CIS 5935 Introductory Seminar on Research

Securing Systems by Threat Mitigation and Adaptive Live Patching

Yue Chen

http://YueChen.me

Outline

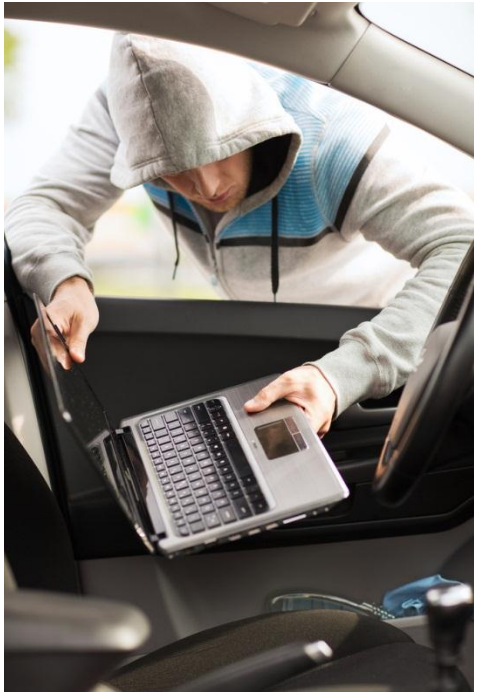
- Hack your PC
- Hack your phone
- Hack your server

And how to protect them…

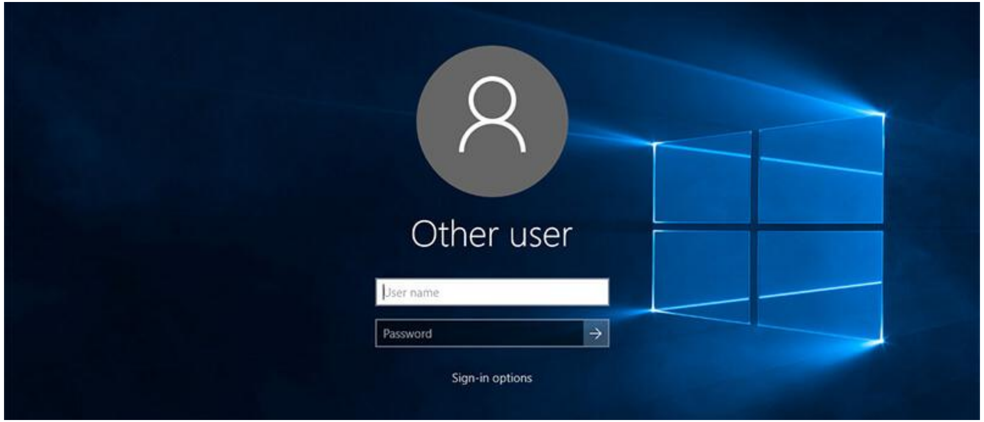
and win cash.

