Securing Systems by Threat Mitigation and Adaptive Live Patching

Yue Chen

http://YueChen.me
Outline

• Hack your PC
• Hack your phone
• Hack your server

And how to protect them... and win cash.
Hack your PC

physically
Stole a PC

Screen Locked

Disk Encrypted
Has a Bitcoin wallet inside

with the BTC amount that can buy two pizzas on

May 22, 2010
Cold Boot Attack

Freeze the memory
Cold Boot Attack

Transplant the memory
Cold Boot Attack

Extract the disk decryption key from the memory

Decrypt the disk

Get the Bitcoins
Protect your PC technically
Cold Boot Attack – Protection

• Sensitive memory content in plaintext can be extracted easily
Cold Boot Attack – Protection

- Sensitive memory content in plaintext can be extracted easily
Our Solution – EncExec

- Sensitive memory content cannot be read with encryption

Memory
“XXXXXXXXXXXXX”
Our Solution – EncExec

• Sensitive memory content cannot be read with encryption
EncExec – Overview

• Data in memory **always encrypted**
• **Decrypted** into the cache **only when accessed**
• Use **reserved** cache as a window over protected data
  – Use L3 (instead of L1 or L2) cache to minimize performance impact
EncExec – Overview

• Decrypted data will *never be evicted* to memory (*no cache conflict*)
  – Extend kernel’s virtual memory management to strictly control access
  – Only data in the window are mapped in the address space
  – If more data than window size -> page replacement
Design: Key Techniques

• Spatial cache reservation
  – Reserves a small part of the L3 cache for its use

• Secure in-cache execution
  – Data encrypted in memory, plaintext view only in cache
Intel Core i7 4790 cache architecture
CPU Cache

2-way set-associative cache, 8 cache lines in 4 sets. Each cache line has 16 bytes.
Design: Spatial Cache Reservation

Cache

Set 0  Set 1  Set 2  Set 3

*0  *0  *4  *4  *8  *8  *C  *C

Memory

00  04  08  0C  10  14  18  1C  20  24  28  2C  30  34  38  3C  40  44  48  4C
Design: Spatial Cache Reservation

Cache

Set 0  Set 1  Set 2  Set 3

*0  *0  *4  *4  *8  *8  *C  *C

: Needs to be reserved

Memory

00  04  08  0C  10  14  18  1C  20  24  28  2C  30  34  38  3C  40  44  48  4C
Design: Spatial Cache Reservation

Cache

- Set 0
- Set 1
- Set 2
- Set 3

Memory

- 00
- 04
- 08
- 0C
- 10
- 14
- 18
- 1C
- 20
- 24
- 28
- 2C
- 30
- 34
- 38
- 3C
- 40
- 44
- 48
- 4C

: Needs to be reserved
Design: Spatial Cache Reservation

Cache

Set 0  Set 1  Set 2  Set 3

*: Needs to be reserved

Memory

00 04 08 0C 10 14 18 1C 20 24 28 2C 30 34 38 3C 40 44 48 4C
Design: Secure In-Cache Execution

Desynchronize memory (encrypted) and cache (plaintext)

• Cache in write-back mode
  – Guaranteed by hardware and existing kernels (in most OS’es)

• L3 cache is inclusive of L1 and L2 caches
  – Guaranteed by hardware and existing kernels

• No conflict in the reserved cache
  – No more protected data at a time than the reserved cache size
More data to protect?

• Demand paging
  – Access unmapped data -> page fault
  – Allocate a plaintext page (for securing data)
  – If no page available, select one for replacement
    • Encrypt the plaintext page, copy it back
    • Decrypt faulting page into plaintext, update page table if necessary
Performance Evaluation

Overhead of common cryptographic algorithms

Mode 1: Choose data to encrypt
Mode 2: Encrypt all the data
Test with 15 or 31 plaintext pages
Performance Evaluation

Overhead of RSA and DH handshakes

Mode 1: Choose data to encrypt
Mode 2: Encrypt all the data
Test with 15 or 31 plaintext pages
Protect your phone
Problem

- Dogspectus ransomware reported on April 2016
- It contains the code for the futex or Towelroot exploit that was first disclosed at the end of 2014
Problem

- Ghost Push malware still a major threat in October 2016
- Over 600,000 Android user affected per day
- Affected 14,847 phone types and 3,658 brands
- Known to use VRoot (CVE-2013-6282) and Towelroot (CVE-2014-3153)
Why?
New system software available!
New version: MPIS24.241-2.35-1-13

- Android Security updates.

