



USENIX Security Symposium 2017



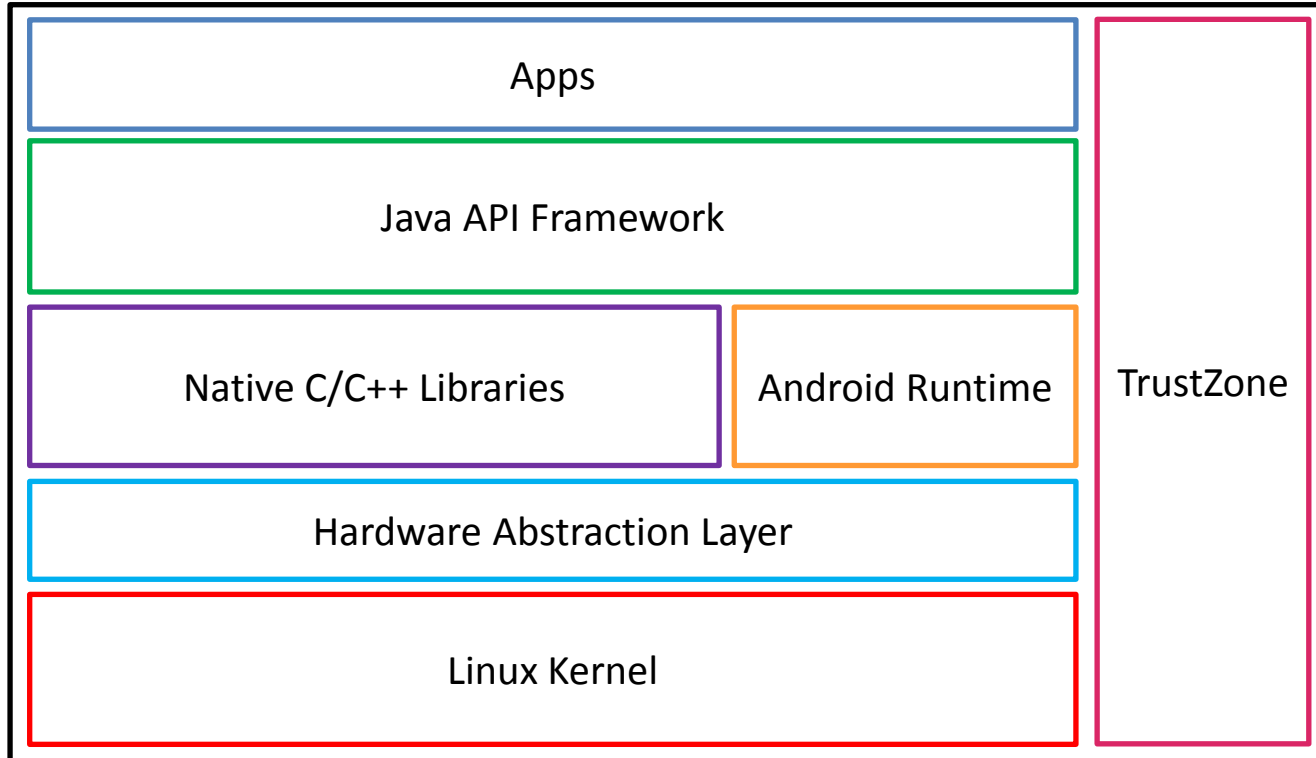
# Adaptive Android Kernel Live Patching

Yue Chen<sup>1</sup>, Yulong Zhang<sup>2</sup>, Zhi Wang<sup>1</sup>, Liangzhao Xia<sup>2</sup>, Chenfu Bao<sup>2</sup>, Tao Wei<sup>2</sup>

