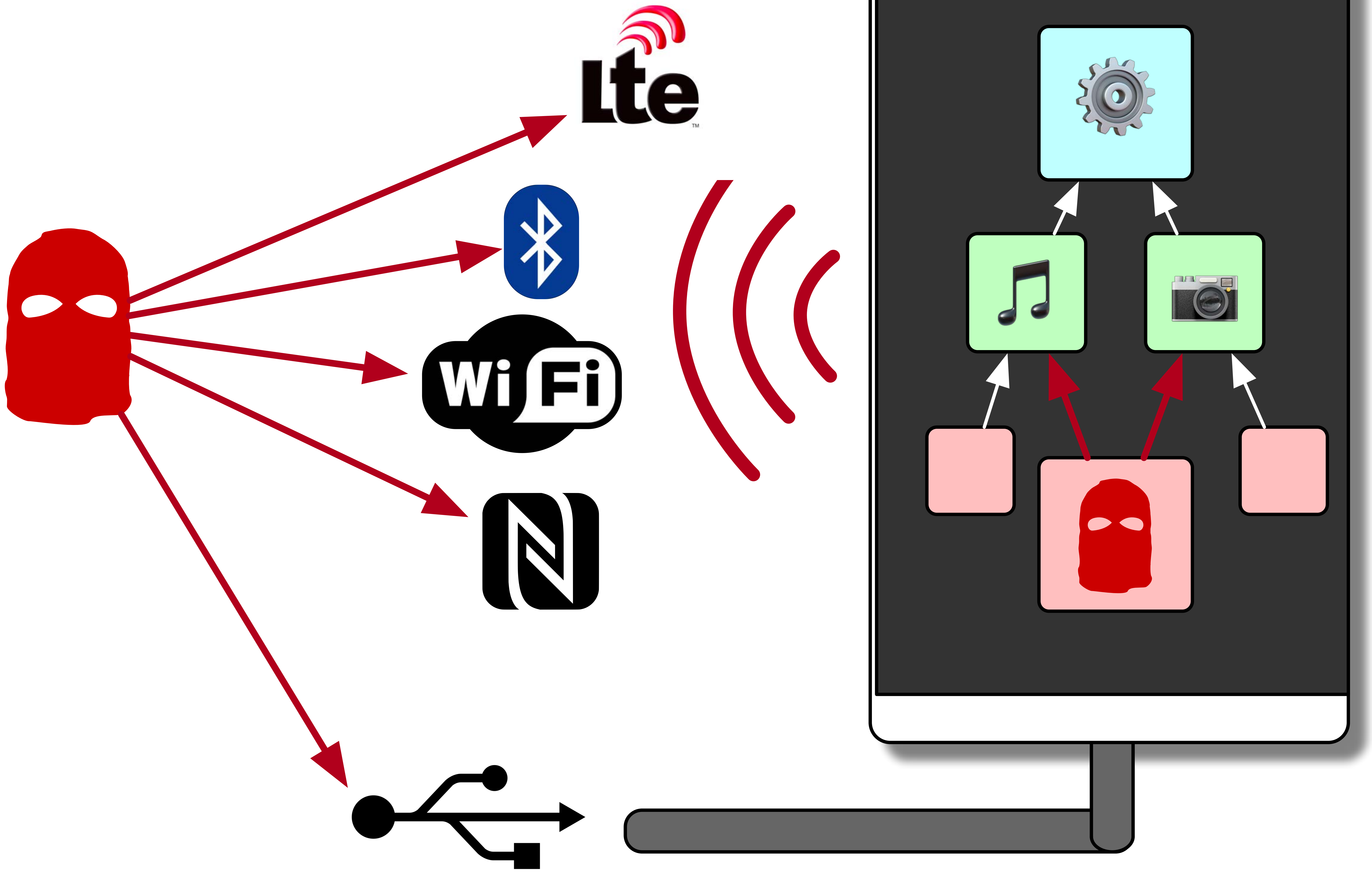
BIGMAC: Fine-Grained Policy Analysis of Android Firmware

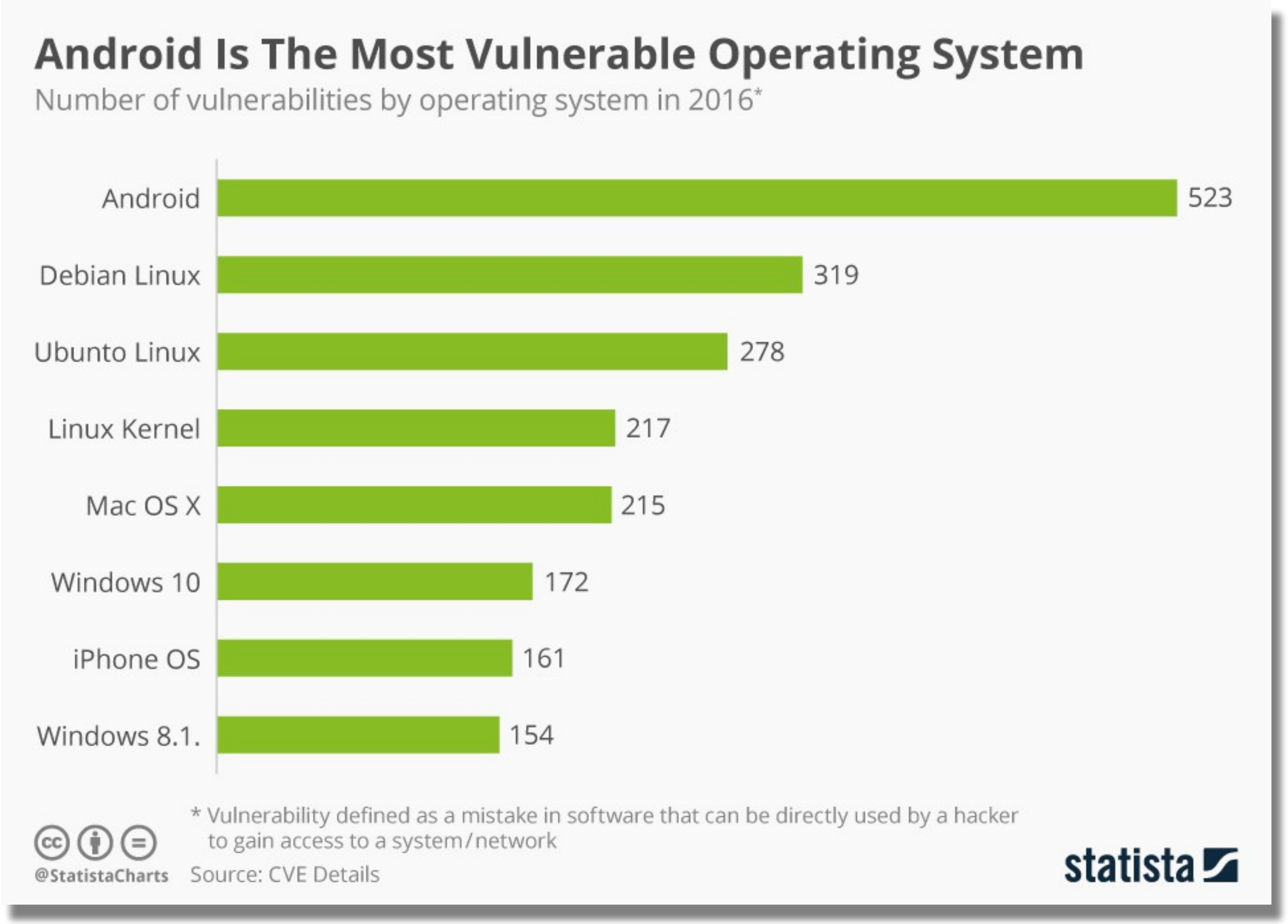
Grant Hernandez [†], Dave (Jing) Tian [‡], Anurag Swarnim Yadav [†], Byron J. Williams [†], and Kevin R. B. Butler [†]

[†] — University of Florida [‡] — Purdue University

Android Attack-surface



High Impact Bugs


Android Is The Most Vulnerable Operating System

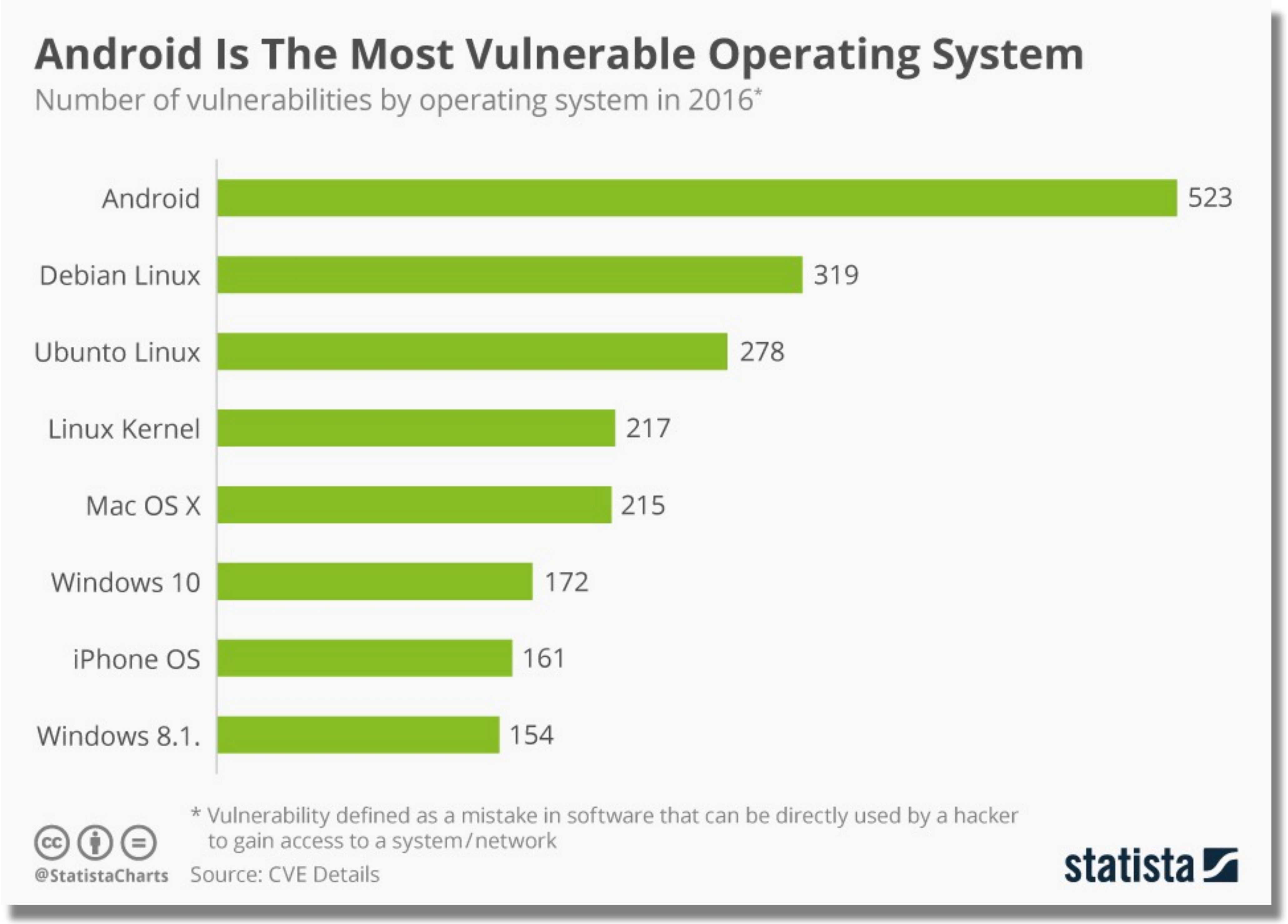
Number of vulnerabilities by operating system in 2016*

