Android Resiliency Defense Strategy

Felipe Boeira
Samsung Research Institute

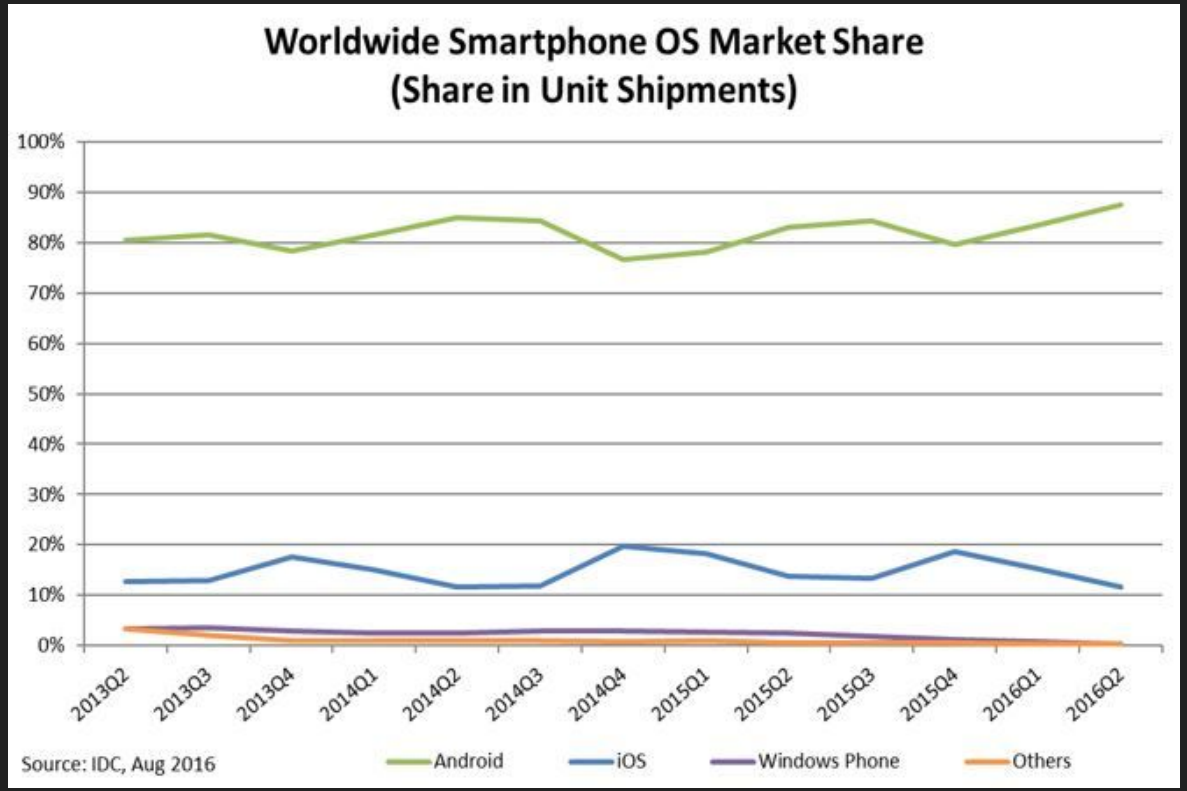
whoami

- Penetration Tester
- Cyberfraud Incident Response
- Security Reseacher
- Mobile, Linux Kernel & IoT
- Security Team: Breno Silva, Felipe Boeira e Pedro Minatel