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

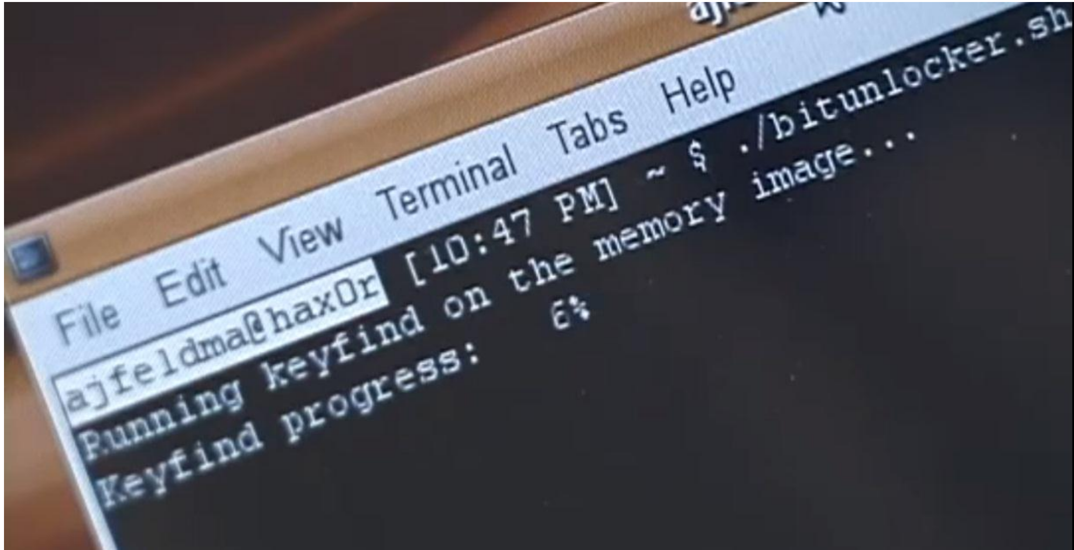
Transplant the memory

Cold Boot Attack

Extract the disk decryption key from the memory

Decrypt the disk

Get the Bitcoins

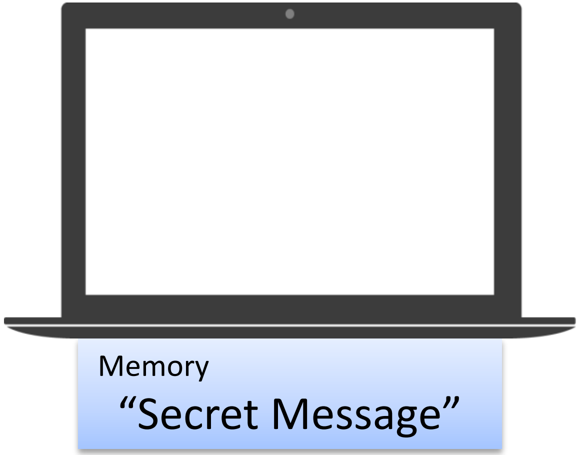


Protect your PC

technically

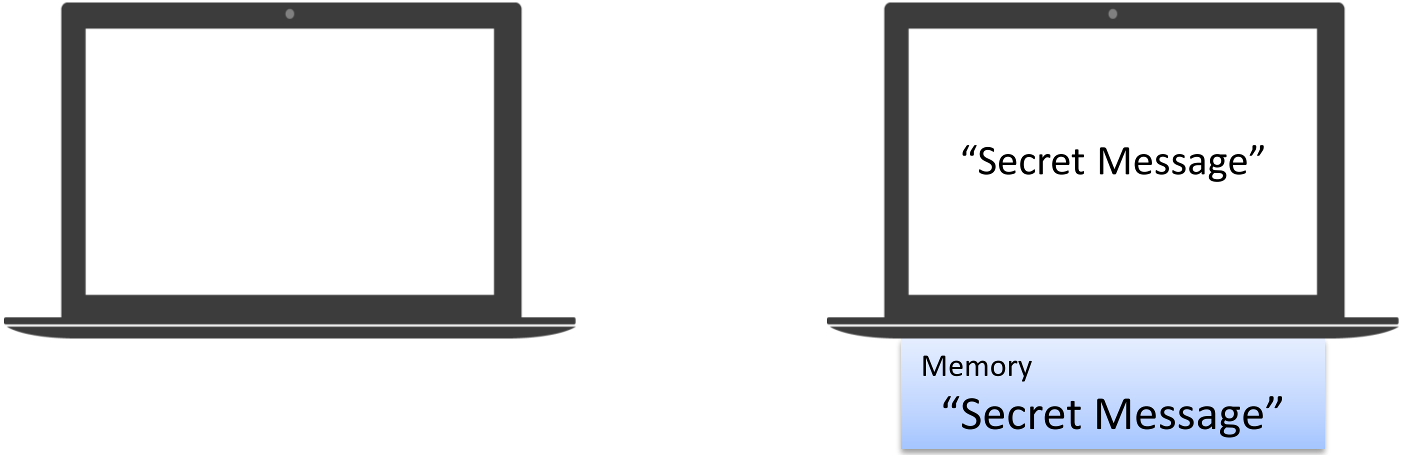
Cold Boot Attack – Protection

- Sensitive memory content in plaintext can be extracted easily



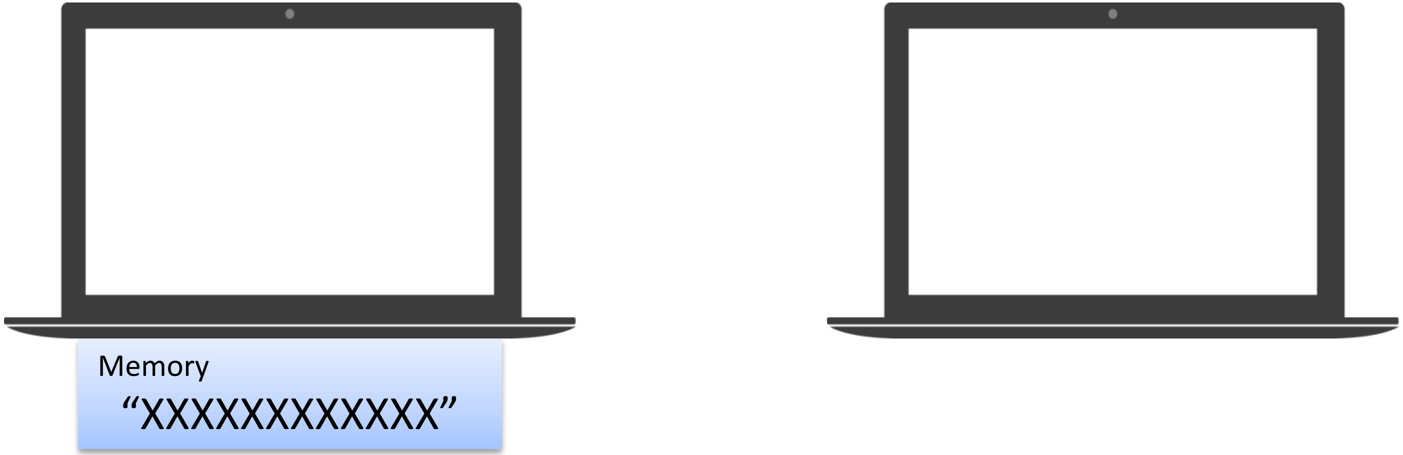
Cold Boot Attack – Protection

- Sensitive memory content in plaintext can be extracted easily



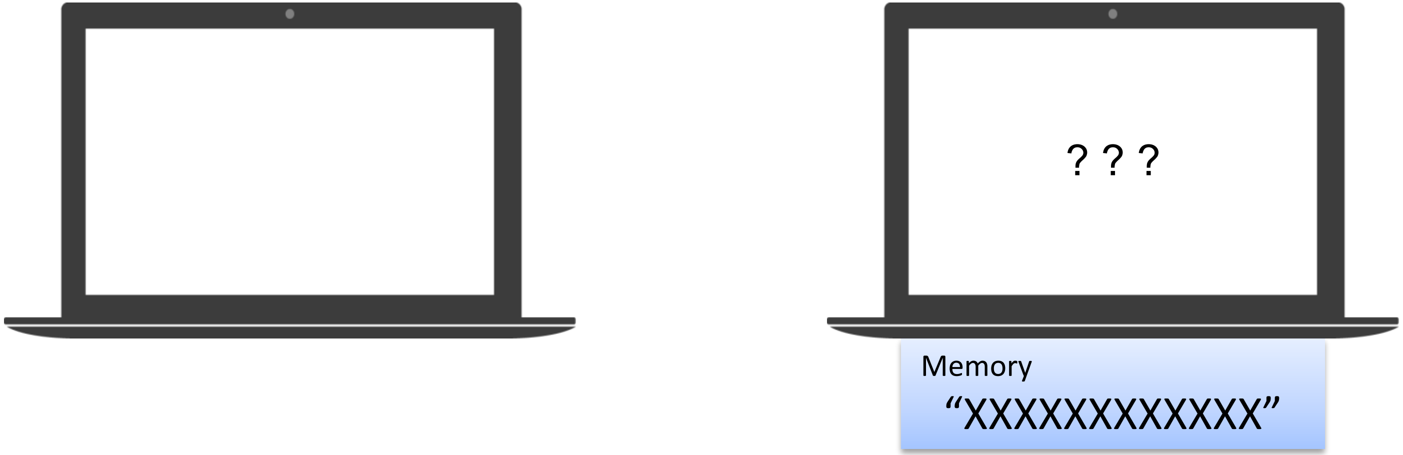
Our Solution – EncExec

- Sensitive memory content cannot be read with encryption



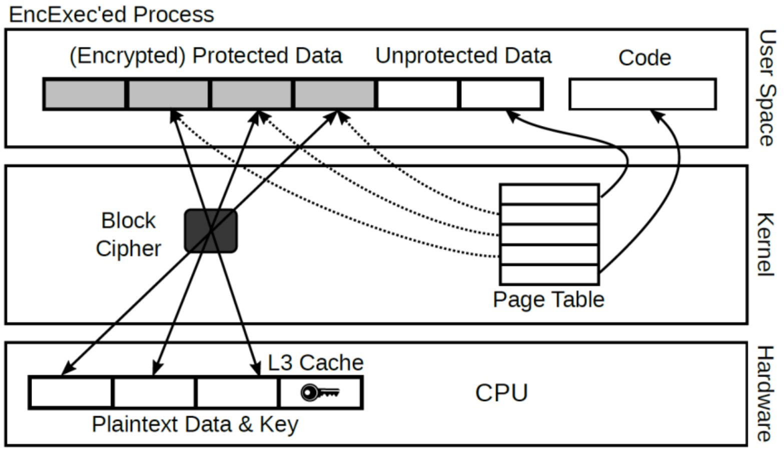
Our Solution – EncExec

- Sensitive memory content cannot be read with encryption



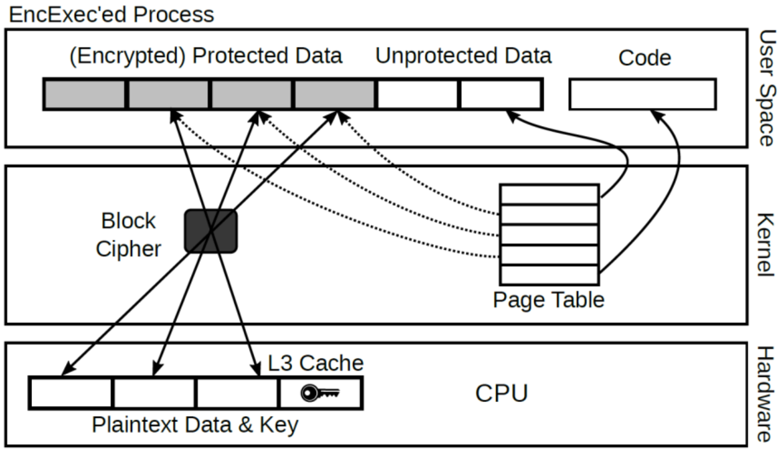
EncExec – Overview

- 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

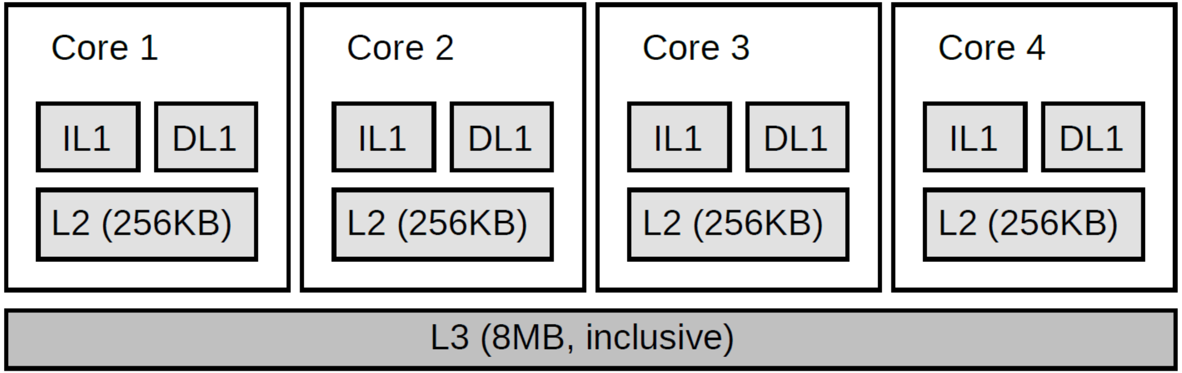
- Decrypted data will *never* be evicted to memory (no cache conflict)
  - Extend kernel's virtual memory management to strictly control access
  - Only data in the window are mapped in the address space
  - If more data than window size -> page replacement



Design: Key Techniques

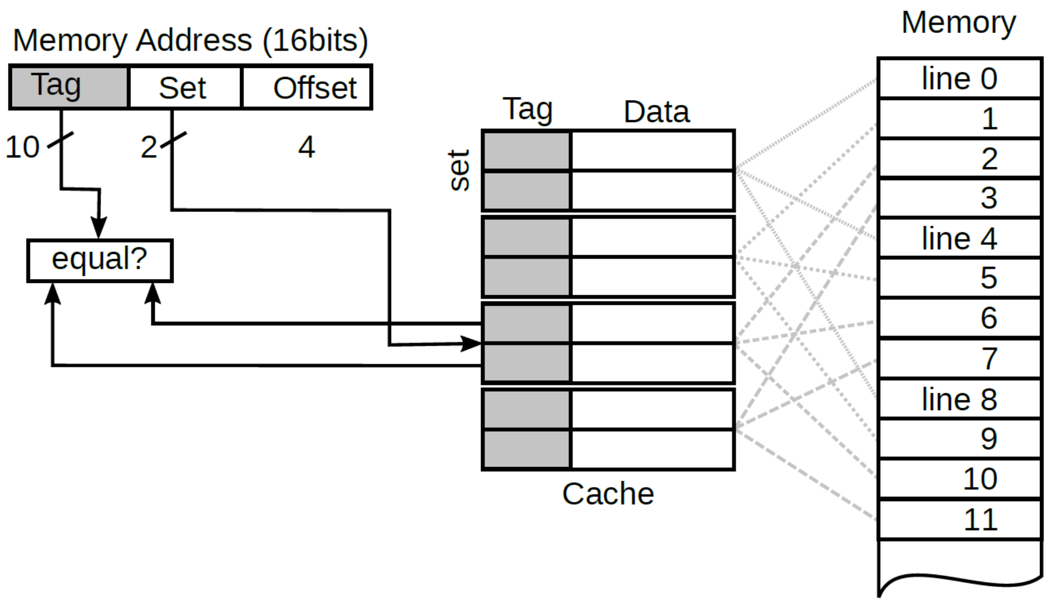
- Spatial cache reservation
  - Reserves a small part of the L3 cache for its use
- Secure in-cache execution
  - Data encrypted in memory, plaintext view only in cache

