InstaGuard: Instantly Deployable Hot-patches for Vulnerable System Programs on Android

Yaohui Chen, Yuping Li, Long Lu, Yueh-Hsun Lin,
Hayawardh Vijayakumar, Zhi Wang, Xinming Ou

Northeastern University
Cybersecurity and Privacy Institute

USF
UNIVERSITY OF SOUTH FLORIDA

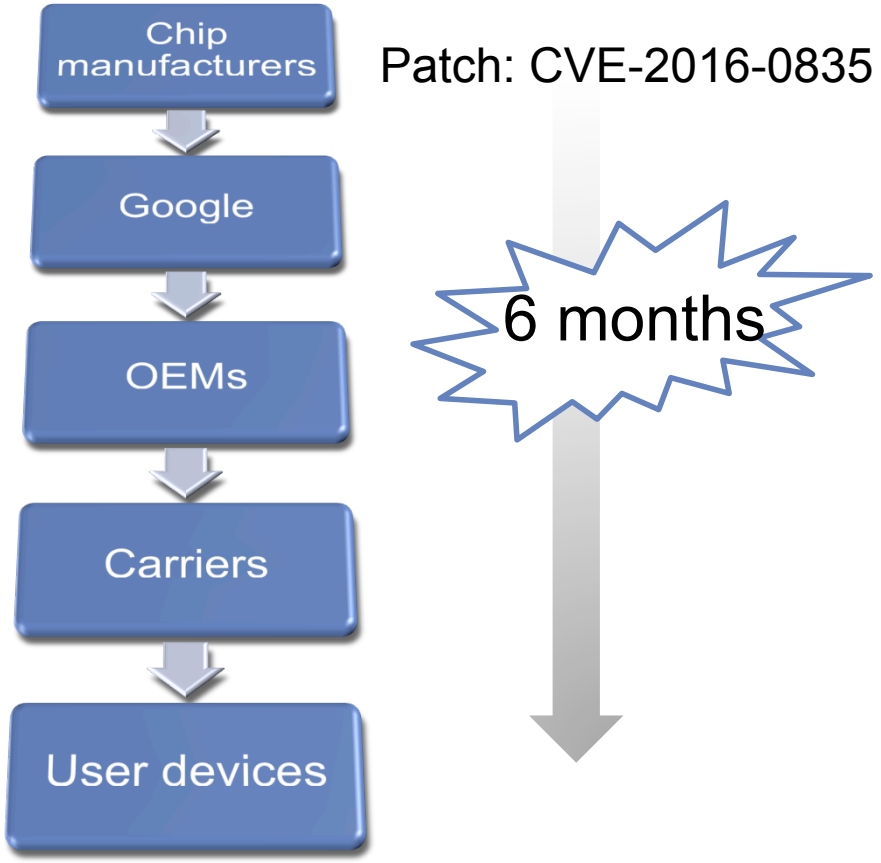
FSU

SAMSUNG
SAMSUNG RESEARCH AMERICA

JD.COM 京东

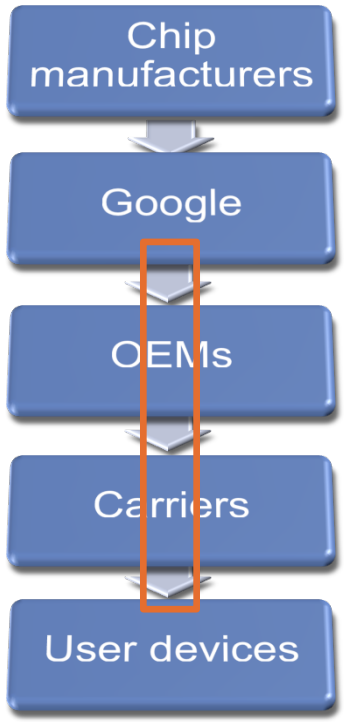
Prolonged Android System Security Updates

- Require lengthy tests from various parties
  - Security
  - Compatibility
- Done by each go-to-market partners
  - It is justified, but too time-consuming



To Accelerate the Process

- Monthly Security Maintenance Release (SMR)
- Project Treble
  - Isolate SoC vendors from Google and OEMs when preparing new OS updates
  - OS updates still need to be done and tested by OEMs and carriers
- Hot patches
  - KARMA [Chen et al, Usenix Security'17]
  - Patchdroid [Mulliner et al, ACSAC'13]
  - **InstaGuard**