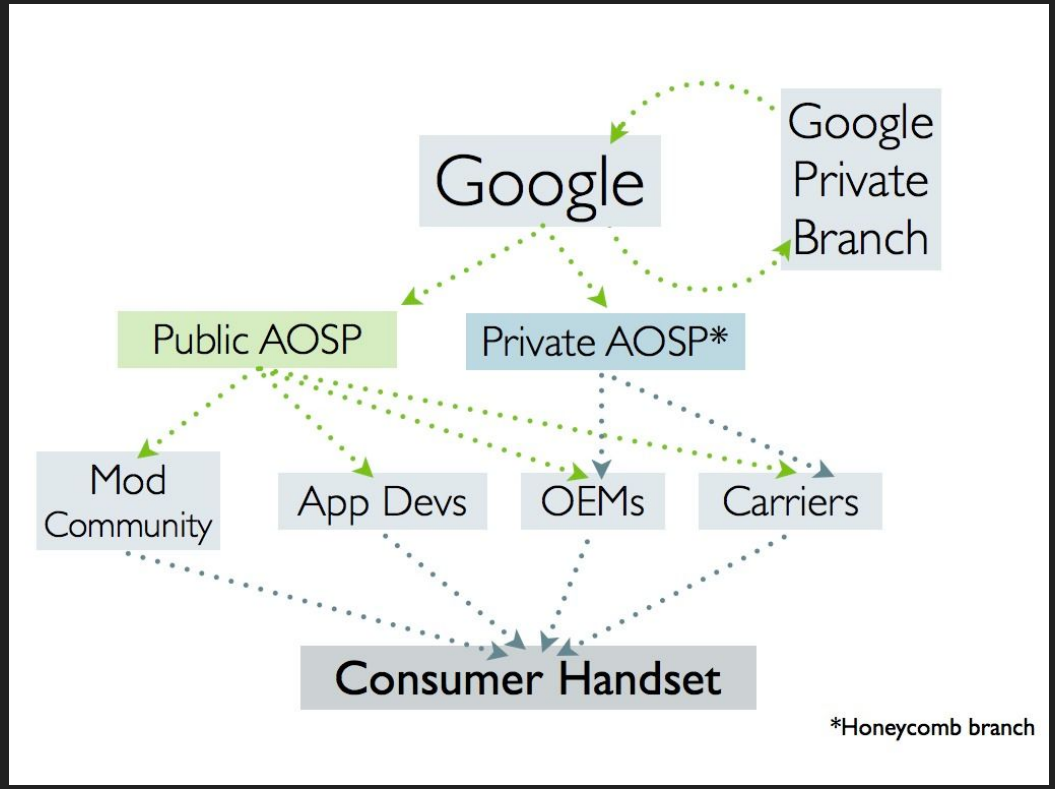
Agenda

- Motivação
- Background
- Visão Geral de Ataques ao Kernel
- Host Resiliency Defense Strategy
- Considerações Finais

