Lecture 6

RECENT RESEARCH IN MOBILE PLATFORM SECURITY

You will be learning:

- What are some current challenges in (mobile) systems security research?

Recap: software platsec

- Permission-based security architecture
  - principle of least privilege
- Granted based on code-signing and/or user-query

Threats

- Malware in general
- Privilege escalation

How prevalent is mobile malware?

domains. We make several important observations. The mobile malware found by the research community thus far appears in a minuscule number of devices in the network: 3,492 out of over 380 million (less than 0.0009%) observed during the course of our analysis.

The Core of the Matter:
Analyzing Malicious Traffic in Cellular Carriers

Charles Lever, *Georgia Institute of Technology*, chazlever@gatech.edu
Manos Antonakakis, *Damballa*, manos@damballa.com
Brad Reaves, *Georgia Institute of Technology*, brad.reaves@gatech.edu
Patrick Traynor, *Georgia Institute of Technology*, traynor@cc.gatech.edu
Wenke Lee, *Georgia Institute of Technology*, wenke@cc.gatech.edu

NDSS 2013

?

Lookout


techland.time.com/2013/04/17/study-32-8-million-android-phones-infected-with-malware/

TIME Tech

SECTIONS

SECURITY

Study: 32.8 Million Android Phones Infected with Malware

By Techlicious / Fox Van Allen | April 17, 2013 | 9 Comments

Do you have an anti-virus app on your Android phone yet? If not, a new study conducted by security firm NQ Mobile suggests you're playing with fire: The number of malware threats to your Android phone has increased 163% over the past year alone.

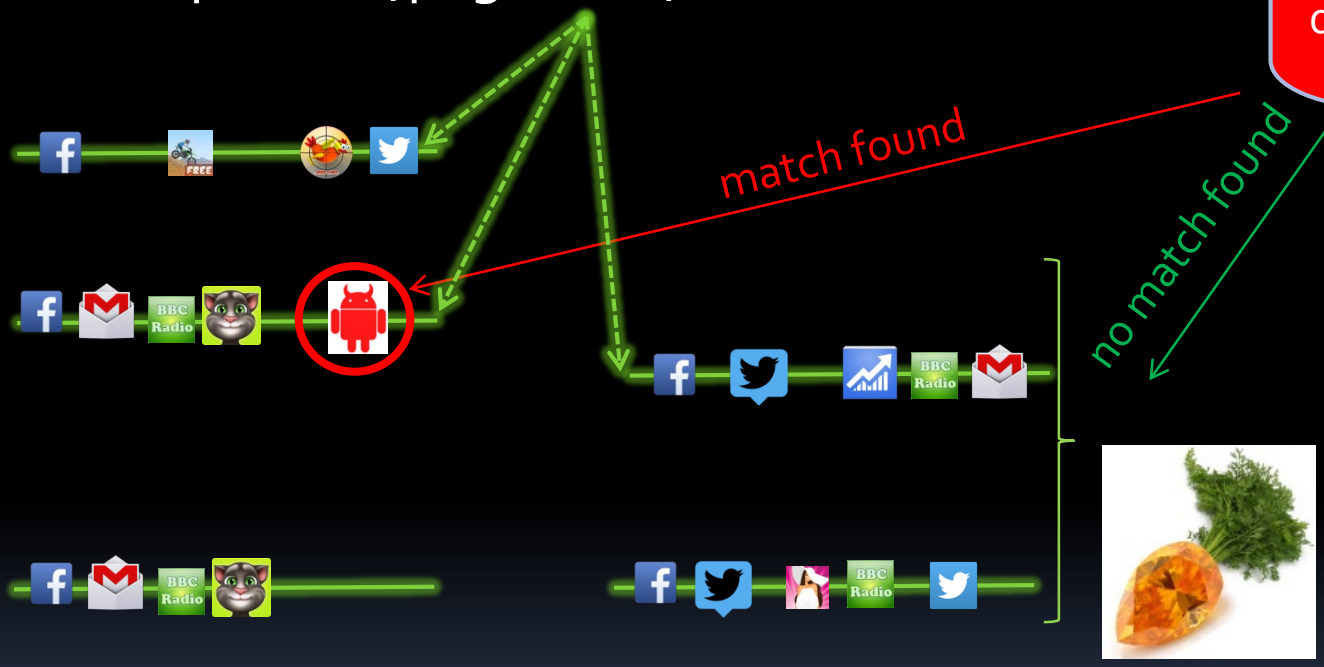
The study, which looked at over 5.3 million apps available in 406 different online stores, identified 65,227 different pieces of potentially dangerous malware last year. A quick look at the trend suggests that malware is growing at an exponential rate – there were only 1,649 such malware discoveries in 2009.

In total, 32.8 million Android phones were infected with malware in 2012 – more than triple the number of the year before. The majority of these infections involve spyware or adware, while about a quarter are designed to steal and profit off of your personal data. A smaller minority is designed to make your phone permanently unusable, something we'd all no doubt like to

Estimating infection rate

time

set of tuples:
<developerCert,pkgName,versionCode>



Device 1

"infected"

Device 2

"clean"

…

Incidence of infection

# Infected Devices	Mobile Sandbox	McAfee	Union
coarse-grained: matching developerCert	37,355 (38%)	32,323 (33%)	40,334 (40%)
fine-grained: full match	263 (0.26%)	255 (0.26%)	477 (0.48%)

Data collected from 99414 devices over one year

More information at the Malware Insights project page

Common malware patterns

In the wild

- Excessive permission requests
  - Ad libraries needing sensitive permissions
    - location, network, …
- Device rooting
- Monitoring apps

Common malware patterns

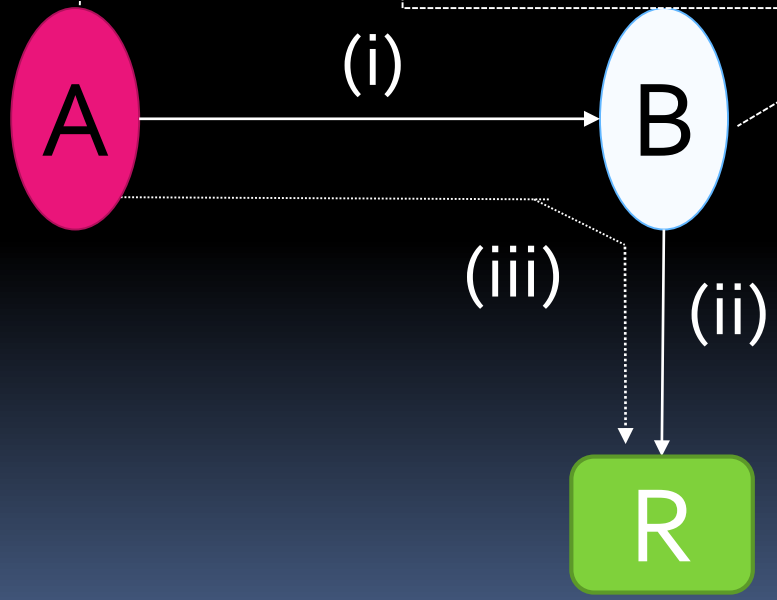
In research papers

- Sensory malware
- Privilege escalation

Privilege escalation

Allow communication to B
Deny access to R

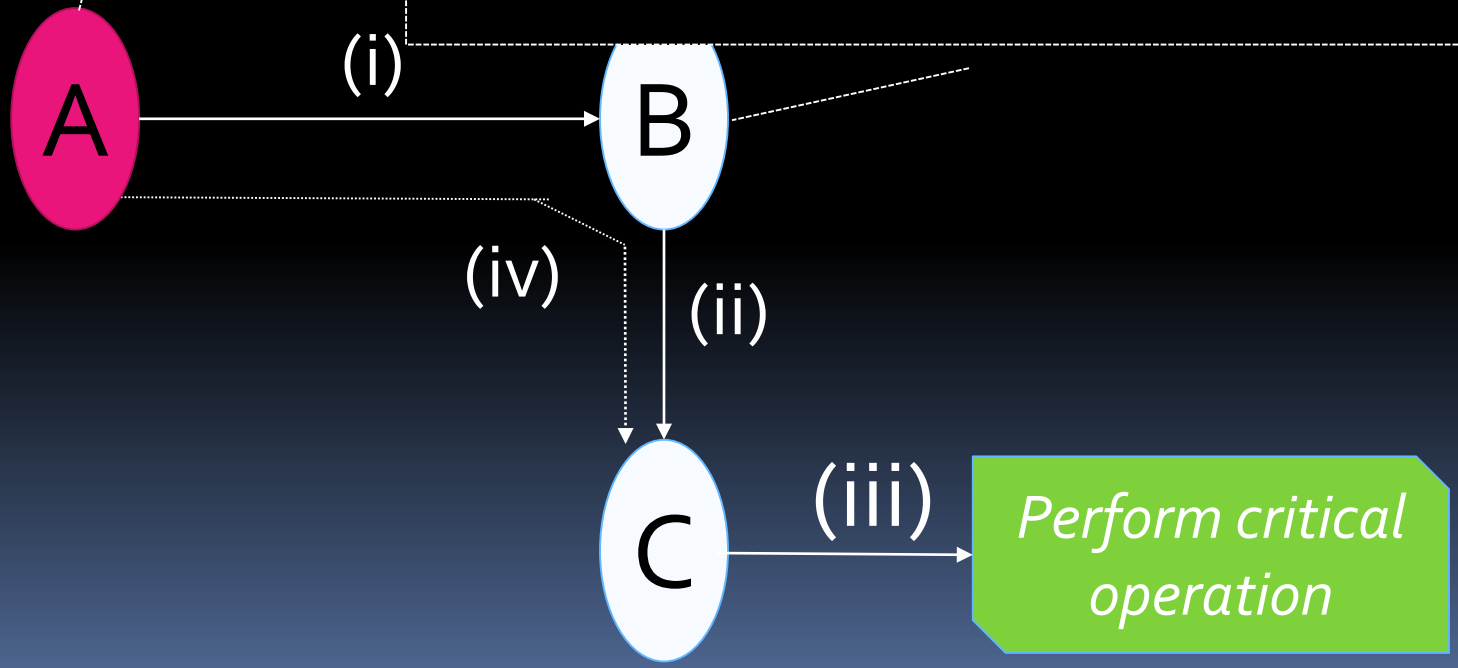
Allow communication to A
Allow access to R