CPU Cache



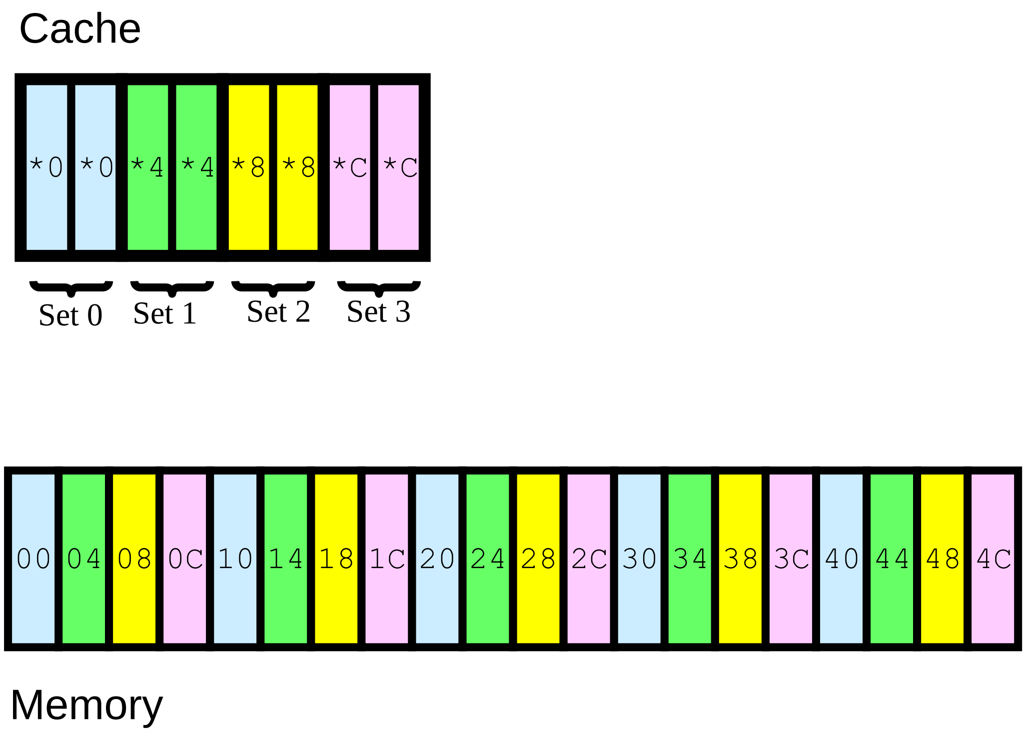
Intel Core i7 4790 cache architecture

CPU Cache

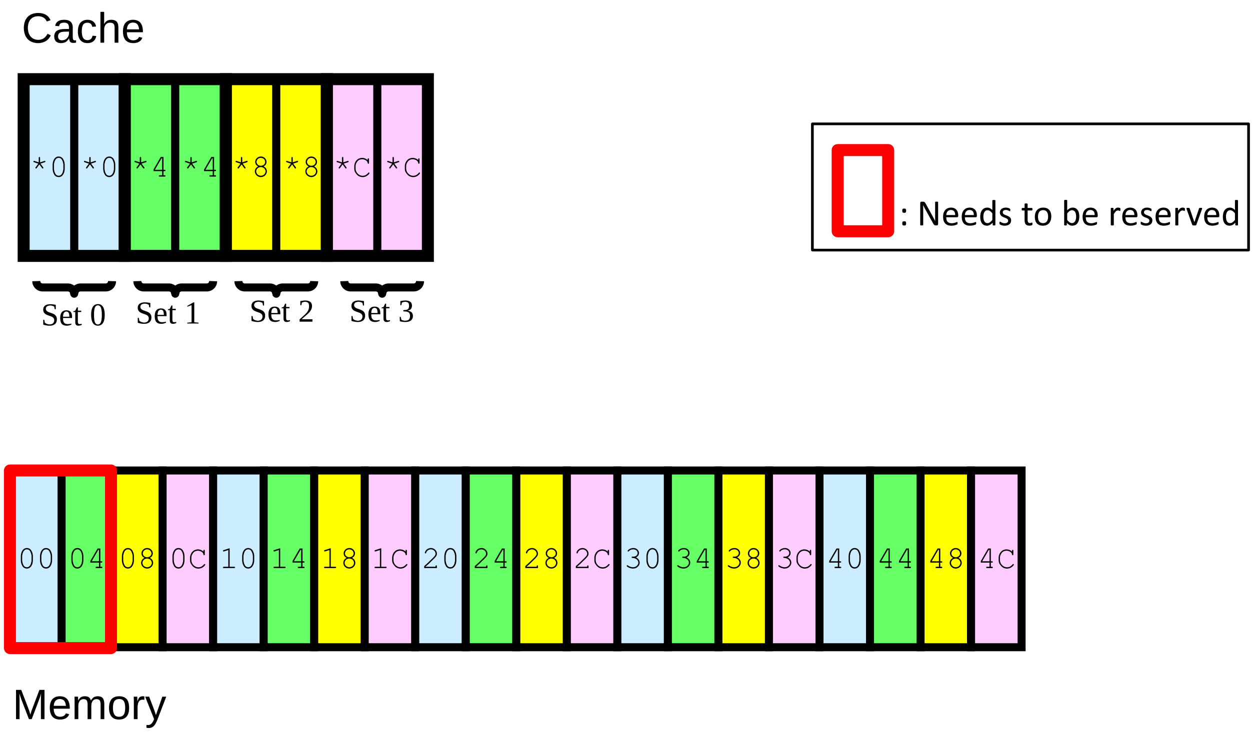


2-way set-associative cache, 8 cache lines in 4 sets. Each cache line has 16 bytes.

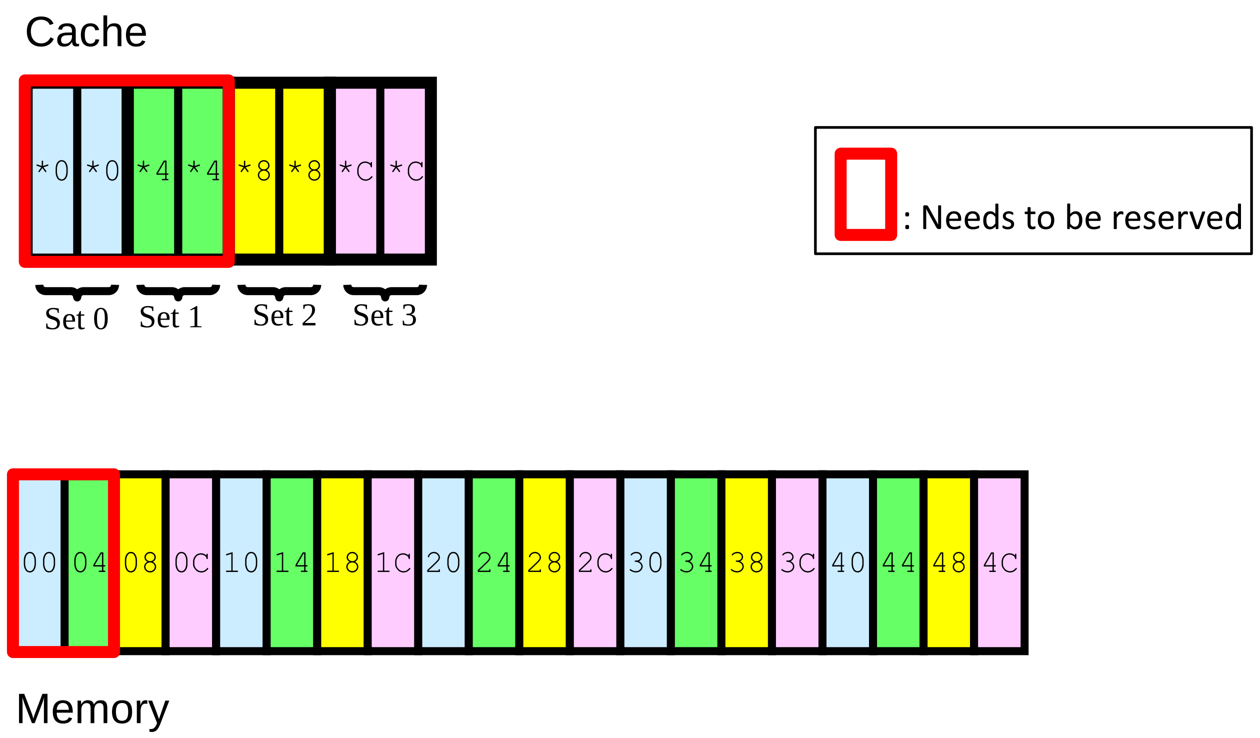
Design: Spatial Cache Reservation



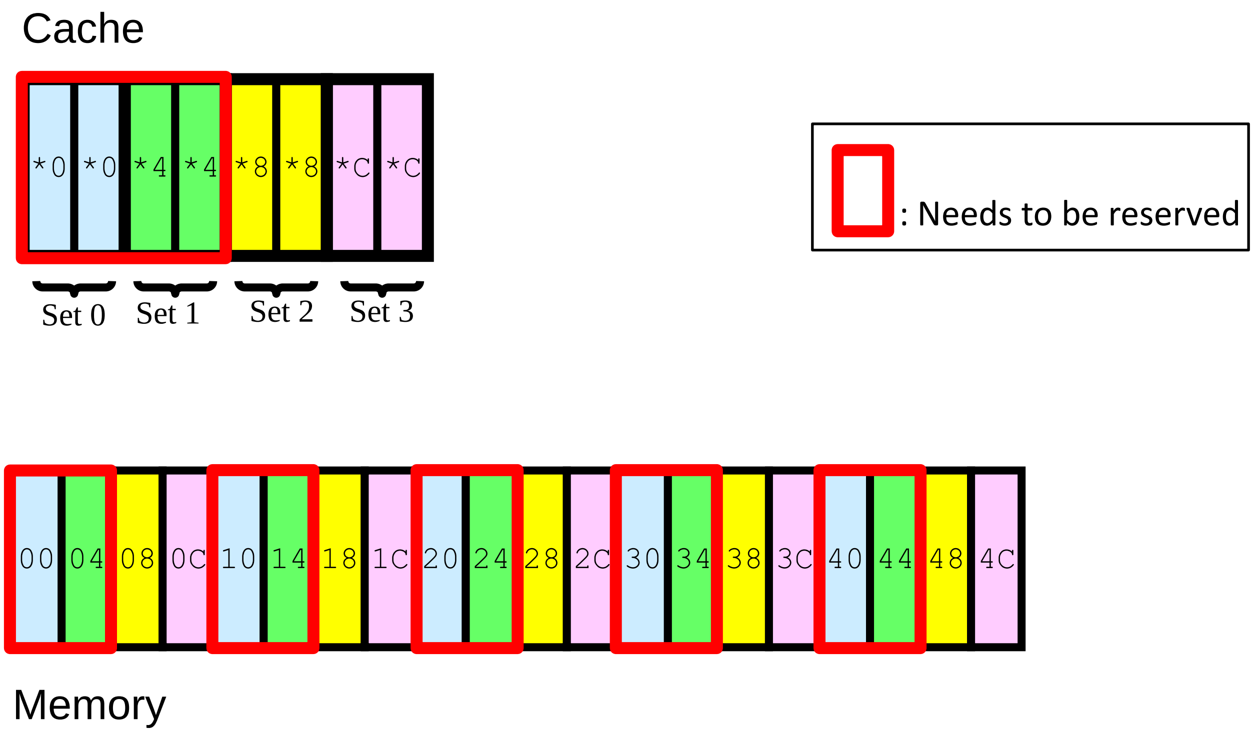
Design: Spatial Cache Reservation



Design: Spatial Cache Reservation



Design: Spatial Cache Reservation



Design: Secure In-Cache Execution

Desynchronize memory (encrypted) and cache (plaintext)

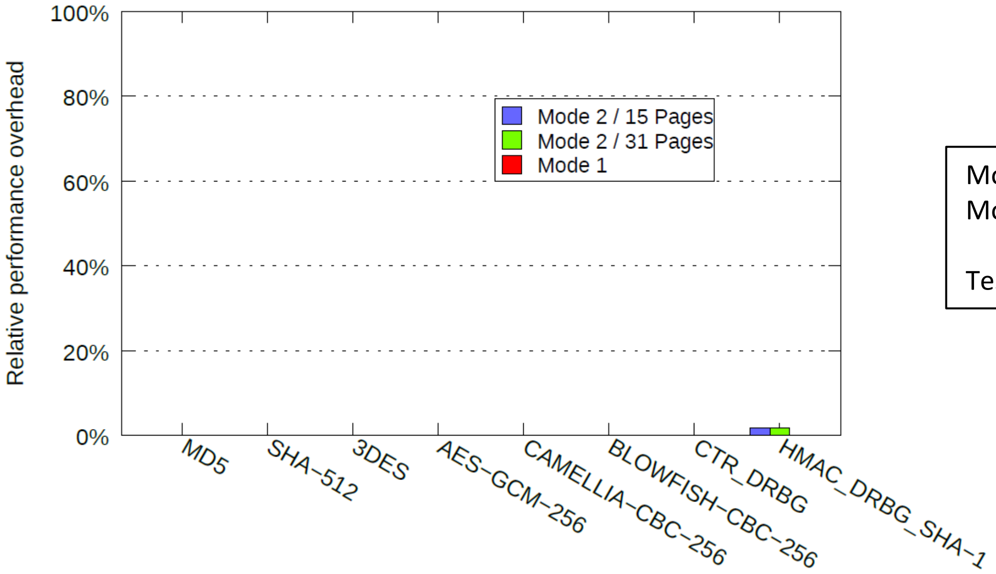
- Cache in write-back mode
  - Guaranteed by hardware and existing kernels (in most OS'es)
- L3 cache is inclusive of L1 and L2 caches
  - Guaranteed by hardware and existing kernels
- No conflict in the reserved cache
  - No more protected data at a time than the reserved cache size

