Fuzzing Tools

Jenny Kang

High-level overview

A pretty slide deck that does a good job of explaining browser fuzzing approaches on a high level

Another slide deck on DOM fuzzing

Peach

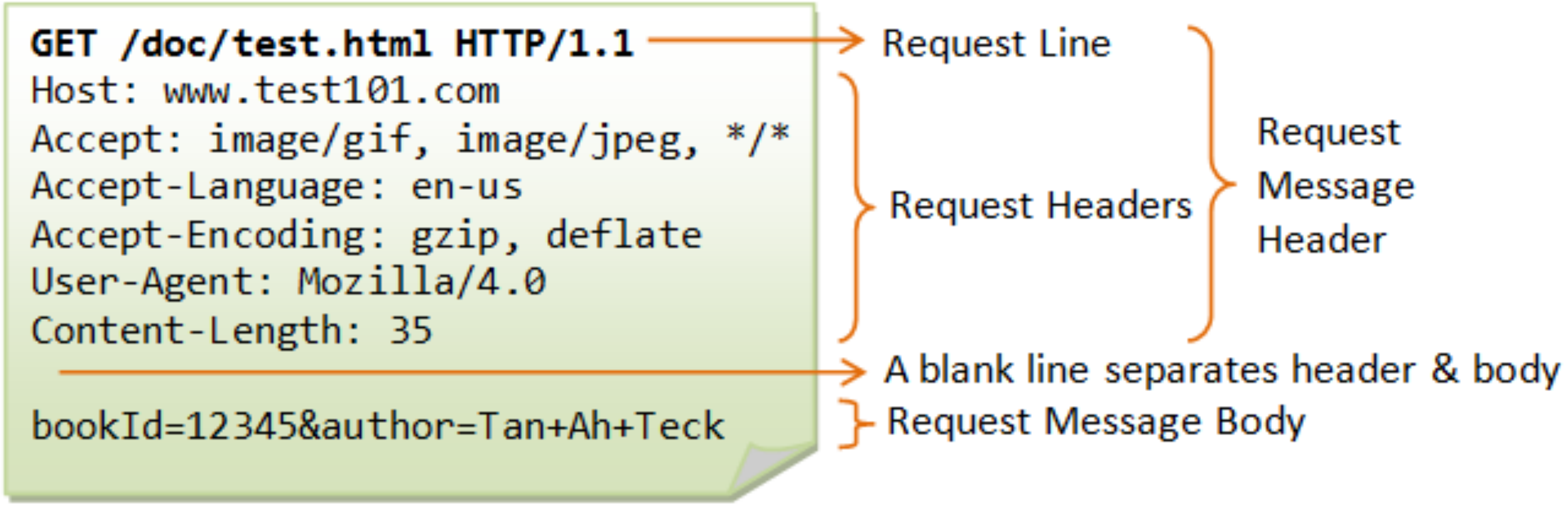
(Windows version)

PeachPit

- is an XML file
  - describes the data type/relationship (Data Model)
  - describes the strategy for fuzzing (State Model)
  - specifies the test environment (publishers, agents, loggers, etc.)
    - sets the target we’d like to fuzz