Mobile OS Market Share

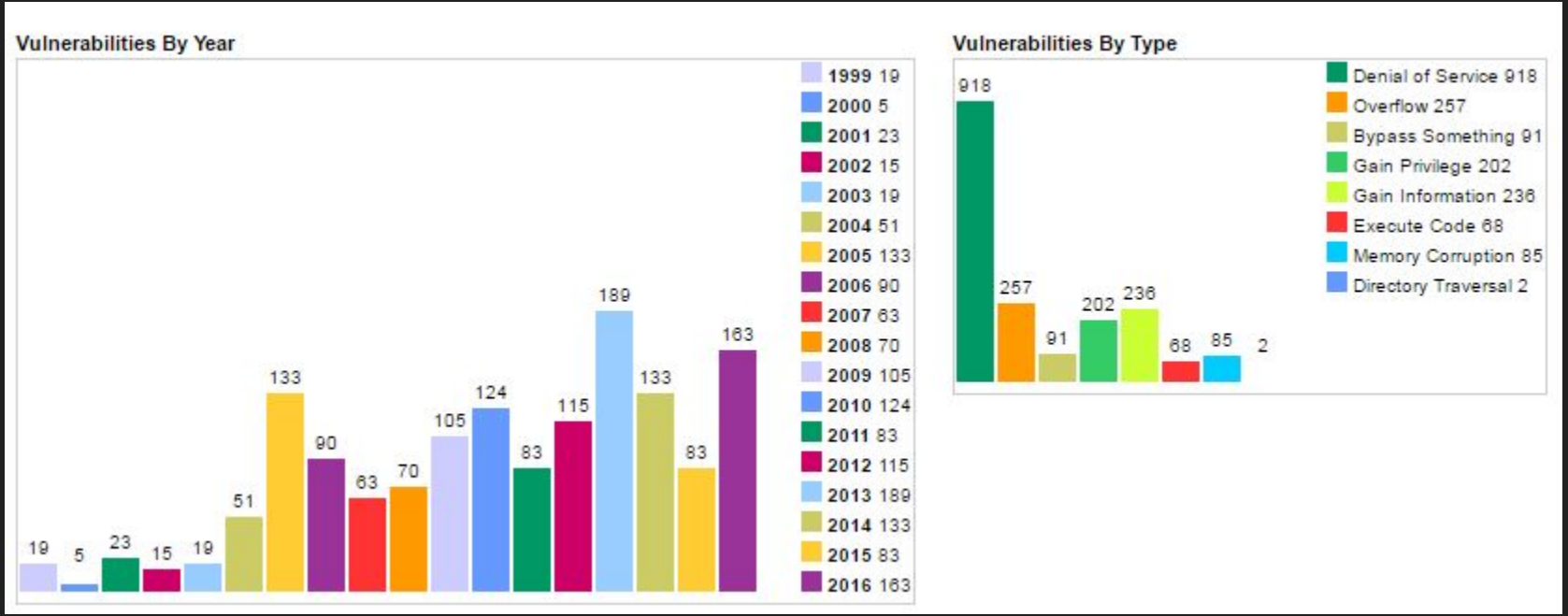


Android Patching Lifecycle



Source: blog.lookout.com

Linux Kernel CVE Statistics



Source: cvedetails.com

Android Security Background

Kernel Memory Protection

- Address Space Layout Randomization (ASLR)
- Mmap_min_addr
- Kptr_restrict
- Dmesg_restrict

Code Execution Protection

- Stack Protector
- FORTIFY SOURCE
- ELF RELRO / BIND_NOW
- NX bit
- Privileged eXecute Never - PXN (aarch64)

Additional Protections

- SELinux Enforcing
- UID Sandbox & Privilege Separation
- Dm-verity

NOT ENOUGH AGAINST
KERNEL EXPLOITATION!

Linux Task Struct

```
1460 struct task_struct {
1461         volatile long state;    /* -1 unrunnable, 0 runnable, >0 stopped */
1462         void *stack;
1463         atomic_t usage;
1464         unsigned int flags;     /* per process flags, defined below */
1465         unsigned int ptrace;
…
1633 /* process credentials */
1634         const struct cred __rcu *real_cred; /* objective and real subjective task
1635                                              * credentials (COW) */
1636         const struct cred __rcu *cred;  /* effective (overridable) subjective task
1637                                              * credentials (COW) */
1638         char comm[TASK_COMM_LEN]; /* executable name excluding path
1639                                      - access with [gs]et_task_comm (which lock
1640                                        it with task_lock())
1641                                      - initialized normally by setup_new_exec */

...
```

Linux Struct Cred

```
118 struct cred {
119         atomic_t        usage;
120 #ifdef CONFIG_DEBUG_CREDENTIALS
121         atomic_t        subscribers;    /* number of processes subscribed */
122         void            *put_addr;
123         unsigned        magic;
124 #define CRED_MAGIC      0x43736564
125 #define CRED_MAGIC_DEAD 0x44656144
126 #endif
127         kuid_t          uid;            /* real UID of the task */
128         kgid_t          gid;            /* real GID of the task */
129         kuid_t          suid;           /* saved UID of the task */
130         kgid_t          sgid;           /* saved GID of the task */
131         kuid_t          euid;           /* effective UID of the task */
132         kgid_t          egid;           /* effective GID of the task */
133         kuid_t          fsuid;          /* UID for VFS ops */
134         kgid_t          fsgid;          /* GID for VFS ops */
```

...

Privilege Escalation

- commit_creds(prepare_kernel_cred(0));
- Kernel Memory Leak! (SP & ~(THREAD_SIZE -1))
- is_cpu_timer_valid()
- Addr_limit
- Arbitrary Kernel read/write

OWNAGE!

Kernel struct cred overwrite!