Design: Secure In-Cache Execution

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

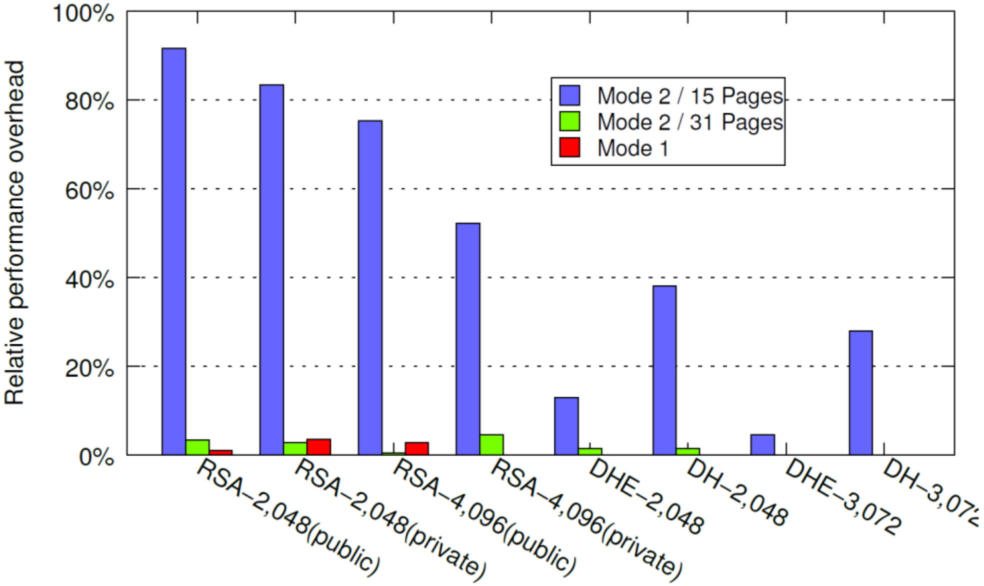


Overhead of common cryptographic algorithms

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

~~Hack~~ 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**

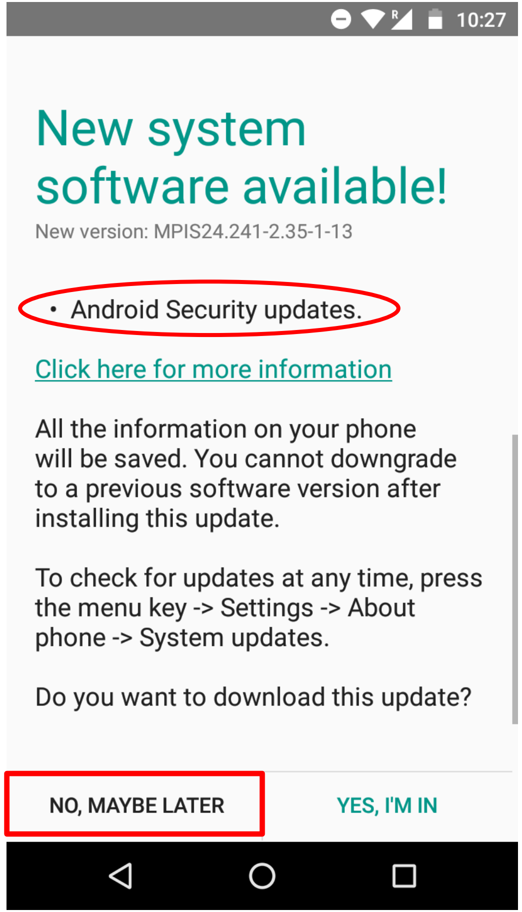
Problem

- 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)

Android Malware: Ghost Push Trojan Still Threatens More Than Half Of Android Devices

17 October 2016, 1:09 pm EDT By Rachel Ranosa Tech Times

Why?


10:27
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

●●○○○ T-Mobile LTE 1:16 PM 100%

[Grads] IOS 11.1 released a few minutes ago Inbox

Yu Wang 11:36 AM

to faculty@cs.fsu.edu, staff@cs.fsu.edu, grads...

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

Reply Reply all Forward

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.

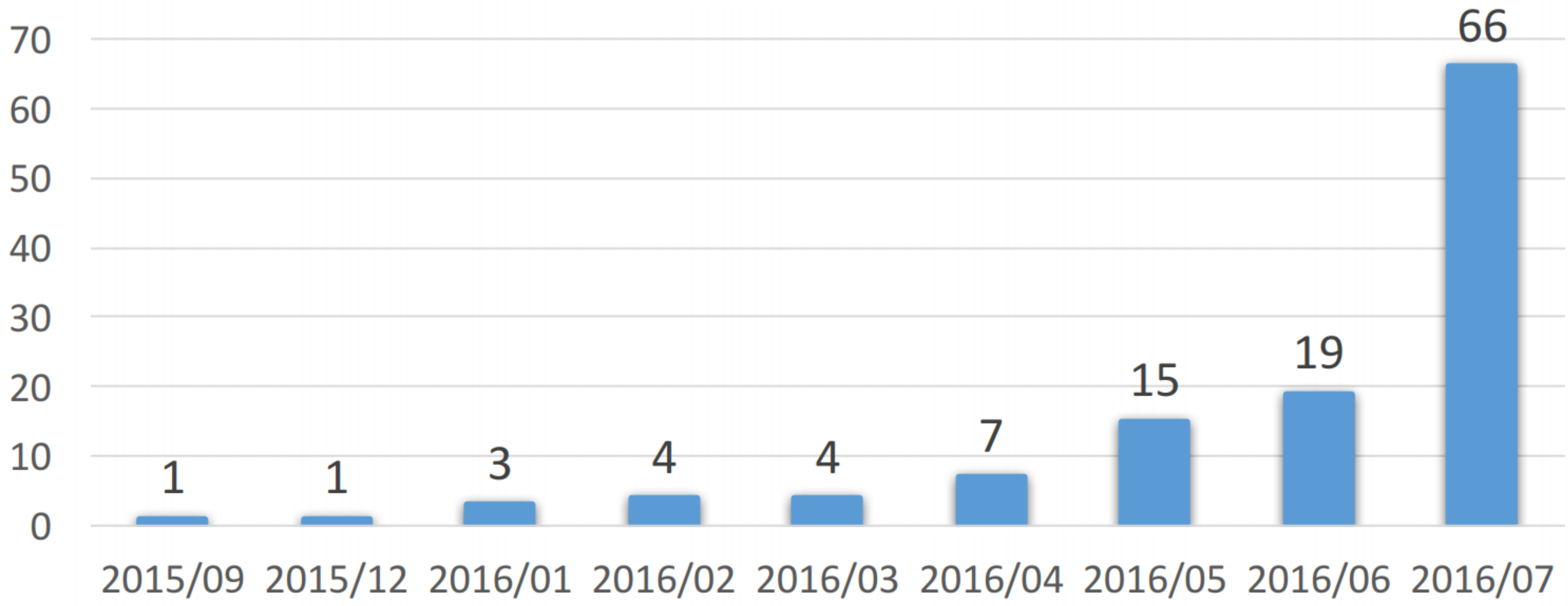
https://bugs.chromium.org/p/project-zero/issues/detail?id=734

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.

https://www.howtogeek.com/163958/why-do-carriers-delay-updates-for-android-but-not-iphone

Monthly disclosed number of Android kernel vulnerabilities



PoC exploits are publicly disclosed

Vulnerability/Exploit Name	CVE ID
mempodipper	CVE-2012-0056
exynos-abuse/Framaroot	CVE-2012-6422
diagexploit	CVE-2012-4221
perf_event_exploit	CVE-2013-2094
fb_mem_exploit	CVE-2013-2596
msm_acdb_exploit	CVE-2013-2597
msm_cameraconfig_exploit	CVE-2013-6123
get/put_user_exploit	CVE-2013-6282
futex_exploit/Towelroot	CVE-2014-3153
msm_vfe_read_exploit	CVE-2014-4321
pipe exploit	CVE-2015-1805
PingPong exploit	CVE-2015-3636
f2fs_exploit	CVE-2015-6619
prctl_vma_exploit	CVE-2015-6640
keyring_exploit	CVE-2016-0728
……	……

iOS More Secure?

?

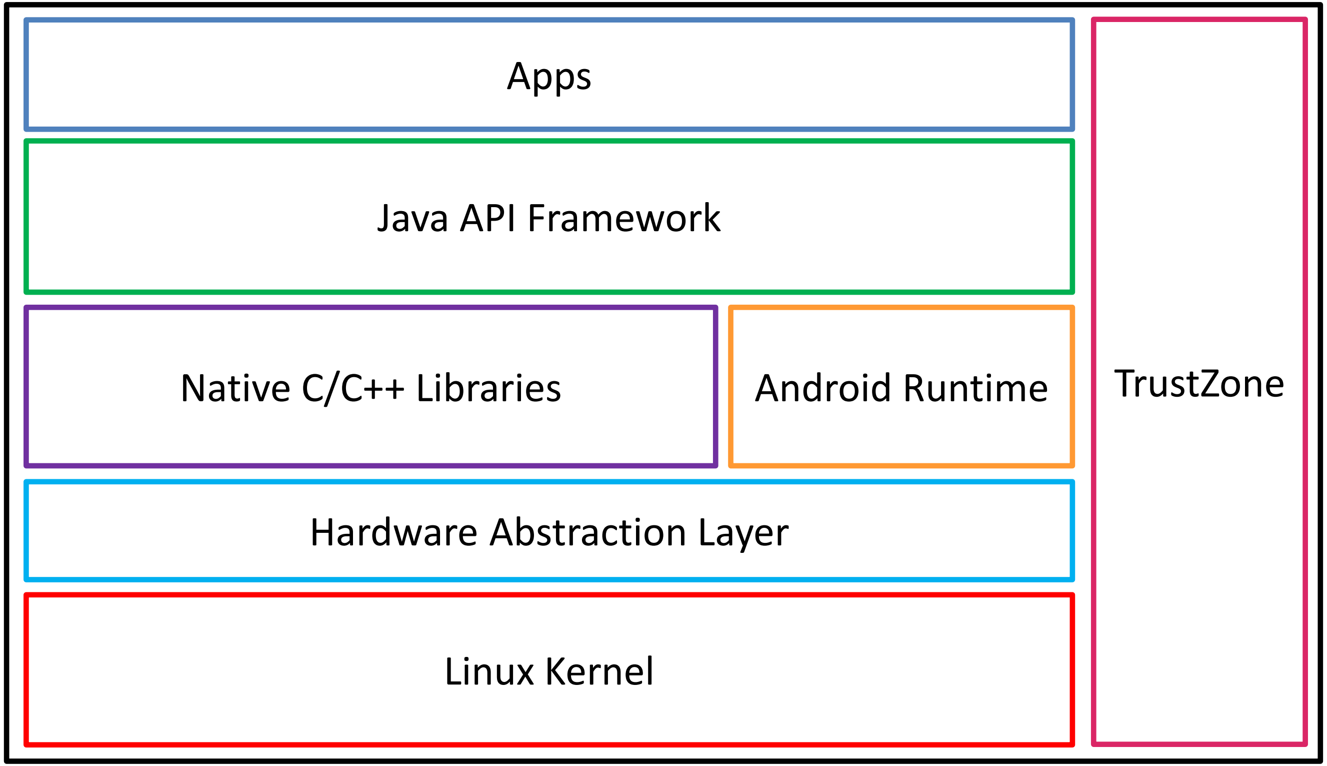
V.S.