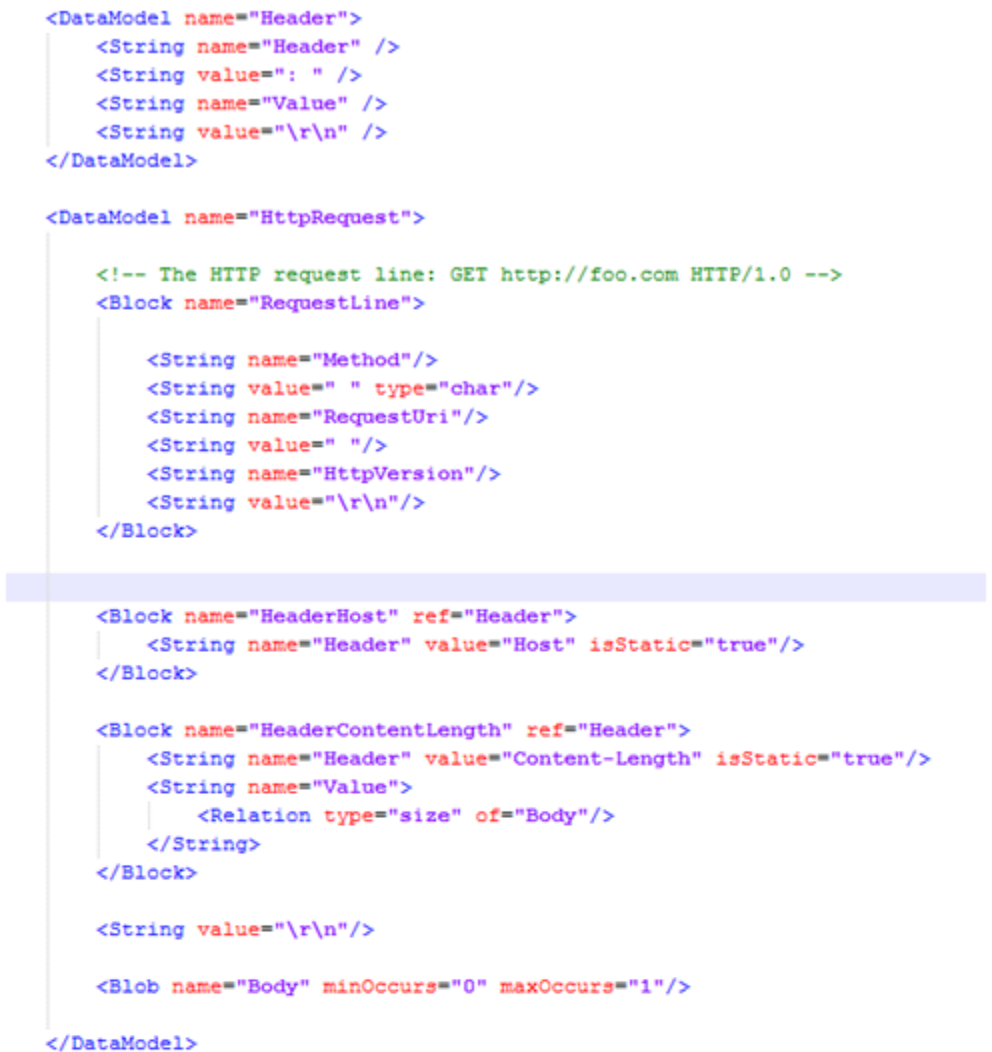
PeachPit Data Models

- PeachPit contains **Data Model(s)** to describe the structure of the data used in fuzzing
    - to be reused when generating new test inputs
    - Can further be split into blocks
    - defines structure of data, including child elements



```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

Source



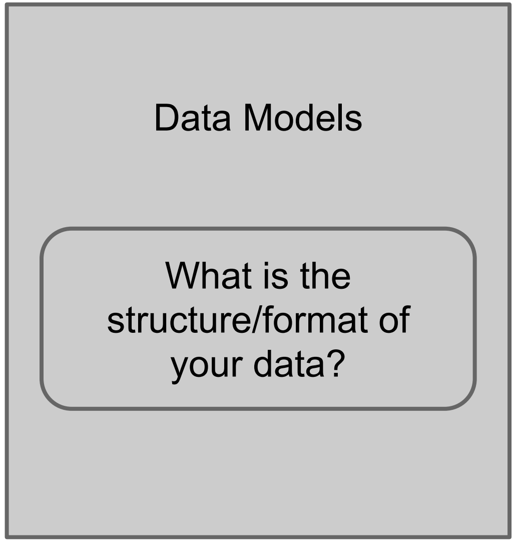
```
<DataModel name="Header">
    <String name="Header" />
    <String value=": " />
    <String name="Value" />
    <String value="\r\n" />
</DataModel>

<DataModel name="HttpRequest">

    <!-- The HTTP request line: GET http://foo.com HTTP/1.0 -->
    <Block name="RequestLine">

        <String name="Method"/>
        <String value=" " type="char"/>
        <String name="RequestUri"/>
        <String value=" "/>
        <String name="HttpVersion"/>
        <String value="\r\n"/>
    </Block>


    <Block name="HeaderHost" ref="Header">
        <String name="Header" value="Host" isStatic="true"/>
    </Block>

    <Block name="HeaderContentLength" ref="Header">
        <String name="Header" value="Content-Length" isStatic="true"/>
        <String name="Value">
            <Relation type="size" of="Body"/>
        </String>
    </Block>