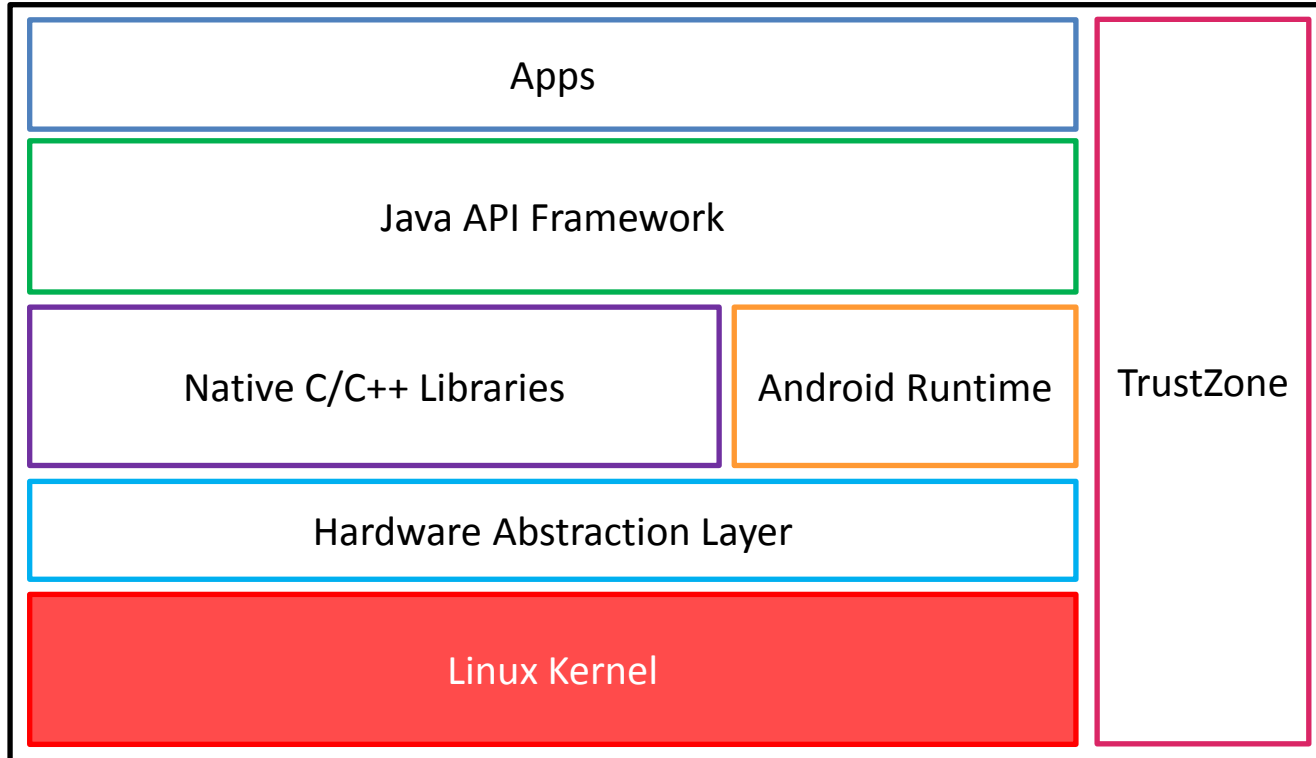
Florida State University<sup>1</sup>

Baidu X-Lab<sup>2</sup>

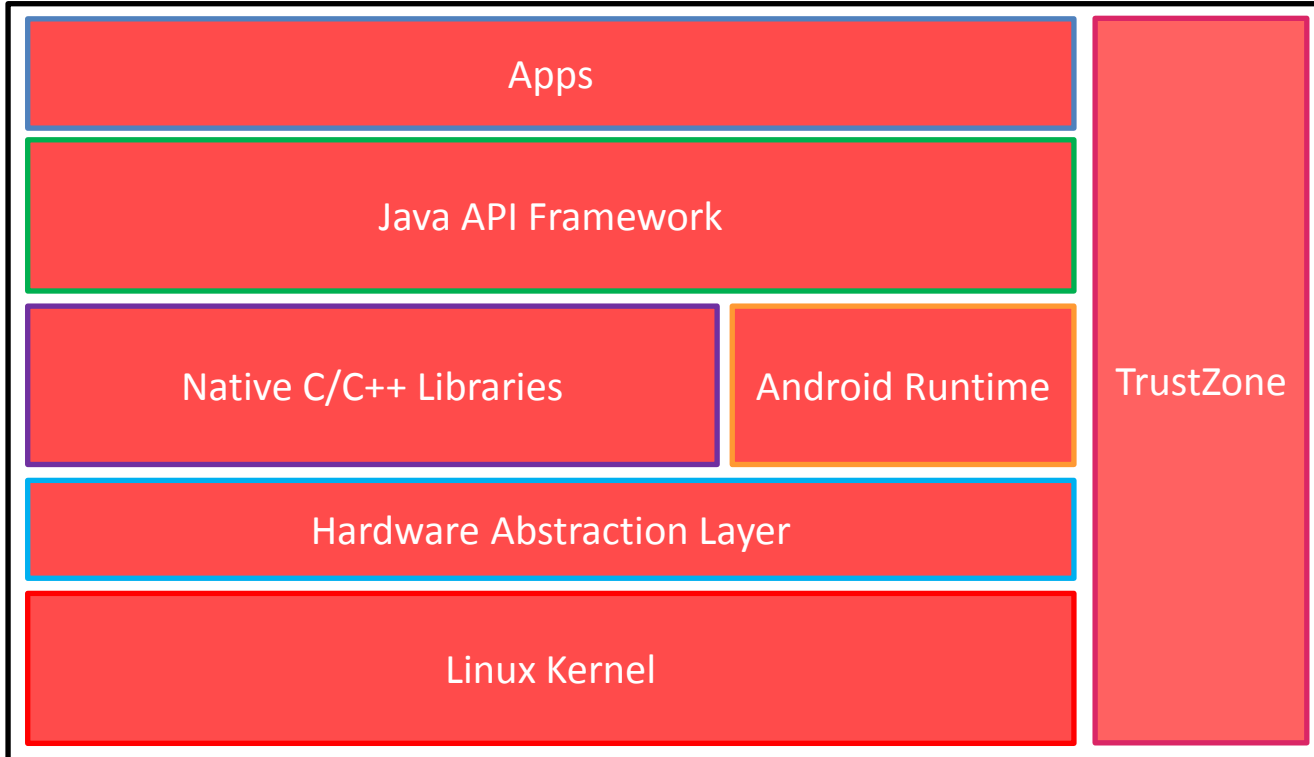
# Android Kernel Vulnerabilities



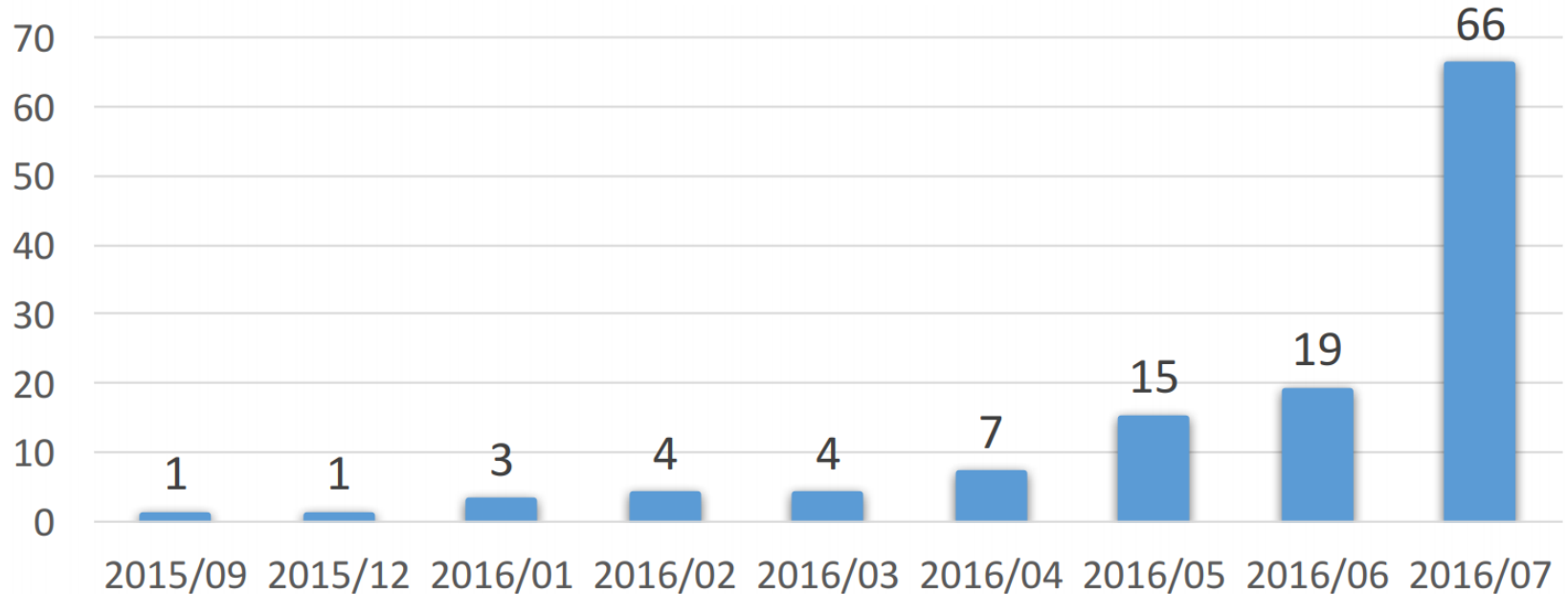
# Android Kernel Vulnerabilities



# Android Kernel Vulnerabilities



# Number of Disclosed Android Kernel Vulnerabilities



# Problem: Old Exploits Remain Effective

<b>CVE ID</b>	<b>Release Date</b>	<b>Months</b>	<b>% Vulnerable Devices</b>
CVE-2015-3636	Sep. 2015	14	30%
CVE-2015-1805	Mar. 2016	8	47%

Number of devices vulnerable to two root exploits as of Nov. 2016

- Android 5.0 released in November **2014**
- **46.3%** of devices run an older version in September **2016**

# Challenges

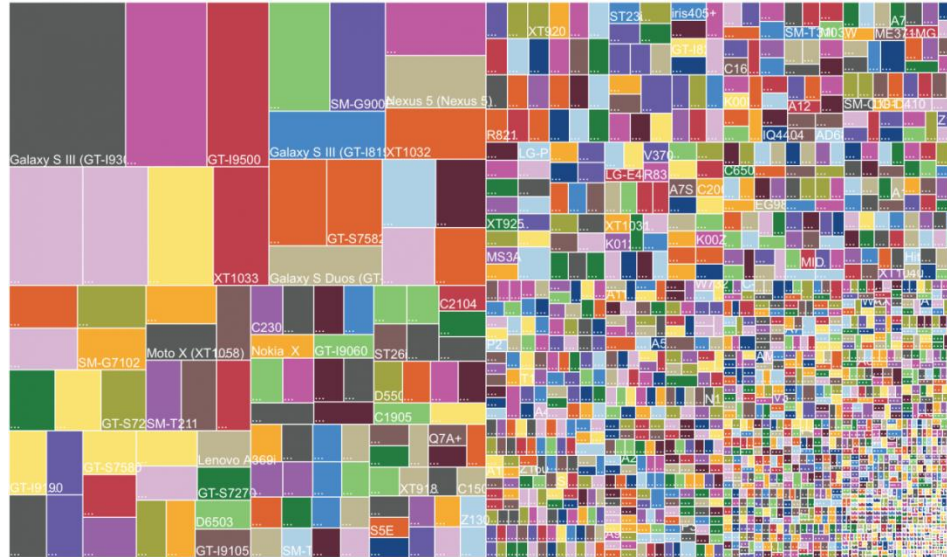
- *Officially* patching an Android device is a **long** process → **Third-party**