Fast patch development

Carrier pass-through

Background – Code update v.s Policy update

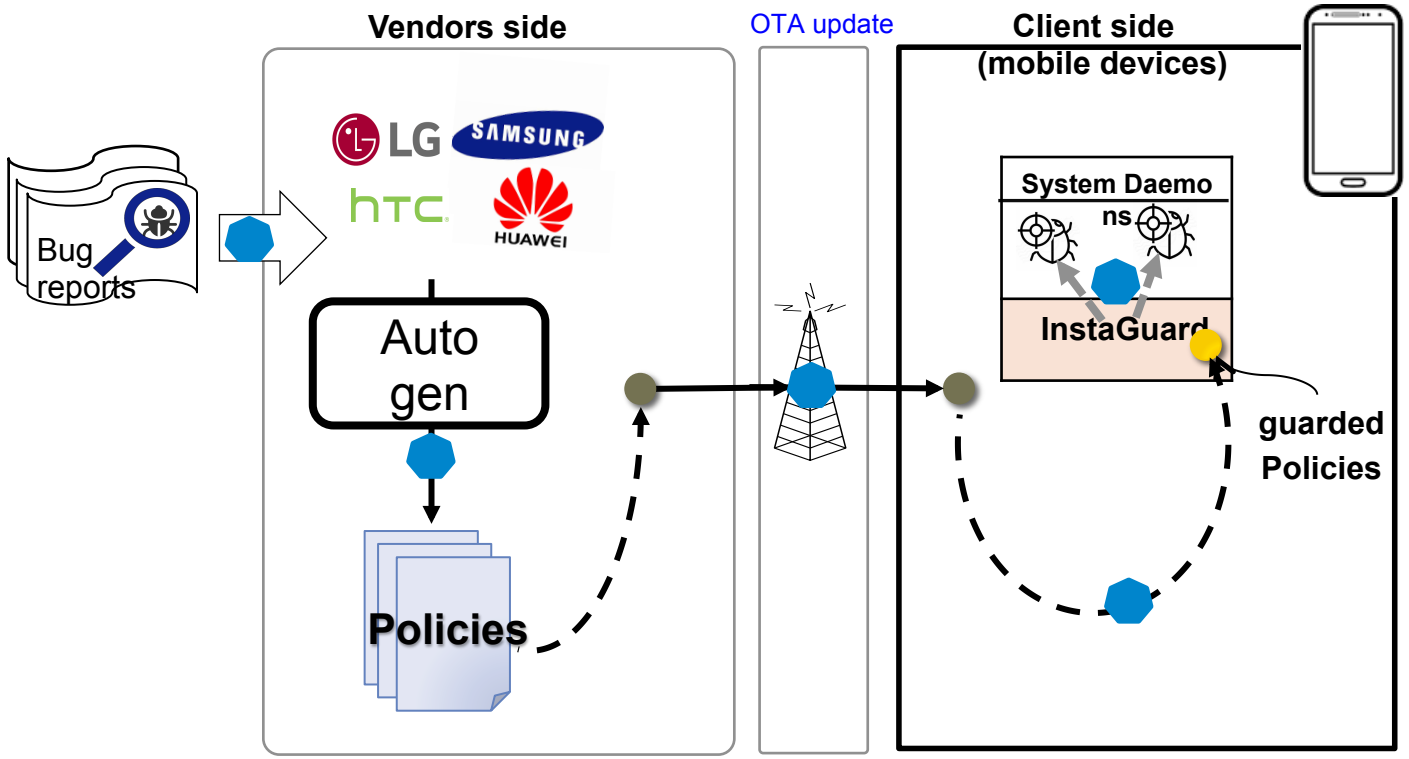
- Matured and commercialized **code update based** hot patch solutions
  - Microsoft Hotfix
  - Ubuntu Livepatch
  - Not carrier-passthrough

SPD SE Android Policy Distribution

ve than code
through
is in-place and

Introducing InstaGuard

- Approach
  - Utilize **policy-driven update** to timely mitigate critical **user-level system** vulnerabilities
- Key difference to traditional hot patches
  - Non-code update
  - Restrictiveness (fail-safe)



