CodeTickler: Automated Software Testing as a Service

Cristian Zamfir, Vitaly Chipounov, George Candea
Wouldn’t it be nice to have reliable software?
Vision

• Machines should
  • find corner cases
  • do tricky security testing
  • generate test suites

• Developers (a.k.a. sentient beings) should
  • build cool new software
  • tell machines how software should behave
Outline

• Status quo
• Our approach
• The road ahead
Outline

• Status quo
• Our approach
• The road ahead
Status Quo

• Automate the **running** of tests
  • *do not automate test generation*

• Automatically **finding** bugs
  • **static analysis**
    • false positives
  • **random fuzzing**
    • inefficient at finding corner cases
  • **white-box fuzzing**
    • hard to use, requires models of the environment
Software bugs cost US economy $59B/year
50% of development budgets go to testing

* US National Institute of Standards and Technology
Outline

• Status quo
• Our approach
• The road ahead
Test Report
15 bugs found

SQLLite DB
Library
Details »

Memcached
Application
Details »

Realtek RTL8029
Network device driver
Details »
Technology

No false positives ➔ Higher quality testing

Tests lots of types of binaries ➔ Source code not required

Scales in the cloud ➔ No upfront costs

CodeTickler
autoShift (int rpm)

if (rpm > 1000)
    gear = gear + 1
    rpm = 0.5 * rpm
if (rpm < 700)
    gear = 0
return

rpm ∈ {0, 350, 944, 1200, 1800}
autoShift (int rpm)
    if (rpm > 1000)
        gear = gear + 1
        rpm = 0.5*rpm
    if (rpm < 700)
        gear = 0
    return
autoShift (int rpm)
  if (rpm > 1000)
    gear = gear+1
    rpm = 0.5*rpm
  if (rpm < 700)
    gear=0
  return
Generated tests contain:

- program inputs
- system events
- thread schedules
- faults

Developer: one test
CodeTickler: many tests
in vitro

in vivo

mockup environment

program

real environment

(libraries, operating system, drivers, etc.)

program
*paths* \( \approx 2^{\text{system size}} \)
real environment

mockup environment

program
paths \approx 2^{\text{program size}}
real environment

program
The S²E Platform

Runs unmodified x86 binaries (incl. proprietary/obfuscated/encrypted binaries)

Customized virtual machine

Selection done at runtime

Most code runs “natively”
Summary

• Test any binary program
  • running at any layer of the software stack

• In-vivo analysis
  • no modeling of the environment

• No false positives
  • generates tests that developers can run

• From lab to real world
  • ASPLOS’11, TOCS’12, USENIX’10
Outline

• Status quo
• Our approach
• The road ahead
Which Software Needs Fixing?

Software that runs at the highest privilege level

- **CVE-2013-1287**: USB kernel-mode drivers in Microsoft Windows 8 allow attackers to execute arbitrary code.
- **CVE-2012-2119**: Buffer overflow in the macvtap Linux device driver.
- **CVE-2006-5882**: Stack-based buffer overflow in the Broadcom wireless device driver used in Cisco Linksys.

Third-party software

Third-party testing is increasingly more important. (Veracode)

“Closed-source software is more vulnerable to backdoors” (Bruce Schneier)
Goal

• Get rid of low-level bugs in device drivers
• Target binary of closed-source software
• Open service that everyone can use
• Started building service for device drivers
<table>
<thead>
<tr>
<th>Device driver</th>
<th>Bug type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel 82801AA AC97</td>
<td>Race condition</td>
</tr>
<tr>
<td>Atheros WiFi</td>
<td>Kernel crash (Blue screen of death)</td>
</tr>
<tr>
<td>Broadcom NetLink Gigabit</td>
<td>Kernel crash (Blue screen of death)</td>
</tr>
<tr>
<td>Undisclosed network driver</td>
<td>Kernel crash (Blue screen of death)</td>
</tr>
<tr>
<td>Intel Pro/1000</td>
<td>Kernel crash (Blue screen of death)</td>
</tr>
<tr>
<td>3Com EtherLink Server</td>
<td>Kernel crash (Blue screen of death)</td>
</tr>
<tr>
<td>3Com Ethernet</td>
<td>Kernel crash (Blue screen of death)</td>
</tr>
<tr>
<td>Silicom FastEthernet</td>
<td>Kernel crash (Blue screen of death)</td>
</tr>
<tr>
<td>NDIS subsystem</td>
<td>Kernel crash (Blue screen of death)</td>
</tr>
<tr>
<td>RTL8029</td>
<td>Multiple kernel crashes and resource leaks</td>
</tr>
<tr>
<td>Winbond PCI Ethernet</td>
<td>Concurrency bug</td>
</tr>
<tr>
<td>Linksys Gigabit Ethernet</td>
<td>Incorrect use of API</td>
</tr>
<tr>
<td>Ensoniq AudioPCI</td>
<td>Multiple kernel crashes and resource leaks</td>
</tr>
<tr>
<td>AMD PCNet</td>
<td>Multiple resource leaks</td>
</tr>
</tbody>
</table>
Next Challenges

• Usability
• Scale
• Provenance
• SaaS

We want feedback on these.
Usability

• We want to improve developer productivity

• Challenges

  • *integrate with software development processes*
  
  • *produce easy-to-understand test reports*
  
  • *integrate with IDEs*
Scale

- Re-use test results
  - libraries
  - frameworks
- Big-data problem
  - GBs of data for each device driver
Provenance

How to identify when bad guys are using the service?
SaaS?

How to securely store program binaries?
Conclusion

• CodeTickler
  • *open, automated cloud-based testing service*

• First step
  • *get rid of low-level bugs in device drivers*

• Join us
  • [http://www.codetickler.org](http://www.codetickler.org)

Make software testing as easy as tweeting!