iOS Version	Release Date	Kernel Vulnerability #	Android # In This Period
8.4.1	8/13/15	3	-
9	9/16/15	12	1
9.1	10/21/15	6	-
9.2	12/8/15	5	1
9.2.1	1/19/16	4	3
9.3	3/21/16	9	8
9.3.2	5/16/16	11	22

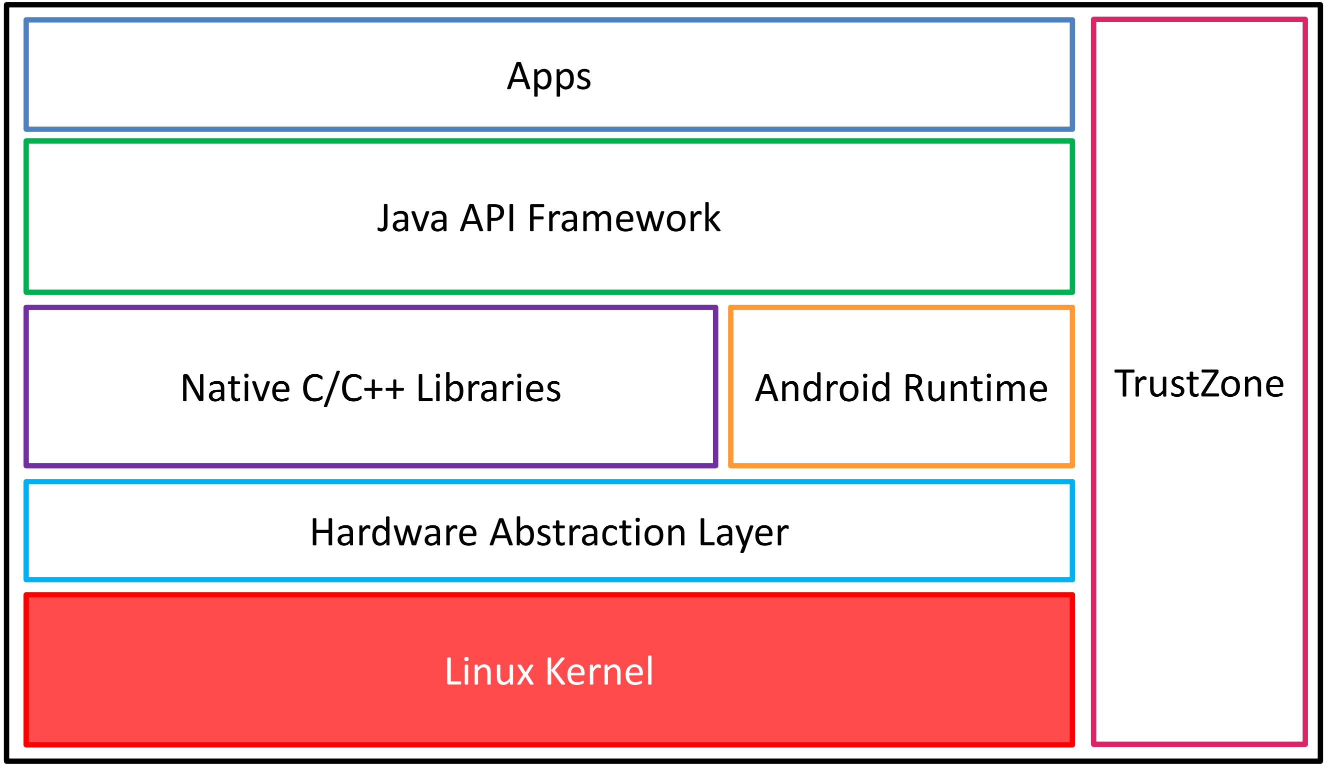
So the problem is: ~~*Android has MORE vulnerabilities*~~

Vulnerabilities remain UNFIXED over a long time

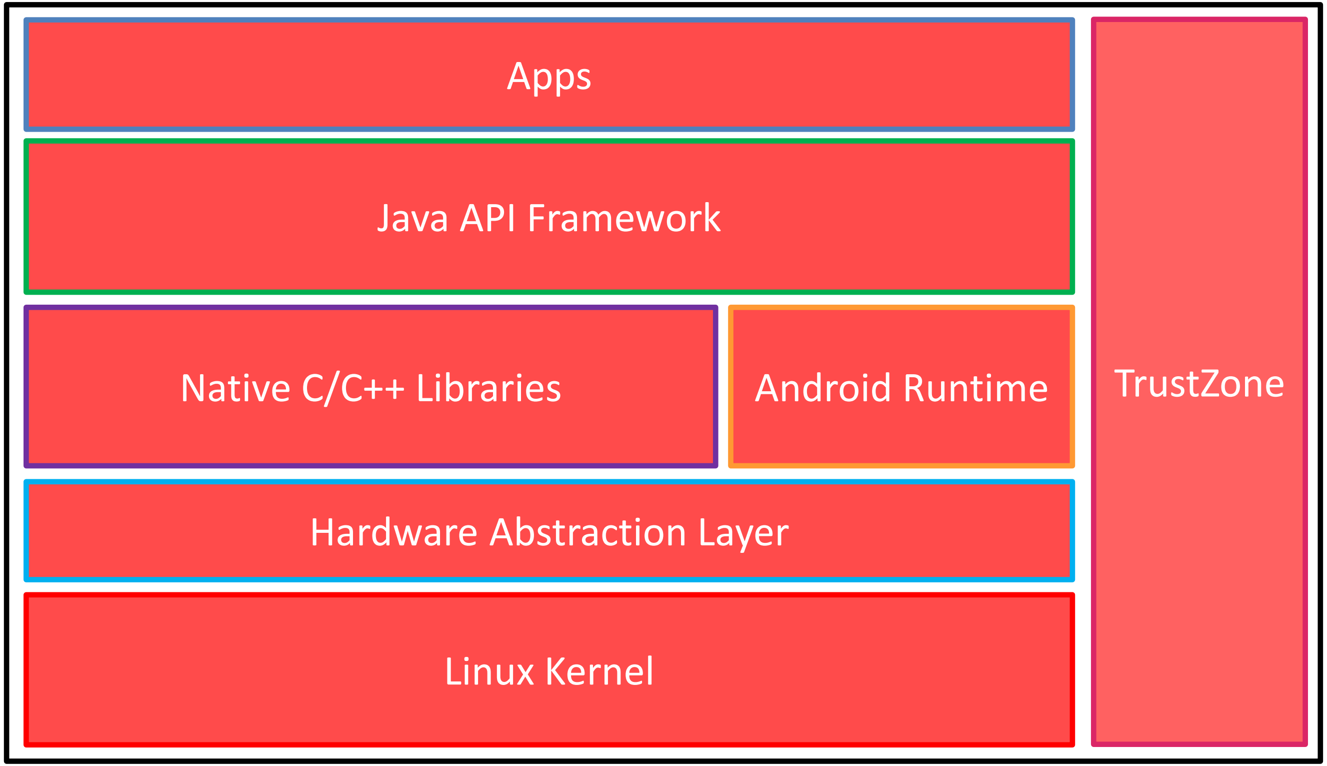
Let's Start from the Kernel



Let’s Start from the Kernel

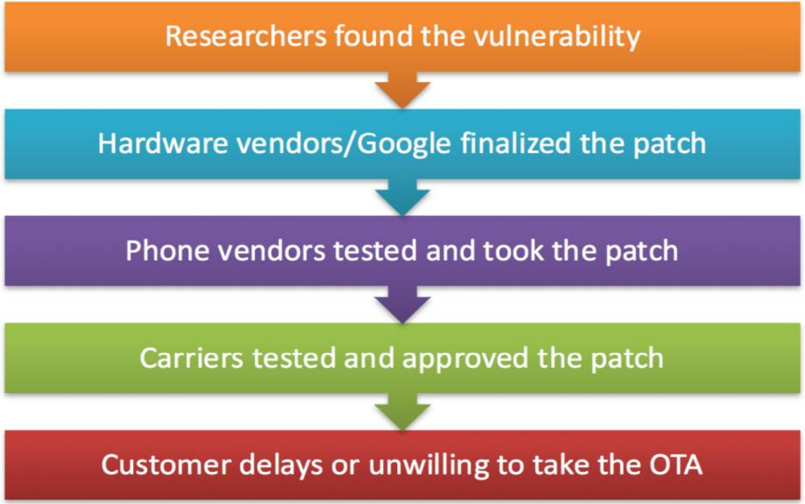


Let's Start from the Kernel



Challenges

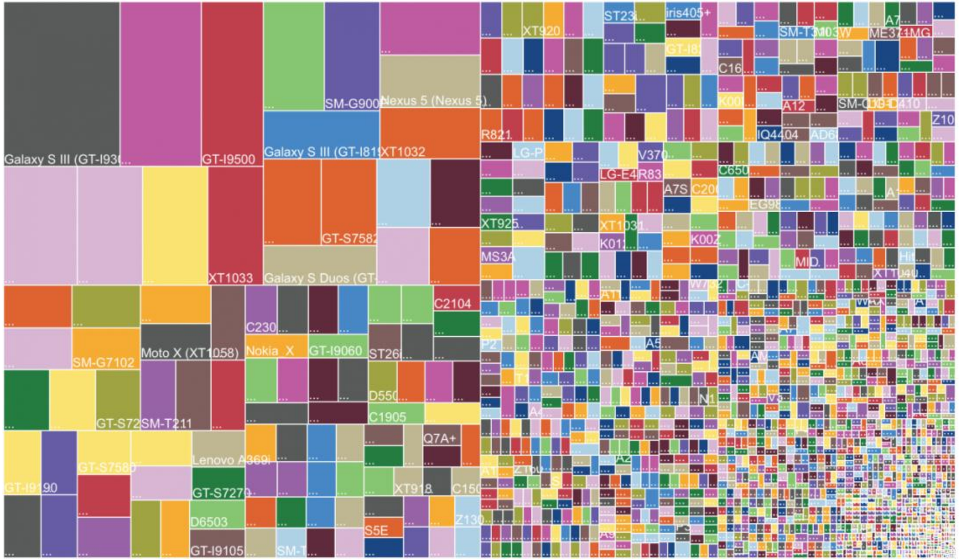
- *Officially* patching an Android device is a long process → **Third-party**