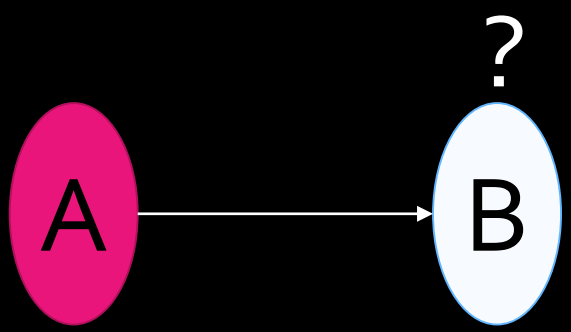
Privilege escalation

Allow communication to B
Deny communication to C

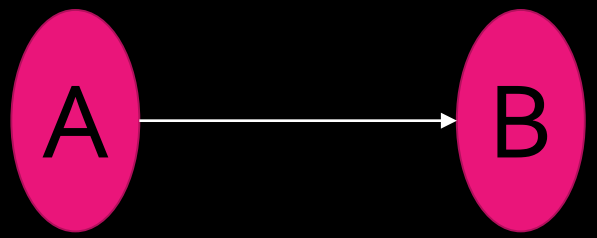
Allow communication to A
Allow communication to C



Classes of privilege escalation

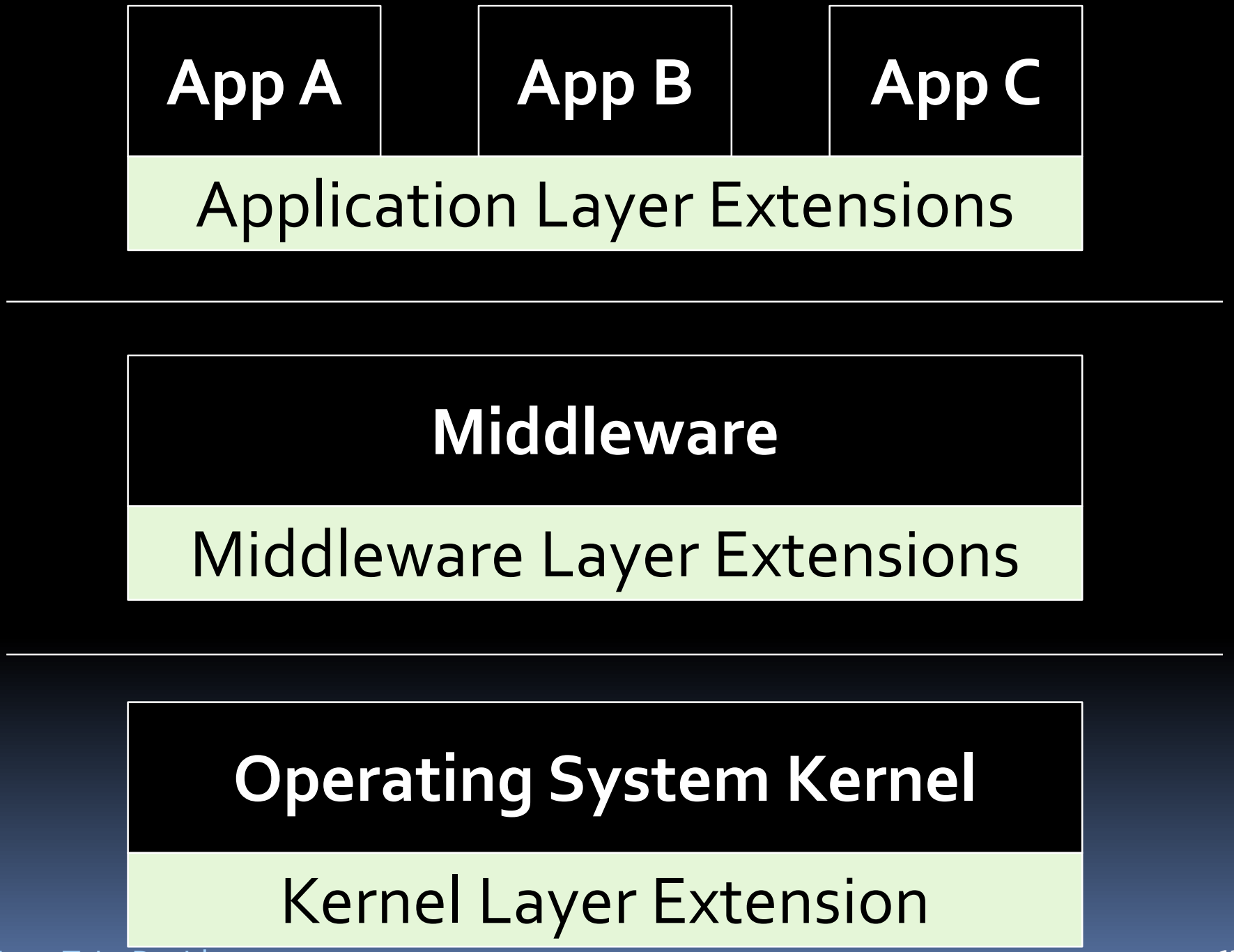


Confused deputy



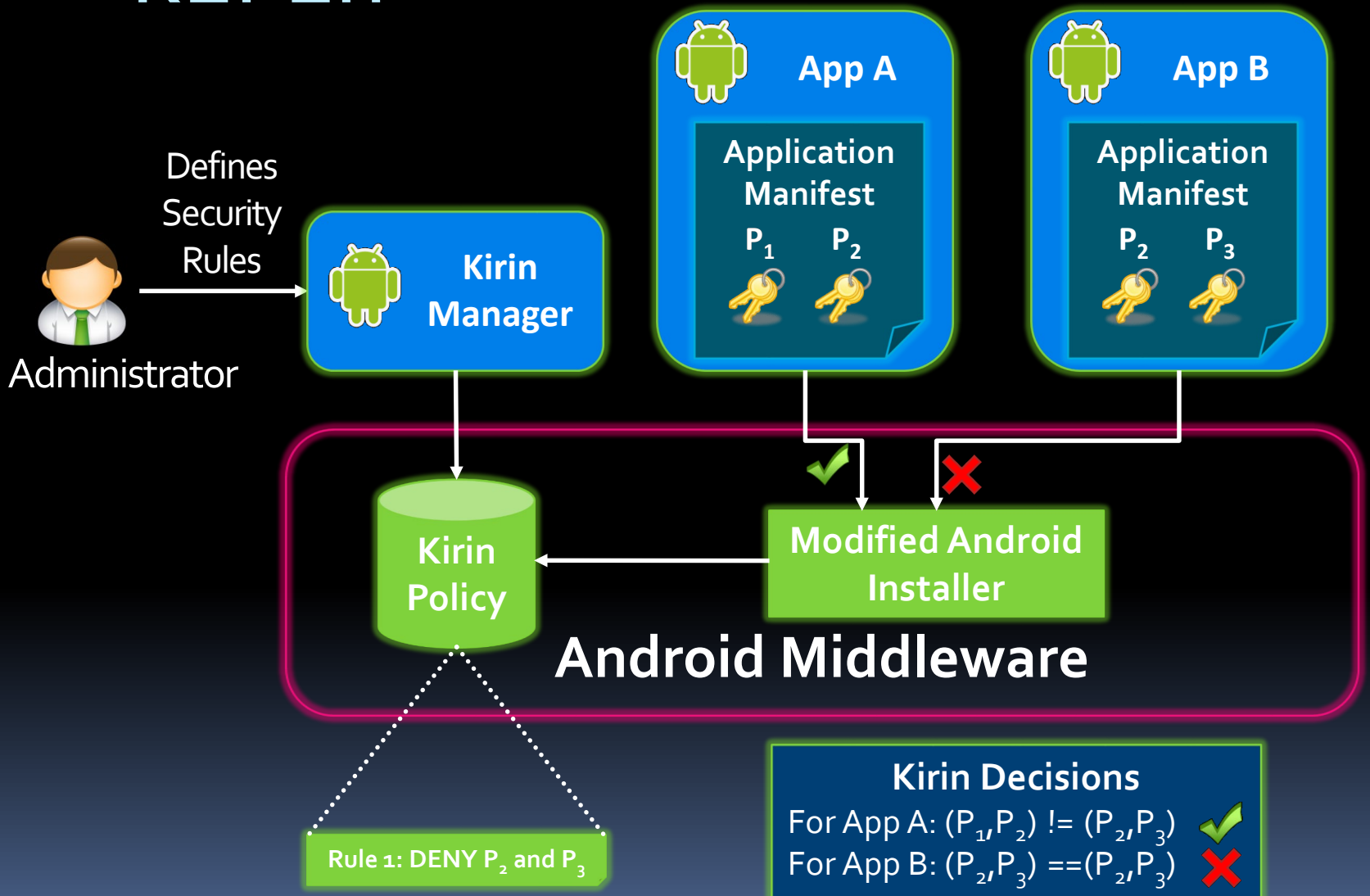
Collusion

Skip to Android summary



Skip to TaintDroid
Skip to Android summary

Kirin



Enck et al., ACM CCS 2009

SAINT

Develops App and defines SAINT policy

Developer

SAINT policies consist of install-time and runtime rules targeting

- Permissions held by client applications
- Configuration (e.g., app version) and Signature by client applications and by new apps to be installed
- Context (e.g., Location, Bluetooth, WiFi) of client applications