Click here for more information

All the information on your phone will be saved. You cannot downgrade to a previous software version after installing this update.

To check for updates at any time, press the menu key -> Settings -> About phone -> System updates.

Do you want to download this update?

NO, MAYBE LATER

YES, I'M IN
[Grads] iOS 11.1 released a few minutes ago

Yu Wang

FYI, If you have issue connecting to either CSWLAN or FSUSecure after your iPhone was updated to iOS 11.0.x, Apple just released IOS 11.1 to fix the issue.

Yu Wang

Grads mailing list
Grads@cs.fsu.edu
http://mail.cs.fsu.edu/mailman/listinfo/grads
Exploits made public but **not** reported

“... We are able to identify at least **10** device driver exploits (from a famous root app) that are **never reported** in the public...”

*Android Root and its Providers: A Double-Edged Sword*

*H. Zhang, D. She, and Z. Qian, CCS 2015*
Exploits disclosed but **not** timely patched

Note that this patch was not applied to all msm branches at the time of the patch release (July 2015) and no security bulletin was issued, so the majority of Android kernels based on 3.4 or 3.10 are still affected despite the patch being available for **6 months**.

https://bugs.chromium.org/p/project-zero/issues/detail?id=734
Exploits patched but *delayed* by carriers

It’s each carrier’s job to test all the different updates for all their different smartphones, and they may take many months to do so. They may even *decline* to do the work and *never* release the update.

Monthly disclosed number of Android kernel vulnerabilities
PoC exploits are publicly disclosed

<table>
<thead>
<tr>
<th>Vulnerability/Exploit Name</th>
<th>CVE ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>mempodipper</td>
<td>CVE-2012-0056</td>
</tr>
<tr>
<td>exynos-abuse/Framaroot</td>
<td>CVE-2012-6422</td>
</tr>
<tr>
<td>diagexploit</td>
<td>CVE-2012-4221</td>
</tr>
<tr>
<td>perf_event_exploit</td>
<td>CVE-2013-2094</td>
</tr>
<tr>
<td>fb_mem_exploit</td>
<td>CVE-2013-2596</td>
</tr>
<tr>
<td>msm_acdb_exploit</td>
<td>CVE-2013-2597</td>
</tr>
<tr>
<td>msm_cameraconfig_exploit</td>
<td>CVE-2013-6123</td>
</tr>
<tr>
<td>get/put_user_exploit</td>
<td>CVE-2013-6282</td>
</tr>
<tr>
<td>futex_exploit/Towelroot</td>
<td>CVE-2014-3153</td>
</tr>
<tr>
<td>msm_vfe_read_exploit</td>
<td>CVE-2014-4321</td>
</tr>
<tr>
<td>pipe exploit</td>
<td>CVE-2015-1805</td>
</tr>
<tr>
<td>PingPong exploit</td>
<td>CVE-2015-3636</td>
</tr>
<tr>
<td>f2fs_exploit</td>
<td>CVE-2015-6619</td>
</tr>
<tr>
<td>prctl_vma_exploit</td>
<td>CVE-2015-6640</td>
</tr>
<tr>
<td>keyring_exploit</td>
<td>CVE-2016-0728</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>
iOS More Secure?
<table>
<thead>
<tr>
<th>iOS Version</th>
<th>Release Date</th>
<th>Kernel Vulnerability #</th>
<th>Android # In This Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4.1</td>
<td>8/13/15</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>9/16/15</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>9.1</td>
<td>10/21/15</td>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>9.2</td>
<td>12/8/15</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>9.2.1</td>
<td>1/19/16</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>9.3</td>
<td>3/21/16</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>9.3.2</td>
<td>5/16/16</td>
<td>11</td>
<td>22</td>
</tr>
</tbody>
</table>
So the problem is: *Android has MORE vulnerabilities*

*Vulnerabilities remain UNFIXED over a long time*
Let’s Start from the Kernel

- Apps
- Java API Framework
- Native C/C++ Libraries
- Android Runtime
- Hardware Abstraction Layer
- Linux Kernel
- TrustZone
Let’s Start from the Kernel

- Apps
- Java API Framework
- Native C/C++ Libraries
- Android Runtime
- Hardware Abstraction Layer
- Linux Kernel
- TrustZone
Let’s Start from the Kernel

- Apps
- Java API Framework
- Native C/C++ Libraries
- Android Runtime
- Hardware Abstraction Layer
- Linux Kernel
- TrustZone
Challenges

- *Officially* patching an Android device is a **long** process → **Third-party**