* Vulnerability defined as a mistake in software that can be directly used by a hacker to gain access to a system/network

Source: CVE Details

High Impact Bugs

- **CVE-2017-0737** - libstagefright (remote MMS triggerable)



Android Is The Most Vulnerable Operating System

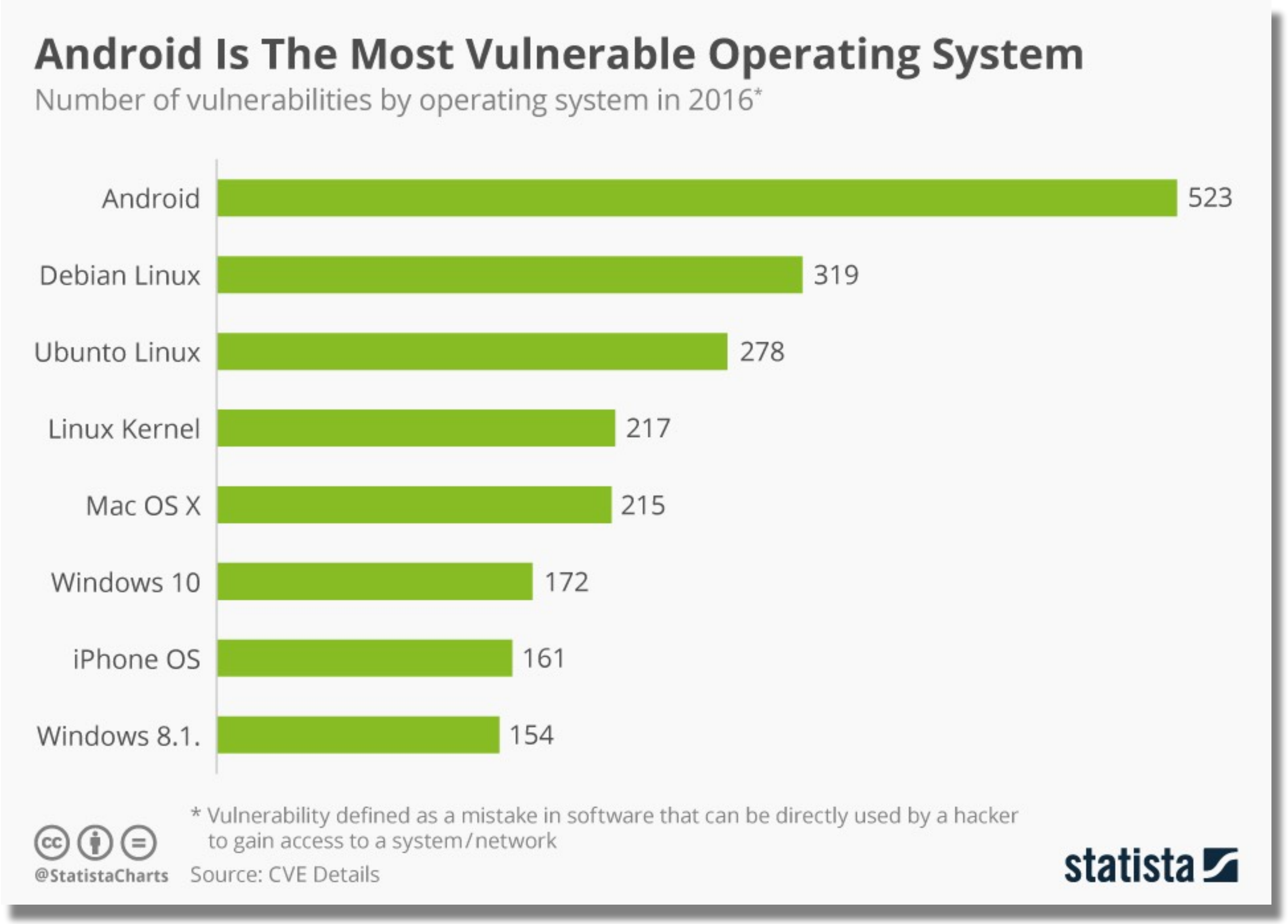
Number of vulnerabilities by operating system in 2016*

* Vulnerability defined as a mistake in software that can be directly used by a hacker to gain access to a system/network

Source: CVE Details

High Impact Bugs

- **CVE-2017-0737** - libstagefright (remote MMS triggerable)
- **CVE-2018-9488** - Privilege escalation to full root compromise (USB)

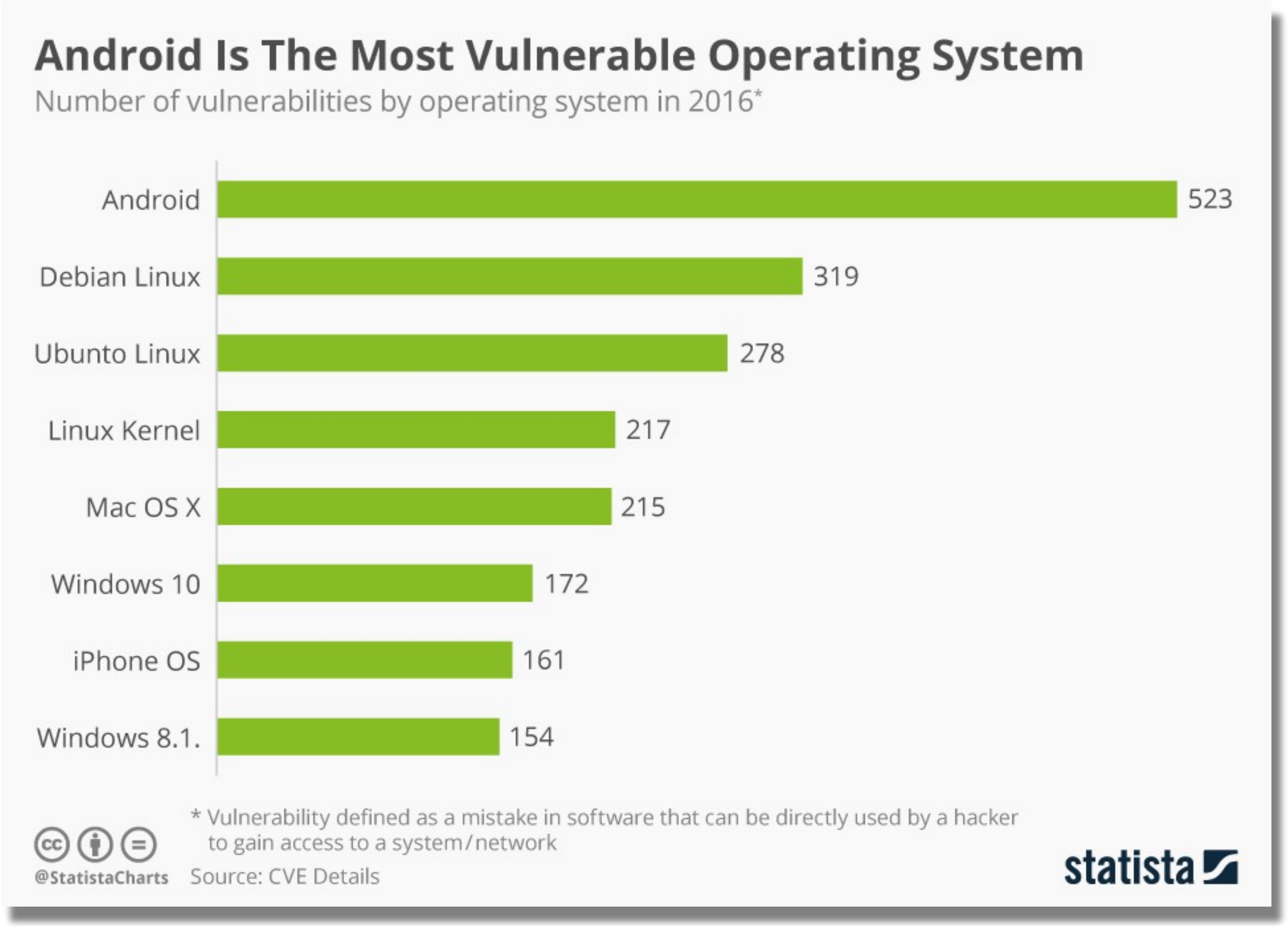


* Vulnerability defined as a mistake in software that can be directly used by a hacker to gain access to a system/network

Source: CVE Details

High Impact Bugs

- **CVE-2017-0737** - libstagefright (remote MMS triggerable)
- **CVE-2018-9488** - Privilege escalation to full root compromise (USB)
- **CVE-2019-2215** - Binder Use After Free (app reachable)



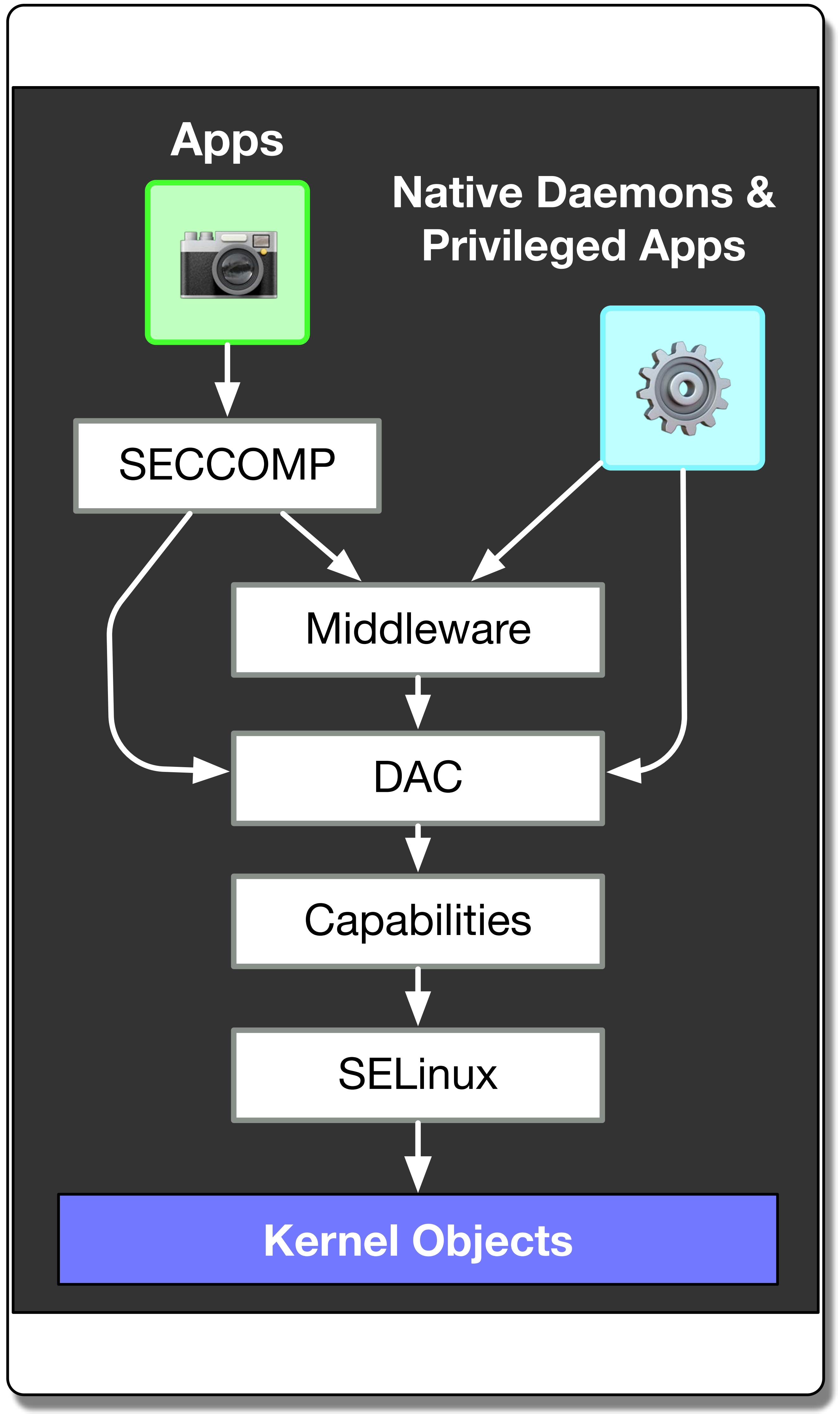
Android Is The Most Vulnerable Operating System