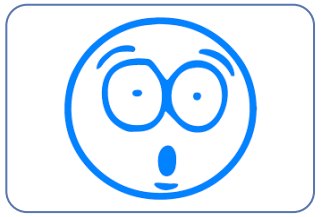
Threat Model

Trusted

- Kernel
- Hardware

Benign but vulnerable

- User-level system daemons
- User-level system libraries



Can’t handle

- Zero-day vulnerabilities
- Compromised system daemons

Unique Challenges

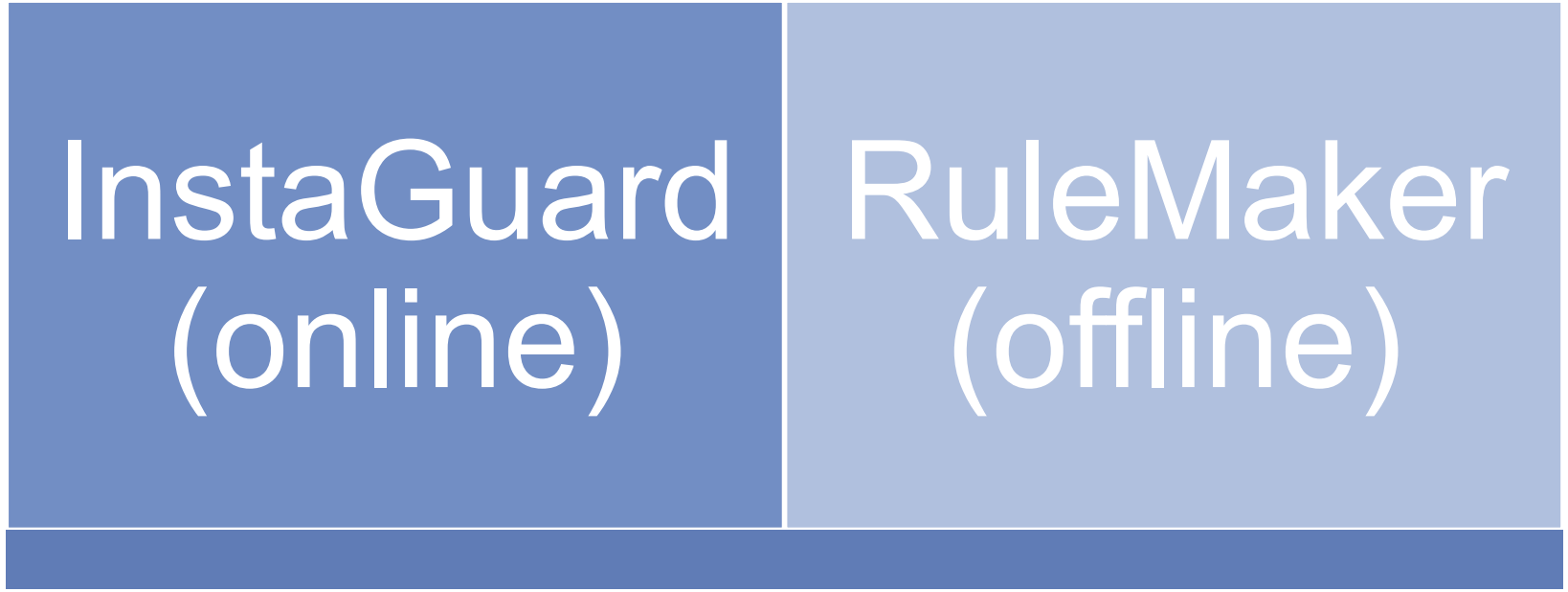
Policy language

- Expressive to describe various kinds of vulnerabilities
- Restrictive to be fail-safe