SAINT extended App A

Application Manifest

requested: P_1 P_2 | declared: P_3

SAINT Policy

Install Rule 1: Apps requesting P_3 must also request P_1 and P_2

Runtime Rule 1: Apps using interface X must have P_5 and have to be signed by Google

Runtime Rule 2: Bluetooth and GPS must be disabled if apps aim to use my interfaces

App B

Application Manifest

P_1 P_2 P_3

Runtime Rule 1 not satisfied

IPC Message to A

Modified Android Installer

Saves SAINT Policy and checks if new apps satisfy existing policies

SAINT App Policy Provider

Binder IPC

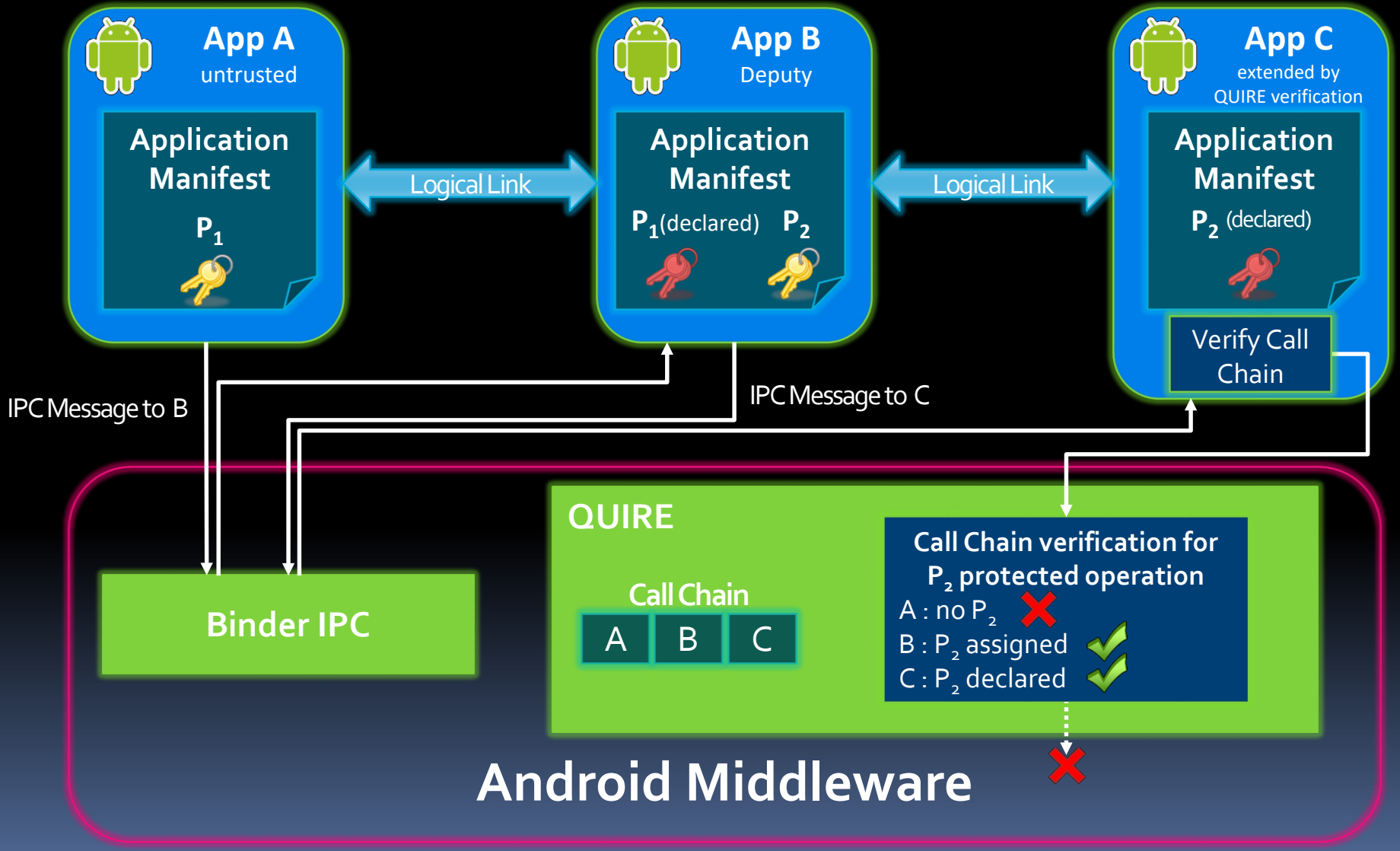
SAINT Mediator

Checks if SAINT policies are satisfied for IPC messages

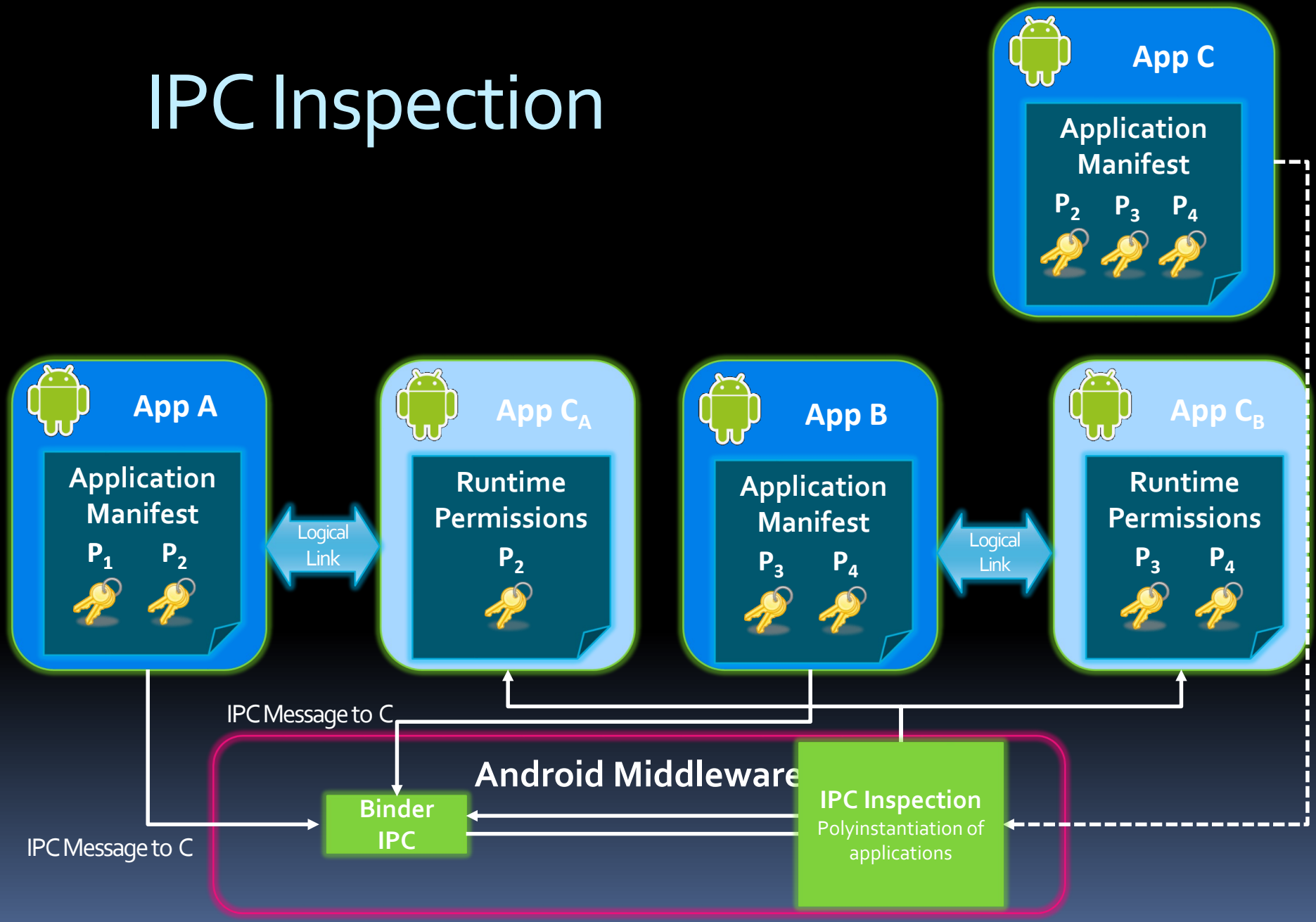
Android Middleware

Ongtang et al., SCN 2012; ACSAC 2010

QUIRE


App A
untrusted
Application Manifest
P1
Logical Link
App B
Deputy
Application Manifest
P1(declared) P2
Logical Link
App C
extended by QUIRE verification
Application Manifest
P2 (declared)
Verify Call Chain
IPC Message to B
IPC Message to C
Binder IPC
QUIRE
Call Chain
A B C
Call Chain verification for P2 protected operation
A : no P2
B : P2 assigned
C : P2 declared
Android Middleware