- **Delayed/non-existing** kernel source code → **Binary-based**

![Diagram showing the process of officially patching an Android device](image)
Challenges

- Severely fragmented Android ecosystem → Adaptive
Key requirements:

- **Adaptiveness**
  - It should be adaptive to various device kernels

- **Safety**
  - Patches should be easy to audit
  - Their behaviors must be technically confined

- **Timeliness**
  - Response time should be short, after disclosed vulnerability or exploit

- **Performance**
  - The solution should not incur non-trivial performance overhead
Feasibility Study: Dataset

• Studied **1139** Android kernels

<table>
<thead>
<tr>
<th>Vendor</th>
<th>#Models</th>
<th>#Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samsung</td>
<td>192</td>
<td>419</td>
</tr>
<tr>
<td>Huawei</td>
<td>132</td>
<td>217</td>
</tr>
<tr>
<td>LG</td>
<td>120</td>
<td>239</td>
</tr>
<tr>
<td>Oppo</td>
<td>74</td>
<td>249</td>
</tr>
<tr>
<td>Google Nexus</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>520</strong></td>
<td><strong>1139</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Countries</td>
<td>67</td>
</tr>
<tr>
<td>Carriers</td>
<td>37</td>
</tr>
<tr>
<td>Android Versions</td>
<td>4.2.x, 4.3.x, 4.4.x, 5.0.x, 5.1.x, 6.0.x, 7.0.x</td>
</tr>
<tr>
<td>Kernel Versions</td>
<td>2.6.x, 3.0.x, 3.4.x, 3.10.x, 3.18.x</td>
</tr>
<tr>
<td>Kernel Architectures</td>
<td>ARM (77%), AArch64 (23%)</td>
</tr>
</tbody>
</table>
Feasibility Study: Observations

• Most kernel functions are **stable** across devices and Android releases
• Most vulnerabilities triggered by **malicious inputs**
• Many functions return **error codes**
  – Return a pointer → ERR_PTR
Feasibility Study: Observations

• Most kernel functions are **stable** across devices and Android releases
• Most vulnerabilities triggered by **malicious inputs**
• Many functions return **error codes**
  – Return a pointer → ERR_PTR

Gracefully return

Filter them
Overall Approach: Input Validation

![Diagram showing input validation process with normal and malicious inputs]

- **Arguments**
  - External Inputs
  - Normal control flow

- **Arguments**
  - External Inputs
  - Pwned

- **Arguments**
  - External Inputs
  - Normal control flow

Note: The term "malicious input" indicates the points where input validation is applied to protect against unauthorized access or control.
KARMA: Kernel Adaptive Repair for Many Androids

- **Adaptive** – Automatically adapt to various device kernels
- **Memory-safe** – Protect kernel from malicious (misused) patches
- **Multi-level** – Flexible for different vulnerabilities
KARMA Design: Safety

- Patches are written in Lua, confined by Lua VM at runtime
- A patch can only be placed at designated locations
- Patched functions must return error codes or void
  - Use existing error handling to recover from attacks
- A patch can read but not write the kernel memory
  - Confined by KARMA APIs
  - Prevent malicious (misused) patches from changing the kernel
  - Prevent information leakage
if (requeue_pi) {
  /*
   * Requeue PI only works on two distinct uaddr. This
   * check is only valid for private futexes. See below.
   */
  if (uaddr1 == uaddr2)
    return -EINVAL;
  /*
   * requeue_pi requires a pi_state, try to allocate it now
   * without any locks in case it fails.
   */
KARMA Patch Example

```c
function kpatcher(patchID, sp, cpsr, r0, r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12, r14)
  if patchID == 0xca5269db50f4 then
    uaddr1 = r0
    uaddr2 = r2
    if uaddr1 == uaddr2 then
      return -22
    else
      return 0
    end
  end
kpatch.hook(0xca5269db50f4,"futex_requeue")
```

More complex examples in the paper
## KARMA API

<table>
<thead>
<tr>
<th>API</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>hook</td>
<td>Hook a function for live patching</td>
</tr>
<tr>
<td>subhook</td>
<td>Hook the calls to sub-functions for live patching</td>
</tr>
<tr>
<td>alloc_mem</td>
<td>Allocate memory for live patching</td>
</tr>
<tr>
<td>free_mem</td>
<td>Free the allocated memory for live patching</td>
</tr>
<tr>
<td>get_callee</td>
<td>Locate a callee that can be hooked</td>
</tr>
<tr>
<td>search_symbol</td>
<td>Get the kernel symbol address</td>
</tr>
<tr>
<td>current_thread</td>
<td>Get the current thread context</td>
</tr>
<tr>
<td>read_buf</td>
<td>Read raw bytes from memory with the given size</td>
</tr>
<tr>
<td>read_int_8</td>
<td>Read 8 bits from memory as an integer</td>
</tr>
<tr>
<td>read_int_16</td>
<td>Read 16 bits from memory as an integer</td>
</tr>
<tr>
<td>read_int_32</td>
<td>Read 32 bits from memory as an integer</td>
</tr>
<tr>
<td>read_int_64</td>
<td>Read 64 bits from memory as an integer</td>
</tr>
</tbody>
</table>
## KARMA API