- **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

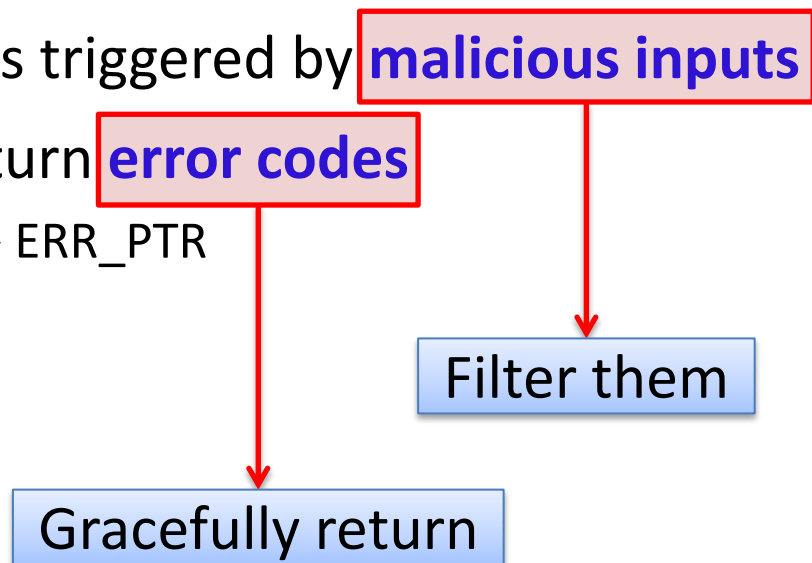
# Feasibility Study: Observations

---

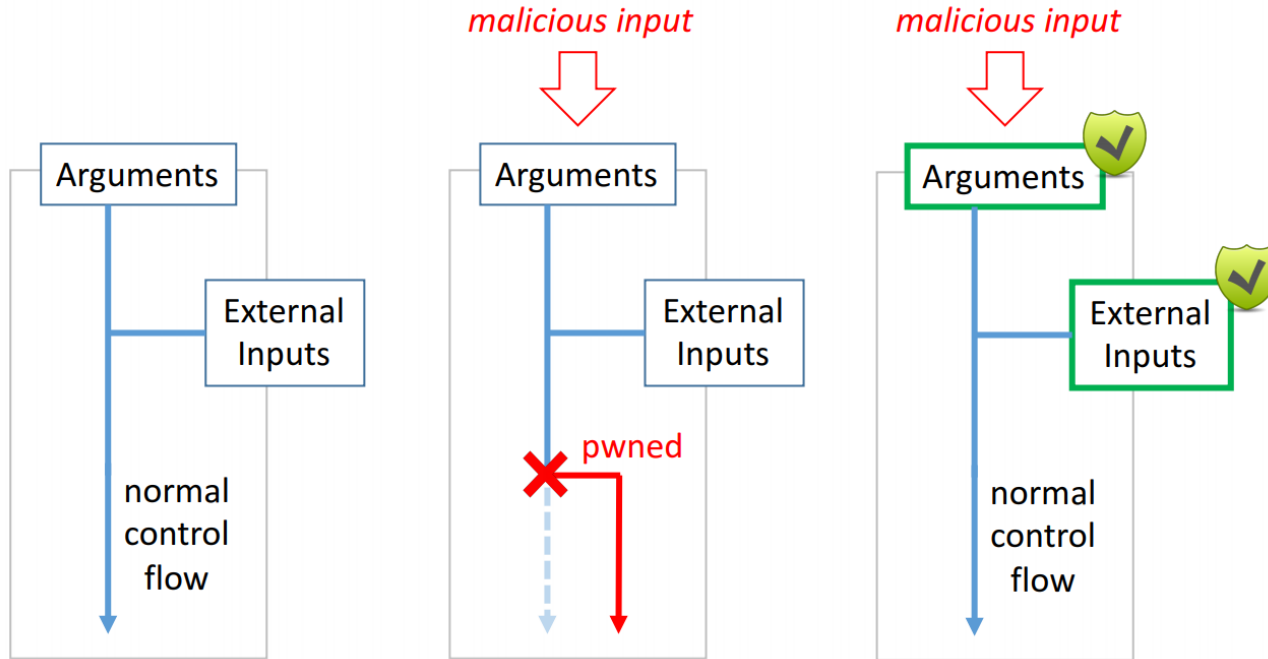
- Most kernel functions are **stable** across devices and Android releases
- Most vulnerabilities triggered by **malicious inputs**
- Many functions return **error codes**
  - Return a pointer → ERR\_PTR

# 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



# 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 Design: Multi-level Patching

- A patch can only be placed at **designated locations**

**Level 1:** Entry or return point of a (vulnerable) function

**Level 2:** Before or after the call site to a callee

e.g., `copy_from_user`

**Level 3:** Binary-based patch

- 76 critical Android kernel vulnerabilities

**Level 1:** 49/76 (64.5%)

**Level 2:** 22/76 (28.9%)

**Level 3:** 5/76 (6.6%)



# KARMA Patch Example

```
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.
+     */
```

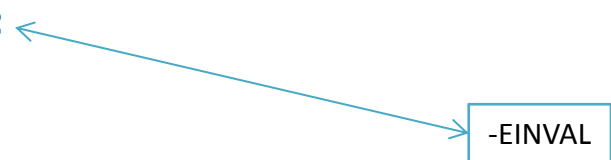
Part of the official patch of CVE-2014-3153 (Towelroot)

# KARMA Patch Example

---

```
1 function kpatcher(patchID, sp, cpsr, r0, r1,
   r2, r3, r4, r5, r6, r7, r8, r9, r10, r11,
   r12, r14)
2   if patchID == 0xca5269db50f4 then
3     uaddr1 = r0
4     uaddr2 = r2
5     if uaddr1 == uaddr2 then
6       return -22
7     else
8       return 0
9     end
10  end
11 end
12 kpatch.hook(0xca5269db50f4, "futex_requeue")
```

---



More *complex* examples in the paper

# KARMA API

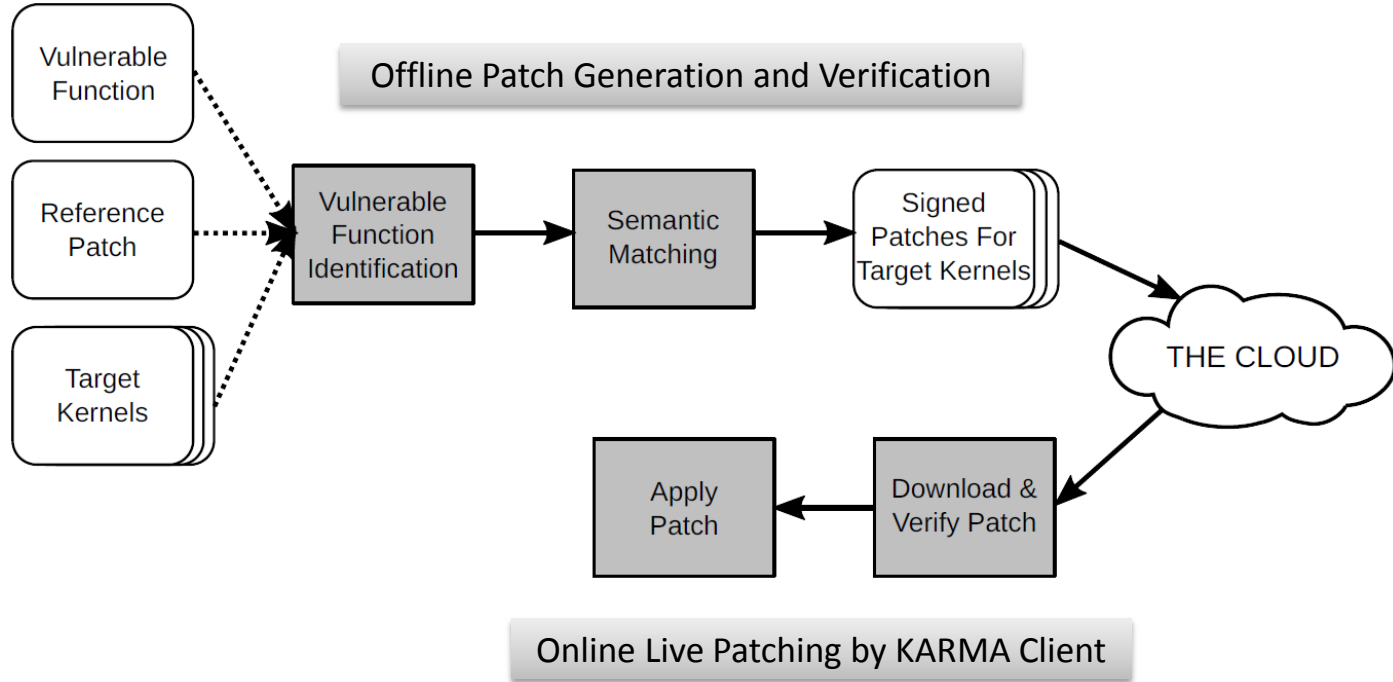
API	Functionality
hook	Hook a function for live patching
subhook	Hook the calls to sub-functions for live patching
alloc_mem	Allocate memory for live patching
free_mem	Free the allocated memory for live patching
get_callee	Locate a callee that can be hooked
search_symbol	Get the kernel symbol address
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

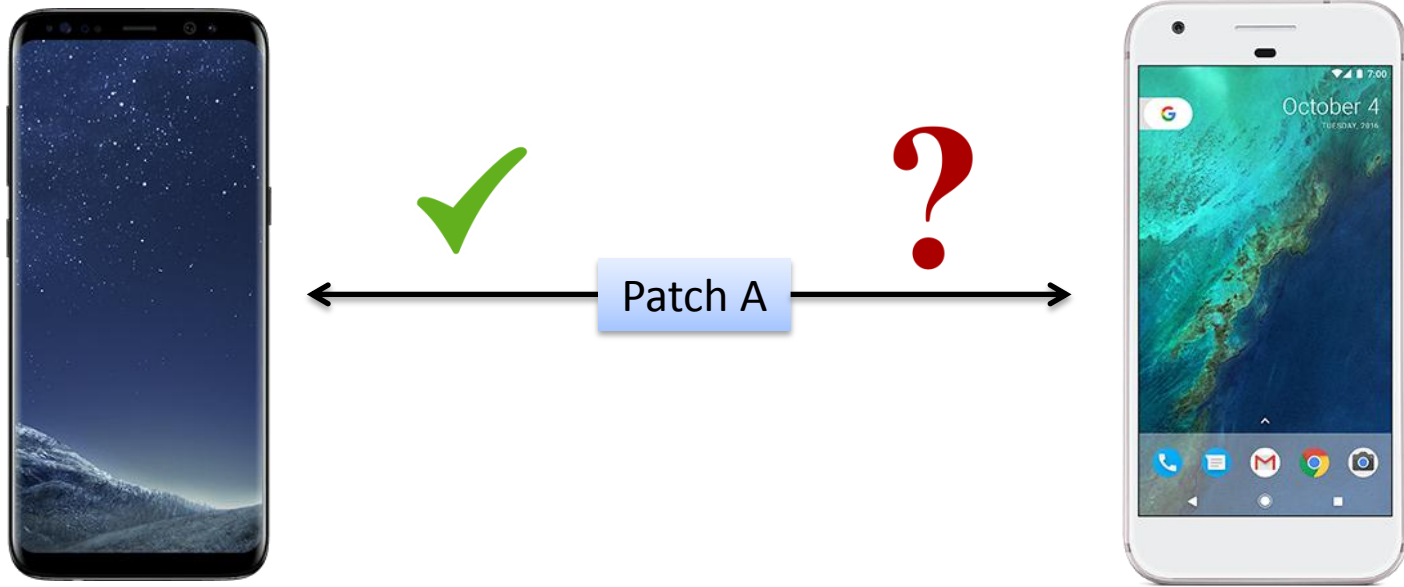
API	Functionality
hook	Hook a function for live patching
subhook	Hook the calls to sub-functions for live patching
alloc_mem	Allocate memory for live patching
free_mem	Free the allocated memory for live patching
get_callee	Locate a callee that can be hooked
search_symbol	Get the kernel symbol address
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