Policy generation

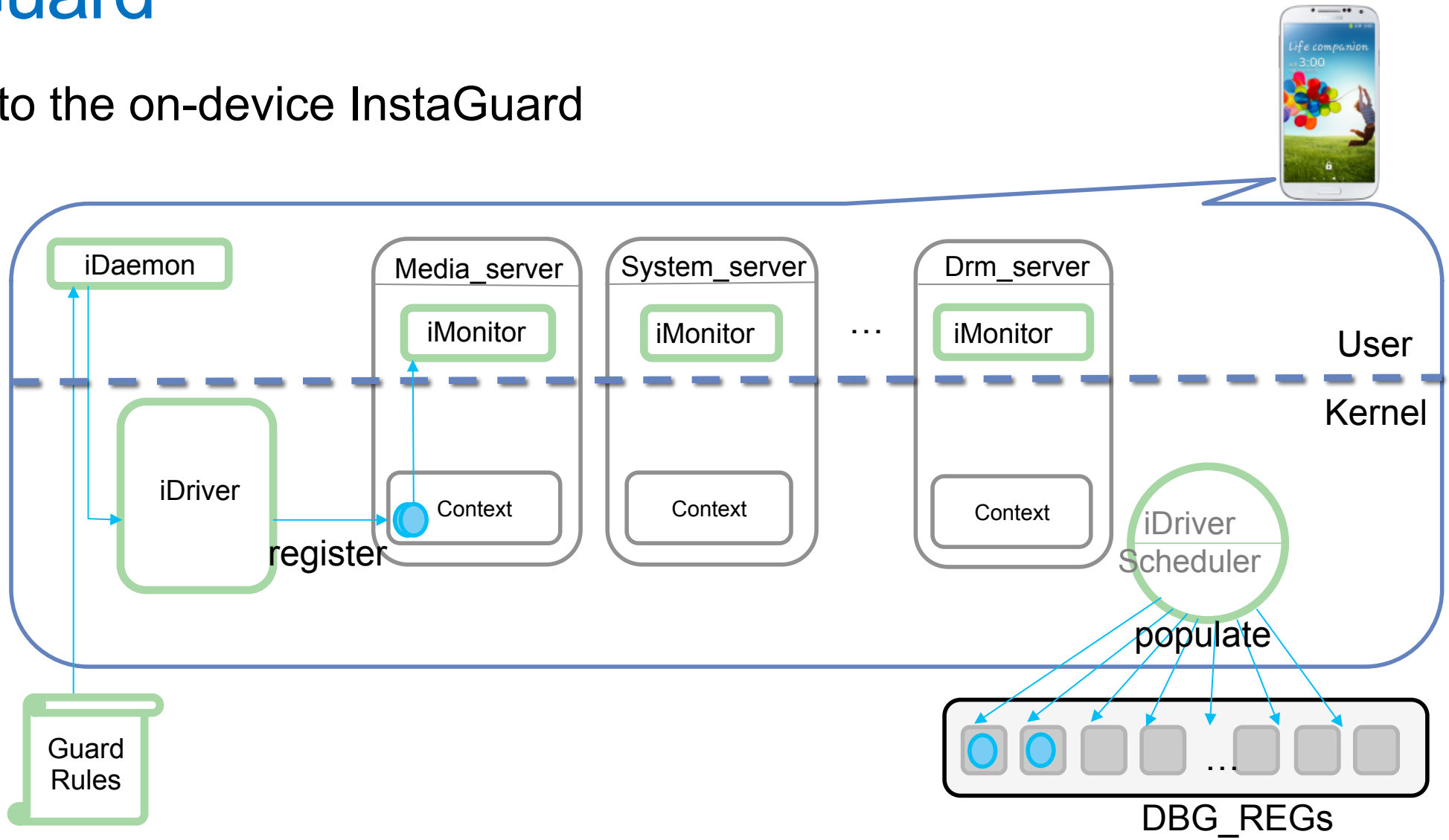
- Automated policy generation

System Overview

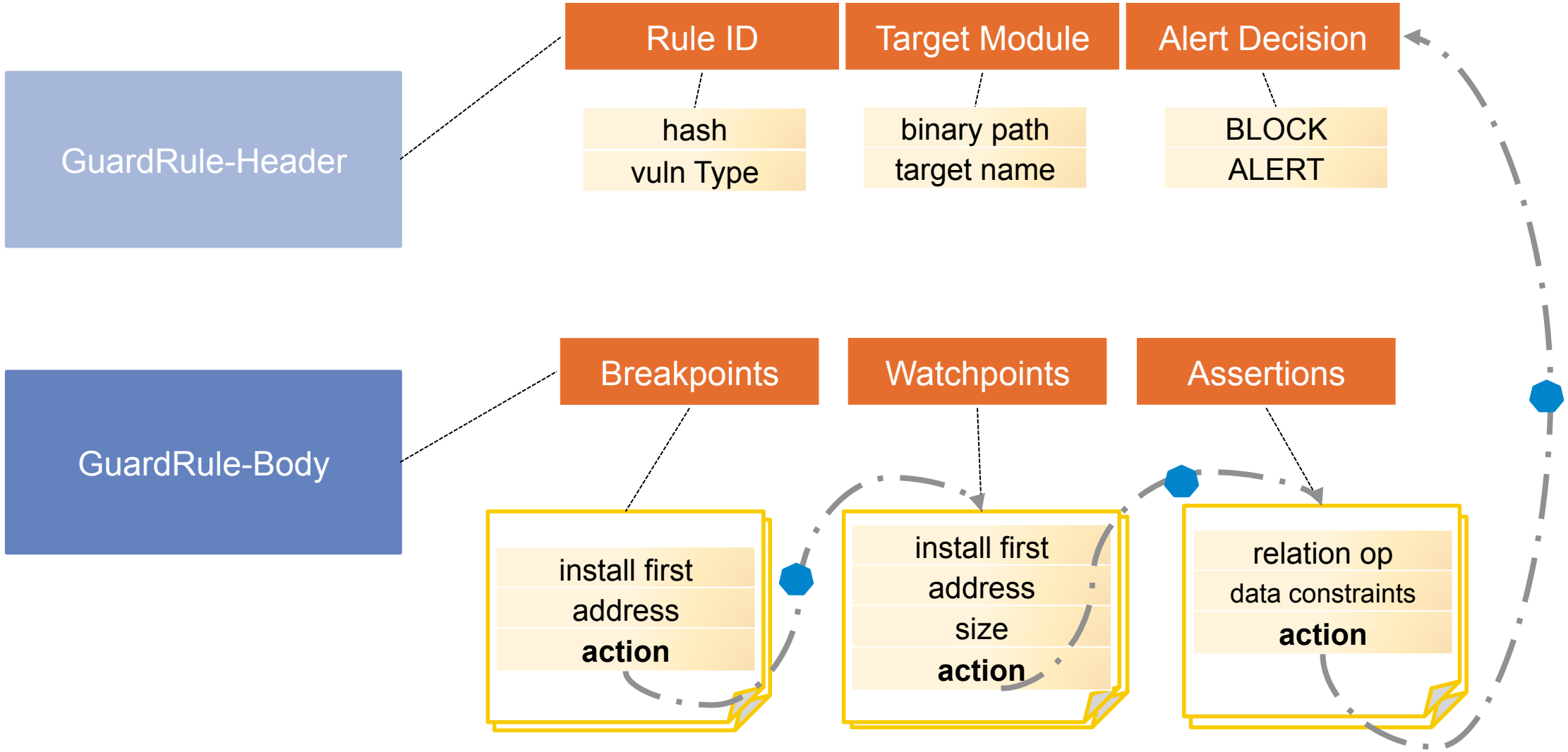


InstaGuard

- Zoom in to the on-device InstaGuard



GuardRule Basic Format



GuardRule

Expressive: precisely capture control and data properties

- Precisely describe various kinds of vulnerabilities
  - logic bug, integer overflow, buffer overflow, out-of-bound access, use-after-free and race conditions

Fail-safe: buggy (or malicious) rules can only cause DoS

- The InstaGuard mechanism simply do not support any intrusive operations.

InstaGuard in Action

- CVE-2016-3861 (logic bug)

```
1  ssize_t utf16_to_utf8_length(const char16_t *src,
     ↪  size_t src_len)
2  {
3    ...//sanity checks
4    size_t ret = 0;
5    const char16_t* const end = src+src_len;
6    while (src < end) {
7      if ((*src & 0xFC00) == 0xD800
8          && (src + 1) < end
9          && (*++src & 0xFC00) == 0xDC00) {
10       // surrogate pairs are always 4 bytes.
11         ret += 4;
12         src++;
13     } else {
14         ret += utf32_codepoint_utf8_length
             ↪  ((char32_t) *src++);
15     }
16   }
17     return ret;
18 }
```

```
1  <rules>
2    <rule cve="CVE-2016-3861">
3      <module_name>libutils.so</module_name>
4      <decision>BLOCK</decision>
5      <binary_path>/system/bin/mediaserver</binary_path>
6      <break_points>
7        <break_point first=true, id=0>
8          <!--binary address corresponding to line 9
               in Listing 1-->
9          <address>0x08055000</address>
10         <!--next action: activating assertion
               primitive with id 0-->
11         <action> VERIFY AS#0</action>
12       </break_point>
13     </break_points>
14     <assertions>
15       <!--if assertion evaluate to true InstaGuard
             BLOCK the execution as node ?decision?
             speicify-->
16       <assertion id=0, action=decision>
17         <data_constraints>
18           <data_constraint>
19             <ops>NE</ops>
20             <left_exp>
21               <!--retrieval rule for *src-->
22               <node id=0>reg_2_32</node>
23               <node id=1>const_0xFC00</node>
24               <node id=2>bitwise_and</node>
25             </left_exp>
26             <right_exp>
27               <node>const_0xDC00</node>
28             </right_exp>
29           </data_constraint>
30         </data_constraints>
31       </assertion>
32     </assertions>
33   </rule>
34 </rules>
```

InstaGuard in Action Cont.