<table>
<thead>
<tr>
<th>API</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>hook</td>
<td>Hook a function for live patching</td>
</tr>
<tr>
<td>subhook</td>
<td>Hook the calls to sub-functions for live patching</td>
</tr>
<tr>
<td>alloc_mem</td>
<td>Allocate memory for live patching</td>
</tr>
<tr>
<td>free_mem</td>
<td>Free the allocated memory for live patching</td>
</tr>
<tr>
<td>get_callee</td>
<td>Locate a callee that can be hooked</td>
</tr>
<tr>
<td>search_symbol</td>
<td>Get the kernel symbol address</td>
</tr>
<tr>
<td>current_thread</td>
<td>Get the current thread context</td>
</tr>
<tr>
<td>read_buf</td>
<td>Read raw bytes from memory with the given size</td>
</tr>
<tr>
<td>read_int_8</td>
<td>Read 8 bits from memory as an integer</td>
</tr>
<tr>
<td>read_int_16</td>
<td>Read 16 bits from memory as an integer</td>
</tr>
<tr>
<td>read_int_32</td>
<td>Read 32 bits from memory as an integer</td>
</tr>
<tr>
<td>read_int_64</td>
<td>Read 64 bits from memory as an integer</td>
</tr>
</tbody>
</table>

Available to patches
KARMA Architecture

Offline Patch Generation and Verification

Vulnerable Function Identification → Semantic Matching → Signed Patches For Target Kernels → Download & Verify Patch → Apply Patch

Online Live Patching by KARMA Client

THE CLOUD
Offline Patch Adaptation
Offline Patch Adaptation

Three steps:

1. **Identify** the vulnerable functions in the target kernel
   - Same function but different names
   - Inlined

2. **Check** if the reference patch works for the target kernel
   - Same function but different semantics

3. **Adapt** the reference patch for the target kernel
Vulnerable Function Identification Example

CVE-2015-3636 (PingPong Root)

**Device A:** ping_unhash

- Func_A
- Func_B
- Func_C
- Func_D
- Func_E

**Device B:** ping_v4_unhash

- Func_A
- Func_B
- Func_C
- Func_D
- Func_E

Call graph based similarity comparison
Semantic Matching

- Check if two functions are semantically equivalent
- If so, adapt the reference patch to the target kernel
- Syntactic matching is too strict
  - Different compilers can generate different code with same semantics
    - Instruction order, register allocation, instruction selection, code layout
Semantic Matching

Same semantics with different syntax
Semantic Matching

• Check if two functions are semantically equivalent
• If so, adapt the reference patch to the target kernel
• Syntactic matching is too strict
  – Different compilers can generate different code with same semantics
    • Instruction order, register allocation, instruction selection, code layout
• Use *symbolic execution* to abstract these differences and adapt patches
  – Use approximation to improve scalability (details in the paper)
Online Patch Application

Function entry point hooking
Prototype Implementation

• Lua engine in kernel (11K SLOC)
  – Simple
  – Memory-safe
  – Easy to embed and extend
  – 24 years of development

• Semantic matching
  – angr
Evaluation: Applicability

- Evaluated **76** critical vulnerabilities in the last three years
### Evaluation: Adaptability