---

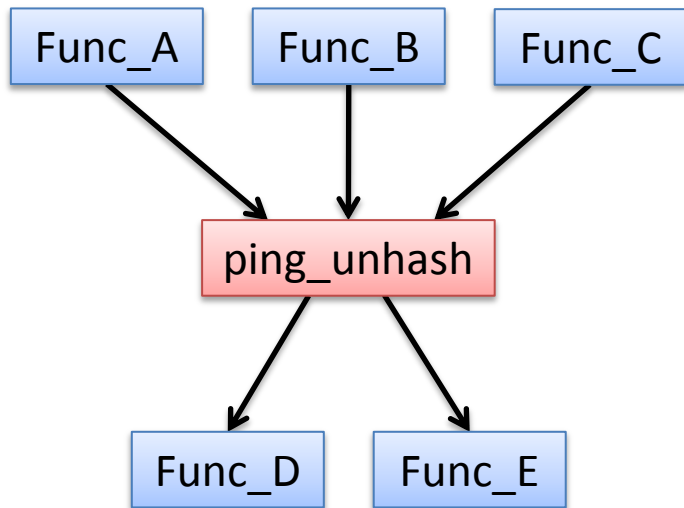
Three steps:

- 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

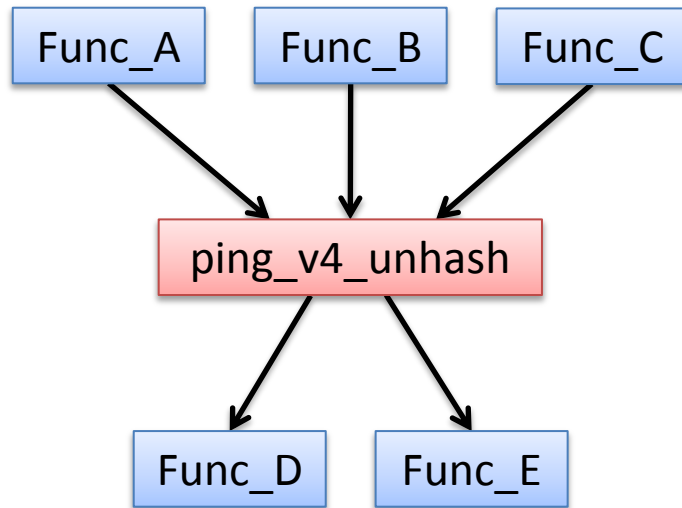
# Vulnerable Function Identification Example

CVE-2015-3636 (PingPong Root)

**Device A:** ping\_unhash



**Device B:** ping\_v4\_unhash



Call graph based similarity comparison

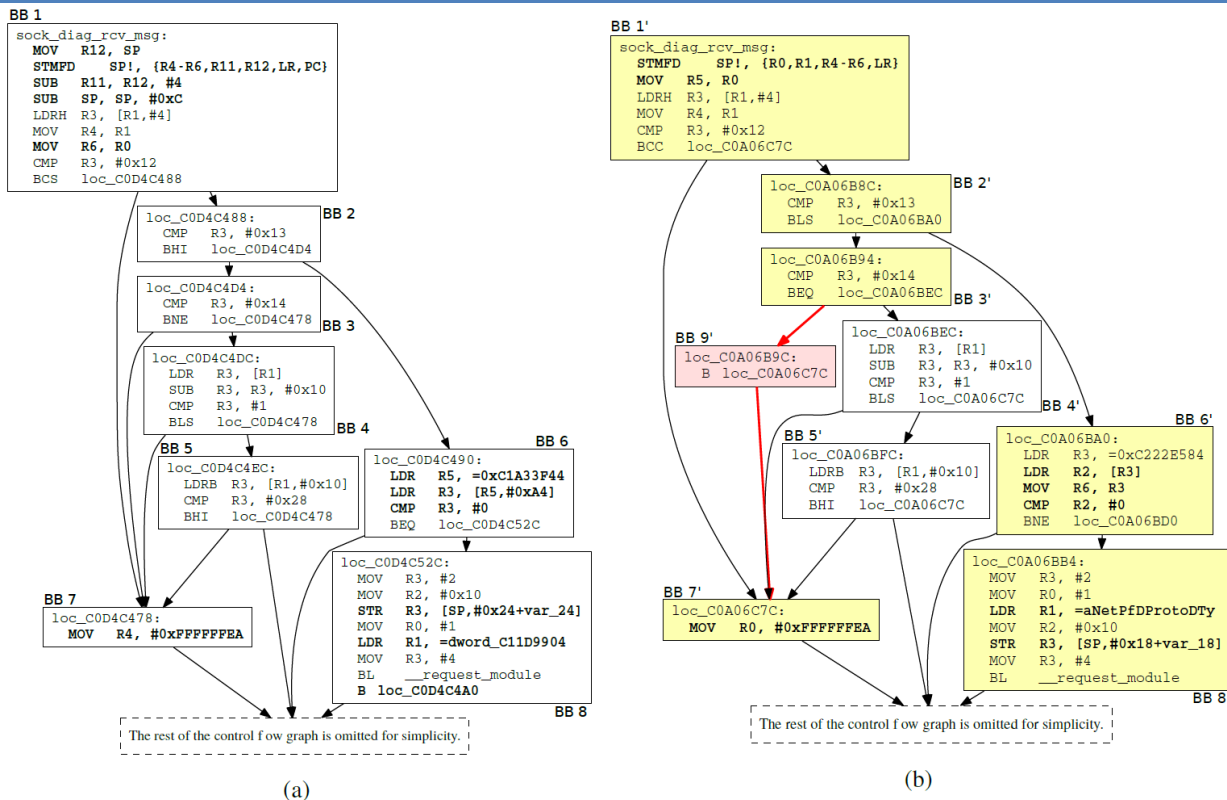


# Semantic Matching

---

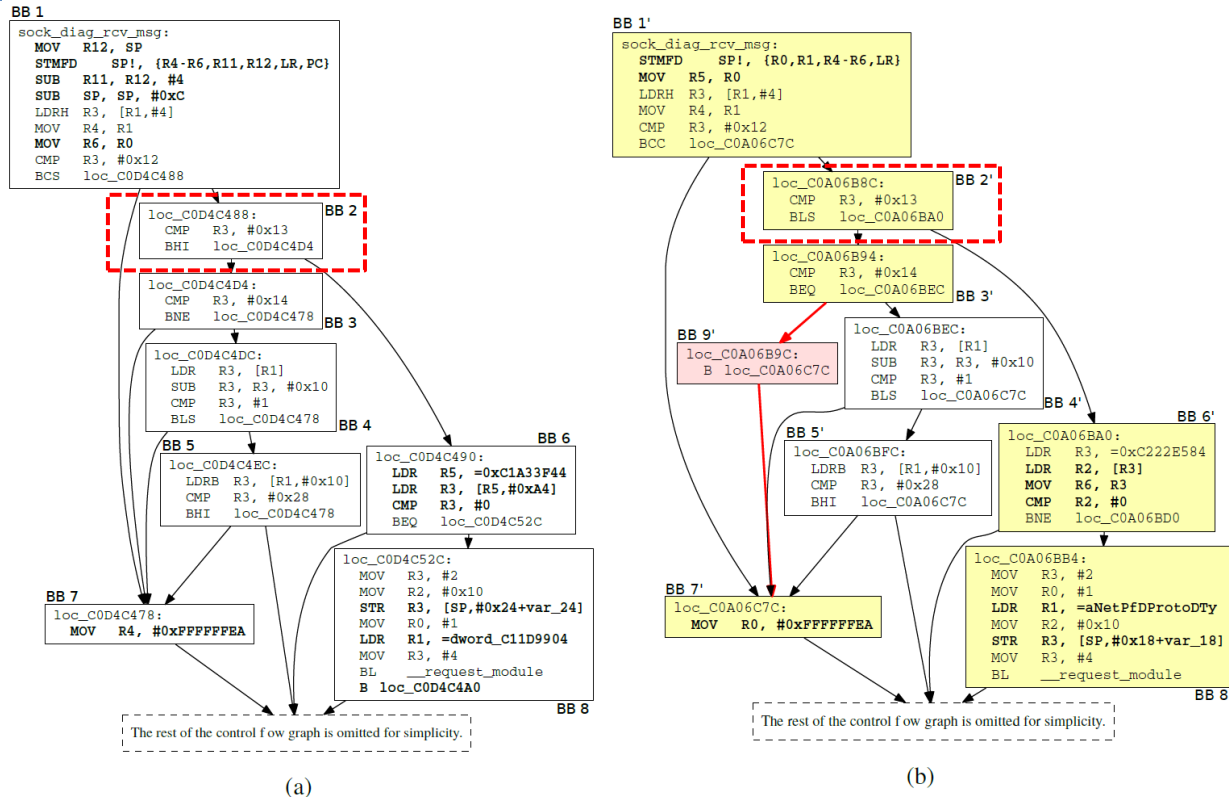
- Check if two functions are semantically equivalent
- If so, adapt the reference patch to the target kernel
- Syntactic matching is **too strict**
  - Different compilers can generate different code with same semantics
    - Instruction order, register allocation, instruction selection, code layout