- CVE-2016-7911 (race condition)

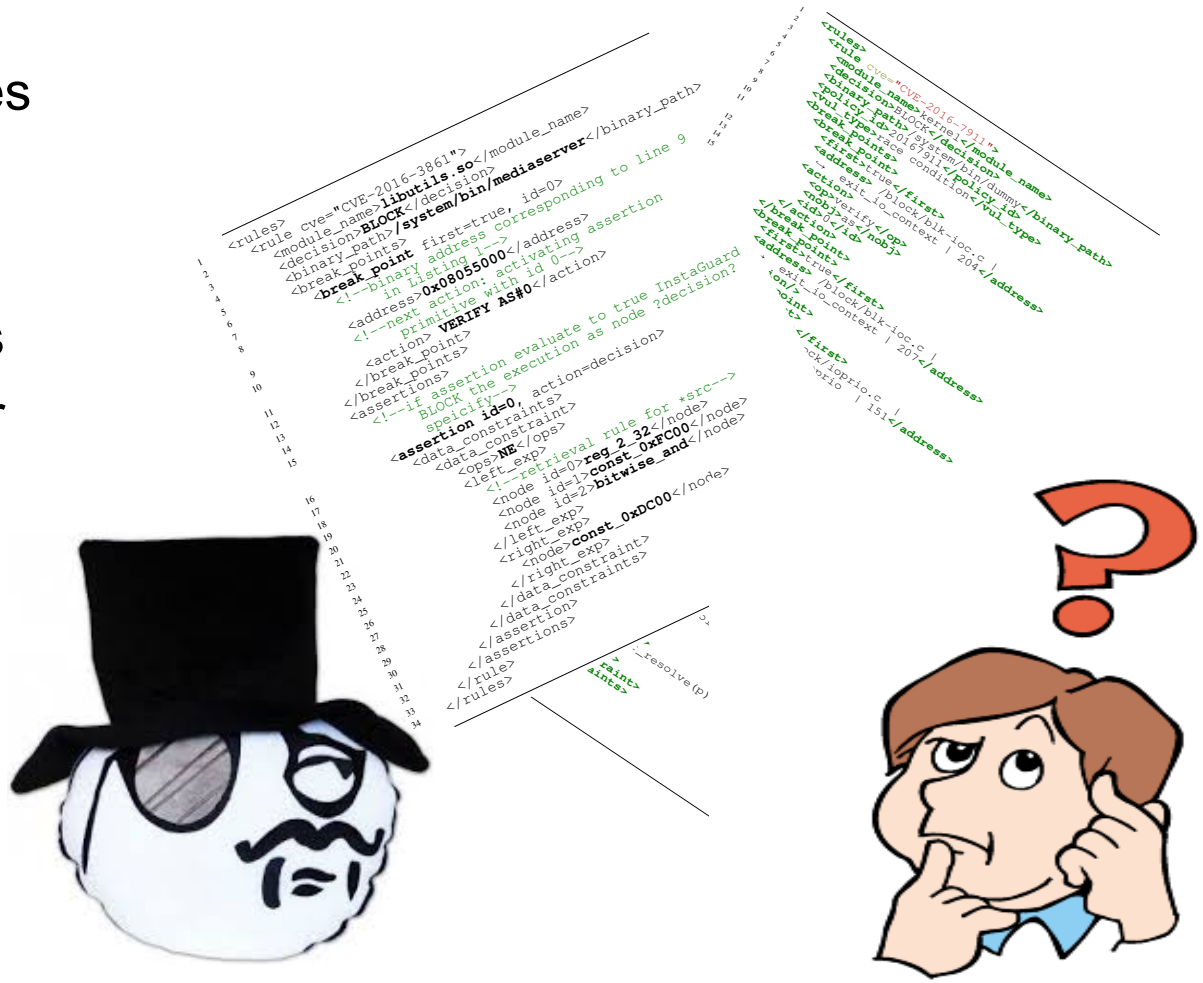
```
1  // src:/block/blk-ioc.c
2  void
3  exit_io_context(struct task_struct *task)
4  {
5          struct io_context *ioc;
6
7          task_lock(task);
8          ioc = task->io_context;
9          task->io_context = NULL;
10         task_unlock(task);
11
12         atomic_dec(&ioc->nr_tasks);
13         put_io_context_active(ioc);
14 }
15
16
17 // src:/block/ioprio.c
18 static int
19 get_task_ioprio(struct task_struct *p)
20 {
21         int ret;
22
23         ret = security_task_getioprio(p);
24         if (ret)
25                 goto out;
26         ret = IOPRIO_PRIO_VALUE(IOPRIO_CLASS_NONE,
             ↪  IOPRIO_NORM);
27         if (p->io_context)
28                 ret = p->io_context->ioprio;
29 out:
30         return ret;
31 }
```