<table>
<thead>
<tr>
<th>Kernel Function</th>
<th>CVE ID</th>
<th># of Opcode Clusters</th>
<th>% of the Largest Opcode Cluster</th>
<th># of Syntax Clusters</th>
<th>% of the Largest Syntax Cluster</th>
<th># of Semantic Clusters</th>
<th>% of Largest Semantic Cluster</th>
<th>Semantic Matching Time Cost</th>
<th># of Instructions</th>
<th># of Basic Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>sock_diag_rcv_msg</td>
<td>2013-1763</td>
<td>35</td>
<td>25.0%</td>
<td>7</td>
<td>73.5%</td>
<td>3</td>
<td>75.5%</td>
<td>10.5s</td>
<td>72</td>
<td>16</td>
</tr>
<tr>
<td>perf_swevent_init</td>
<td>2013-2094</td>
<td>9</td>
<td>55.9%</td>
<td>5</td>
<td>55.9%</td>
<td>2</td>
<td>96.3%</td>
<td>24.6s</td>
<td>81</td>
<td>22</td>
</tr>
<tr>
<td>fb_mmap</td>
<td>2013-2596</td>
<td>26</td>
<td>20.2%</td>
<td>7</td>
<td>44.4%</td>
<td>5</td>
<td>66.9%</td>
<td>12.2s</td>
<td>102</td>
<td>15</td>
</tr>
<tr>
<td>__get_user_1</td>
<td>2013-6282</td>
<td>3</td>
<td>92.4%</td>
<td>2</td>
<td>92.4%</td>
<td>2</td>
<td>98.0%</td>
<td>3.2s</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>futex_requeue</td>
<td>2014-3153</td>
<td>54</td>
<td>14.8%</td>
<td>9</td>
<td>71.0%</td>
<td>3</td>
<td>99.3%</td>
<td>35.8s</td>
<td>459</td>
<td>107</td>
</tr>
<tr>
<td>msm_isp_proc_cmd</td>
<td>2014-4321</td>
<td>42</td>
<td>22.0%</td>
<td>5</td>
<td>66.5%</td>
<td>3</td>
<td>42.8%</td>
<td>8.8s</td>
<td>385</td>
<td>68</td>
</tr>
<tr>
<td>send_write Packing test_read</td>
<td>2014-9878</td>
<td>12</td>
<td>57.6%</td>
<td>4</td>
<td>61.2%</td>
<td>1</td>
<td>100%</td>
<td>4.9s</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>msm_cci_validate_queue</td>
<td>2014-9890</td>
<td>6</td>
<td>59.5%</td>
<td>4</td>
<td>84.9%</td>
<td>2</td>
<td>72.4%</td>
<td>6.7s</td>
<td>77</td>
<td>8</td>
</tr>
<tr>
<td>ping_unhash</td>
<td>2015-3636</td>
<td>36</td>
<td>12.5%</td>
<td>5</td>
<td>75.7%</td>
<td>3</td>
<td>50.5%</td>
<td>4.6s</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>q6lsm_snd_model_buf_alloc</td>
<td>2015-8940</td>
<td>29</td>
<td>34.0%</td>
<td>9</td>
<td>36.6%</td>
<td>5</td>
<td>44.2%</td>
<td>9.9s</td>
<td>104</td>
<td>20</td>
</tr>
<tr>
<td>sys_perf_event_open</td>
<td>2016-0819</td>
<td>22</td>
<td>36.3%</td>
<td>6</td>
<td>46.9%</td>
<td>6</td>
<td>84.2%</td>
<td>34.6s</td>
<td>569</td>
<td>118</td>
</tr>
<tr>
<td>kgsl_ioctl_slpumem_alloc</td>
<td>2016-3842</td>
<td>16</td>
<td>35.4%</td>
<td>3</td>
<td>88.8%</td>
<td>4</td>
<td>46.0%</td>
<td>4.7s</td>
<td>79</td>
<td>11</td>
</tr>
<tr>
<td>is_ashmem_file</td>
<td>2016-5340</td>
<td>6</td>
<td>89.6%</td>
<td>2</td>
<td>93.9%</td>
<td>2</td>
<td>98.1%</td>
<td>0.8s</td>
<td>23</td>
<td>3</td>
</tr>
</tbody>
</table>
### Evaluation: Adaptability