IPC Inspection



Felt et al., USENIX SEC 2011

XManDroid

Adminstrator

Defines system-centric XManDroid policy

Android Middleware

XManDroid Policy

IPC Message to B

Binder IPC

Rule 1 not satisfied

XManDroid Policy Engine

Rule 1: DENY communication if Sender has P_2 and Receiver P_3

Send data to B via files

Files

File-System

XManDroid TOMOYO

Network package to B

Network Sockets

Network

XManDroid iptables

TaintDroid

App A

Location → var1

Contacts → var2

Logical Link

App B

var1 → var3

var2 → var4

Network

var3

var4

var3 and var4 leave the system

Contact Database

TaintDroid

integrated in Dalvik VM

Taint Storage

var1

var2

Taint Storage

var3

var4

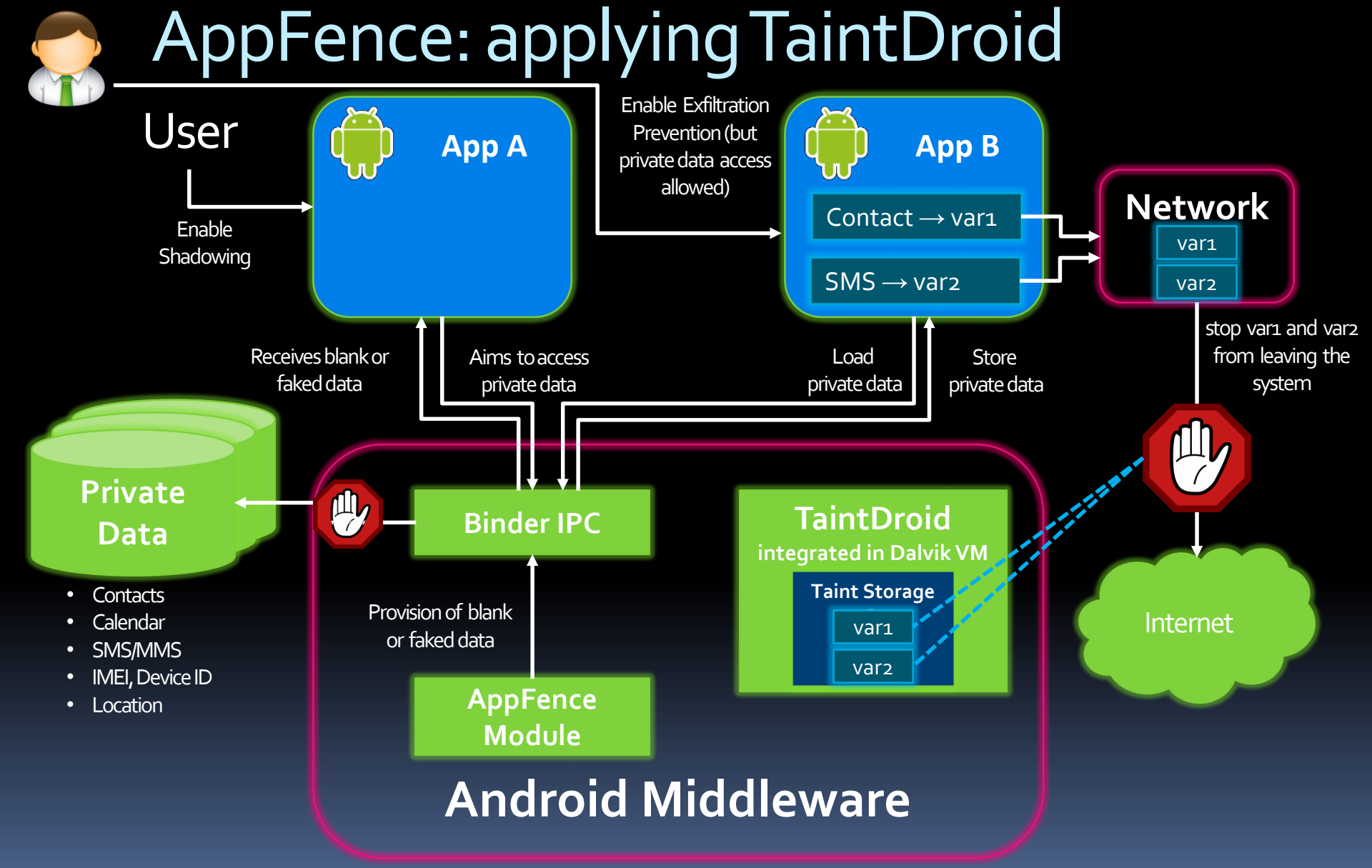
Binder IPC

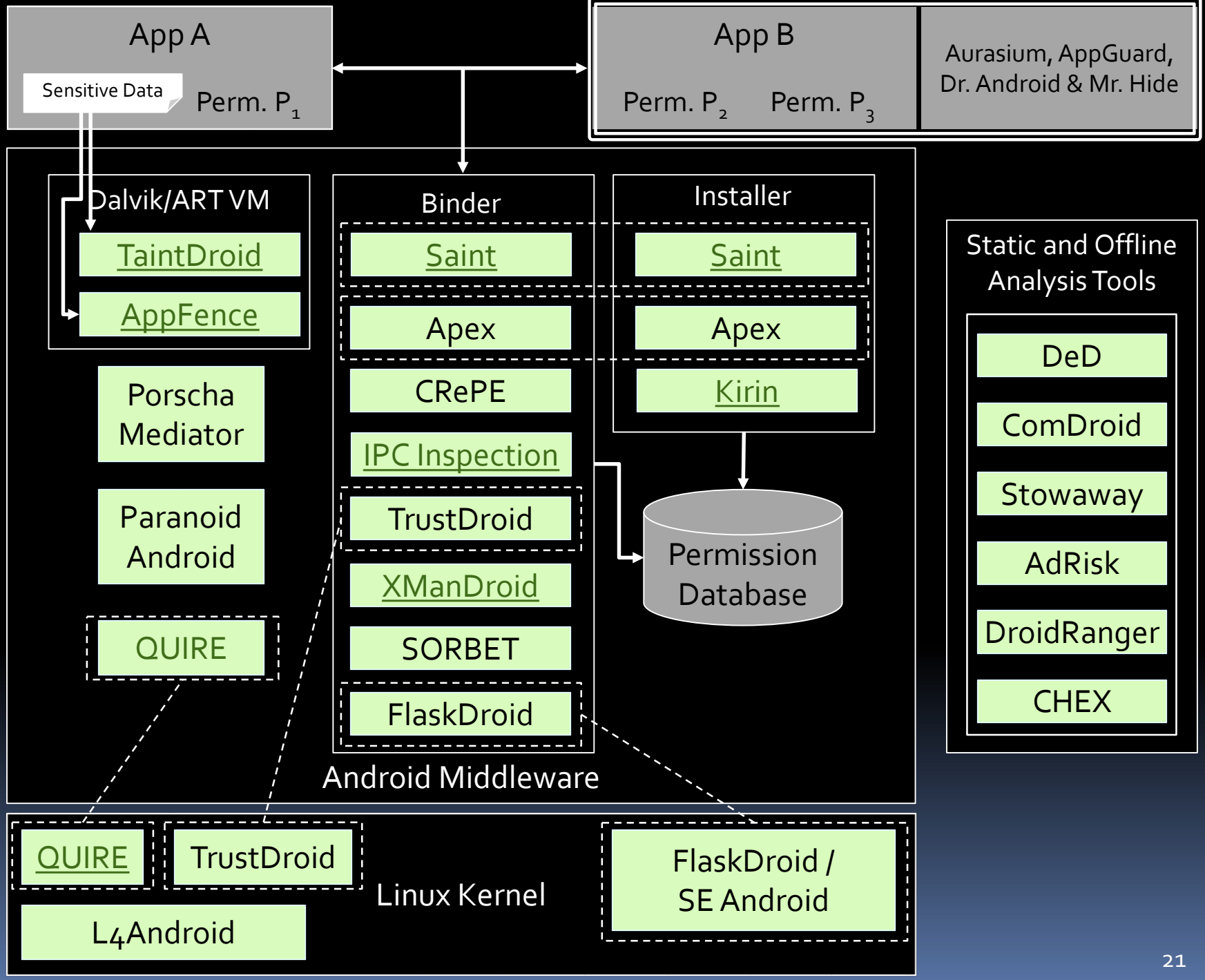
Android Middleware

Internet

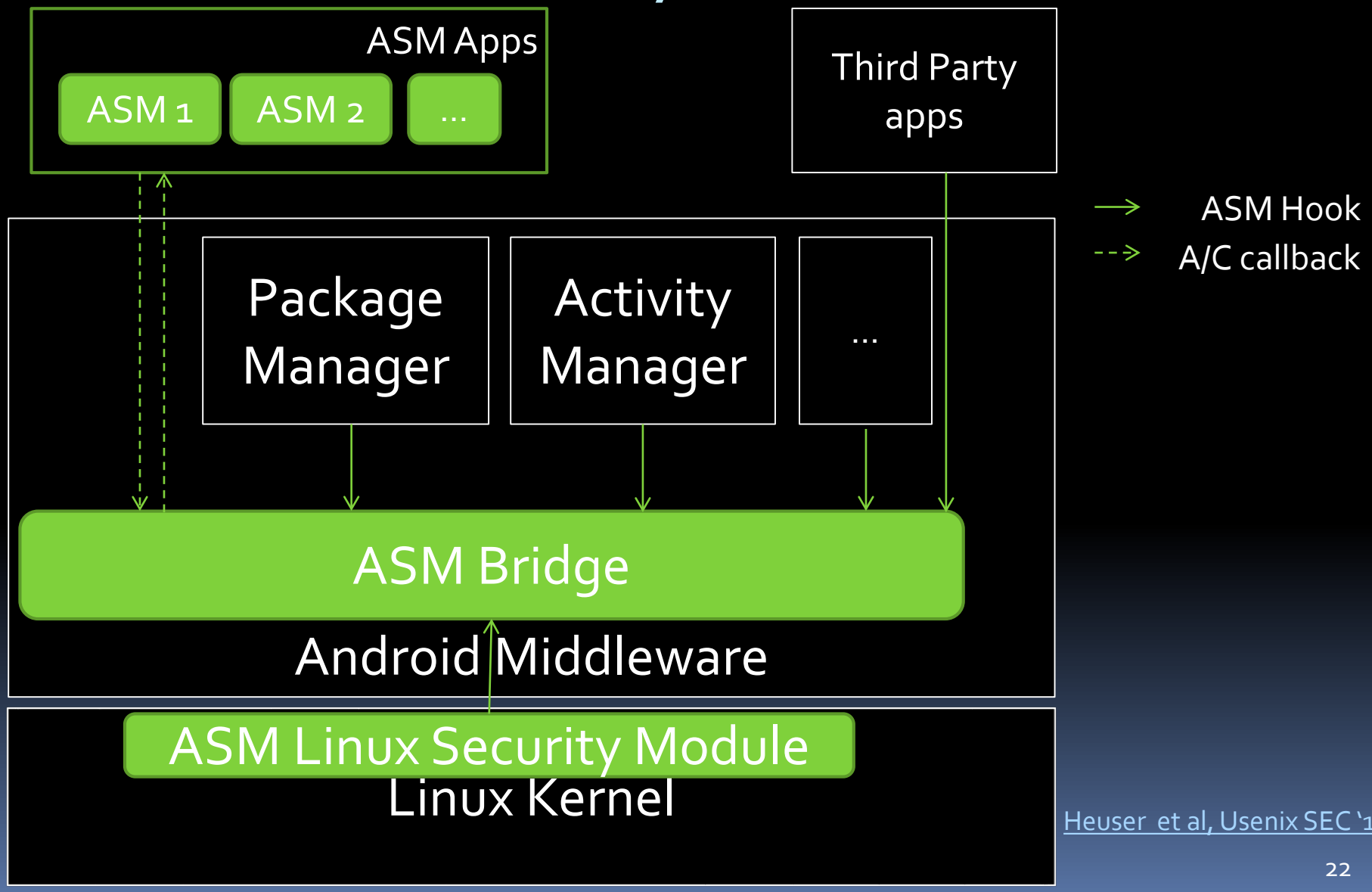
Inform user that sensitive data has been leaked

AppFence: applying TaintDroid




App A
Sensitive Data
Perm. P1
App B
Perm. P2
Perm. P3
Aurasium, AppGuard, Dr. Android & Mr. Hide
Dalvik/ART VM
TaintDroid
AppFence
Porscha Mediator
Paranoid Android