# Semantic Matching



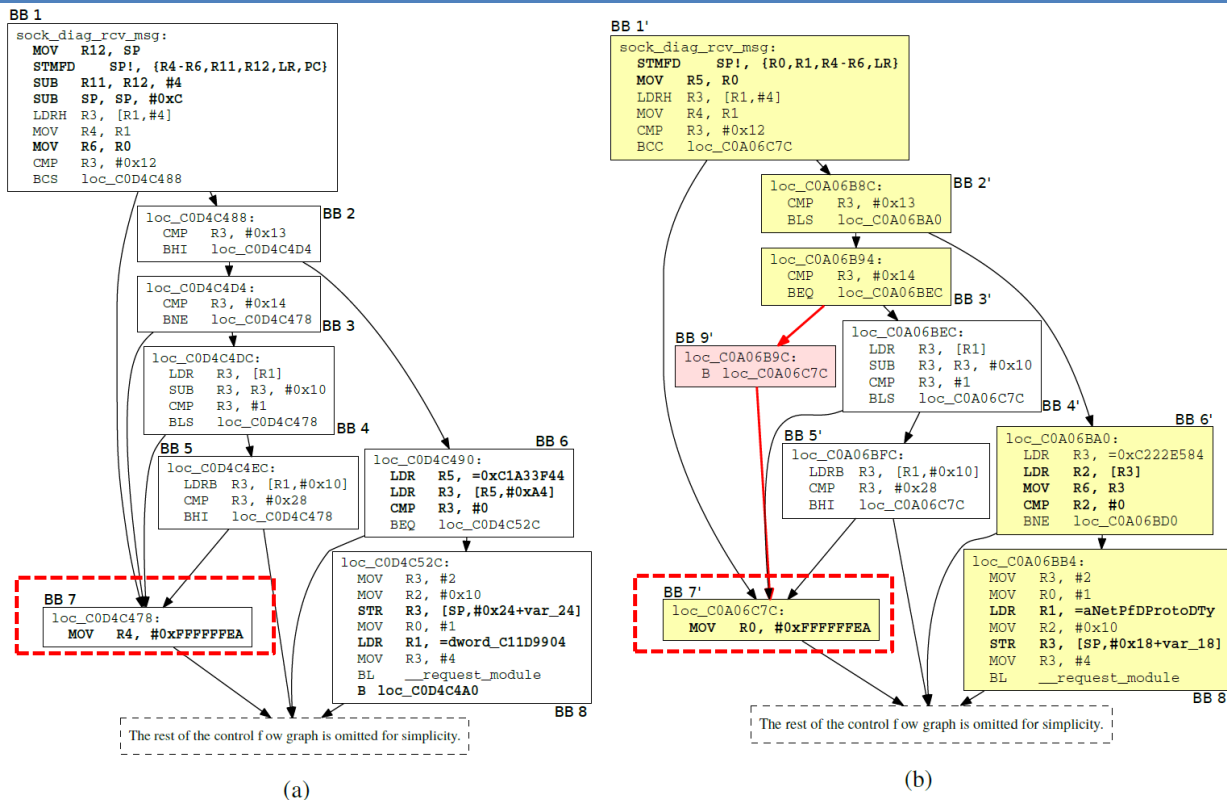
Same semantics with different syntax

# Semantic Matching



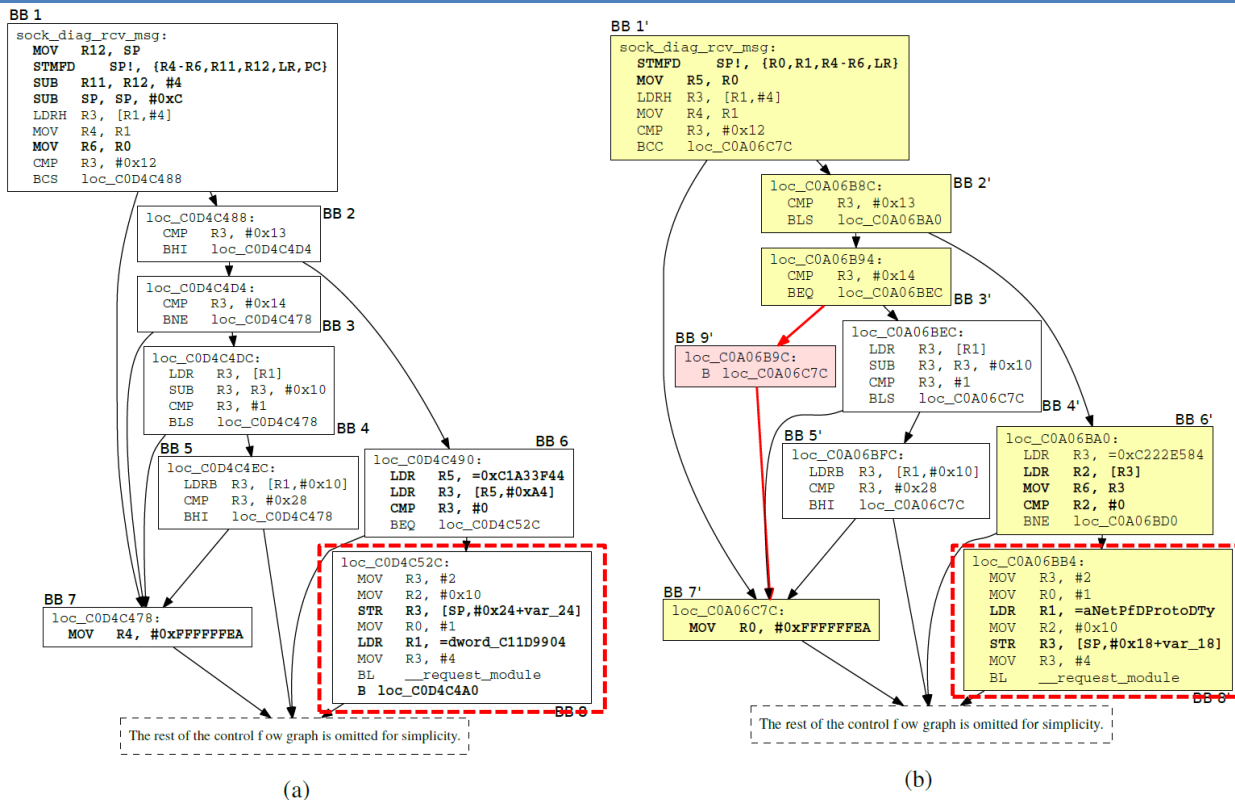
Same semantics with different syntax

# Semantic Matching



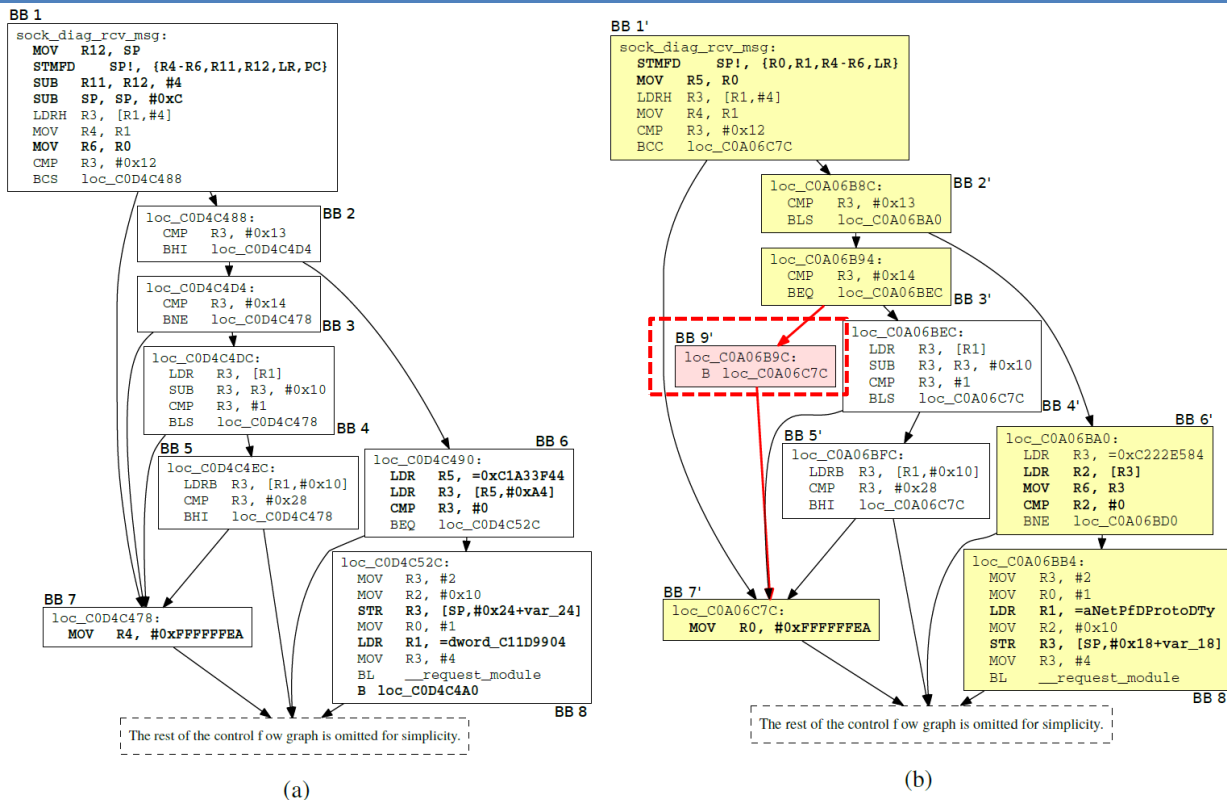
Same semantics with different syntax

# Semantic Matching



Same semantics with different syntax

# Semantic Matching



(a)

(b)

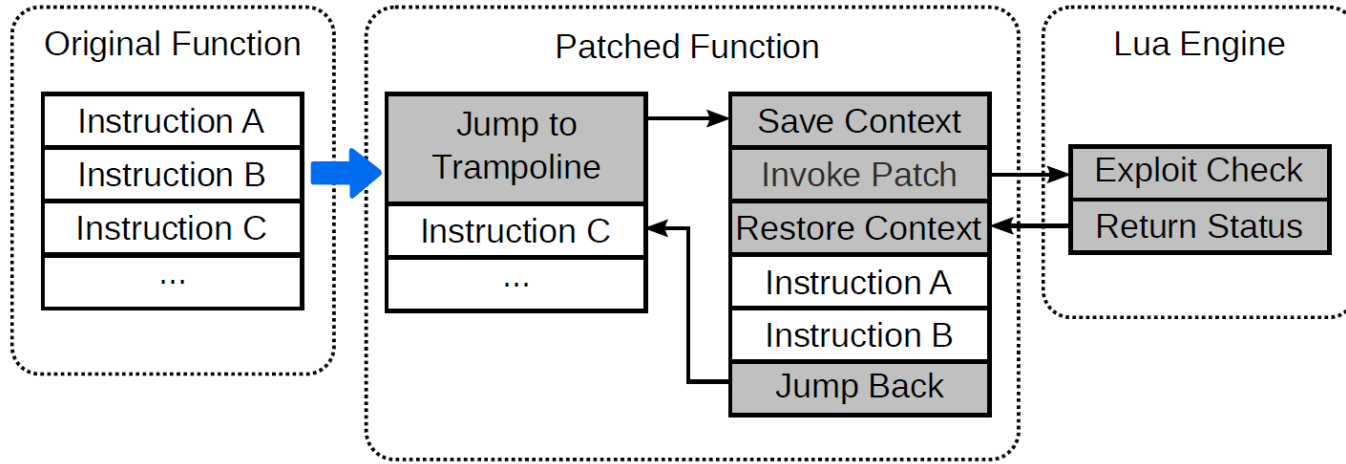
Same semantics with different syntax

# 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
- Patch level:
  - Level-1: 49
  - Level-2: 22
  - Level-3: 5

Vulnerability	Hotpatching Using KARMA
CVE-2016-7117	Hook <code>__sys_recvmmsg</code> and its invocation of <code>fput</code> . On returning of <code>fput</code> , check if <code>__sys_recvmmsg's err</code> is not equal to 0 and not equal to <code>-EAGAIN</code> . If so, return <code>err</code> and skip the rest execution.
CVE-2016-5340	Hook <code>is_ashmem_file</code> and check the full path of the input file. Only return <code>True</code> if the full path is <code>/dev/ashmem</code> . Otherwise return <code>False</code> .
CVE-2016-4470	Hook <code>key_reject_and_link</code> and its invocation of <code>__key_link_end</code> . Check if <code>link_ret</code> is 0 before calling <code>key_reject_and_link</code> is void
CVE-2016-3951	It requires writing to kernel memory
CVE-2016-3841	Hook <code>do_ipv6_setsockopt</code> if it exceeds <code>MAX_RW_COUNT</code> . If so, use the same socket fd.
CVE-2016-3775	Hook <code>aio_setup_single_vec</code> if it exceeds <code>MAX_RW_COUNT</code> . If so, use the same socket fd.
CVE-2016-3768	It requires to skip some instructions not an allowed operation by <code>Kernel</code>
CVE-2016-3767	Hook <code>mtk_p2p_wext_discov</code> are deleted by the official patch
CVE-2016-3134	Android does not enable <code>CONFIG_ANDROID</code> to Android devices. But <code>KARMA</code> requires to check if <code>next_offset</code> to check if it is 0
CVE-2016-2503	It requires to reorder the instructions not an allowed operation by <code>Kernel</code>
CVE-2016-2474	Hook <code>hdd_parse_ese beacon</code> argument <code>pValue</code> . If it exceeds <code>MAX_RW_COUNT</code> .
CVE-2016-2468	Hook <code>kgs1_shardmem_page</code> and its invocation of <code>__key_link_end</code> . Check if <code>link_ret</code> is 0 before calling <code>key_reject_and_link</code> is void