<table>
<thead>
<tr>
<th>Kernel Function</th>
<th>CVE ID</th>
<th># of Opcode Clusters</th>
<th>% of the Largest Opcode Cluster</th>
<th># of Syntax Clusters</th>
<th>% of the Largest Syntax Cluster</th>
<th># of Semantic Clusters</th>
<th>% of Largest Semantic Cluster</th>
<th>Semantic Matching Time Cost</th>
<th># of Instructions</th>
<th># of Basic Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>sock_diag_rcv_msg</td>
<td>2013-1763</td>
<td>35</td>
<td>25.0%</td>
<td>7</td>
<td>73.5%</td>
<td>3</td>
<td>75.5%</td>
<td>10.5s</td>
<td>72</td>
<td>16</td>
</tr>
<tr>
<td>perf_swevent_init</td>
<td>2013-2094</td>
<td>9</td>
<td>55.9%</td>
<td>5</td>
<td>55.9%</td>
<td>2</td>
<td>96.3%</td>
<td>24.6s</td>
<td>81</td>
<td>22</td>
</tr>
<tr>
<td>fb_mmap</td>
<td>2013-2596</td>
<td>26</td>
<td>20.2%</td>
<td>7</td>
<td>44.4%</td>
<td>5</td>
<td>66.9%</td>
<td>12.2s</td>
<td>102</td>
<td>15</td>
</tr>
<tr>
<td>__get_user_1</td>
<td>2013-6282</td>
<td>3</td>
<td>92.4%</td>
<td>2</td>
<td>92.4%</td>
<td>2</td>
<td>98.0%</td>
<td>3.2s</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>futex_requeue</td>
<td>2014-3153</td>
<td>54</td>
<td>14.8%</td>
<td>9</td>
<td>71.0%</td>
<td>3</td>
<td>99.3%</td>
<td>35.8s</td>
<td>459</td>
<td>107</td>
</tr>
<tr>
<td>msm_isp_proc_cmd</td>
<td>2014-4321</td>
<td>42</td>
<td>22.0%</td>
<td>5</td>
<td>66.5%</td>
<td>3</td>
<td>42.8%</td>
<td>8.8s</td>
<td>385</td>
<td>68</td>
</tr>
<tr>
<td>send_write_packing_test_read</td>
<td>2014-9878</td>
<td>12</td>
<td>57.6%</td>
<td>4</td>
<td>61.2%</td>
<td>1</td>
<td>100%</td>
<td>4.9s</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>msm_cci_validate_queue</td>
<td>2014-9890</td>
<td>6</td>
<td>59.5%</td>
<td>4</td>
<td>84.9%</td>
<td>2</td>
<td>72.4%</td>
<td>6.7s</td>
<td>77</td>
<td>8</td>
</tr>
<tr>
<td>ping_unhash</td>
<td>2015-3636</td>
<td>36</td>
<td>12.5%</td>
<td>5</td>
<td>75.7%</td>
<td>3</td>
<td>50.5%</td>
<td>4.6s</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>q61sm_snd_model_buf_alloc</td>
<td>2015-8940</td>
<td>29</td>
<td>34.0%</td>
<td>9</td>
<td>36.6%</td>
<td>5</td>
<td>44.2%</td>
<td>9.9s</td>
<td>104</td>
<td>20</td>
</tr>
<tr>
<td>sys_perf_event_open</td>
<td>2016-0819</td>
<td>22</td>
<td>36.3%</td>
<td>6</td>
<td>46.9%</td>
<td>6</td>
<td>84.2%</td>
<td>34.6s</td>
<td>569</td>
<td>118</td>
</tr>
<tr>
<td>kgsl_ioctl_spumem_alloc</td>
<td>2016-3842</td>
<td>16</td>
<td>35.4%</td>
<td>3</td>
<td>88.8%</td>
<td>4</td>
<td>46.0%</td>
<td>4.7s</td>
<td>79</td>
<td>11</td>
</tr>
<tr>
<td>is_ashmem_file</td>
<td>2016-5340</td>
<td>6</td>
<td>89.6%</td>
<td>2</td>
<td>93.9%</td>
<td>2</td>
<td>98.1%</td>
<td>0.8s</td>
<td>23</td>
<td>3</td>
</tr>
</tbody>
</table>