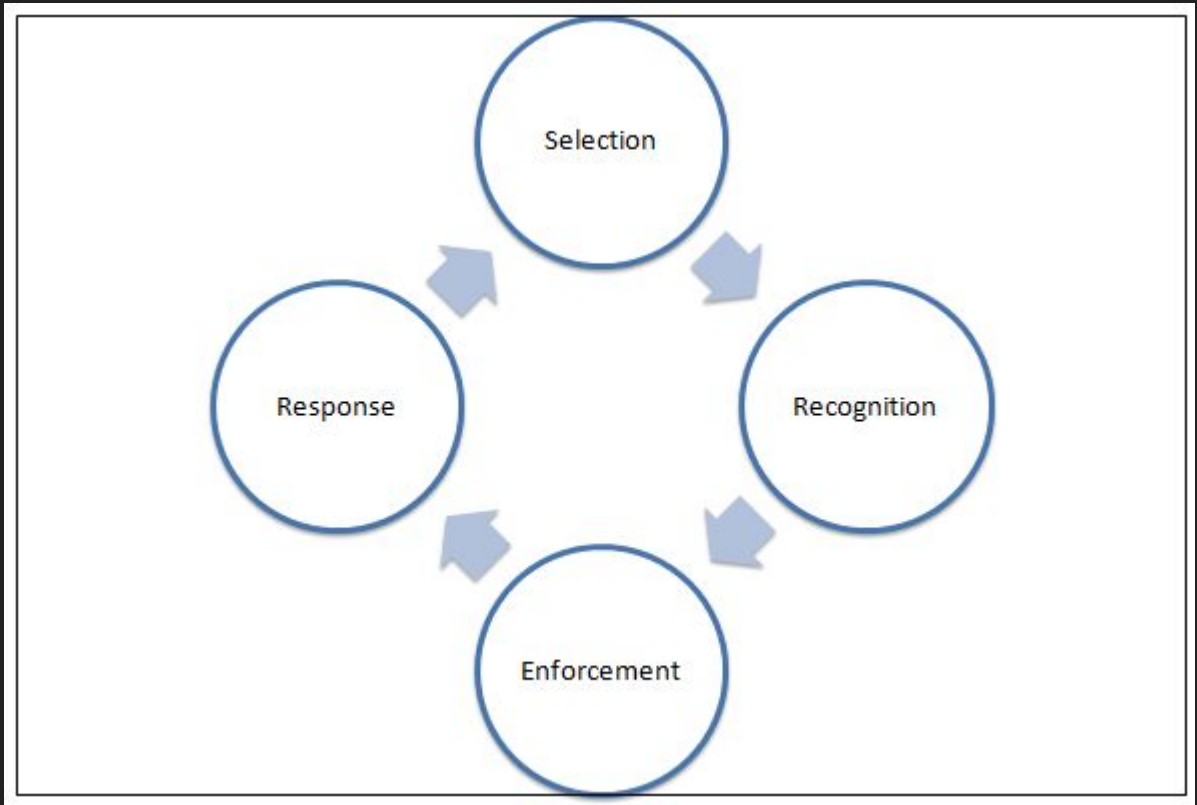
```
cred->uid = 0;
cred->gid = 0;
cred->suid = 0;
cred->sgid = 0;
cred->euid = 0;
cred->egid = 0;
cred->fsuid = 0;
cred->fsgid = 0;
```

```
cred->cap_inheritable.cap[0] = 0xffffffff;
cred->cap_inheritable.cap[1] = 0xffffffff;
cred->cap_permitted.cap[0] = 0xffffffff;
cred->cap_permitted.cap[1] = 0xffffffff;
cred->cap_effective.cap[0] = 0xffffffff;
cred->cap_effective.cap[1] = 0xffffffff;
cred->cap_bset.cap[0] = 0xffffffff;
cred->cap_bset.cap[1] = 0xffffffff;
```