Number of vulnerabilities by operating system in 2016*

* Vulnerability defined as a mistake in software that can be directly used by a hacker to gain access to a system/network

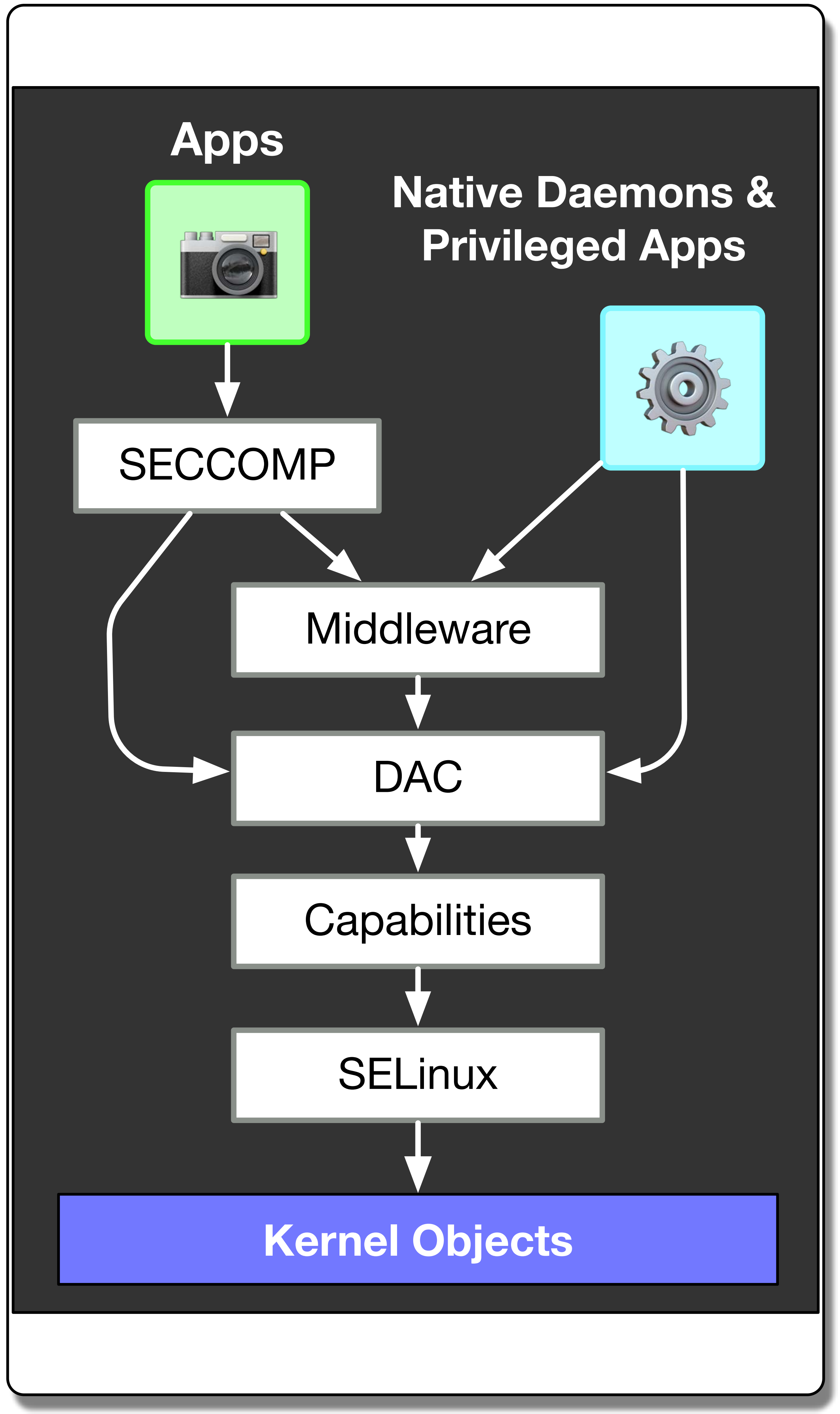
Source: CVE Details

Android Security Mechanisms



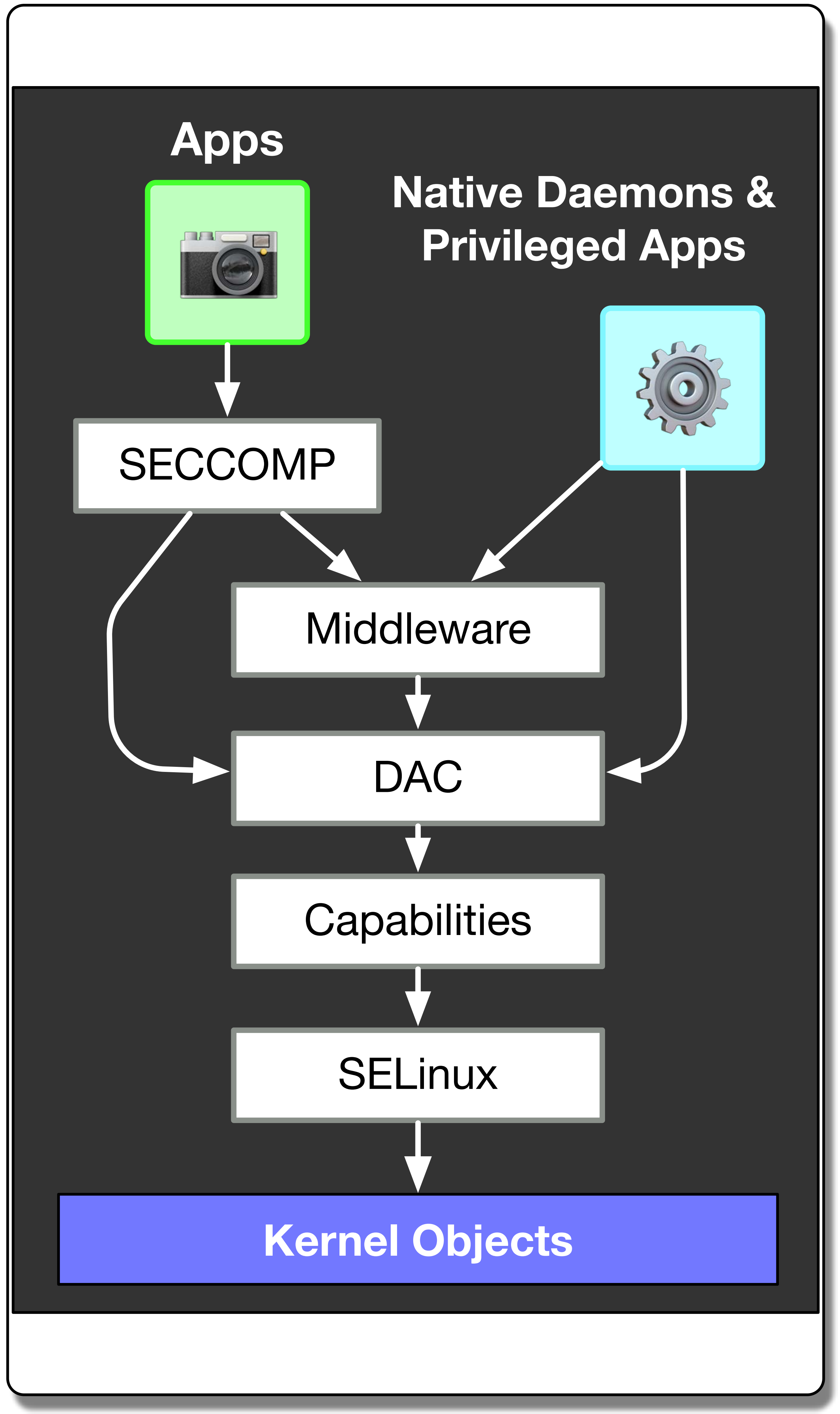
Android Security Mechanisms

- **Primary Access Control**
  - Linux DAC
  - Linux Capabilities
  - SELinux / SEAndroid (MAC)



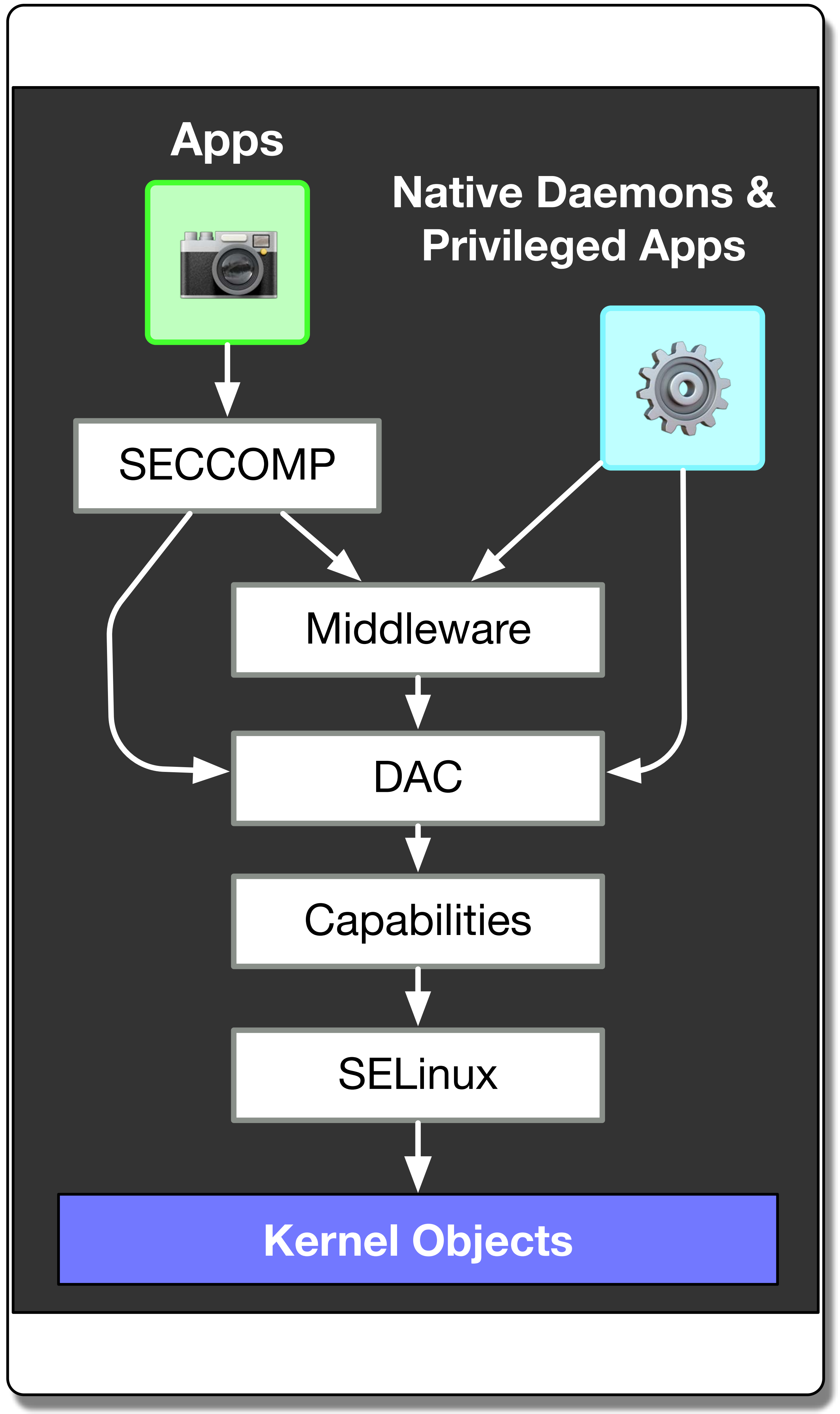
Android Security Mechanisms

- **Primary Access Control**
  - Linux DAC
  - Linux Capabilities
  - SELinux / SEAndroid (MAC)
- **Other**
  - SECCOMP
  - Android Middleware

