О фаззинге подробно и со вкусом

привет Зеро Найтс -- 19.20.11.2012
атте кеттунен & мяузаебись
Багс we found in 2012

<table>
<thead>
<tr>
<th>Location</th>
<th>Chrome:</th>
<th>Firefox:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Оулу, Улуруу</td>
<td>25</td>
<td>8</td>
</tr>
<tr>
<td>Хельсинки, Хельсинки</td>
<td>50</td>
<td>5</td>
</tr>
</tbody>
</table>

Багс we found in 2012.
Easy to get started

Enough bugs for novices to find a proper one before they lose interest
First bug report gets very encouraging response from cevans and mozilla

- miaubiz started after aki helin presentation on radamsa at t2 '10
- Atte Kettunen started after joining OUSPG in the summer of 2011
АдрессСанитазизер

- он охуенный
- Clang compiler plugin
- Similar to Valgrind
- Very fast (2x slowdown)
- Originally made by Chromium devs
- Came out May 2011
- Firefox now supported quite well
- Linux & OSX
ERROR: AddressSanitizer heap-buffer-overflow on address 0x1ab53c4c at pc 0x9eaf2ec bp 0xbff9a808 sp 0xbff9a804
READ of size 1 at 0x1ab53c4c thread T0
  #0 0x9eaf2eb (Chromium Framework+0x8d3f2eb)
  #1 0x9f9b89e (Chromium Framework+0x8e2b89e)
  #2 0x9f9dc24 (Chromium Framework+0x8e2dc24)
==79269== ERROR: AddressSanitizer heap-buffer-overflow on address 0x1ab1bc4c at pc 0x9e792ec bp 0xbfffd27e8 sp 0xbfffd27e4 READ of size 1 at 0x1ab1bc4c thread T0
  #0 0x9e792eb in SkA1_Blitter::blitH(int, int, int) (in Chromium Framework) + 539
  #1 0x9f6589e in sk_fill_path(SkPath const&, SkIRect const*, SkBlitter*, int, int, int, SkRegion const&) (in Chromium Framework) + 3182
0x1ab1bc4c is located 0 bytes to the right of 1637388-byte region [0x1a9c040, 0x1ab1bc4c) allocated by thread T0 here:

#0 0x1fbbb in __asan::ASAN_OnSIGSEGV (int, __siginfo*, void*) (in Chromium Helper) + 123

#1 0x93b9954a in malloc_zone_malloc (in libc.dylib) + 74

#2 0x93b99f86 in malloc (in libc.dylib) + 52
==2978== ERROR: AddressSanitizer unknown-crash on address 0x8000e033f080 at pc 0x555555f2a4310 bp 0x7fffffffff7550 sp 0x7fffffffff7308
READ of size 1 at 0x8000e033f080 thread T0
  #0 0x555555f2a430f in __interceptor_memcpypy ???:0
  #1 0x7fffe95934c6 in ?? ???:0

==2978== AddressSanitizer CHECK failed:
/usr/local/google/chrome/src/third_party/llvm/projects/compiler-rt/lib/asan/asan_report.cc:136 "((0 && "Address is not in memory and not in shadow?"))) != (0)"
(0x0, 0x0)
  #0 0x555555f2a923e in __sanitizer::CheckFailed(char const*, int, char const*, unsigned long long, unsigned long long) ???:0
  #1 0x555555f2a83a9 in __asan::

ERROR: AddressSanitizer heap-use-after-free on address 0x7fffffff7ecbfa0 at pc 0x5555559bf1131 bp 0x7fffffff7950 sp 0x7fffffff7948
WRITE of size 8 at 0x7fffffff7ecbfa0 thread T0
  #0 0x5555559bf1130 in WebCore::BaseMultipleFieldsDateAndTimeInputType::~BaseMultipleFieldsDateAndTimeInputType()
  #1 0x5555559bfd95d in WebCore::DateInputType::~DateInputType()
  #2 0x555555995cc6b in WebCore::HTMLInputElement::updateType()
0x7ffffff7ecbfa0 is located 96 bytes inside of 184-byte region [0x7ffffff7ecbf40, 0x7ffffff7ecbff8) freed by thread T0 here:

  #0 0x55555fade730 in operator delete (void*) ???:0

  #1 0x555555589c18f5 in WebCore::ContainerNode::removeAllChildren() ???:0

  #2 0x5555555a19387 in WebCore::InputType::destroyShadowSubtree() ???:0

  #3 0x5555555a487bd in WebCore::
previously allocated by thread T0 here:
  #0 0x55555fade5b0 in operator new (unsigned long) ???:0
  #1 0x555559baf27d in WebCore::SpinBoxElement::create(WebCore::Document*, WebCore::SpinBoxElement::SpinBoxOwner&) ???:0
  #2 0x555559bf1a5d in WebCore::BaseMultipleFieldsDateAndTimeInputType::createShadowSubtree() ???:0
  #3 0x55555995ccc4 in WebCore::HTMLInputElement::updateType() ???:0
  #4 0x555559960466 in WebCore::HTMLInputElement::parseAttribute(WebCore::Attribute const&) ???:0
  #5 0x555558a75803 in WebCore::Element::attributeChanged(WebCore::QualifiedName const&, WTF::AtomicString const&) ???:0
• Makes this all possible
• Awesome with use-after-free
• Very good for buffer оверфлоу / out of bounds access
• Good on type confusion
• Annoying on wild pointer
  (unknown 0xffffffffffebe38a68 @ pc 0x7fff7ad9c58)
If you like sysadmining..