CVE-2016-2467	Hook <code>msm_compr_ioctl</code> and check if the <code>params_length</code> exceeds <code>MAX_AC3_PARAM_SIZE</code> . If so, without executing into it.
CVE-2016-2466	Hook <code>adm_get_params</code> and <code>ADM_GET_PARAMETER_LENGTH</code> return <code>-EINVAL</code> .
CVE-2016-2465	Hook the concerned functions patched in the original patch, and return <code>-EINVAL</code> and count, and return <code>-EINVAL</code> .

CVE-2015-0570	Hook <code>__iw_softap_setwpsie</code> and check if <code>ioctl</code> arguments have improper length, same as the official patch. The check list is long so omitted here.
CVE-2014-9902	Hook <code>dot11fUnpackCountry</code> and <code>dot11fUnpackTeSuppChannels</code> to validate the value of the input <code>ielen</code> .
CVE-2014-9891	Hook <code>__qseecom_process_rpmv_svc_cmd</code> and validate if the input <code>req_ptr</code> fields passed in from user space are out of range.
CVE-2014-9890	Hook <code>msm_cci_validate_queue</code> and validate if <code>cmd_size</code> extracted from the inputs is larger than 10.
CVE-2014-9887	Hook <code>qseecom_send_modfd_cmd</code> and its invocation of <code>__copy_from_user</code> . Validate <code>req.cmd_req_len</code> obtained from user space.
CVE-2014-9884	Hook <code>qseecom_register_listener</code> etc. handlers to validate pointers passed in from user space, same as the official patch.

```

1 void dhd_rx_frame(...)
2 {
3     ...
4     dhd_wl_host_event(dhd, &fididx,
5                       skb_mac_header(skb),
6                       skb->mac_raw,
7                       len - 2,
8                       &event, &data);
9     ...
10 }
11
12 static int dhd_wl_host_event(...)
13 {
14     ...
15     if (dngl_host_event(dhd_pub, pktdata) ==
16         BCME_OK) {
17         if (dngl_host_event(dhd_pub, pktdata,
18                             pktlen) == BCME_OK) {
19             ...
20         }
21     }
22     int dngl_host_event(...)
23     {
24         ...
25         if (datalen > pktlen)
26             return (BCME_ERROR);
27         ...
28     }

```

...er space, same as the official patch.

tract\_dci\_log and check for the integer overflow condition of the length.

iris\_vidioc\_s\_ctrl. If the input `ctrl->id` is `PRIVATE_IRIS_RIVA_ACCS_LEN_POKE`, validate if the copied `len` exceeds `MAX_RIVA_PEEK_RSP_SIZE`.

s\_vidioc\_s\_ext\_ctrls and perform range/overflow check on the input `len`.

lenc\_ioctl and its invocation of `__copy_from_user`. Validate `req` fetched from user space.