    <String value="\r\n"/>

    <Blob name="Body" minOccurs="0" maxOccurs="1"/>

</DataModel>
```

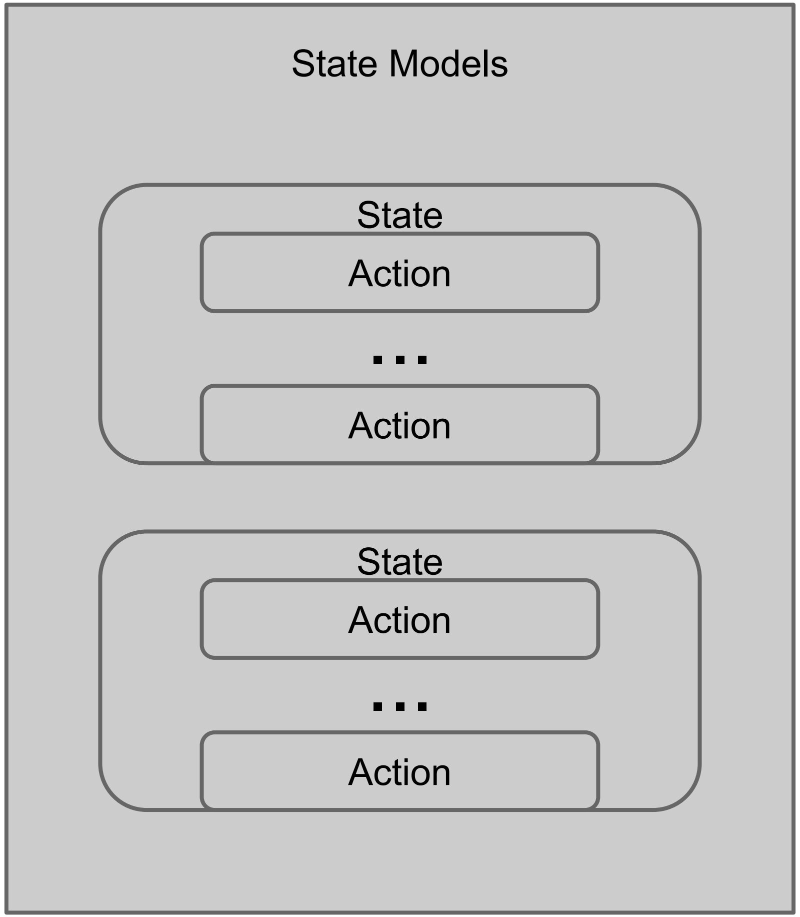
PeachPit State Models

- PeachPit contains **State Model(s)**
  - <State> is a building block consisting of <Actions>
    - at least one state (ex. an ‘initial state’) and one model
  - <Action> actually performs some action such as sending a request or reading data
    - <Data> child element of <Action> can specify default dataset to use in model

```
<DataModel name="TestTemplate">
        <String name="TheString" value="Hello World!" />
</DataModel>

<StateModel name="State" initialState="Initial">
        <State name="Initial">
                <!-- <Action type="connect" /> -->

                <Action name="SendValue1" type="output">
                        <DataModel ref="TestTemplate" />
                </Action>

                <Action name="SendValue2" type="output">
                        <DataModel ref="TestTemplate" />
                </Action>

                <!-- <Action type="close" /> -->
        </State>
</StateModel>
```

To Review...


Data Models
What is the structure/format of your data?


State Models
State
Action
...
Action
State
Action
...
Action

Other components of PeachPit

- **Agents** -- host local or remote **Monitors**, which are responsible for monitoring what's going on while fuzzing (i.e. logging crashes)
- **Publishers** -- think of them as I/O managers responsible for sending/receiving data.
    - <Action> in a state model sends commands to the publisher
- **Test Block** -- Configuration for a test case such as specifying agents, publishers, loggers, etc.
- **Run Block** -- deprecated in Peach 3?

```
<StateModel name="TheStateModel" initialState="TheState">
        <State name="TheState">
                <Action type="output">
                        <DataModel ref="HttpRequest" />
                </Action>
        </State>
</StateModel>

<!-- Agents that run localy will be started automatically by Peach -->
<Agent name="RemoteAgent" location="tcp://192.168.1.190:9001">
        <Monitor name="Debugger" class="WindowsDebugger">
                <Param name="CommandLine" value="CrashableServer.exe 192.168.1.190 4242"/>
        </Monitor>

        <Monitor name="Network" class="PcapMonitor">
                <Param name="filter" value="tcp"/>
        </Monitor>
</Agent>

<Test name="Default">
        <Agent ref="RemoteAgent" />
        <StateModel ref="TheStateModel"/>

        <Publisher class="TcpClient">
                <Param name="Host" value="192.168.1.190" />
                <Param name="Port" value="4242" />
        </Publisher>

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

A couple commands...

- To validate an xml file:
  - `C:/peach/peach.exe -t` *`<some xml file>`*
  - or from the peach directory:
    - `peach -t` *`<some xml file>`*
  - -t flat parses the .xml file

- To run:
  - `peach -1 --debug` *`<some xml file>`*
  - runs one iteration with debug enabled

If you see...

Peach.Core.PeachException: Error, could not load platform assembly 'Peach.Core.OS.Windows.dll'. The assembly is part of the Internet Security Zone and loading has been blocked.

then do this...

Find that .dll file in your peach directory

-> right click and open Properties

-> Under the “General” tab, go to the bottom where it says “This file came from another computer…” and click “Unblock”, then “Apply”

see this and this for more info

Versions!

aka “I copied the tutorial but why does nothing work....”

“Cracking Data”

“The process of interpreting valid data according to a provided DataModel is what Peach calls "cracking" data.”