QUIRE
Binder
Saint
Apex
CRePE
IPC Inspection
TrustDroid
XManDroid
SORBET
FlaskDroid
Installer
Saint
Apex
Kirin
Permission Database
Android Middleware
Static and Offline Analysis Tools
DeD
ComDroid
Stowaway
AdRisk
DroidRanger
CHEX
QUIRE
TrustDroid
L4Android
Linux Kernel
FlaskDroid / SE Android

Android Security Modules



Heuser et al, Usenix SEC '14

Privilege escalation via run-time attacks

Software written in memory unsafe languages such as C/C++

- Suffer from various memory-related errors

Memory errors may allow run-time attacks to compromise program behaviour

- *Control-flow hijacking* / *code injection*
- *Return-Oriented Programming* (ROP)
- *Non-control-data attacks*
- *Data-Oriented Programming* (DOP)

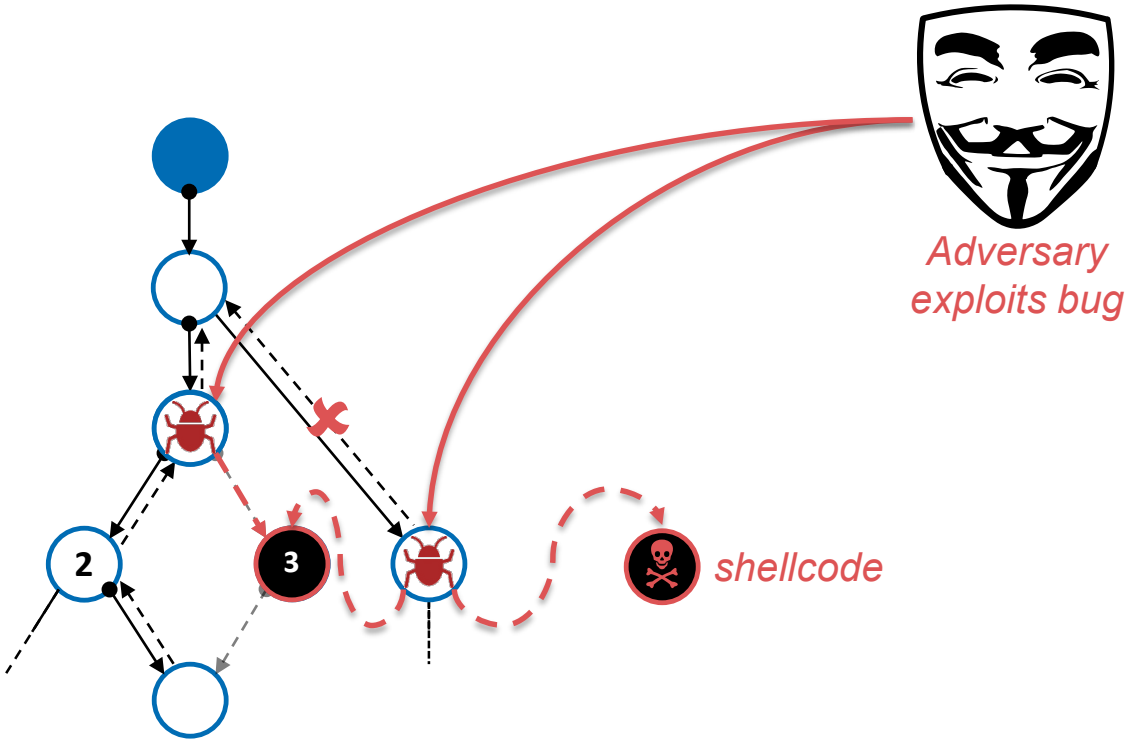
Run-time attacks compromise program behaviour

(i) Code-injection attack

(ii) Code-reuse attack

(iii) Non-control-data attack

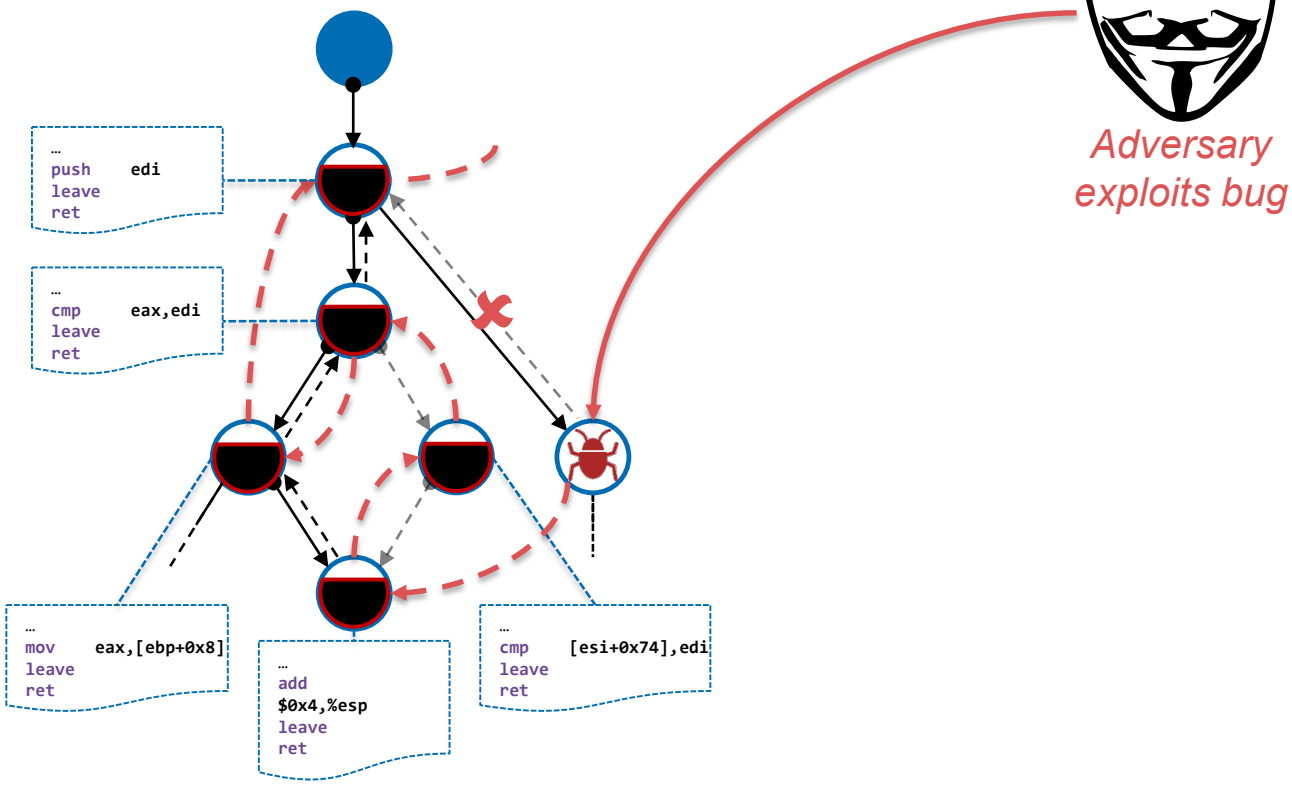
```
1 if (authenticated != true)
      then: call unprivileged()
      else: call   privileged()
…

2 unprivileged() { … }
3   privileged() { … }
…
```