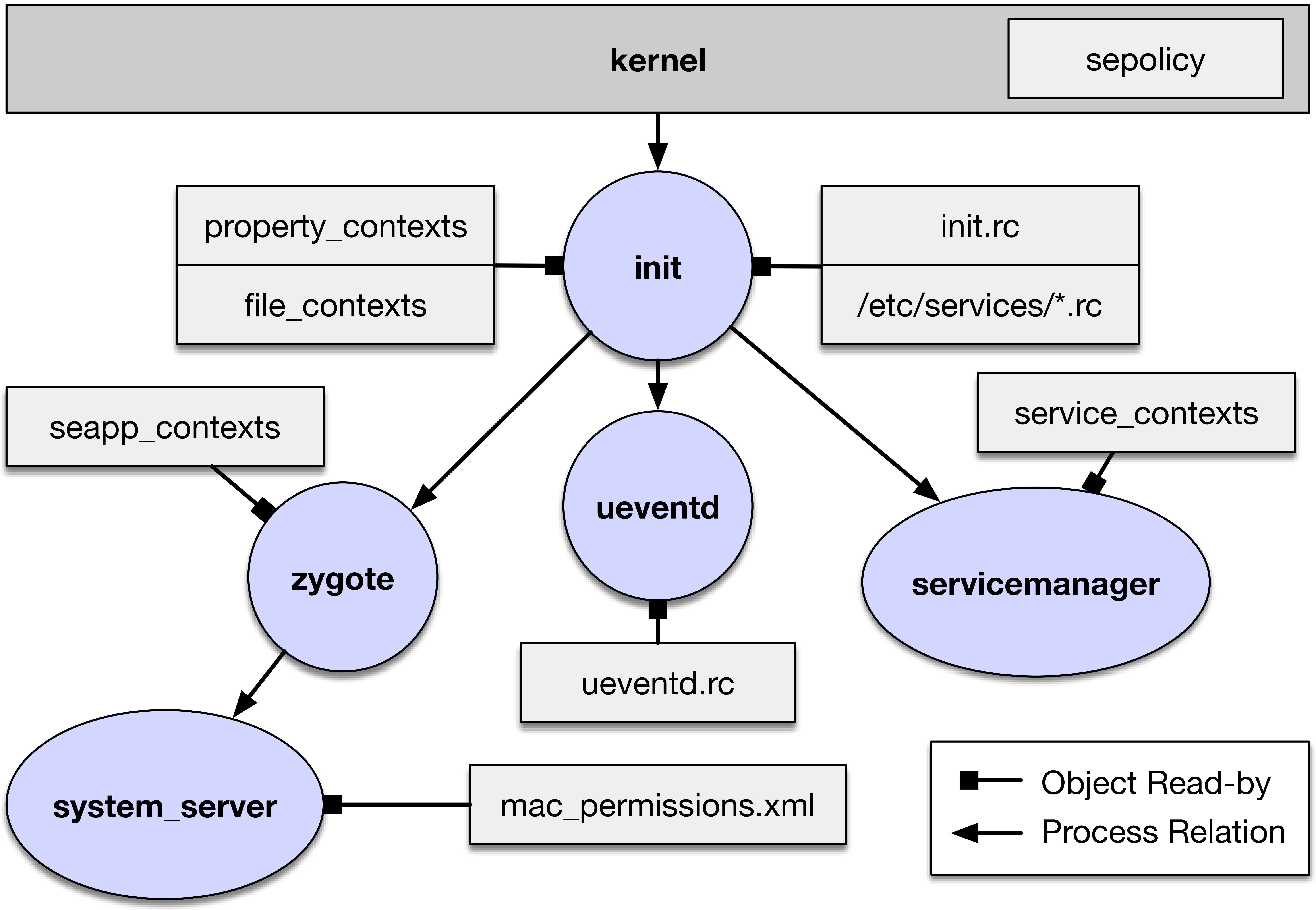


Android Security Mechanisms

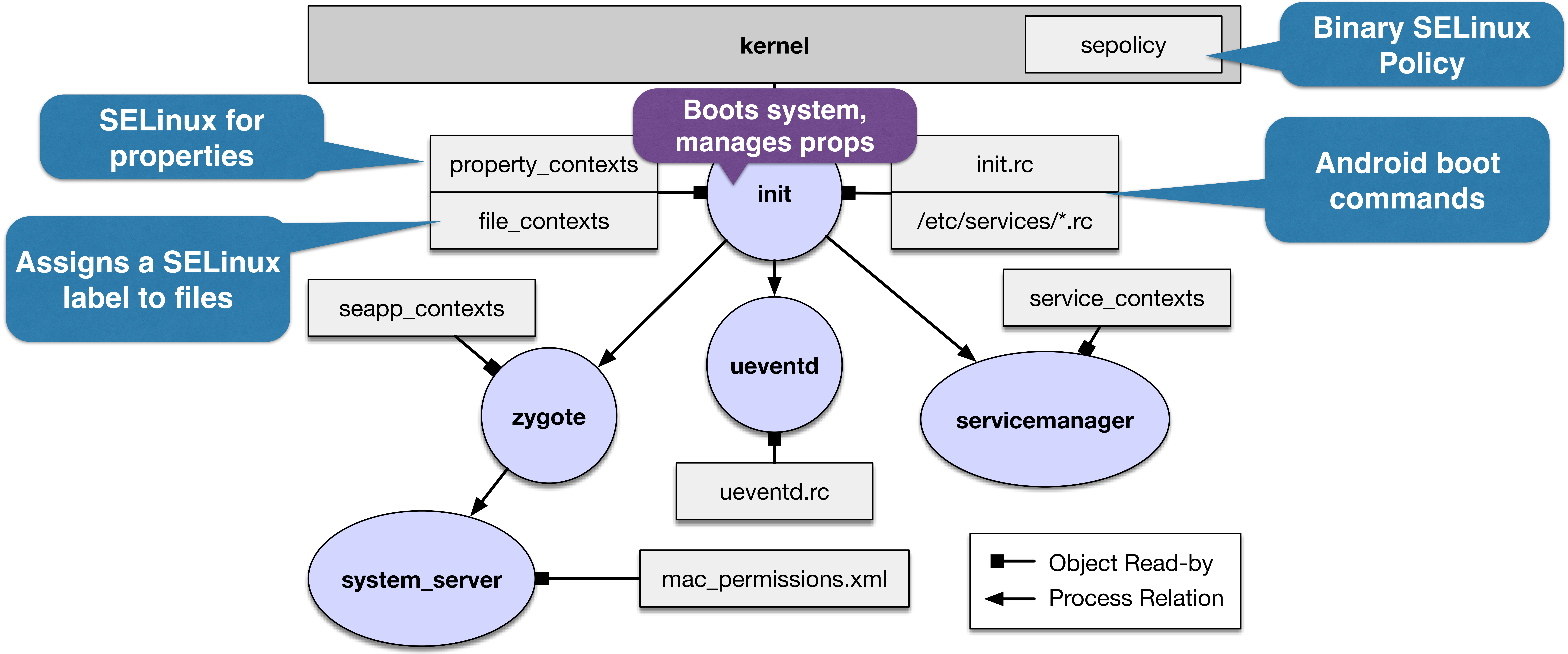
- **Primary Access Control**
  - Linux DAC
  - Linux Capabilities
  - SELinux / SEAndroid (MAC)
- **Other**
  - SECCOMP
  - Android Middleware