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

Challenges

- Severely fragmented Android ecosystem → **Adaptive**



http://d.ibtimes.co.uk/en/full/1395443/android-fragmentation-2014.png

Solution

Third-party Binary-based Adaptive Kernel Live Patching

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

Vendor	#Models	#Images
Samsung	192	419
Huawei	132	217
LG	120	239
Oppo	74	249
Google Nexus	2	15
Total	520	1139

Category	Statistics
Countries	67
Carriers	37
Android Versions	4.2.x, 4.3.x, 4.4.x, 5.0.x, 5.1.x, 6.0.x, 7.0.x
Kernel Versions	2.6.x, 3.0.x, 3.4.x, 3.10.x, 3.18.x
Kernel Architectures	ARM (77%), AArch64 (23%)
Kernel Build Years	2012, 2013, 2014, 2015, 2016

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

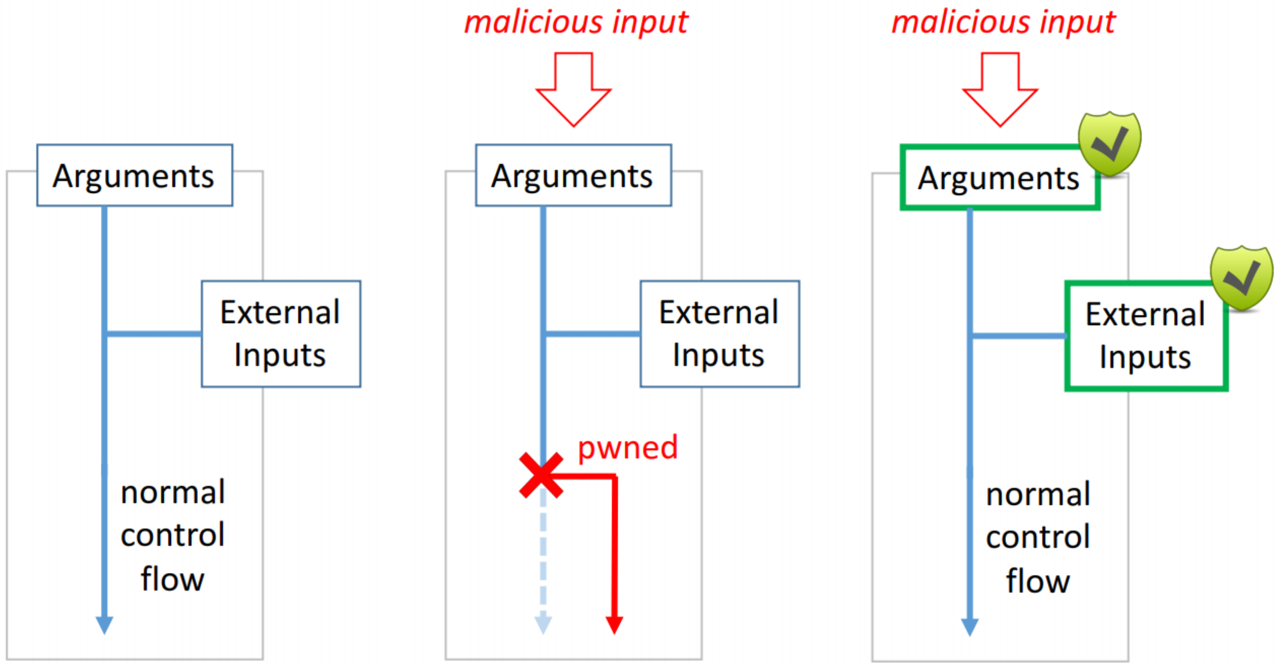
Feasibility Study: Observations

- 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

Filter them

Gracefully return

Overall Approach: Input Validation



KARMA

KARMA: **K**ernel **A**daptive **R**epair for **M**any **A**ndroids

- ✓ **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

KARMA Patch Example

```
        if (requeue_pi) {
                /*
+                * Requeue PI only works on two distinct uaddrs. 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.
                 */
```

Part of the official patch of CVE-2014-3153 (Towelroot)

KARMA Patch Example

```
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
end
kpatch.hook(0xca5269db50f4,"futex_requeue")
```

-EINVAL

More *complex* examples in the paper

KARMA API

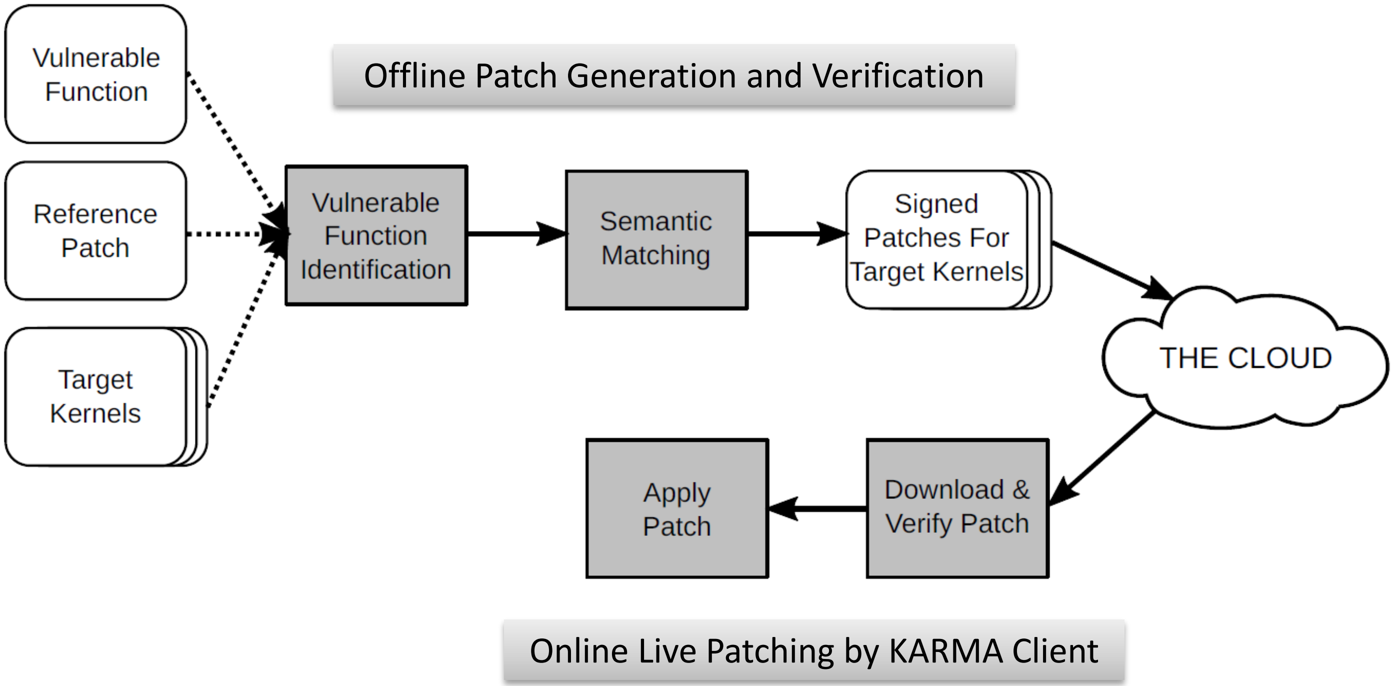
API	Functionality
hook	Hook a function for live patching
subhook	Hook the calls to sub-functions for live patching
alloc_mem	Allocate memory for live patching
free_mem	Free the allocated memory for live patching
get_callee	Locate a callee that can be hooked
search_symbol	Get the kernel symbol address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

KARMA API

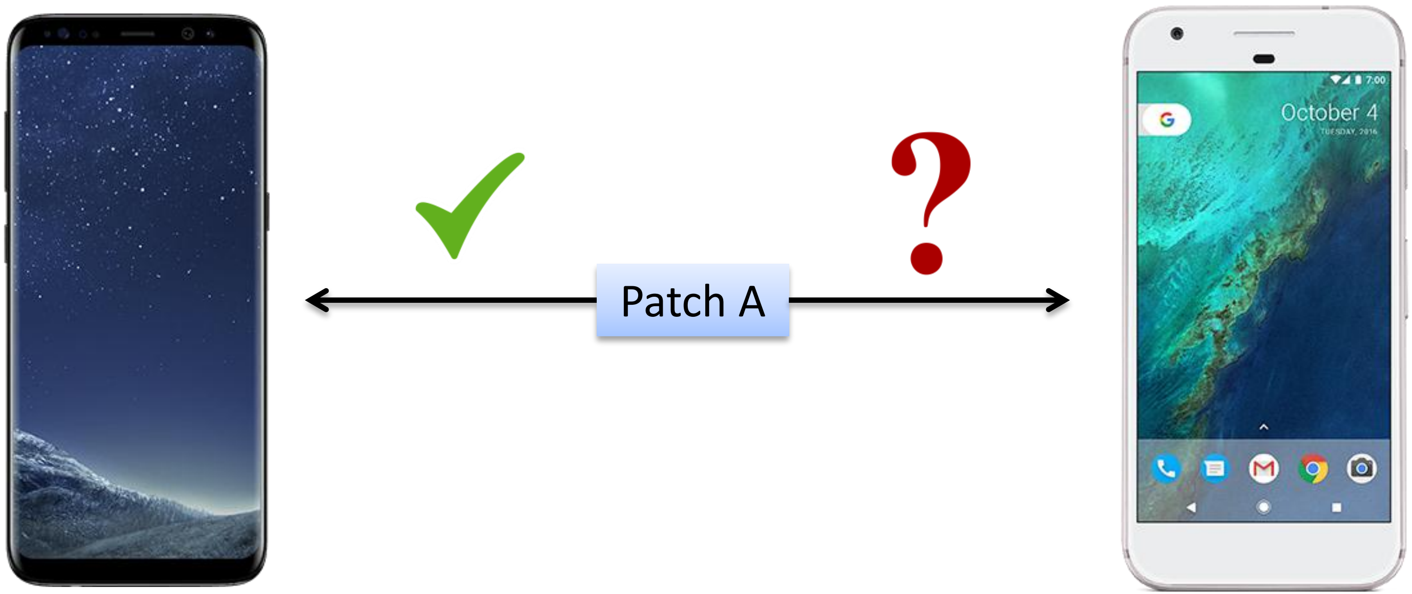
API	Functionality
hook	Hook a function for live patching
subhook	Hook the calls to sub-functions for live patching
alloc_mem	Allocate memory for live patching
free_mem	Free the allocated memory for live patching
get_callee	Locate a callee that can be hooked
search_symbol	Get the kernel symbol address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