- Con
  - BP
  - BP
  - BP
  - BP
  - AS

```
<rules>
  <rule cve="CVE-2016-7911">
    <module_name>kernel</module_name>
    <decision>BLOCK</decision>
    <binary_path>/system/bin/dummy</binary_path>
    <policy_id>20167911</policy_id>
    <vul_type>race condition</vul_type>
    <break_points>
      <break_point>
        <first>true</first>
        <address> /block/blk-ioc.c |
          ↪  exit_io_context | 204</address>
        <action>
          <op>verify</op>
          <nobj>as</nobj>
          <id>0</id>
        </action>
      </break_point>
      <break_point>
        <first>true</first>
        <address> /block/blk-ioc.c |
          ↪  exit_io_context | 207</address>
        <action/>
      </break_point>
      <break_point>
        <first>true</first>
        <address> /block/ioprio.c  |
          ↪  get_task_ioprio  | 151</address>
        <action>
          <op>verify</op>
          <nobj>as</nobj>
          <id>0</id>
        </action>
      </break_point>
      <break_point>
        <first>true</first>
        <address> /block/ioprio.c  |
          ↪  get_task_ioprio  | 154</address>
        <action/>
      </break_point>
    </break_points>
    <watch_points/>
    <assertions>
      <assertion>
        <num>1</num>
        <relation>AND</relation>
        <data_constraints>
          <data_constraint>
            <ops>NE</ops>
            <left_exp>
              <node>
                <id>0</id>
                <type>var_resolve(task)</type>
              </node>
            </left_exp>
            <right_exp>
              <node>
                <id>0</id>
                <type>var_resolve(p)</type>
              </node>
            </right_exp>
          </data_constraint>
        </data_constraints>
      </assertion>
    </assertions>
  </rule>
</rules>
```

GuardSpec and RuleMaker

- GuardSpec
  - High-level **vulnerability description**
  - Hides the details about InstaGuard primitives
- RuleMaker
  - **Automatically** synthesizes GuardRules
  - Conversion based on empirical experiences
  - Several implementation challenges in paper



GuardSpec Showcases

CVE-2016-3861 (logic bug)

```
[common]
ID = CVE-2016-3861
binary_path = /system/bin/mediaserver
module_name =libutils.so
decision = BLOCK

[logic bug]
vul_location = system/core/libutils/Unicode.cpp |
  ↪  utf16_to_utf8_length | 411
lexp = *src & 0xFC00
rexp = 0xDC00
relation_op = NE
```

CVE-2016-7911 (race condition)

```
[common]
ID = CVE-2016-7911
binary_path = /system/bin/dummy
module_name =kernel
decision = BLOCK

[race condition]
racer_location1 = task : /block/blk-ioc.c |
  ↪  exit_io_context | 204 : /block/blk-ioc.c |
  ↪  exit_io_context | 207
racer_location2 = p : /block/ioprio.c  |
  ↪  get_task_ioprio  | 151 : /block/ioprio.c  |
  ↪  get_task_ioprio  | 154
```

CVE-2016-3871 (buffer overflow)

```
[common]
ID = CVE-2016-3871
binary_path = /system/bin/mediaserver
module_name =libstagefright_soft_avcenc.so
decision = BLOCK

[buffer overflow]
buf_name = outHeader->pBuffer
buf_size = outHeader->nAllocLen
vul_location =
  ↪  libstagefright/codecs/mp3dec/SoftMP3.cpp
  ↪  |SoftMP3::internalGetParameter | 303
```

CVE-2016-3895 (integer overflow)

```
[common]
ID = CVE-2016-3895
binary_path = /system/bin/surfaceflinger
module_name = libui.so
decision = BLOCK

[integer overflow]
involved_vars = numRects,Rect
overflow_exp =  numRects * Rect
overflow_dir = MAX
trigger_value = 0xffffffff
vul_location = frameworks/native/libs/ui/Region.cpp
  ↪  | Region:flatten | 794
```

Evaluation

- Expressiveness
  - Can InstaGuard generically block different kinds of real-world vulnerabilities?
- Ease of use
  - How easy is it to make use of the InstaGuard framework?
- Overhead
  - What are the runtime and memory overheads of generated policies?