**Types and frequencies of instruction opcodes**
### Evaluation: Adaptability

<table>
<thead>
<tr>
<th>Kernel Function</th>
<th>CVE ID</th>
<th># of Opcode Clusters</th>
<th>% of the Largest Opcode Cluster</th>
<th># of Syntax Clusters</th>
<th>% of the Largest Syntax Cluster</th>
<th># of Semantic Clusters</th>
<th>% of Largest Semantic Cluster</th>
<th>Semantic Matching Time Cost</th>
<th># of Instructions</th>
<th># of Basic Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>sock_diag_rcv_msg</td>
<td>2013-1763</td>
<td>35</td>
<td>25.0%</td>
<td>7</td>
<td>73.5%</td>
<td>3</td>
<td>75.5%</td>
<td>10.5s</td>
<td>72</td>
<td>16</td>
</tr>
<tr>
<td>perf_swevent_init</td>
<td>2013-2094</td>
<td>9</td>
<td>55.9%</td>
<td>5</td>
<td>55.9%</td>
<td>2</td>
<td>96.3%</td>
<td>24.6s</td>
<td>81</td>
<td>22</td>
</tr>
<tr>
<td>fb_mmap</td>
<td>2013-2596</td>
<td>26</td>
<td>20.2%</td>
<td>7</td>
<td>44.4%</td>
<td>5</td>
<td>66.9%</td>
<td>12.2s</td>
<td>102</td>
<td>15</td>
</tr>
<tr>
<td>__get_user_1</td>
<td>2013-6282</td>
<td>3</td>
<td>92.4%</td>
<td>2</td>
<td>92.4%</td>
<td>2</td>
<td>98.0%</td>
<td>3.2s</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>futex_requeue</td>
<td>2014-3153</td>
<td>54</td>
<td>14.8%</td>
<td>9</td>
<td>71.0%</td>
<td>3</td>
<td>99.3%</td>
<td>35.8s</td>
<td>459</td>
<td>107</td>
</tr>
<tr>
<td>msm_isp_proc_cmd</td>
<td>2014-4321</td>
<td>42</td>
<td>22.0%</td>
<td>5</td>
<td>66.5%</td>
<td>3</td>
<td>42.8%</td>
<td>8.8s</td>
<td>385</td>
<td>68</td>
</tr>
<tr>
<td>send_write Packing_test_read</td>
<td>2014-9878</td>
<td>12</td>
<td>57.6%</td>
<td>4</td>
<td>61.2%</td>
<td>1</td>
<td>100%</td>
<td>4.9s</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>msm_cci_validate_queue</td>
<td>2014-9890</td>
<td>6</td>
<td>59.5%</td>
<td>4</td>
<td>84.9%</td>
<td>2</td>
<td>72.4%</td>
<td>6.7s</td>
<td>77</td>
<td>8</td>
</tr>
<tr>
<td>ping_unhash</td>
<td>2015-3636</td>
<td>36</td>
<td>12.5%</td>
<td>5</td>
<td>75.7%</td>
<td>3</td>
<td>50.5%</td>
<td>4.6s</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>q61sm_snd_model_buf_alloc</td>
<td>2015-8940</td>
<td>29</td>
<td>34.0%</td>
<td>9</td>
<td>36.6%</td>
<td>5</td>
<td>44.2%</td>
<td>9.9s</td>
<td>104</td>
<td>20</td>
</tr>
<tr>
<td>sys_perf_event_open</td>
<td>2016-0819</td>
<td>22</td>
<td>36.3%</td>
<td>6</td>
<td>46.9%</td>
<td>6</td>
<td>84.2%</td>
<td>34.6s</td>
<td>569</td>
<td>118</td>
</tr>
<tr>
<td>kgs1_ioctl_gpumem_alloc</td>
<td>2016-3842</td>
<td>16</td>
<td>35.4%</td>
<td>3</td>
<td>88.8%</td>
<td>4</td>
<td>46.0%</td>
<td>4.7s</td>
<td>79</td>
<td>11</td>
</tr>
<tr>
<td>is_ashmem_file</td>
<td>2016-5340</td>
<td>6</td>
<td>89.6%</td>
<td>2</td>
<td>93.9%</td>
<td>2</td>
<td>98.1%</td>
<td>0.8s</td>
<td>23</td>
<td>3</td>
</tr>
</tbody>
</table>

Number of function calls and conditional branches (to abstract CFG)
Evaluation: Adaptability