Available to patches

KARMA Architecture



Offline Patch Adaptation



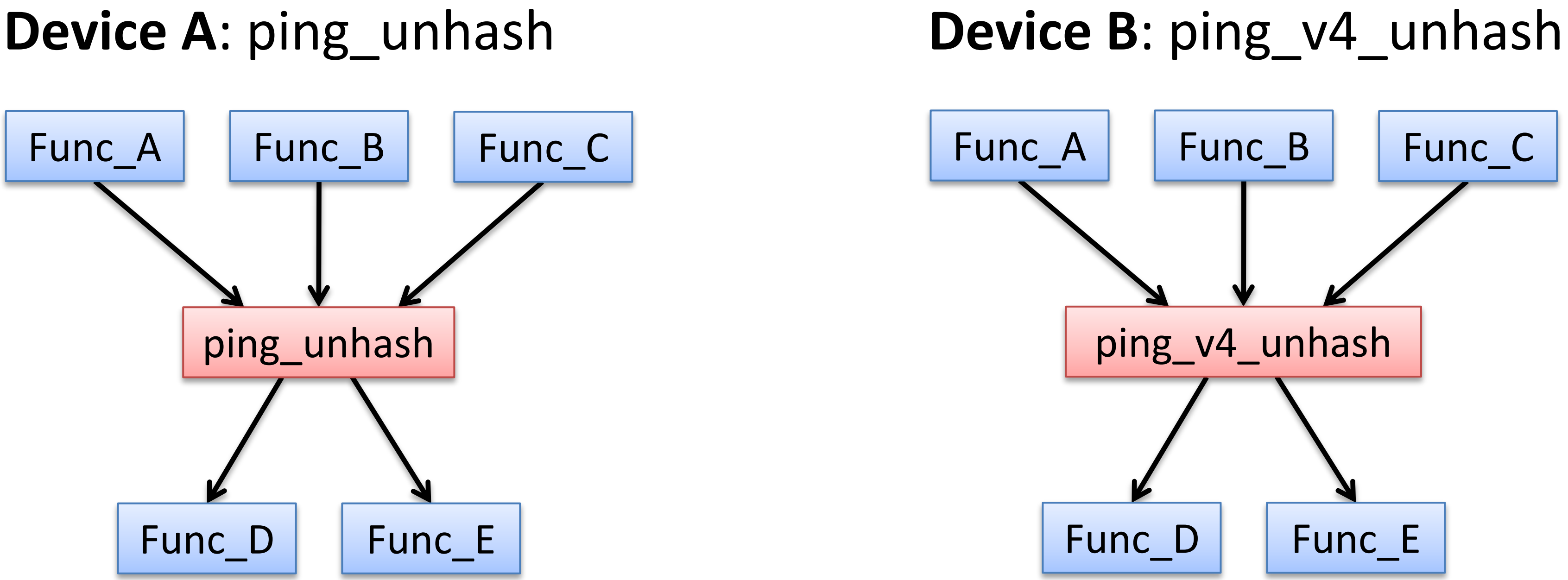
Offline Patch Adaptation

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)

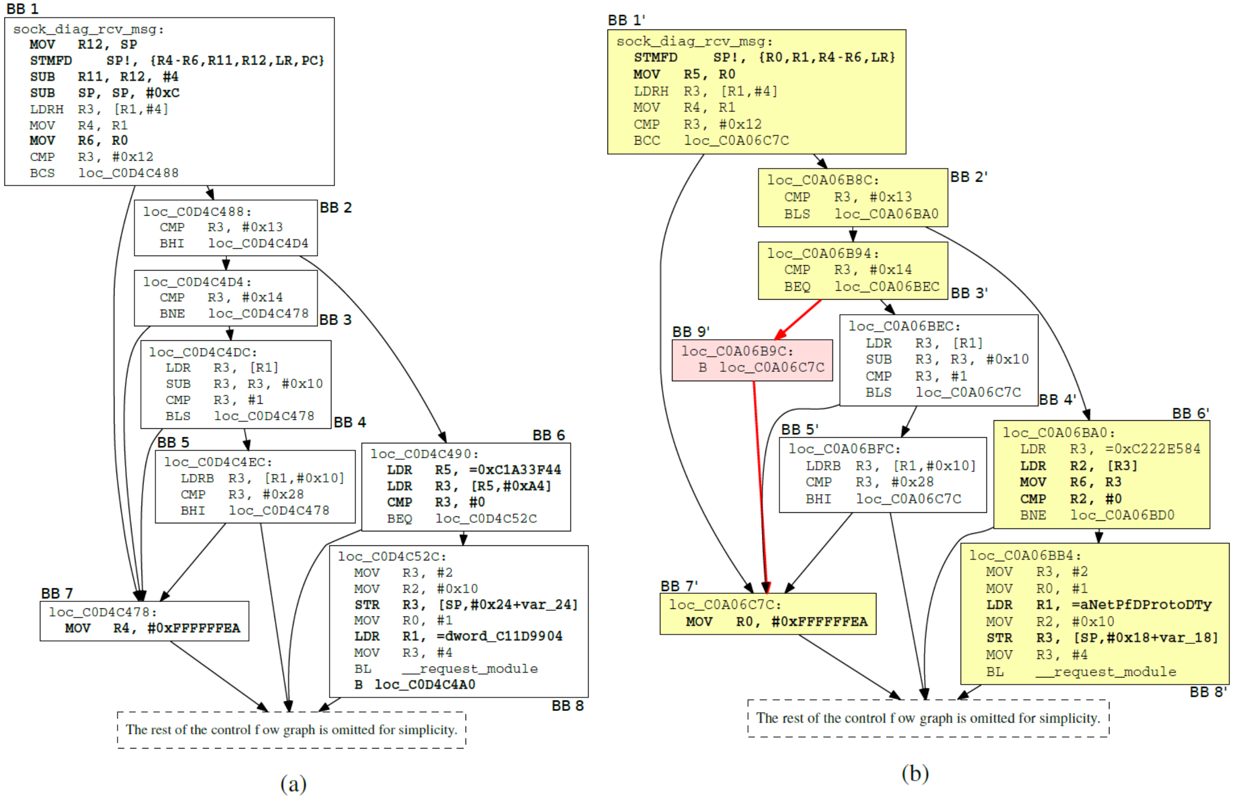


Call graph based similarity comparison

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

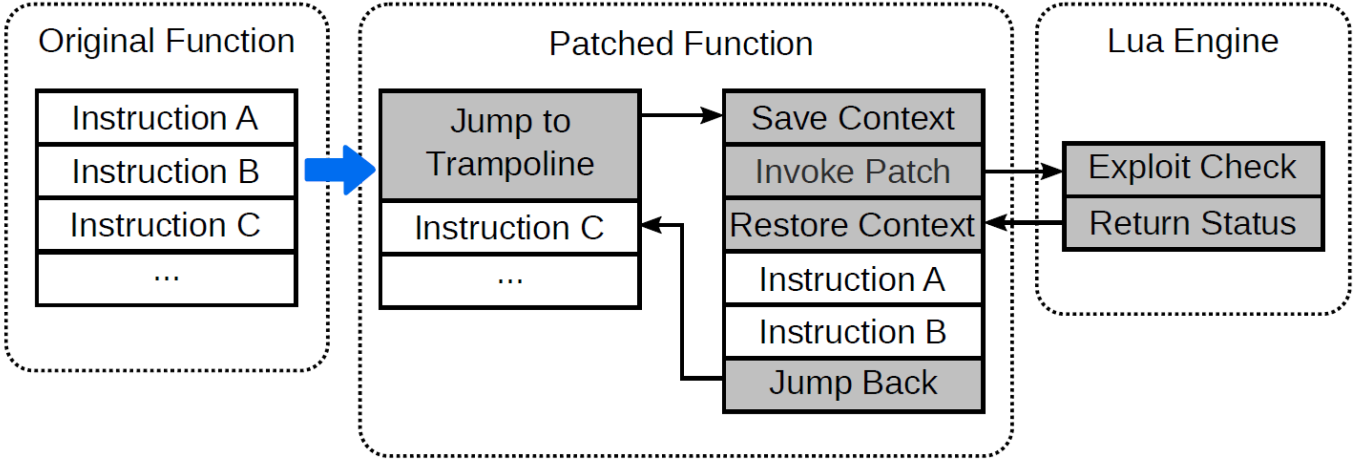


Same semantics with different syntax

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



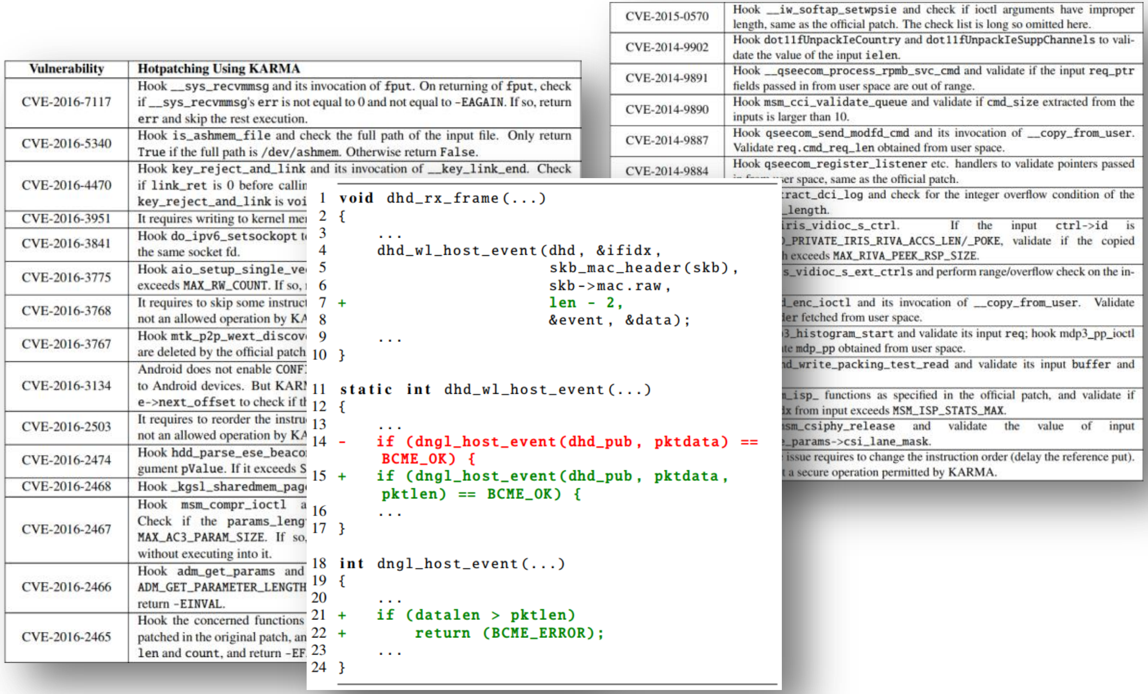
Function entry point hooking

Prototype Implementation

- Lua engine in kernel (11*K* 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