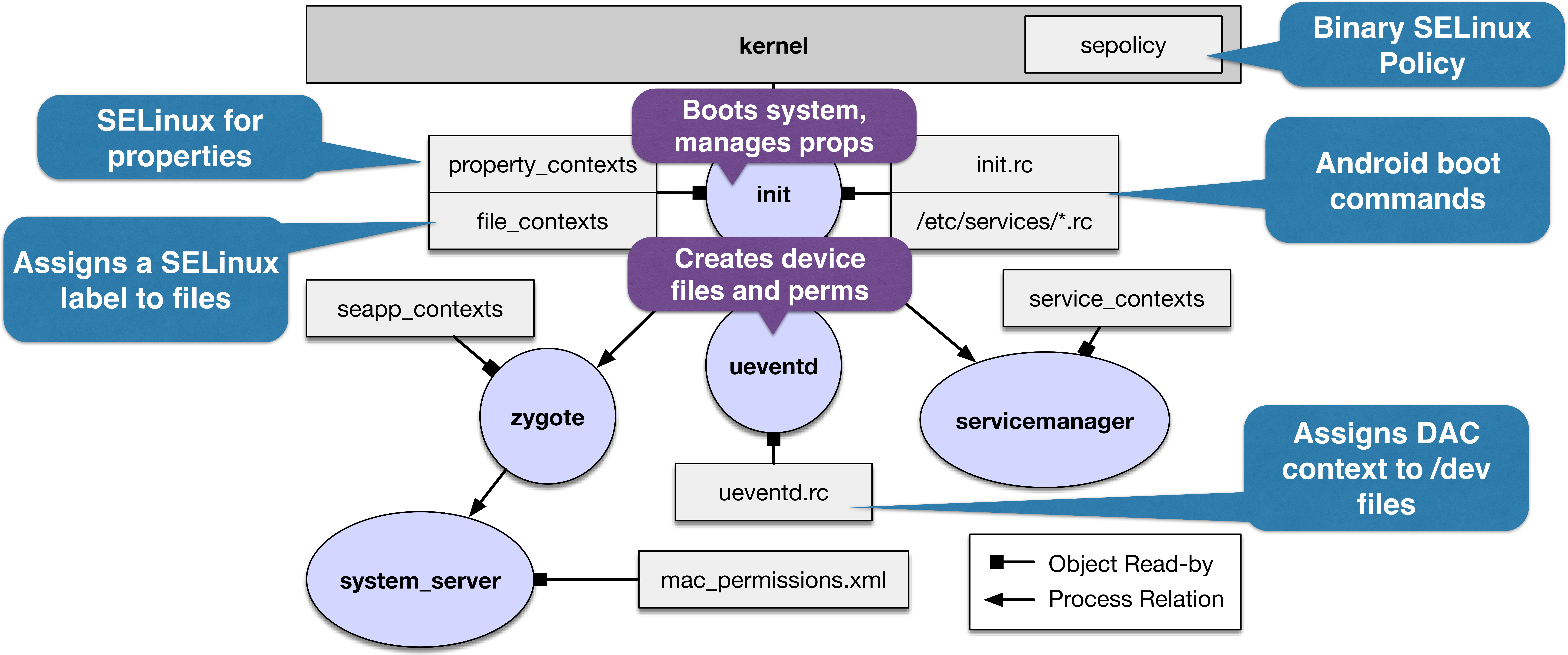
Android Security Core Files



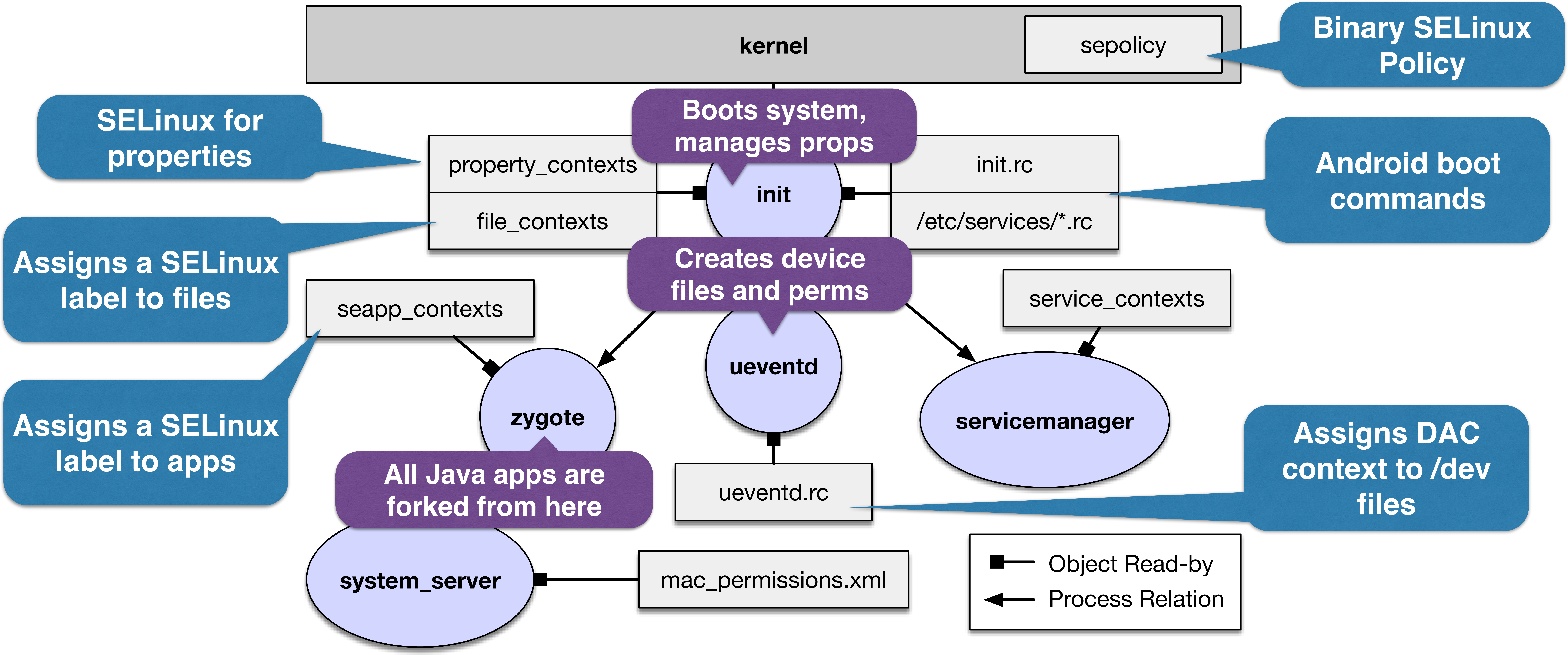
Android Security Core Files



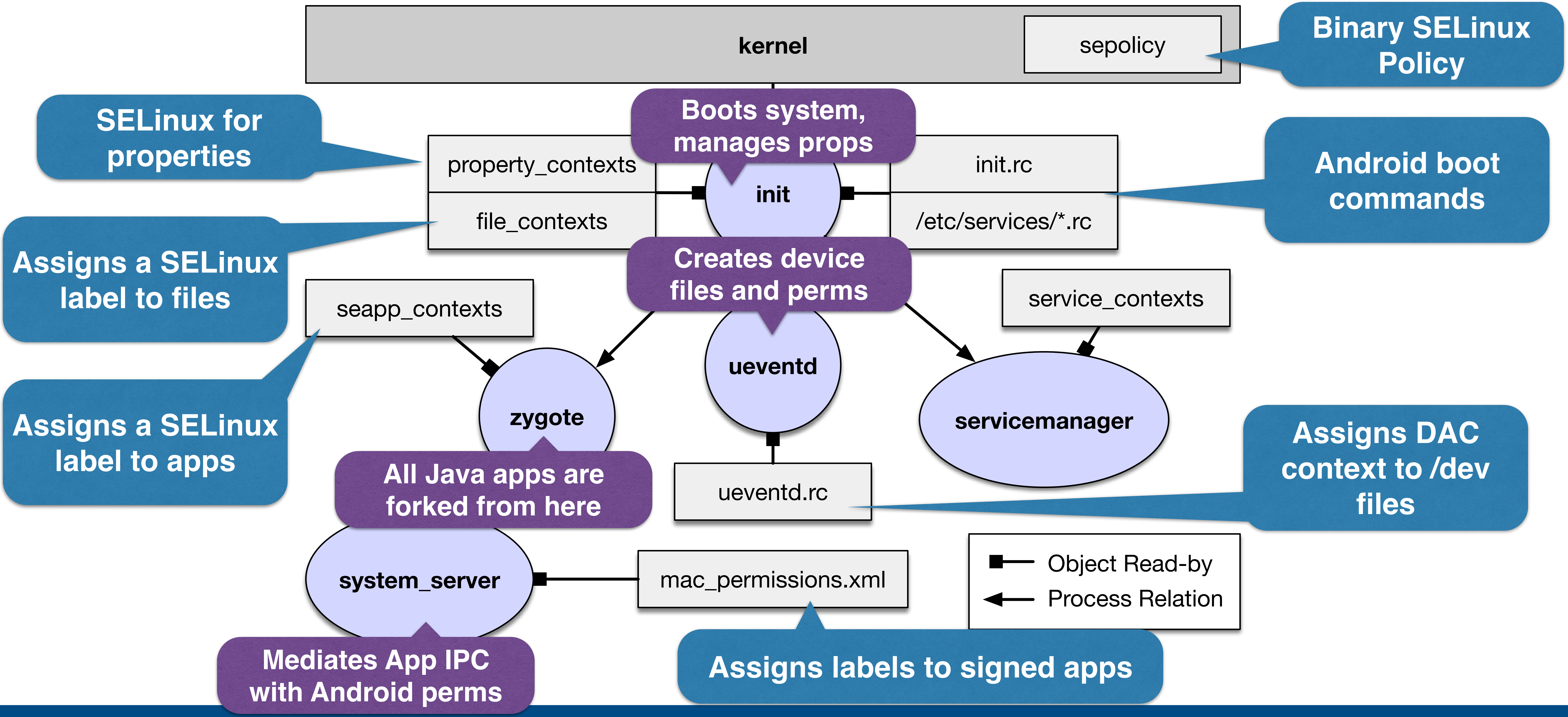
Android Security Core Files



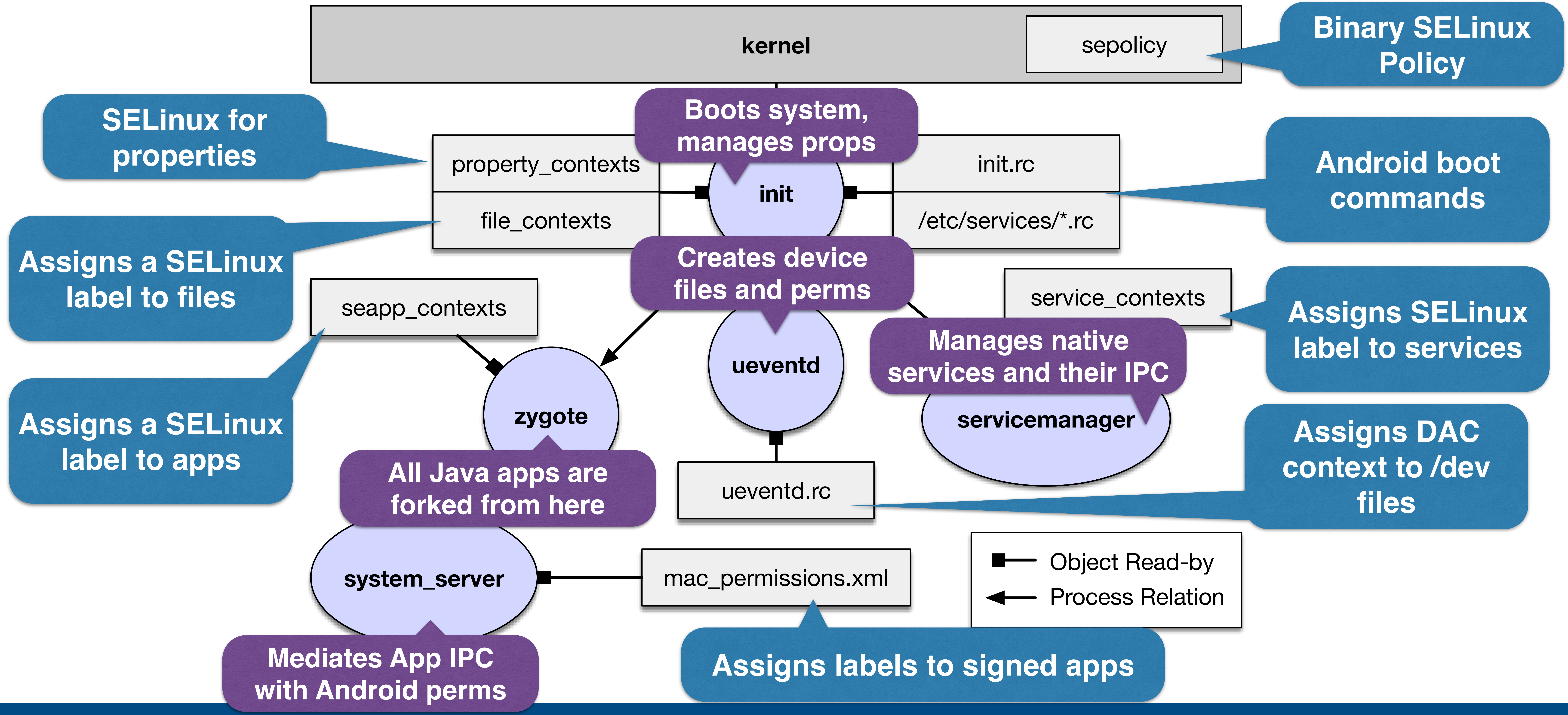
Android Security Core Files



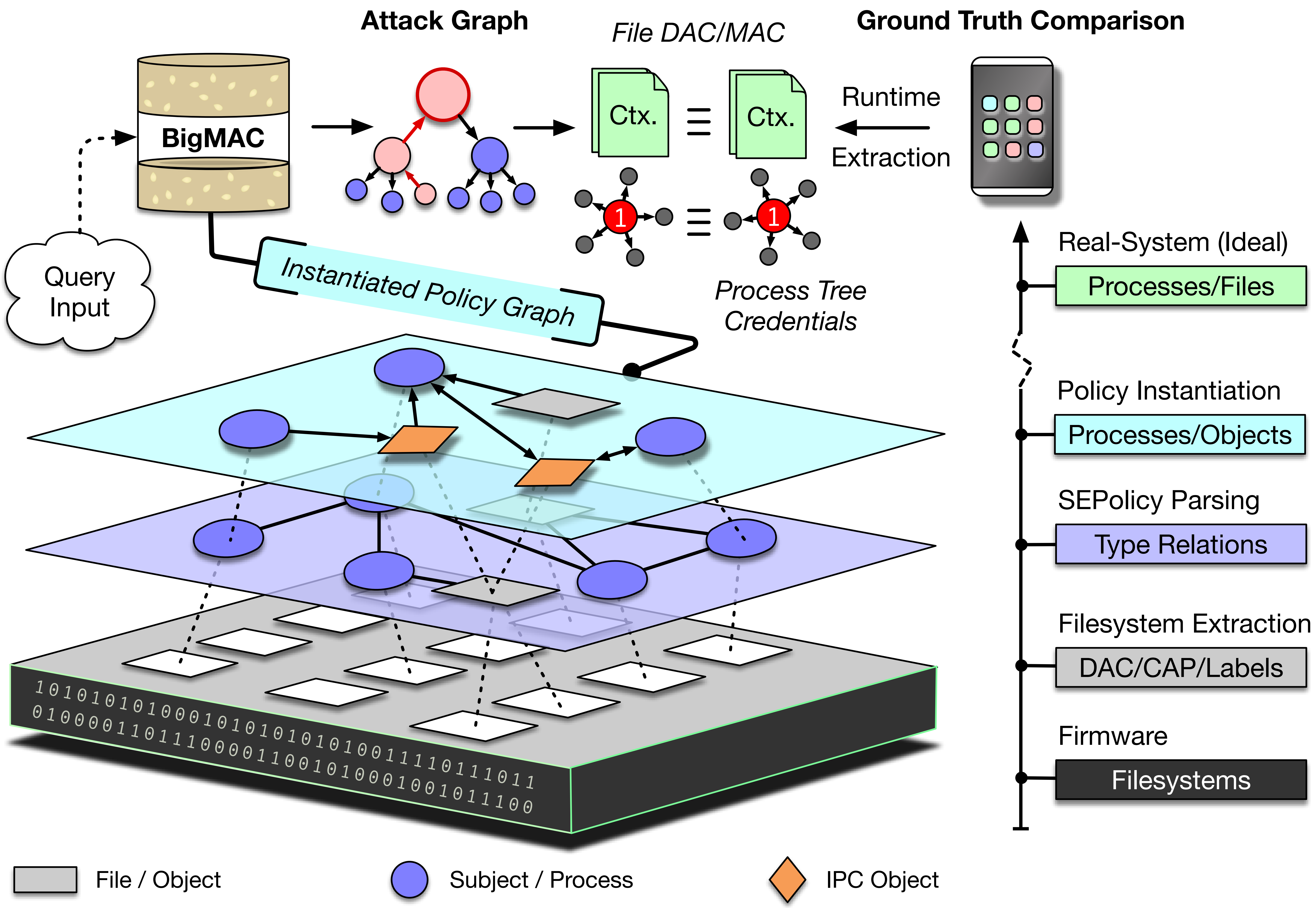
Android Security Core Files



Android Security Core Files

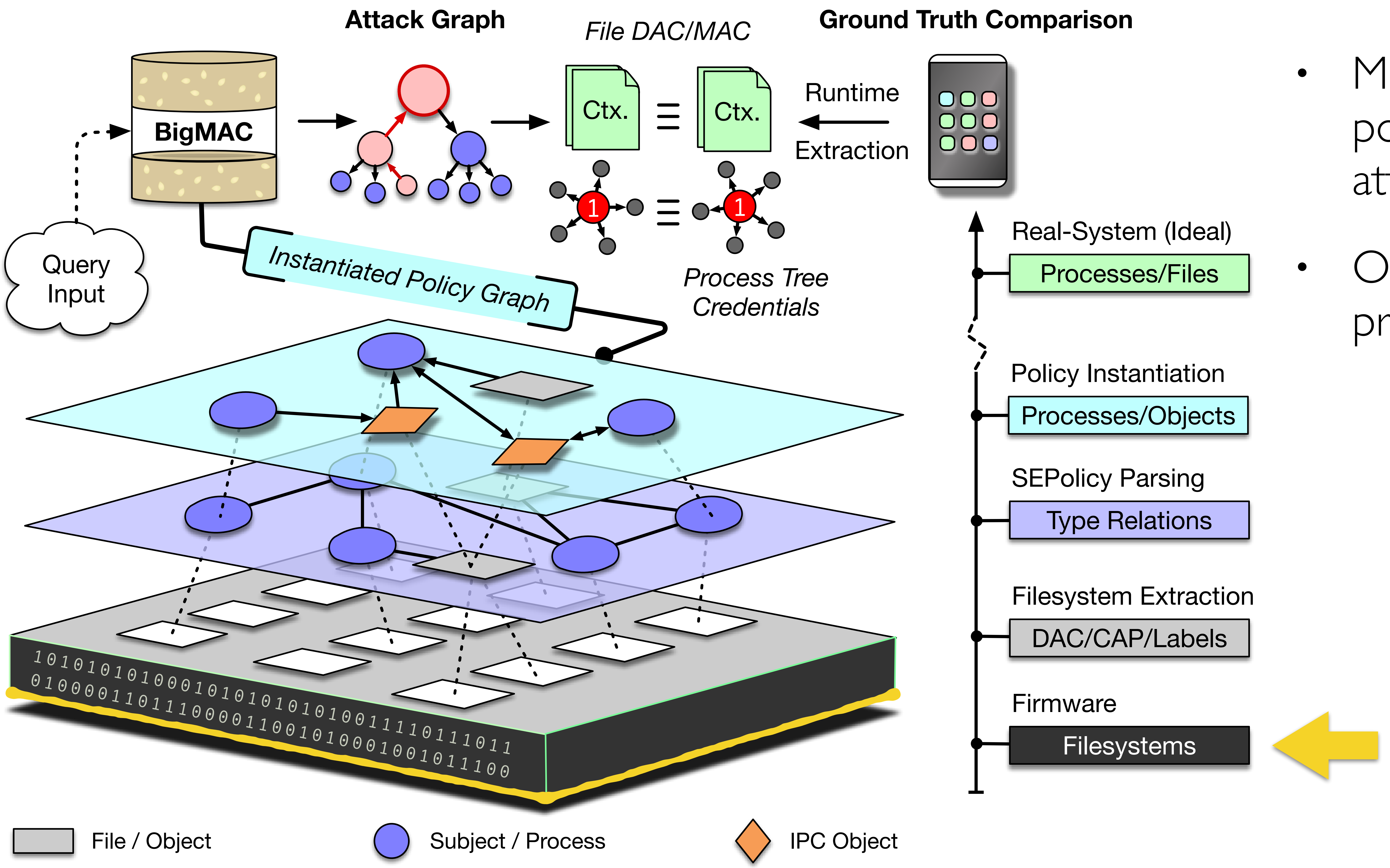

kernel
sepolicy
Binary SELinux Policy
SELinux for properties
Boots system, manages props
property_contexts
init
init.rc
Android boot commands
file_contexts
/etc/services/*.rc
Assigns a SELinux label to files
Creates device files and perms
seapp_contexts
service_contexts
Assigns SELinux label to services
ueventd
Manages native services and their IPC
Assigns a SELinux label to apps
zygote
servicemanager
Assigns DAC context to /dev files
All Java apps are forked from here
ueventd.rc
system_server
mac_permissions.xml
■— Object Read-by
←— Process Relation
Mediates App IPC with Android perms
Assigns labels to signed apps

BigMAC at a High Level



- Maps MAC+DAC+CAP policies onto a fine-grained attack-graph
- Only considers running processes and present files

BigMAC at a High Level



Attack Graph

File DAC/MAC

Ground Truth Comparison

BigMAC

Query Input

Instantiated Policy Graph

Ctx. ≡ Ctx.

Runtime Extraction

Process Tree Credentials

Real-System (Ideal)