Return-oriented programming

Attacker arranges call stack with code pointers to existing code sequences ("*gadgets*")

- **Given a suitable gadget set, arbitrary *return-oriented programs* can be constructed**

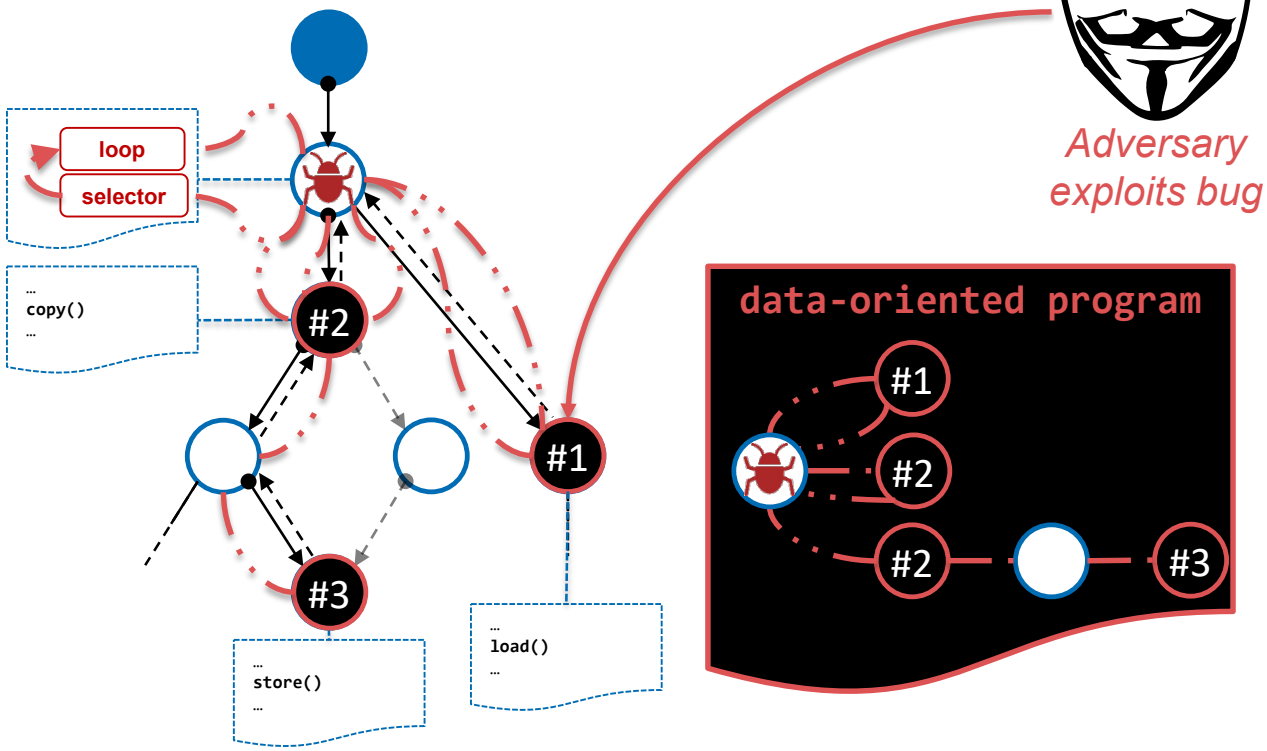


Shacham et al, ROP papers, ACM CCS 2007, 2008, BlackHat 2008
https://hovav.net/ucsd/talks/blackhat08.html

Data-oriented Programming

Enables expressive computation via use of "*data-oriented gadgets*" without diverging from program's benign control-flow

- **Requires a "gadget dispatch" that allows chaining together gadgets at will**



Hu et al., Data-oriented Programming, IEEE S&P 2016
https://huhong-nus.github.io/advanced-DOP/

HardScope: Hardware-assisted Run-time Scope Enforcement

How can variable visibility rules be enforced at run-time to prevent run-time attacks?

Run-time attacks violate data integrity

- data references disallowed at compile time

Variable visibility rules reduce attacks…

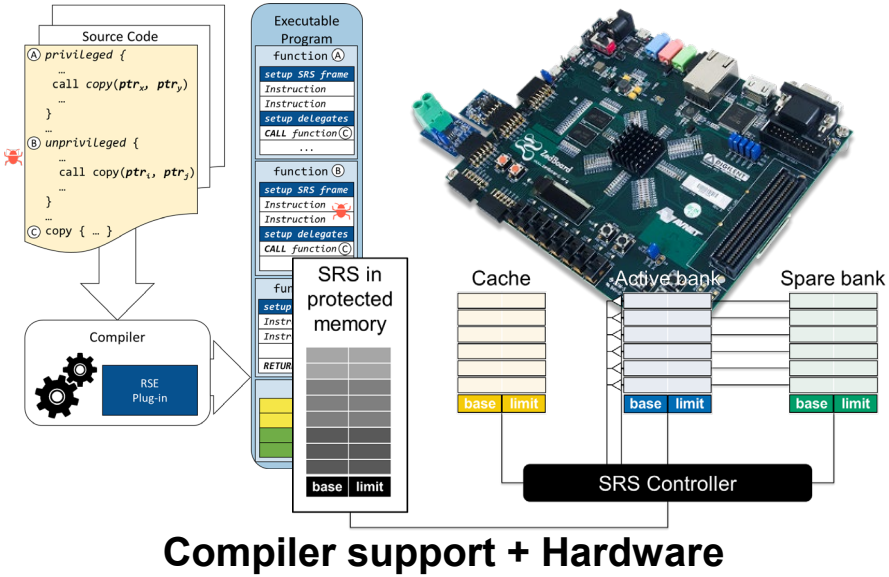
- …but in C/C++ only enforced by compiler

H/W ext. for run-time scope enforcement

- PoC on RISC-V PULPino SoC on FPGA

Low-overhead (~3%) with changes to h/w

- Can apply at different granularities to give resilience against many classes of attacks



Compiler support + Hardware

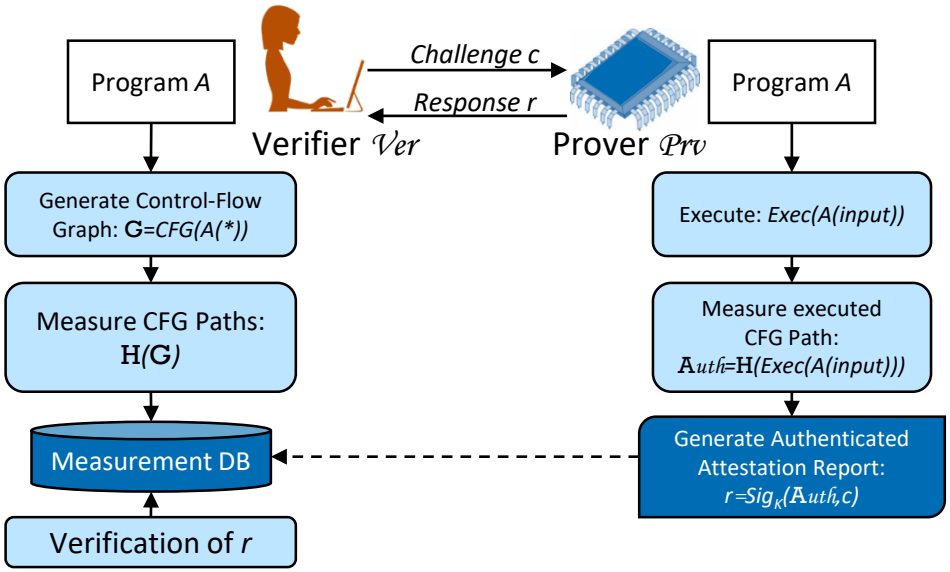
https://ssg.aalto.fi/research/projects/harp/

Nyman et al, DAC 2019

C-FLAT: Attestation for Run-time Behavior (high-level idea)

How can a device convince an external verifier that its run-time behavior is correct?

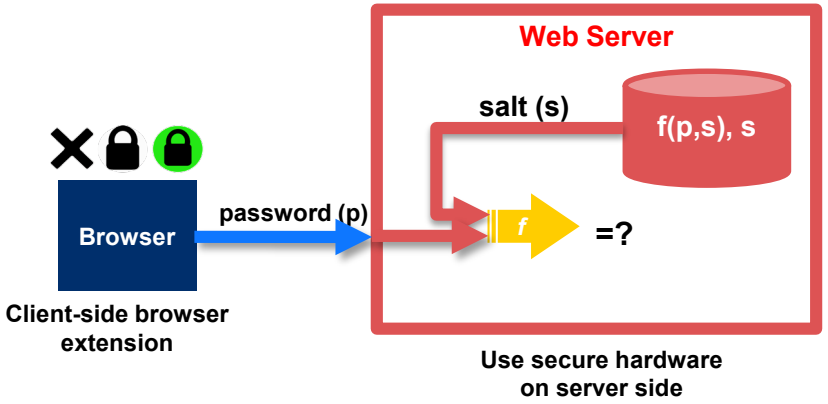
Trace and record control flow of prover and aggregate measurement in *hash-chain*



https://arxiv.org/abs/1605.07763
Abera et al, ACM CCS 2017

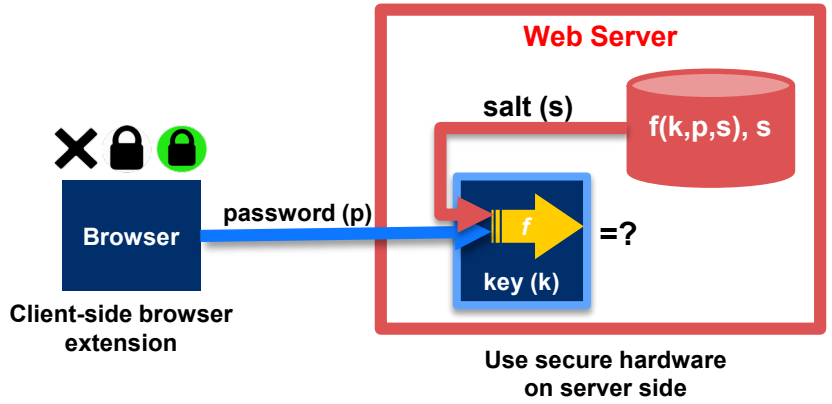
SafeKeeper: Protecting Web Passwords

How can we use widely available hardware security mechanisms to deter password database theft and server compromise?