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

Evaluation: Adaptability

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

Types and frequencies of instruction opcodes

Evaluation: Adaptability

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

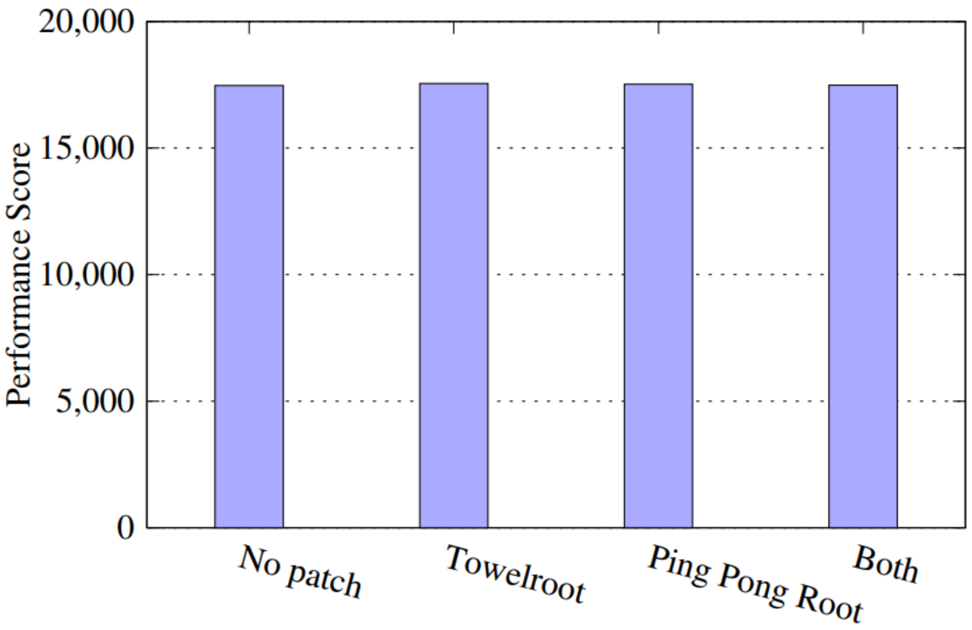
Number of function calls and conditional branches (to abstract CFG)

Evaluation: Adaptability

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

KARMA's semantic matching

Evaluation: Performance

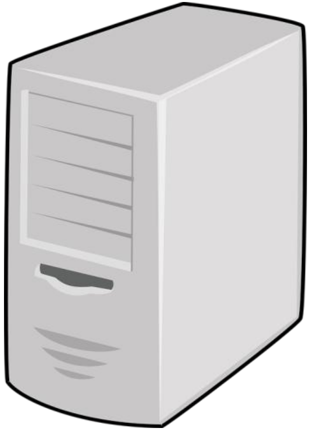


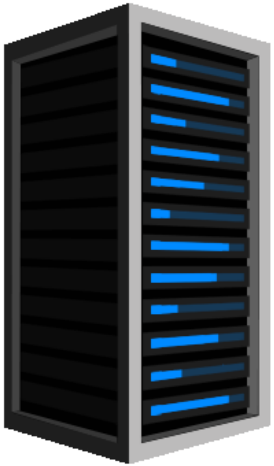
CF-Bench results with different patches

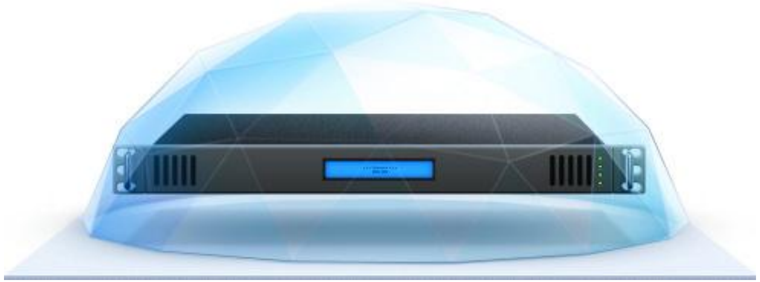
Hack your server

remotely

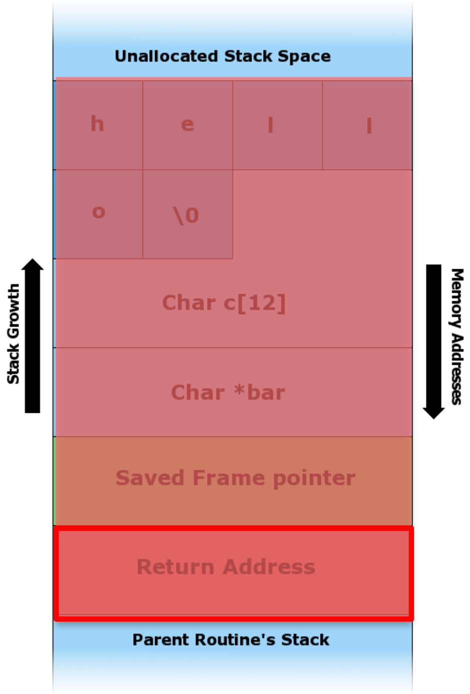
Attackers have limited information



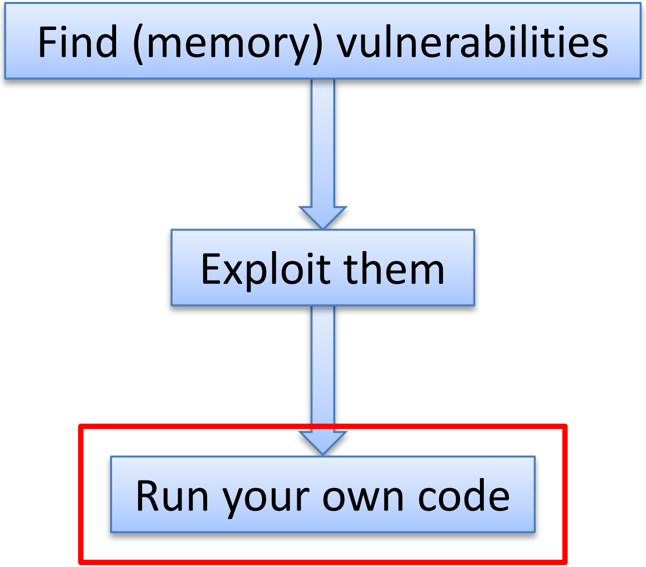




Attack Example: Stack Overflow



Typical Attack Procedure to Take Over the *Whole System*



Data Execution Prevention (DEP)

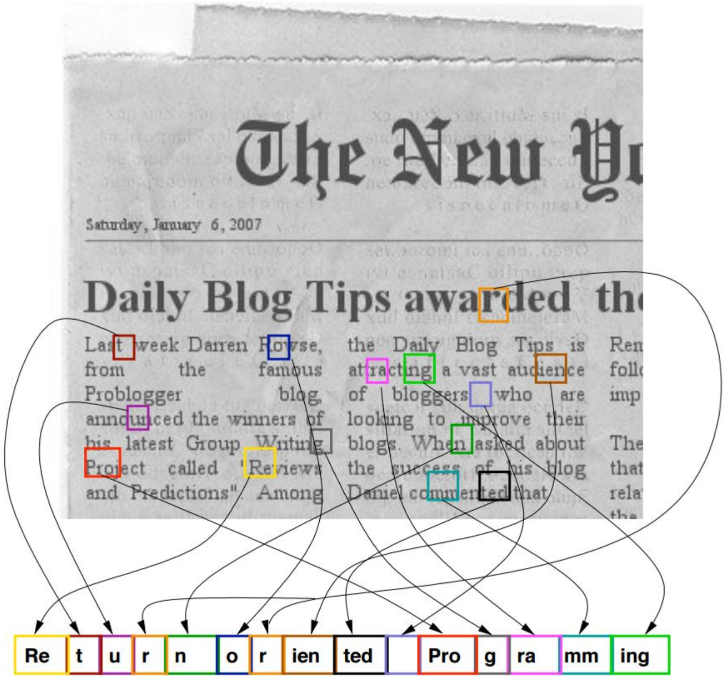
- *Previously*, attackers inject their *own stuff* into the process, and run it
- *Currently*, Data Execution Prevention (DEP) is widely deployed.
- You cannot run what you inject

Code Reuse Attack

Example: Return-Oriented Programming

Existing Code

Chained Gadgets



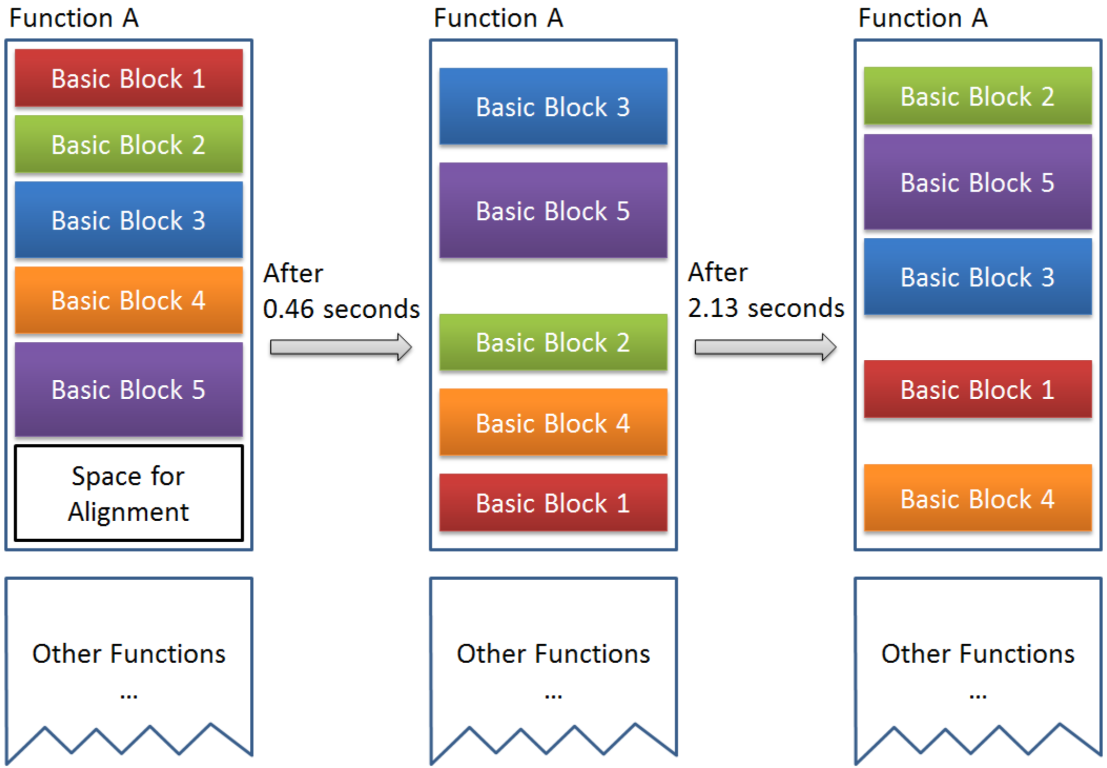
Protect your server

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

SECURITY

After 0 successful submissions, Google doubles top reward for hacking a Chromebook to $100,000

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

Above: An HP Chromebook.

Image Credit: TechnologyGuide TestLab/Flickr

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.

VB Recommendations

Chip industry turned upside down: Broadcom bids $130 billion for Qualcomm, Intel teams up with AMD

The end of the cloud is coming

Overwatch lead Jeff Kaplan on Moira, Mercy's troubles, and fixing toxicity

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