Processes/Files

Policy Instantiation

Processes/Objects

SEPolicy Parsing

Type Relations

Filesystem Extraction

DAC/CAP/Labels

Firmware

Filesystems

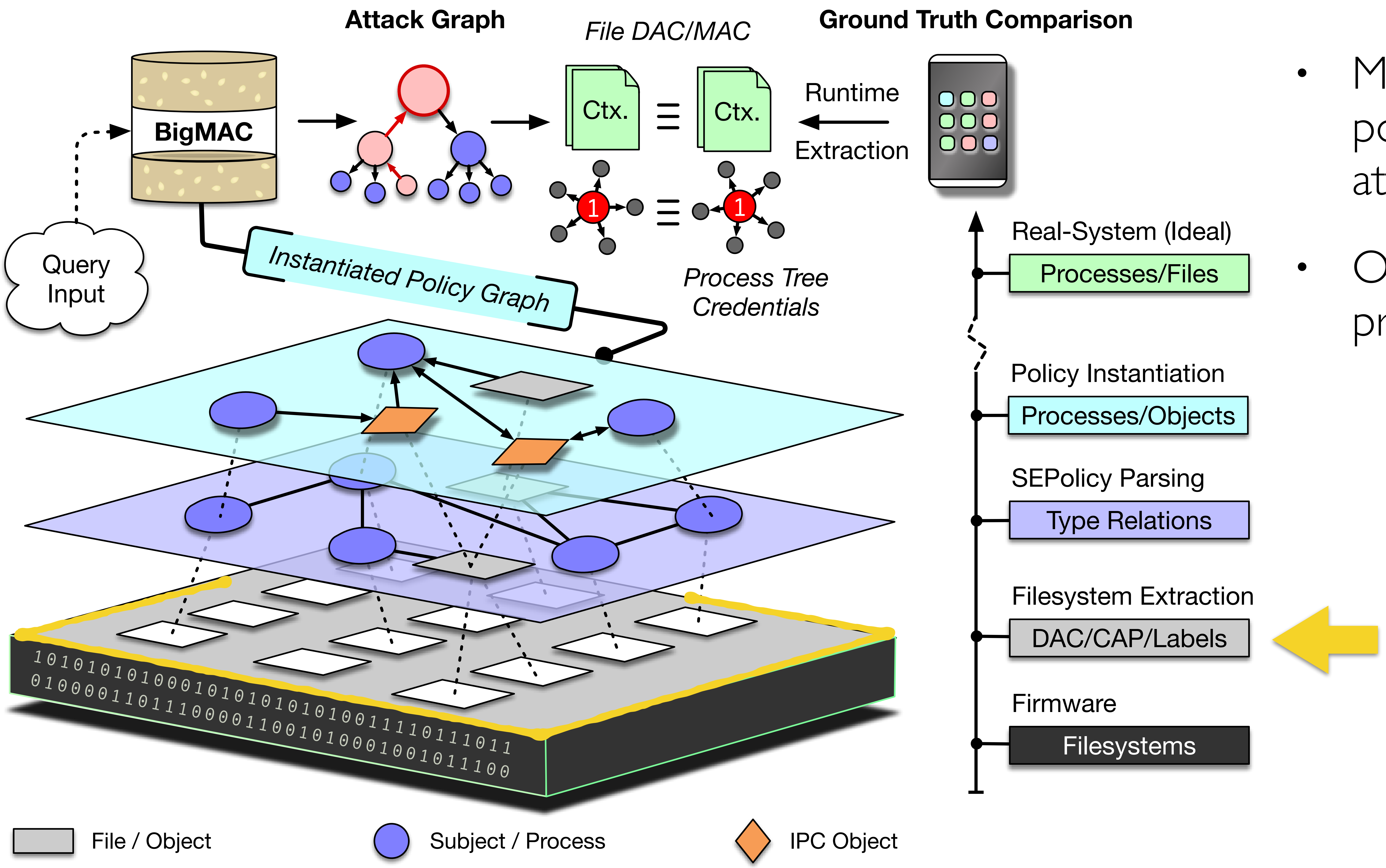
▭ File / Object

● Subject / Process

◆ IPC Object

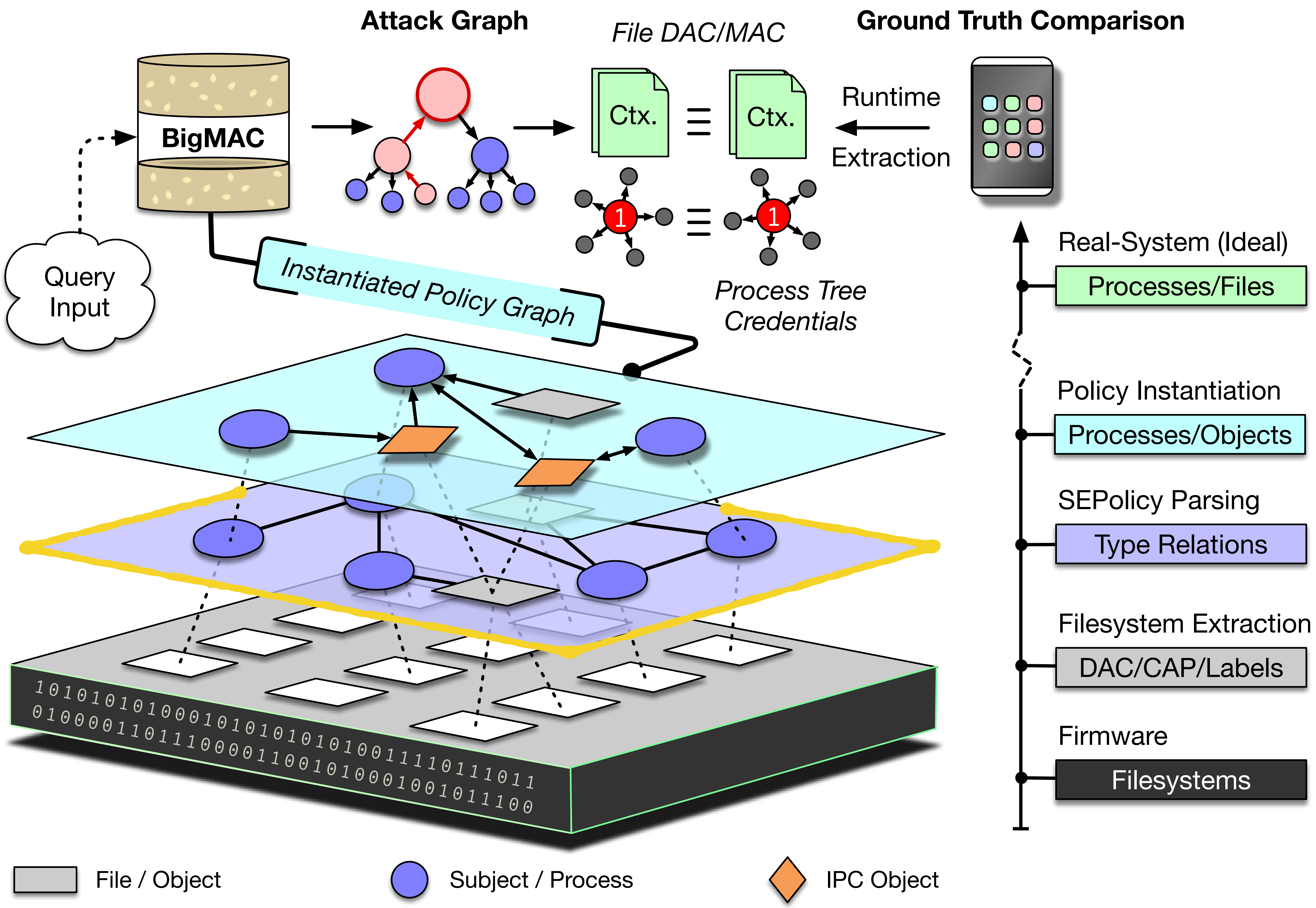
- Maps MAC+DAC+CAP policies onto a fine-grained attack-graph
- Only considers running processes and present files

BigMAC at a High Level



- Maps MAC+DAC+CAP policies onto a fine-grained attack-graph
- Only considers running processes and present files

BigMAC at a High Level



Real-System (Ideal)