3\_histogram\_start and validate its input `req`; hook `mdp3_pp_ioctl` and `mdp_pp` obtained from user space.

d\_write\_packing\_test\_read and validate its input buffer and its invocation of `__copy_from_user`.

isp\_ functions as specified in the official patch, and validate if `len` from input exceeds `MSM_ISP_STATS_MAX`.

sm\_csiiphy\_release and validate the value of input `params->csi_lane_mask`.

issue requires to change the instruction order (delay the reference put).

a secure operation permitted by KARMA.

# Evaluation: Adaptability

Kernel Function	CVE ID	# of Opcode Clusters		# of Syntax Clusters		# of Semantic Clusters		Semantic Matching Time Cost		
		# of the Largest Opcode Cluster	% of the Largest Opcode Cluster	# of the Largest Syntax Cluster	% of the Largest Syntax Cluster	# of the Largest Semantic Cluster	% of the Largest Semantic Cluster	# 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 Syntax Clusters		# of Semantic Clusters		Semantic Matching Time Cost		
		# of the Largest Opcode Cluster	% of the Largest Opcode Cluster	# of the Largest Syntax Cluster	% of the Largest Syntax Cluster	# of the Largest Semantic Cluster	% of the Largest Semantic Cluster	# 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 Syntax Clusters		# of Semantic Clusters		Semantic Matching Time Cost		
		# of the Largest Opcode Cluster	% of the Largest Opcode Cluster	# of the Largest Syntax Cluster	% of the Largest Syntax Cluster	# of the Largest Semantic Cluster	% of the Largest Semantic Cluster	# 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

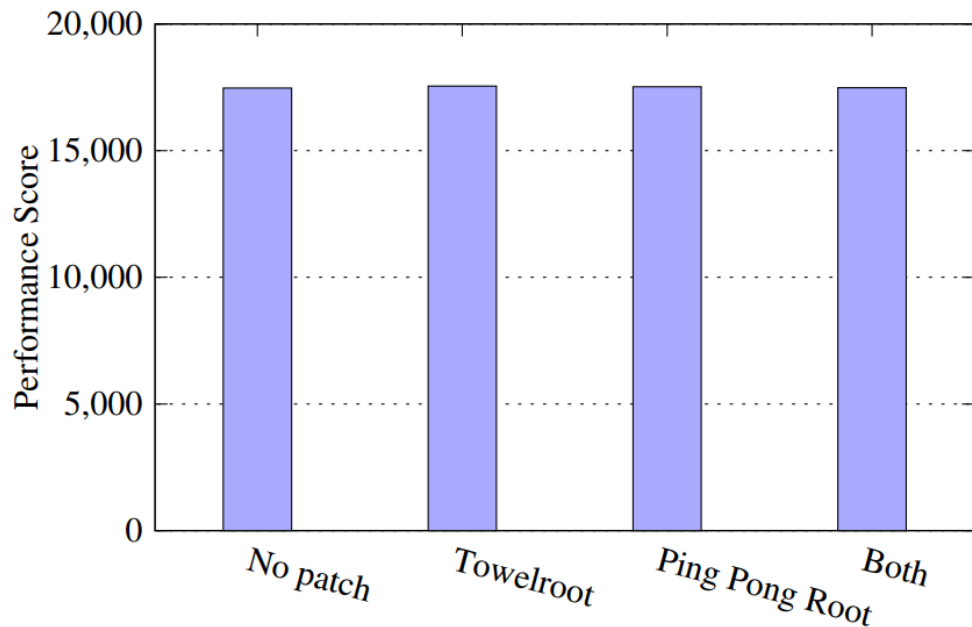
Number of function calls and conditional branches (to abstract CFG)

# Evaluation: Adaptability

Kernel Function	CVE ID	# of Opcode Clusters		# of Syntax Clusters		# of Semantic Clusters		Semantic Matching Time Cost	# of Instructions	# of Basic Blocks
		% of the Largest Opcode Cluster	% of the Largest Syntax Cluster	% of the Largest Semantic Cluster	% of the Largest Semantic Cluster					
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

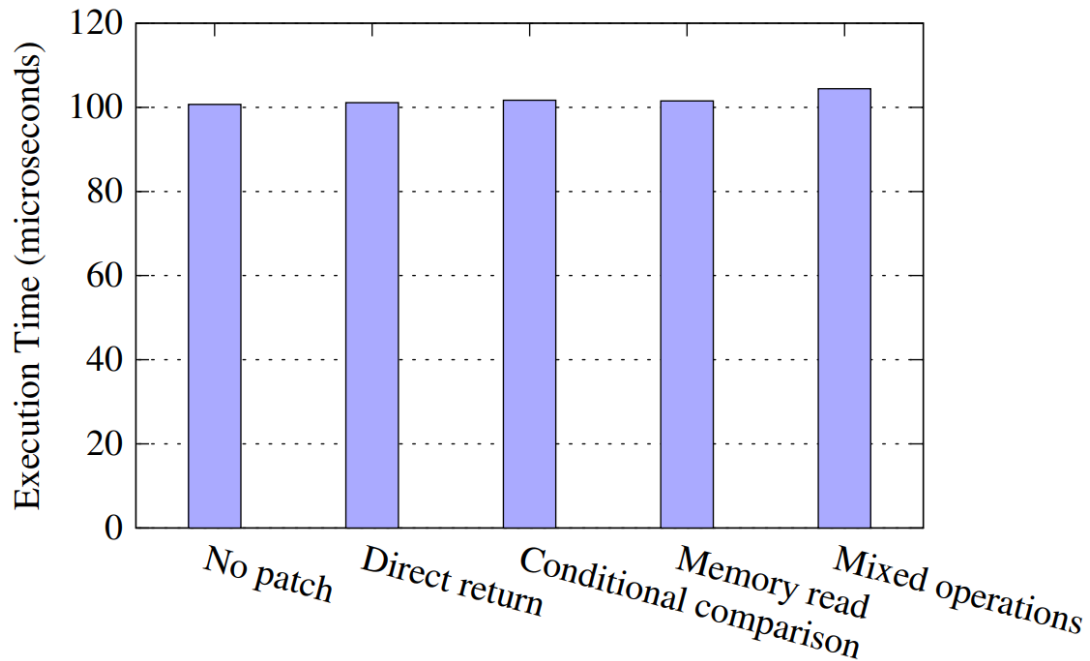
KARMA's semantic matching

# Evaluation: Performance



CF-Bench results with different patches

# Evaluation: Performance



Execution time of `chmod` with different patches



# Future Work

---

- User-space vulnerability protection
  - Project Treble → only partially solve the problem
- Lua engine in the kernel (11K SLOC)
  - Alternative execution engines, like BPF or sandboxed binary patches
- Error handling code could be vulnerable
  - Error injection to detect vulnerable error-handling code
- Improve semantic matching

The background of the slide is a dense, overlapping collage of numerous smartphones. The phones are shown from various angles, some displaying different home screens with various app icons, weather widgets, and maps. The overall color palette is muted, with a light blue and grey tint over the original phone colors.

# Q & A

## Adaptive Android Kernel Live Patching

[www.YueChen.me](http://www.YueChen.me)

Backup Slides

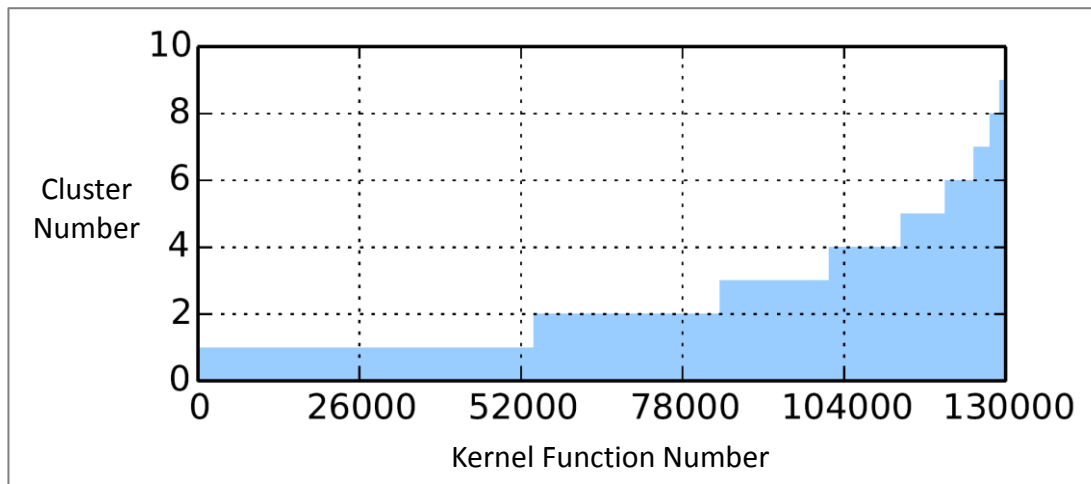
# Attack TrustZone from Kernel

---

- Example:
  - [Downgrade Attack on TrustZone](#) (see its references)

# Observations

- Most kernel functions are **stable** across devices and Android releases.