<table>
<thead>
<tr>
<th>Kernel Function</th>
<th>CVE ID</th>
<th># of Opcode Clusters</th>
<th>% of the Largest Opcode Cluster</th>
<th># of Syntax Clusters</th>
<th>% of the Largest Syntax Cluster</th>
<th># of Semantic Clusters</th>
<th>% of the Largest Semantic Cluster</th>
<th>Semantic Matching Time Cost</th>
<th># of Instructions</th>
<th># of Basic Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>sock_diag_recv_msg</td>
<td>2013-1763</td>
<td>35</td>
<td>25.0%</td>
<td>7</td>
<td>73.5%</td>
<td>3</td>
<td>75.5%</td>
<td>10.5s</td>
<td>72</td>
<td>16</td>
</tr>
<tr>
<td>perf_swevent_init</td>
<td>2013-2094</td>
<td>9</td>
<td>55.9%</td>
<td>5</td>
<td>55.9%</td>
<td>2</td>
<td>96.3%</td>
<td>24.6s</td>
<td>81</td>
<td>22</td>
</tr>
<tr>
<td>fb_mmap</td>
<td>2013-2596</td>
<td>26</td>
<td>20.2%</td>
<td>7</td>
<td>44.4%</td>
<td>5</td>
<td>66.9%</td>
<td>12.2s</td>
<td>102</td>
<td>15</td>
</tr>
<tr>
<td>__get_user_l</td>
<td>2013-6282</td>
<td>3</td>
<td>92.4%</td>
<td>2</td>
<td>92.4%</td>
<td>2</td>
<td>98.0%</td>
<td>3.2s</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>futex_requeue</td>
<td>2014-3153</td>
<td>54</td>
<td>14.8%</td>
<td>9</td>
<td>71.0%</td>
<td>3</td>
<td>99.3%</td>
<td>35.8s</td>
<td>459</td>
<td>107</td>
</tr>
<tr>
<td>msm_isp_proc_cmd</td>
<td>2014-4321</td>
<td>42</td>
<td>22.0%</td>
<td>5</td>
<td>66.5%</td>
<td>3</td>
<td>42.8%</td>
<td>8.8s</td>
<td>385</td>
<td>68</td>
</tr>
<tr>
<td>send_write Packing_test_read</td>
<td>2014-9878</td>
<td>12</td>
<td>57.6%</td>
<td>4</td>
<td>61.2%</td>
<td>1</td>
<td>100%</td>
<td>4.9s</td>
<td>25</td>
<td>4</td>
</tr>
<tr>
<td>msm_cci_validate_queue</td>
<td>2014-9890</td>
<td>6</td>
<td>59.5%</td>
<td>4</td>
<td>84.9%</td>
<td>2</td>
<td>72.4%</td>
<td>6.7s</td>
<td>77</td>
<td>8</td>
</tr>
<tr>
<td>ping_unhash</td>
<td>2015-3636</td>
<td>36</td>
<td>12.5%</td>
<td>5</td>
<td>75.7%</td>
<td>3</td>
<td>50.5%</td>
<td>4.6s</td>
<td>54</td>
<td>8</td>
</tr>
<tr>
<td>q61sm_snd_model_buf_alloc</td>
<td>2015-8940</td>
<td>29</td>
<td>34.0%</td>
<td>9</td>
<td>36.6%</td>
<td>5</td>
<td>44.2%</td>
<td>9.9s</td>
<td>104</td>
<td>20</td>
</tr>
<tr>
<td>sys_perf_event_open</td>
<td>2016-0819</td>
<td>22</td>
<td>36.3%</td>
<td>6</td>
<td>46.9%</td>
<td>6</td>
<td>84.2%</td>
<td>34.6s</td>
<td>569</td>
<td>118</td>
</tr>
<tr>
<td>kgs1_ioctl_gpumem_alloc</td>
<td>2016-3842</td>
<td>16</td>
<td>35.4%</td>
<td>3</td>
<td>88.8%</td>
<td>4</td>
<td>46.0%</td>
<td>4.7s</td>
<td>79</td>
<td>11</td>
</tr>
<tr>
<td>is_ashmem_file</td>
<td>2016-5340</td>
<td>6</td>
<td>89.6%</td>
<td>2</td>
<td>93.9%</td>
<td>2</td>
<td>98.1%</td>
<td>0.8s</td>
<td>23</td>
<td>3</td>
</tr>
</tbody>
</table>
Evaluation: Performance

CF-Bench results with different patches
Hack your server remotely
Attackers have limited information
Attack Example: Stack Overflow
Typical Attack Procedure to Take Over the *Whole System*

1. **Find (memory) vulnerabilities**
2. **Exploit them**
3. **Run your own code**
Data Execution Prevention (DEP)

• *Previously*, attackers inject their *own stuff* into the process, and run it
• *Currently*, Data Execution Prevention (DEP) is widely deployed.
• You cannot run what you inject
Code Reuse Attack

Example: Return-Oriented Programming

Existing Code

Chained Gadgets
Protect your server
magically
Code Reuse Attack

• Need to know the code location
  – Guess the code locations (repeatedly)
• Protect?
  – Make the code locations unpredictable
Remix: On-demand Live Randomization
Win cash decently
After 0 successful submissions, Google doubles top reward for hacking a Chromebook to $100,000

EMIL PROTALINSKI @EPRO  MARCH 14, 2016 10:30 AM

Over the past six years, Google has paid security researchers over $6 million (over $2 million last year alone) since launching its bug bounty program in 2010. The company today expanded its Chrome Reward Program with two changes: increasing its top reward for Chromebooks and adding a new bounty.
References

• Protect your PC
  – Secure In-Cache Execution
• Protect your phone
  – Adaptive Android Kernel Live Patching
• Protect your server
  – Remix: On-demand Live Randomization
Thank you

http://YueChen.me