-- Mozilla Wiki Tutorial

Random woff.xml Demo notes

- This demo used Peach 3.1.124 on Windows 7
- Taken from wiki.mozilla tutorial
  - Note: Mozilla firefox tutorial DOES NOT WORK out of the box for Peach 3.
- WOFF file format is Web Open Font Format
  - you can get a ttf font format from fontsquirrel.com and then use a ttf->woff converter
  - you'll need a "starter file" to feed to your PeachPit
- Read spec carefully! (ex. size = bits; length = bytes)

Websockets.xml demo

What are websockets?

- persistent connection between web browser and server

Note!!: Websockets are just an EXAMPLE here of how to use Peach fuzzer with Firefox. Be open to other uses of Peach Fuzz!!

Websockets.xml Demo

- run from peach-3.1.53\samples directory
- Add ‘WinDbgPath’ to Monitor
- Change path names (for samples_png dir for instance to full dir path)
- Change path to point to your firefox executable
- More info on using Websockets Publisher for browser fuzzing

```
<Agent name="TheAgent">
        <Monitor class="WindowsDebugger">
                <Param name="CommandLine" value="C:\mozilla-central\obj-i686-pc-mingw32\dist\bin\firefox.exe peach_ws_client.html"/>
                <Param name="WinDbgPath" value="C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64\" />
        </Monitor>
</Agent>

<Test name="Default">
        <Agent ref="TheAgent"/>
        <StateModel ref="TheState"/>

        <Publisher class="WebSocket">
                <Param name="Port" value="8080"/>
                <Param name="Template" value="peach_ws_template.html"/>
                <Param name="Publish" value="base64"/>
        </Publisher>
</Test>
```

Where’s my firefox executable?

- **If you’ve downloaded the mozilla-source, navigate to that directory and then go to:**
  - **Windows:** obj-.../dist/bin/firefox.exe
  - **Linux:** obj-.../dist/bin/firefox
  - **OS X:** obj-.../dist/Nightly.
    app/Contents/MacOS/firefox

Other cool Peach tools

Check out the PeachFuzzBang and PeachValidator tools in the peach directory!

A word of encouragement….

Moar resources

- More info on using Websockets Publisher for browser fuzzing
- black hat presentation on mozilla bug hunting
- fuzzing w/ Peach tutorial (uses older version but lists some good tools you can try
- A nice walkthrough of discovering an exploit using Peach Fuzz for a webserver
- A Tutorial using Peach to exploit a vulnerable server (useful to see how Peach is used). And another one
- HotFuzz and Peach overview
- gVim is a nice GUI Vim editor for windows

Memory Inspection Tools

Valgrind, Address Sanitizer, rr

Nicholas Shahan

November 20, 2014

Using a VM?

- Enable code profiling on the CPU.

Your VM software might have an option for this.

- VMware does.

Valgrind Remember me?

- Memory access errors
- Using uninitialized values
- Double-free or mismatched malloc/new/new [] versus free/delete/delete[]
- Overlapping src and dst pointers
- Memory leaks.

When Building Firefox

- Add to mozconfig file：
```
--disable-jemalloc
--enable-valgrind
```

- When running valgrind use the flags:
```
--smc-check=all-non-file --vex-iropt-register-
updates=allregs-at-mem-access
```

Address Sanitizer (ASan)

- Memory error detector
- Looks for:
  - Use-after-free bugs
  - Out-of-bound bugs
- Requires the Clang compiler
- Mozilla has pre-built versions of Firefox for download.

What does Address Sanitizer do?

- Replaces the malloc and free functions
- The memory around malloc-ed regions is poisoned.
- The free-ed memory is placed is also poisoned.

Memory access is transformed by the compiler:

Before:

```
*address = ...;  // or: ... = *address;
```

After:

```
if (IsPoisoned(address)) {
  ReportError(address, kAccessSize, kIsWrite);
}
*address = ...;  // or: ... = *address;
```

Running Firefox & Address Sanitizer

- Download a build from Mozilla
  (or build your own with Clang)
- Run the executable
- Can run in GDB also
  - `break __asan_report_error` or
  - `break AsanDie`
- All errors are fatal, meaning it will only report the first error.

rr

“rr records nondeterministic executions and debugs them deterministically”

NOTE - 32bit only!

Record, Replay, and Debug

- Record a Firefox Session

```
$> rr record <firefox executable>
```

- Replay the Recording

```
$> rr replay
```

Resources

Building Firefox

https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/Build_Instructions

Valgrind

https://developer.mozilla.org/en-US/docs/Mozilla/Testing/Valgrind

Address Sanitizer

https://developer.mozilla.org/en-US/docs/Mozilla/Testing/Firefox_and_Address_Sanitizer

https://code.google.com/p/address-sanitizer/wiki/AddressSanitizer

rr

http://rr-project.org/

https://github.com/mozilla/rr