SafeKeeper: Protecting Web Passwords

How can we use widely available hardware security mechanisms to deter password database theft and server compromise?



https://ssg.aalto.fi/research/projects/passwords/
Krawiecka et al, WebConf 2018 (aka WWW 2018)

Best Infosec thesis award, Tietoturva ry, 2017
Runner up, Best national CS thesis award, 2018

SAFE KEEPER

Breaking & repairing deniable messaging

Attestation can be used to undetectably break deniable messaging

Attestation can help restore deniability in messaging

Deniable messaging is useful…

- whistleblowers, marginalized, politicians,…

and popular

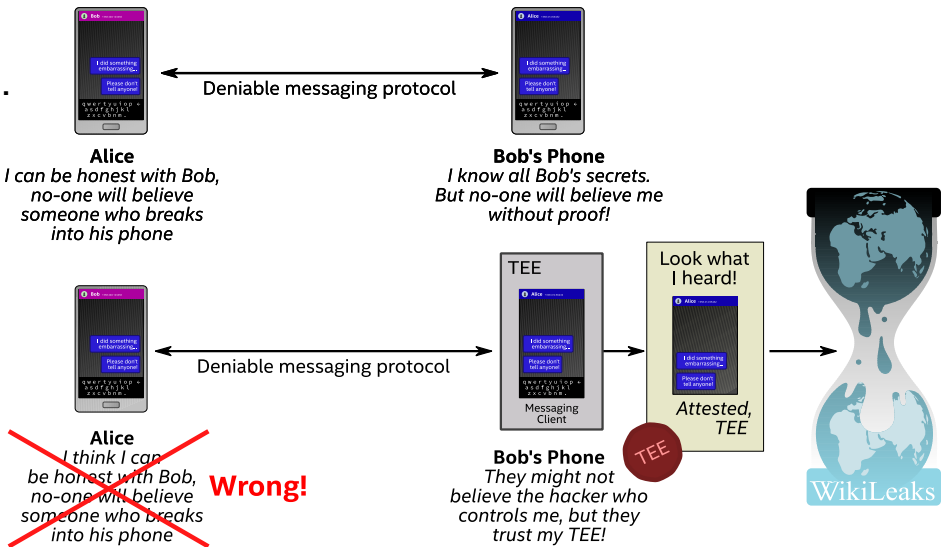
- Signal/WhatsApp, Telegram, OTR, …

Undetectably breaking deniability

- have TEE attest received messages to skeptical verifiers

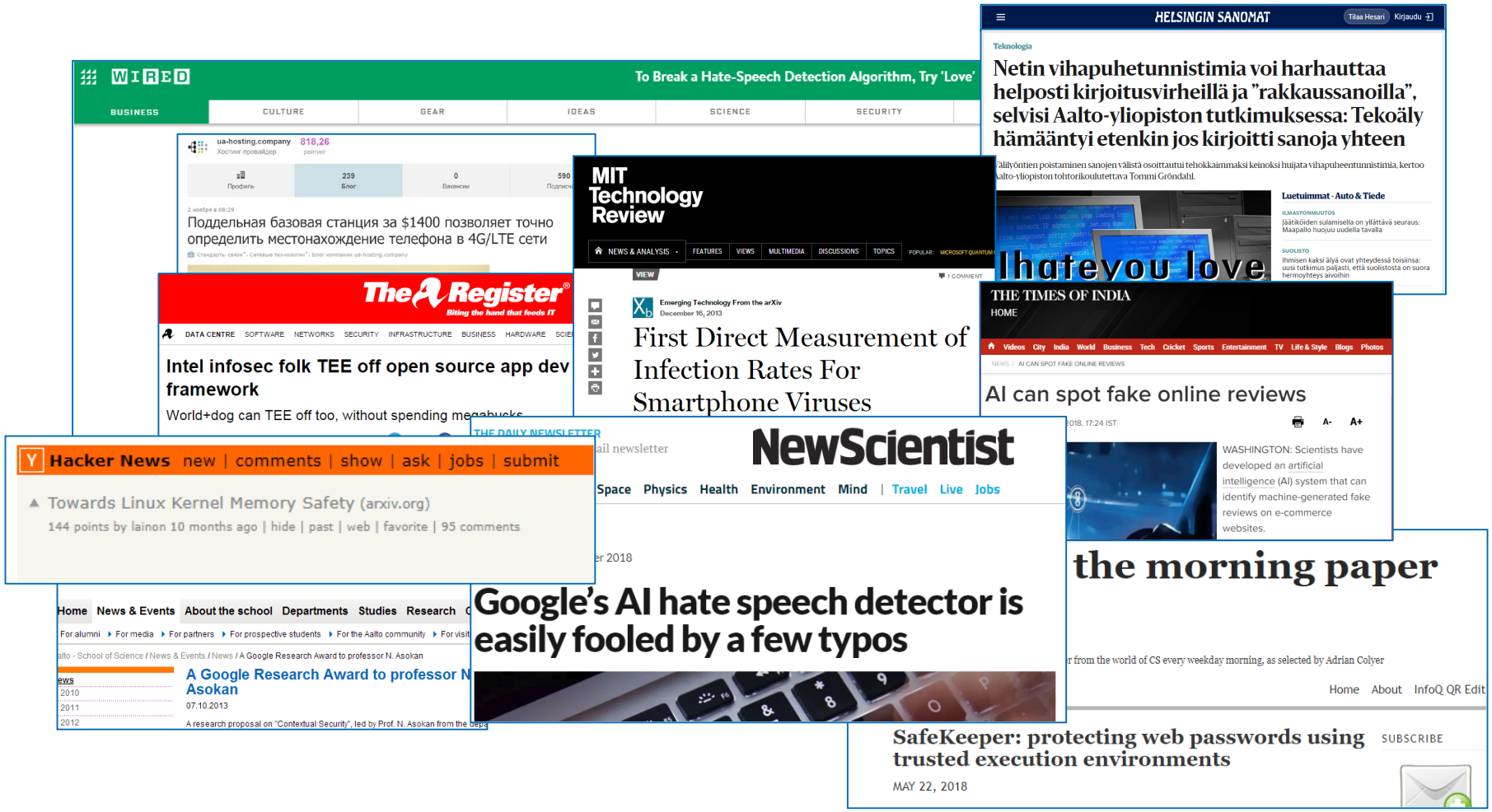
S/W attacker: thwarted using attestation

- H/W attackers are hard to defend against



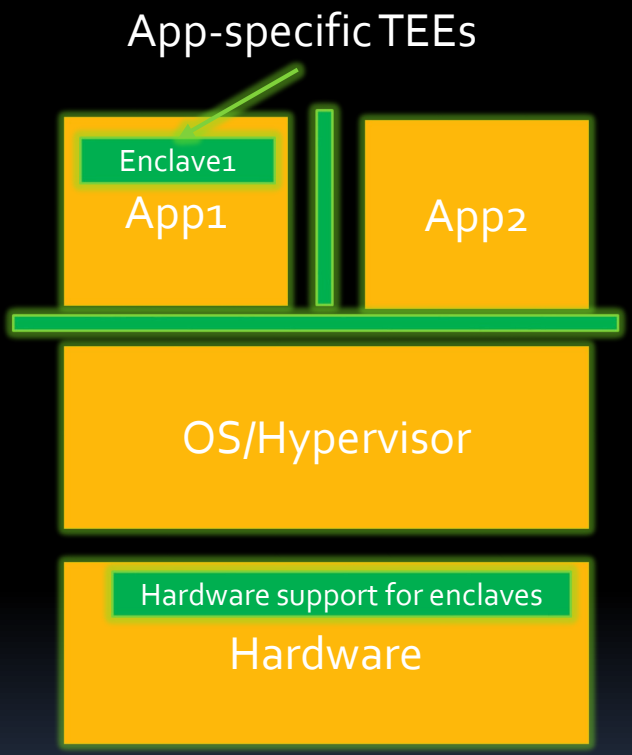
https://eprint.iacr.org/2018/424
Gunn et al, Blackhat Europe 2018, PETS 2019

Media coverage of our research



Skip to “Research @SSG”

