
<DataModel name="GIFHEADER">
    <String name="Signature" length="3" value="GIF" token="true" />
    <String name="Version" length="3" value="89a" token="true" />
    <Block name="LOGICALSCREENDESCRIPTOR">
        <Number name="Width" size="16" endian="little" signed="false" />
        <Number name="Height" size="16" endian="little" signed="false" />
        <Number name="Sort_flag" size="8" value="F7" valueType="hex"/>
        <String name="Color_res" length="3" value="7" />
        <Number name="Background" size="8" endian="little" signed="false" />
        <Number name="pixel_aspect_ration" size="8" endian="little" signed="false" />
    </Block>
    <Block name="COLOR_MAP">
        <Blob name="Color0" length="64" valueType="hex"  />
    </Block>
    <Block name="
        <String na
    </Block>

    ...

</DataModel>
```

beer.gif

00000006	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
00000006	47	49	46	38	39	61	00	05	55	03	f7	00	00	00	00	00	GIF89a..U.÷.....
00000010	ff	ff	ff	fc	fc	fa	f8	c7	c8	4f	08	0b	71	09	10	90	ÿÿÿüüúøÇÈO..q..
00000020	0b	17	f8	38	48	e8	38	48	f7	48	58	d1	6b	74	e5	7e	..ø8Hè8H÷HXÑktå~
00000030	87	b4	17	29	b3	28	37	d1	48	57	51	1c	22	f8	58	68	‡´.)³(7ÑHWQ."øXh

http://pastebin.com/9Y2yENqG