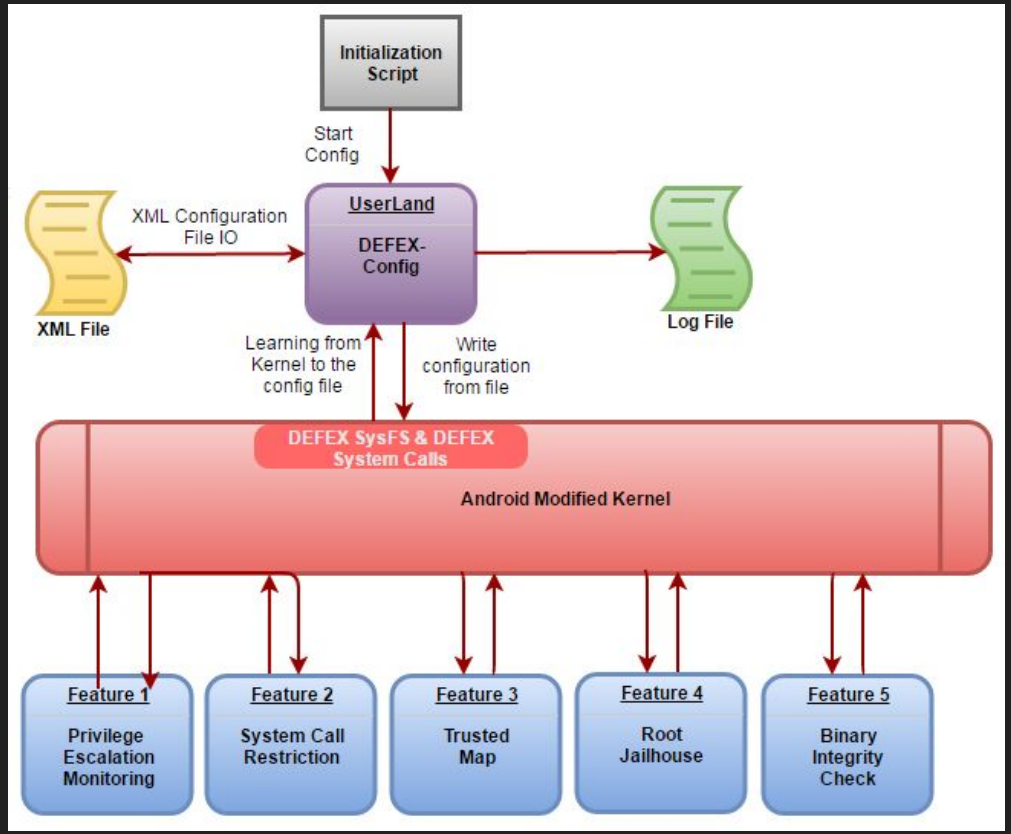
```
# id

uid=0(root) gid=0(root) groups=0(root)
```

Host Resiliency Defense Strategy

Host Resistance



Defex Architecture



Privilege Escalation Monitoring

- Inicia o monitoramento de um processo quando é criado por chamadas clone()/fork() ou exec()
- O UID original do processo é armazenado em uma área de memória do kernel quando ele é criado
- A cada execução de uma chamada de sistema a credencial é comparada com o valor salvo na matriz do Defex
- Caso uma alteração na credencial seja detectada, o processo é morto

System Call Restriction

- Chamadas de sistema são um vetor comum de ataque
- Redução da superfície de ataque
- Restrição de chamadas de sistema de acordo com níveis de execução: root, system e user
- Defex realiza a aprendizagem dos perfis de execução de chamadas de sistema para criação da configuração de enforcement

Trusted Map

- Após a escalação de privilégios, geralmente são executados comandos para persistência da infecção ou é feito o upload de novos binários
- Assinaturas são criadas com base nos parâmetros de execução do comando e das variáveis de ambiente
- A cada chamada a execve(), a assinatura é calculada e verificada

Root Jailhouse

- Identificação da característica da partição *data* no Android
- Bloqueia a execução de chamadas de sistema *execve, open, openat, fork, vfork, clone* pelo usuário *root* na partição *data*
- Mesmo após uma escalação de privilégios com sucesso, um atacante não poderá interagir com a partição de dados do *smartphone*