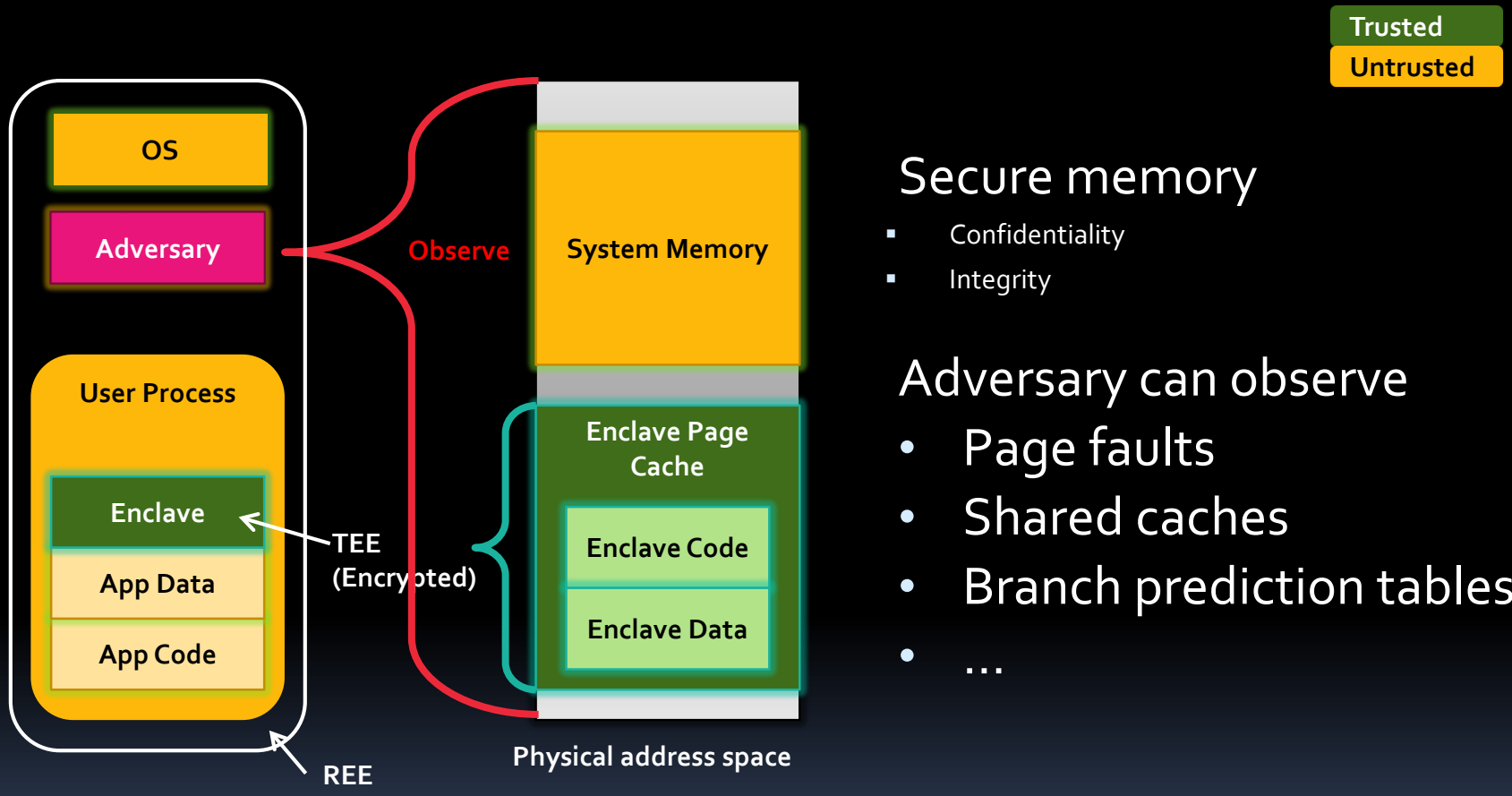
Intel Software Guard Extensions



- SGX provides hw-supported TEE functionality in ring-3
- Enclave code/data encrypted by hardware
- Supports attestation and sealing

https://software.intel.com/sgx

Potential for information leakage



Secure memory

- Confidentiality
- Integrity

Adversary can observe

- Page faults
- Shared caches
- Branch prediction tables
- …

https://software.intel.com/sgx

Research @SSG in general

- Platform security
- ML & security
- Other topics
  - Blockchains and consensus
  - Stylometry and linguistic analysis

https://ssg.aalto.fi/research/available-research-topics/

Did you learn:

- A quick overview of some recent research in (mobile) systems security?

Contributors: Thomas Nyman, Lachlan Gunn, Hien Truong, Andrew Paverd, Luca Davi, Ahmad-Reza Sadeghi, N. Asokan

Secure storage: Apple vs FBI

Skip to end

Lecture 6

SUMMARY OF LECTURES

Objectives of the course:

Expose you to platform security on mobile devices

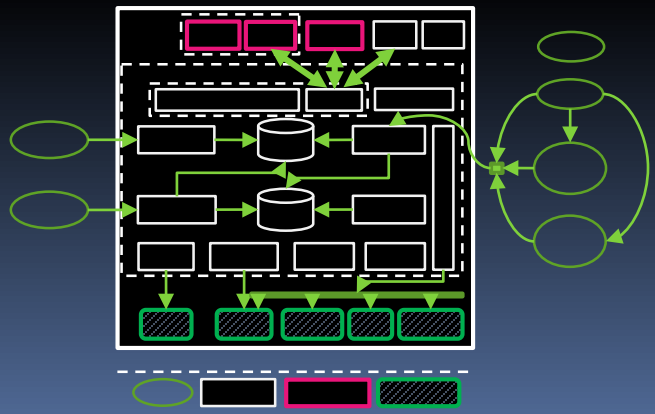
- Cover major themes
- Give hands-on experience to those who want it
- Prepare to learn about current research

Basic concepts

- ACLs/capabilities, MAC/DAC

Software PlatSec

- General model
  - Controlled API access to sensitive functionality
  - Permission-based architectures

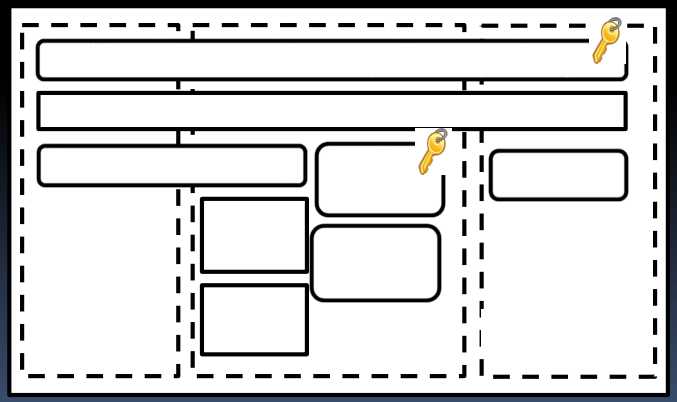


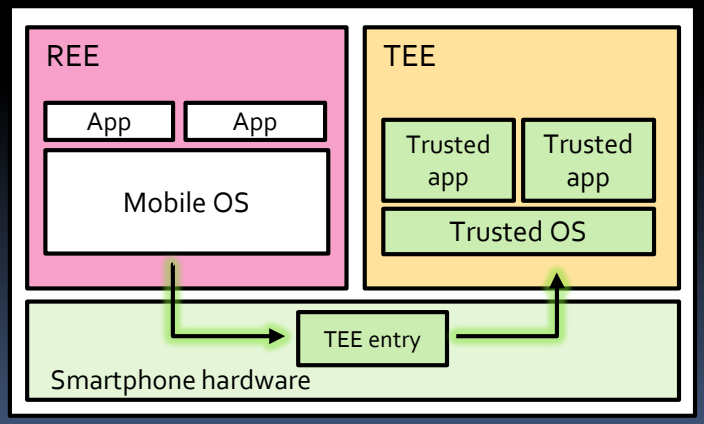
Software PlatSec

- Instantiations and comparison
  - Limitations and challenges

Hardware PlatSec

- Generic model
  - Boot integrity, secure storage, TEEs



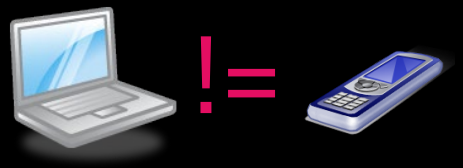


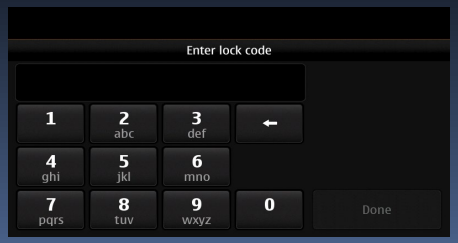
Hardware PlatSec

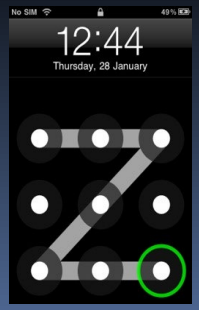
- Instantiations
  - TrustZone
  - Trust Platform Module
    - Authorization in TPM.2

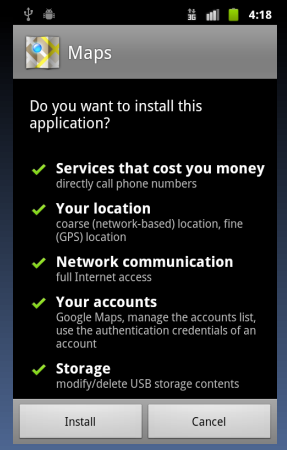
Usability of security

- Challenges in usable mobile security









Usability of security

- Exploiting context to improve usability

Recent research

- Some recent research in mobile security
  - Usability of permission assignment
  - Thwarting privilege escalation
- IoT Security
- Machine Learning and Security
- SE Linux for Android

FIN.