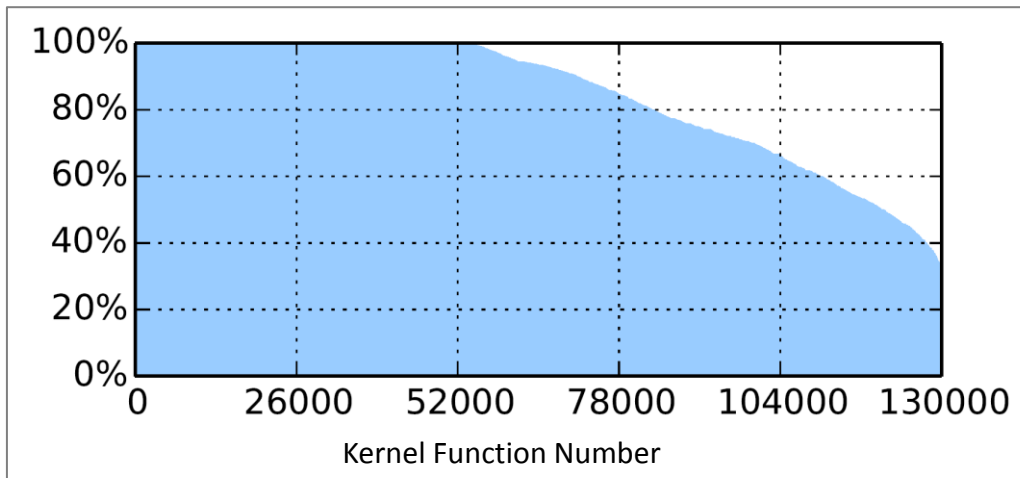


Number of syntax clusters for each function

About 40% of the shared functions have only one cluster, and about 80% of them have 4 clusters or less.

# Observations


- Most kernel functions are **stable** across devices and Android releases.



Percentage of kernels in the largest cluster for each function

For about 60% of shared functions, the largest cluster contains more than 80% of all the kernels that have this function.

# Symbolic Execution

- Challenges
  - Avoid path explosion
  - Impact to the environment
- Practical Solution
  - Non-local memory writes
  - Function calls (and their arguments)
  - Function return values
- Adaptation (e.g., mutate constants or offsets)
  - `foo(symbol_A + 4, 36)`  `foo(symbol_A + 8, 36)`

# Evaluation: Overall Performance

---

- *Complex* patch for *most frequent* syscall (`gettimeofday`) during web browsing
- Overall system performance overhead in this *extreme* situation: **0.9%**



# Example: CVE-2013-6123

---

```
1 static long msm_ioctl_server(struct file *
    file, void *fh, bool valid_prio, int cmd,
    void *arg)
2 {
3     ...
4     if (copy_from_user(&u_isp_event,
5         (void __user *)ioctl_ptr->ioctl_ptr,
6         sizeof(struct msm_isp_event_ctrl))) {
7         ...
8     }
9     ...
10 + if(u_isp_event.isp_data.ctrl.queue_idx<0
11 + || u_isp_event.isp_data.ctrl.queue_idx >=
12 +     MAX_NUM_ACTIVE_CAMERA) {
13 +     pr_err("%s: Invalid index %d\n",
14 +         __func__, u_isp_event.isp_data.
15 +         ctrl.queue_idx);
16 +     rc = -EINVAL;
17 +     return rc;
18 + }
19     ...
20 }
```

---

# Example: CVE-2013-6123

---

```
1 static long msm_ioctl_server(struct file *
    file, void *fh, bool valid_prio, int cmd,
    void *arg)
2 {
3     ...
4     if (copy_from_user(&u_ism_event,
5         (void __user *)ioctl_ptr->ioctl_ptr,
6         sizeof(struct msm_ism_event_ctrl))) {
7         ...
8     }
9     ...
10 + if (u_ism_event.ism_data.ctrl.queue_idx < 0
11 + || u_ism_event.ism_data.ctrl.queue_idx >=
12 +     MAX_NUM_ACTIVE_CAMERA) {
13 +     pr_err("%s: Invalid index %d\n",
14 +         __func__, u_ism_event.ism_data.
15 +         ctrl.queue_idx);
16 +     rc = -EINVAL;
17 +     return rc;
18 + }
19     ...
20 }
```

---

# Example: CVE-2013-6123

---

```
1 static long msm_ioctl_server(struct file *
    file, void *fh, bool valid_prio, int cmd,
    void *arg)
2 {
3     ...
4     if (copy_from_user(&u_isp_event,
5         (void __user *)ioctl_ptr->ioctl_ptr,
6         sizeof(struct msm_isp_event_ctrl))) {
7         ...
8     }
9     ...
10 + if(u_isp_event.isp_data.ctrl.queue_idx<0
11 + || u_isp_event.isp_data.ctrl.queue_idx >=
12 +     MAX_NUM_ACTIVE_CAMERA) {
13 +     pr_err("%s: Invalid index %d\n",
14 +         __func__, u_isp_event.isp_data.
15 +         ctrl.queue_idx);
16 +     rc = -EINVAL;
17 +     return rc;
18 + }
19     ...
20 }
```

---

# Example: CVE-2013-6123

---

```
1 static long msm_ioctl_server(struct file *
    file, void *fh, bool valid_prio, int cmd,
    void *arg)
2 {
3     ...
4     if (copy_from_user(&u_ism_event,
5         (void __user *)ioctl_ptr->ioctl_ptr,
6         sizeof(struct msm_ism_event_ctrl))) {
7         ...
8     }
9     ...
10 + if(u_ism_event.ism_data.ctrl.queue_idx<0
11 + || u_ism_event.ism_data.ctrl.queue_idx >=
12 +     MAX_NUM_ACTIVE_CAMERA) {
13 +     pr_err("%s: Invalid index %d\n",
14 +         __func__, u_ism_event.ism_data.
15 +         ctrl.queue_idx);
16 +     rc = -EINVAL;
17 +     return rc;
18 + }
19     ...
20 }
```

---

# Example: CVE-2013-6123

---

```
1 static long msm_ioctl_server(struct file *
    file, void *fh, bool valid_prio, int cmd,
    void *arg)
2 {
3     ...
4     if (copy_from_user(&u_ism_event,
5         (void __user *)ioctl_ptr->ioctl_ptr,
6         sizeof(struct msm_ism_event_ctrl))) {
7         ...
8     }
9     ...
10 + if(u_ism_event.ism_data.ctrl.queue_idx<0
11 + || u_ism_event.ism_data.ctrl.queue_idx >=
12 +     MAX_NUM_ACTIVE_CAMERA) {
13 +     pr_err("%s: Invalid index %d\n",
14 +         __func__, u_ism_event.ism_data.
15 +         ctrl.queue_idx);
16 +     rc = -EINVAL;
17 +     return rc;
18 + }
19     ...
}
```

---

# Example: CVE-2013-6123

---

```
1 static long msm_ioctl_server(struct file *
    file, void *fh, bool valid_prio, int cmd,
    void *arg)
2 {
3     ...
4     if (copy_from_user(&u_isp_event,
5         (void __user *)ioctl_ptr->ioctl_ptr,
6         sizeof(struct msm_isp_event_ctrl))) {
7         ...
8     }
9     ...
10 + if(u_isp_event.isp_data.ctrl.queue_idx<0
11 + || u_isp_event.isp_data.ctrl.queue_idx >=
12 +     MAX_NUM_ACTIVE_CAMERA) {
13 +     pr_err("%s: Invalid index %d\n",
14 +         __func__, u_isp_event.isp_data.
15 +         ctrl.queue_idx);
16 +     rc = -EINVAL;
17 +     return rc;
18 + }
19     ...
20 }
```

---

# You May Also Like

---

- A time machine to locate vulnerabilities:
  - [Pinpointing Vulnerabilities](#)
- Protect your computer by encrypting memory all the time:
  - [Secure In-Cache Execution](#)
- Fine-grained dynamic ASLR during runtime:
  - [Remix: On-demand Live Randomization](#)