Binary Integrity Check

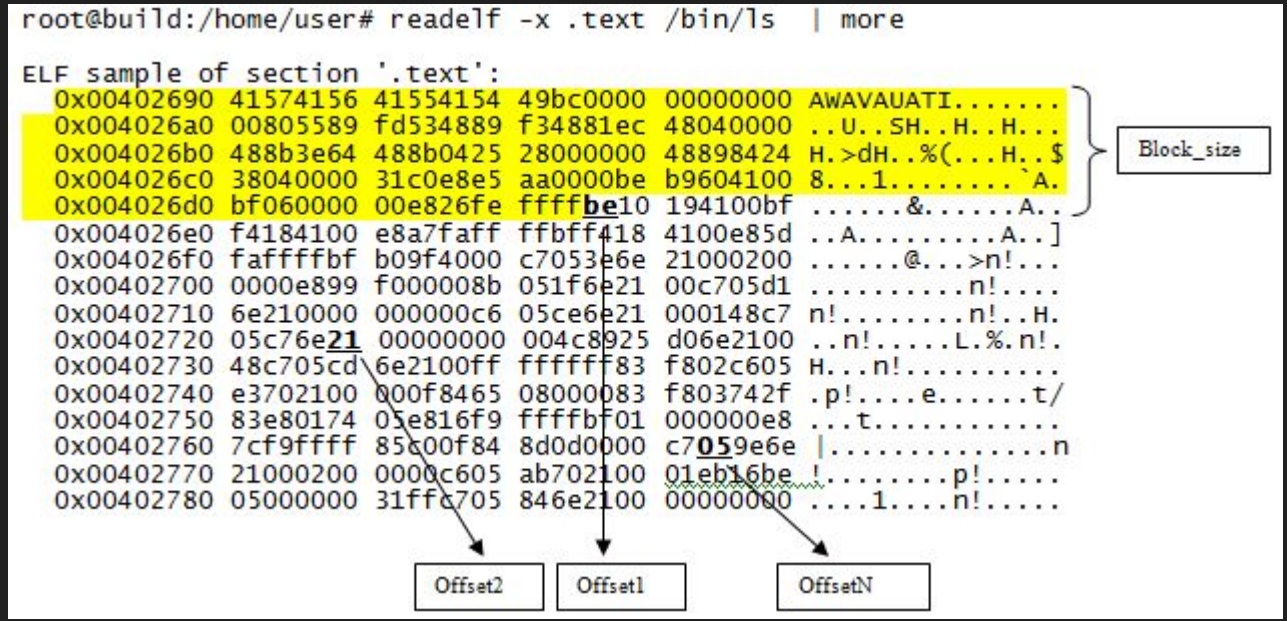
- Estabelecimento de um algoritmo eficiente o qual seja possível aplicar em toda chamada de sistema

- O algoritmo realiza a verificação de integridade de determinados *offsets* da seção de código *.text* do binário ELF

- Caso algum *opcode* seja divergente, o processo é morto. Caso o binário não tenha assinatura, o processo também é morto.

Binary Integrity Check

- O BLOCK_SIZE é gerado por um PRNG desenvolvido pela equipe que utiliza sensores do *smartphone*



Desempenho

Defex Performance

- Aumento no consumo de bateria ~10%
- Aumento no consumo de memória < 10%
- Perfil completo tem tempo de execução médio de 4234 ns (0,0042ms)

Case	Average(ns)
All features off	**1408**
Privilege Escalation enabled only	**2081**
Root Jailhouse enabled only	**1758**
Restrict Syscall enabled only	**1550**
Trusted Map enabled only	**1429**
Binary Integrity enabled only	**2622**
All features on	**4234**

Defex vs SELinux

- Enforcement de integridade *in-memory*
- Aprendizagem do comportamento do sistema como perfis de execução de chamadas de sistema, execução de comandos e assinaturas de binários
- Detecção e reação a escalação de privilégios horizontal e vertical
- Defex não é um MAC!

Ameaças Controladas - Exemplos

- Kingroot
- CVE-2016-0728
- CVE-2014-3153
- Custom Exploits

Defex Defense

- Gap Space alocado para aleatorizar estruturas críticas
- PRNG baseado em sensores do *smartphone*
- *Immutable flag*
- Defex interface *lock*
- Chaves geradas por dispositivo para tornar os perfis únicos

Considerações Finais

- Aumento na resiliência a APT e exploração de vulnerabilidades no Android
- A superfície de ataque é reduzida com a restrição de chamadas de sistema
- Se alguma vulnerabilidade for explorada, o atacante não poderá alterar as credenciais do processo
- Se o *bypass* for possível, ainda assim ele não poderá interagir com a partição de aplicações
- Não poderá executar qualquer comando que não tenha sido autorizado
- Não poderá executar um binário arbitrário ou modificar um binário legítimo
- Complexidade de ataque é aumentada significativamente

Perguntas