Expressiveness Evaluation

- 30 critical framework vulnerabilities from Android security bulletin from 2016
  - 28/30 GuardSpec are less than 10 lines
  - Info leak is not easy to mitigate without risking availability lost
  - UaF and Race conditions mostly find their root causes as logical bugs

2016-0811, 2016-3822, 2016-0803, 2016-3744, 2016-0815, 2016-0837, 2016-2463, 2016-0827, 2016-0849, 2016-2428, 2016-3895, 2016-3872, 2016-3819, 2016-2451, 2016-2484, 2016-3863, 2016-2485, 2016-1474, 2016-3871, 2016-2494, 2016-0836, 2016-0840, 2016-6707, 2016-3861, 2016-0835, ***2016-2417***, ***2016-2419***, 2016-2418, 2016-3826, 2016-0816

Integer overflow (13)
Buffer overflow (7)
Logic bug(7)
OOB(2)
UaF(1)

Ease of Use

- Facilitated by security researchers from Samsung
- Tested 4 different categories of bugs
- Task: write GuardSpec and then synthesize GuardRule using RuleMaker

	Vulnerability type	GuardSpec compose time (mins)	GuardSpec Line#	Synthesized GuardRule Line#
CVE-2016-3895	Integer Overflow	40	9	52
CVE-2016-0836	Out-of-Bound	20	7	45
CVE 2016-3861	Logic Bug	15	8	55
CVE-2015-1474	Buffer Overflow	15	7	94
Avg.		22.5	7.75	61.5

Runtime and Memory Overheads

- Unit tests
- Stacked GuardRules

	Used Primitives	Vulnerability type	Memory Overhead(%)	Runtime Slowdown(%)
CVE-2016-3895	(1x)BP, (1x)AS	Integer Overflow	0.37%	2.89%
CVE-2016-0836	(1x)BP, (1x)AS	Out-of-Bound	4.11%	3.27%
CVE 2016-3861	(1x)BP, (1x)AS	Logic Bug	1.08%	2.70%
CVE-2015-1474	(2x)BP, (1x)WP, (1x)AS	Buffer Overflow	1.19%	1.94%
Avg.			1.69%	2.70%
Wrapper service	**All of the above**	**All of the above**	**0.48%**	**9.73%**

Conclusion

- InstaGuard aims to make timely hot-patches available to users
  - Leverage **policy update** instead of code update
  - GuardRules are **expressive** yet **restrictive**
- RuleMaker synthesizes GuardRule from GuardSpec
  - GuardSpecs are high-level **vulnerability description**
  - Hides low-level InstaGuard primitives from user
- Evaluation on critical system vulnerabilities reveals minor overheads

Yaohui Chen
chen.yaoh@husky.neu.edu

More Details about Data constraints

Necessary Data Constraint Operators

- logical and, or operators: &&, ||
- relational operators: ==, !=, <, >, <=, >=
- binary operators: +, -, *, /, %, &, |, ^
- unary operators: +, -, !(logical negation), ~(bitwise not)

Data Constraint Representation

- primaryExpression:
  - basicOperands [target variables, const]
  - **unaryOps** basicOperands [-,!,~]
  - primaryExpression **binaryOps** primaryExpression [+,-,*,/,%,&,|,^]

- **conditionalExpression**:
  - primaryExpression **relationalOps** primaryExpression [==,!=,>,<,>=,<=]

- Conditional expression format: (type), (ops), (leftExp), (rightExp);

Data constraints format

- <data_constraints>
  - <type>int64</type>
  - <relational_operator>==</relational_operator>
  - <left_exp>
    - <ops>*,+</ops>
    - <var_categories>global,local,local</var_categories>
    - <operands>x,y,z</operands>
  - </left_exp>
  - <right_exp>
    - <ops></ops>
    - <var_categories>const</var_categories>
    - <operands>2</operands>
  - </right_exp>
- <data_constraints>