Fuzzing is a great way to justify your hobby of configuring boxen!

miaubiz: 2x 3930k, 2700k, 3770k, 112 gigs of ram, tons of ssds <3

attekett: 2600k, 2x 1055T, 6x dual-core opterons, and more on the way
Follow the browser developers

- Follow the evolution of tools
- Follow new features that are added
- Follow build environments
- Follow testing methods

Not only to find more bugs, but to keep your environment in a working state.
Where the bugs are

- Юс аfter фри, invalid cast
  - DOM
  - Rendering
  - CSS
- Баффер оверфлоу
  - Media formats
  - Canvas (skia)
- Интежер оверфлоу
  - WebGL
SOME FUNNY 2012 BUGS HAHA

- wk 86531 / ff 789046 - bit flipping in gif
- CVE-2012-2806 - oob write in libjpeg-turbo
- CVE-2012-2896 - integer overflow in SafeAdd() and SafeMultiply()
- crbug 143761 - vulnerable code had just been rewritten to fix previous SVG bug
dumb фаззинг

- бит флипинг still works in 2012
- mashup repros from old bugs together
- radamsa \o/
- feed files to браузер as fast possible...
  ...and still identify winning inputs
smarter fuzzing

1. generate inputs based on something
2. process inputs
3. погладь кота
4. погладь кота, сука
5. hope to reproduce
6. hope to minimize
smartish fuzzing: ВебКит Rendering

- find a bug
- write a script that will randomly find that same bug
- wait for more bugs
2 repros 1 bug

```html
<video>
  <source src="r"
  type='video/mp4; type=' />
</video>

<meta http-equiv="X-WebKit-CSP" content="img-src * .b">
<img src="a">
<html>
<head>
<script>
    var gl = document.createElement("canvas").getContext('experimental-webgl');
    var texture = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(gl.TEXTURE_2D, 0, gl.RGBA, 256, 256, 0, gl.RGBA, gl.UNSIGNED_BYTE, null);
    gl.texSubImage2D(gl.TEXTURE_2D, 0, 0, 0x7fffffff00, 256, 256, gl.RGBA, gl.UNSIGNED_BYTE, new Uint8Array(256 * 256 * 4));
</script>
</head>
</html>
smartish fuzzing: Canvas

- take W3C specification
- group together
  - methods
  - attributes
  - properties
- replace input values with `getRandomValue()`
- written by Aki Helin at OUSPG
- see t2 '10 presentation
- Binary(flips, copy-paste)
- String(format-detection, more copy-paste)
- Колмогоров-Смирнов it just works
for (1..500)
    stuff = random_element_of(weird_stuff)
    stash.push(stuff())

while(x = stash.pop())
    eval(x)

dump with:
    print(x)
NodeFuzz

- Modules
  - e.g. canvas, gif, css
- Samples
  - 20+ filetypes supported by browsers
- Injection into browser via websocket connected to node.js server
reproducibility tips

- use asan
- don't reference global state
  - body.children[5].appendChild(body.children[7])
  - impossible to minimize
- if possible, group stuff
Q: How do you know your fuzzer is working?

A: If it looks like what you'd expect

I tried to fuzz <path>, but I get white boxes
  ● wrong namespace for SVG elements
  ● [] instead of () in function call
Instead of random strings I get 'undefined'

minimizing test cases

manually in text editor

```
$ while true; do
  inotifywait repro.html
  && browser repro.html
done
```
$ gzip -c /bin/bash > sample.gz

$ while true
do
  radamsa sample.gz > fuzzed.gz
  gzip -dc fuzzed.gz > /dev/null
  test $? -gt 127 && break
done

(http://code.google.com/p/ouspg/wiki/Radamsa)
git, rsync, redis, 2>&1

- evolve the infrastructure
- automate as much as possible
- rsync results to master node
- repros on filesystem for easy manipulation
- redis keeps:
  - metadata
  - input queues
  - crash logs
2>&1 | grep "inside|left|right|unknown|pc|offset|frame"

check
- page aligned EIP of crash
- offset and size reported (e.g. 8 inside 144)
- top stack frames of crash
- top stack frames of object free/allocate
infrastructure

- git push new fuzzers
- rsync new browser versions
- asan allows multiple browsers to run at once, no need for VMs
- Xephyr leaks memory
- browsers crash native Xorg
- Xvfb works best for many things
- monitor throughput, load, temp..
Inferno is fuzzing the same stuff we are, with 20,000 Google computers. (Firefox too)

Fuzzing is like high frequency arbitrage
microseconds count1

Atte + miaubiz => over 50 dupes in 2012
What if we run out of bugs?

- Манул идет
- Манул идет за тобой
- Browsers are continuously adding features
- Bounties will go up
- We will learn to write exploits
спасибо