Processes/Files

Policy Instantiation

Processes/Objects

SEPolicy Parsing

Type Relations

Filesystem Extraction

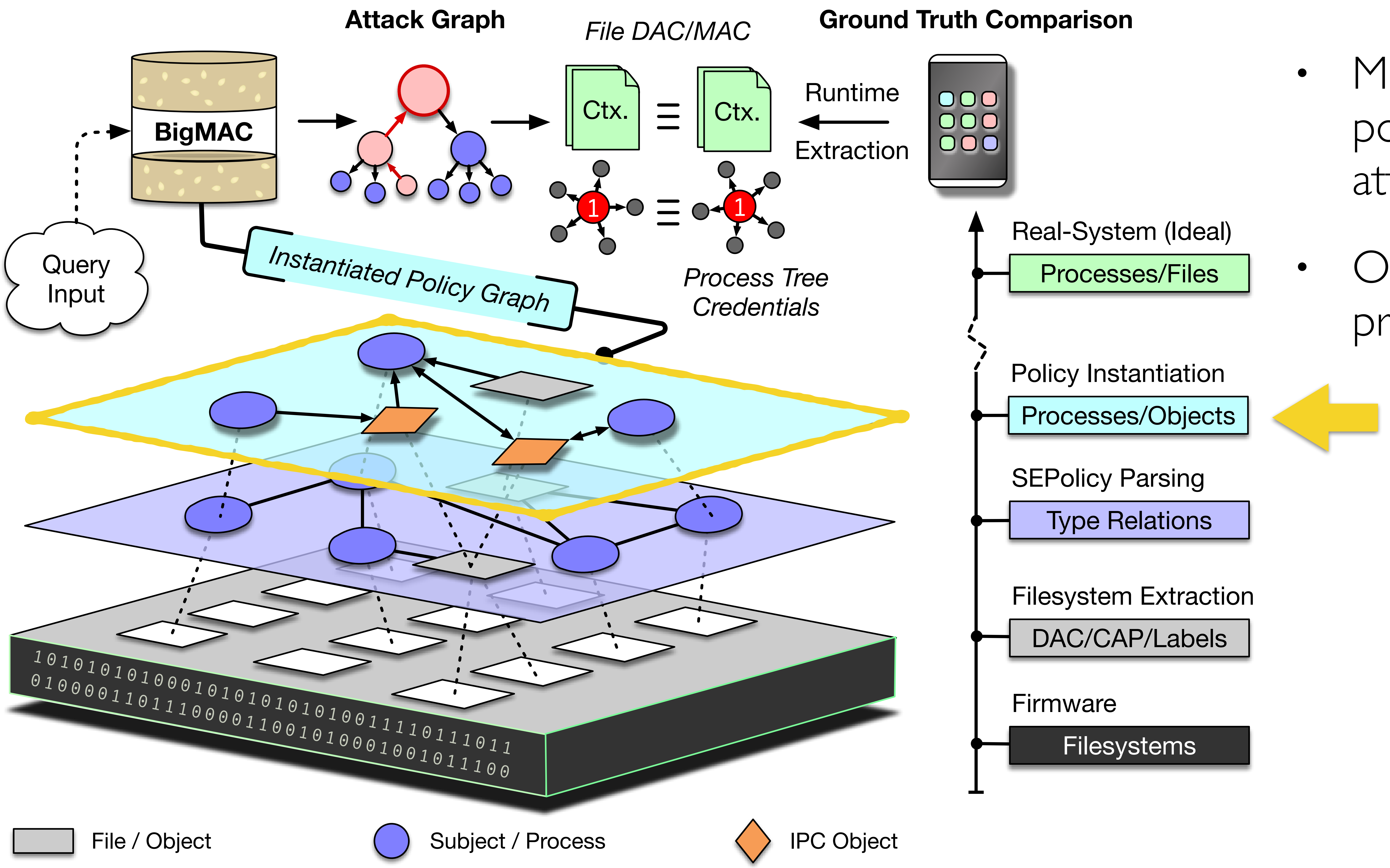
DAC/CAP/Labels

Firmware

Filesystems

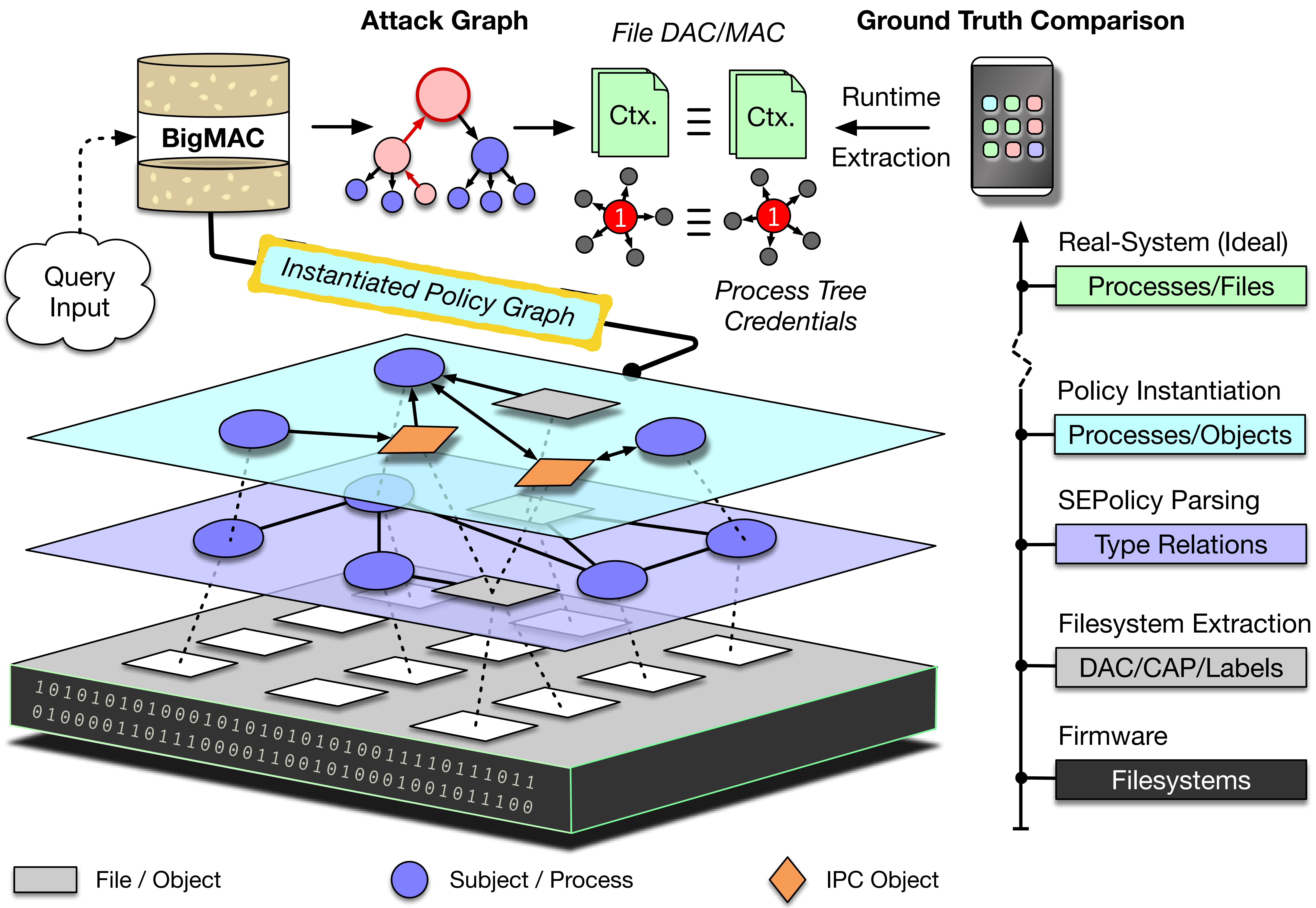
- Maps MAC+DAC+CAP policies onto a fine-grained attack-graph
- Only considers running processes and present files

BigMAC at a High Level



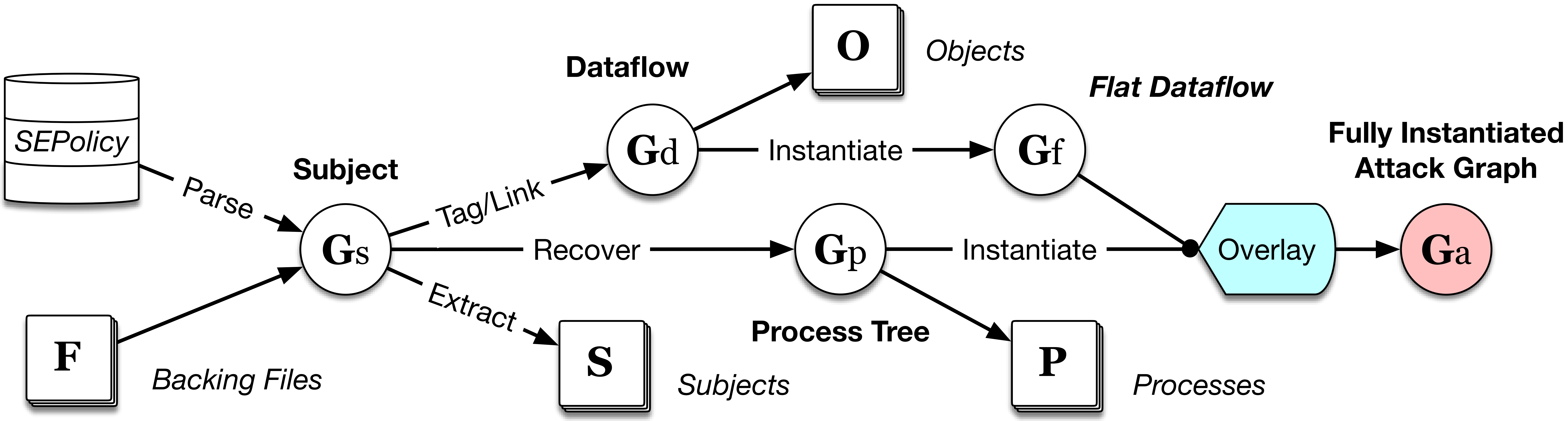
- Maps MAC+DAC+CAP policies onto a fine-grained attack-graph
- Only considers running processes and present files

BigMAC at a High Level

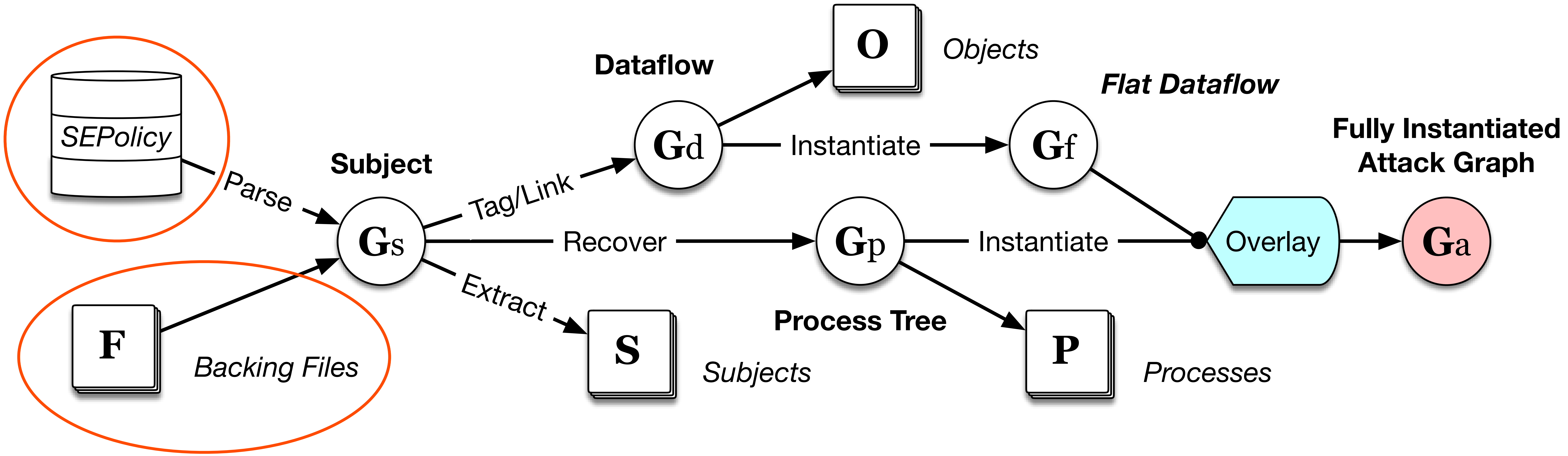


- Maps MAC+DAC+CAP policies onto a fine-grained attack-graph
- Only considers running processes and present files

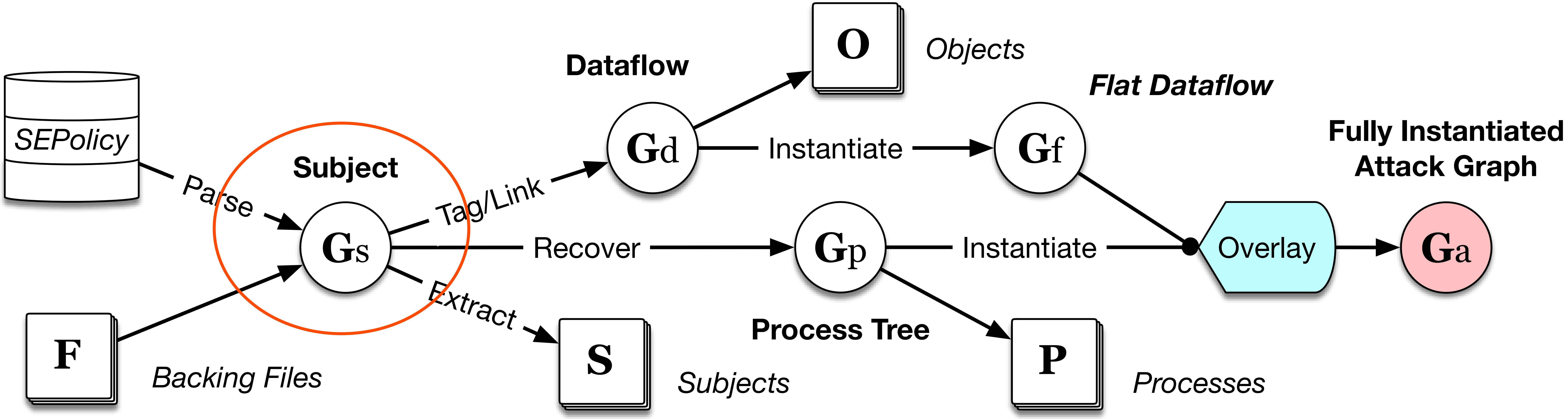
Building an Attack-Graph



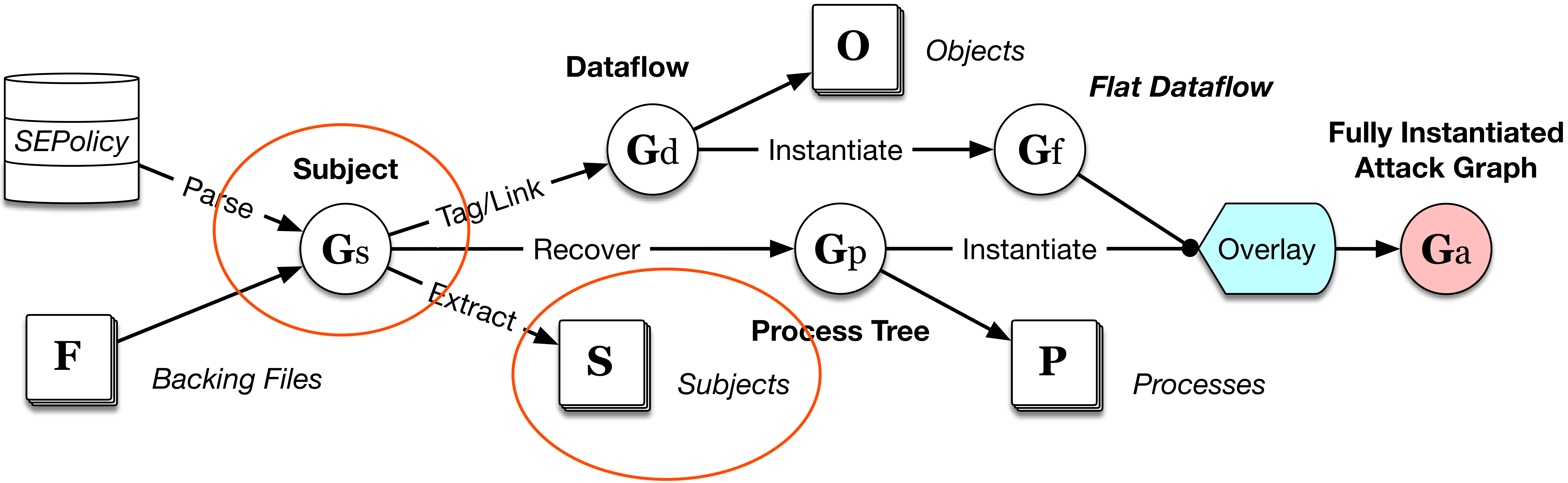
Building an Attack-Graph



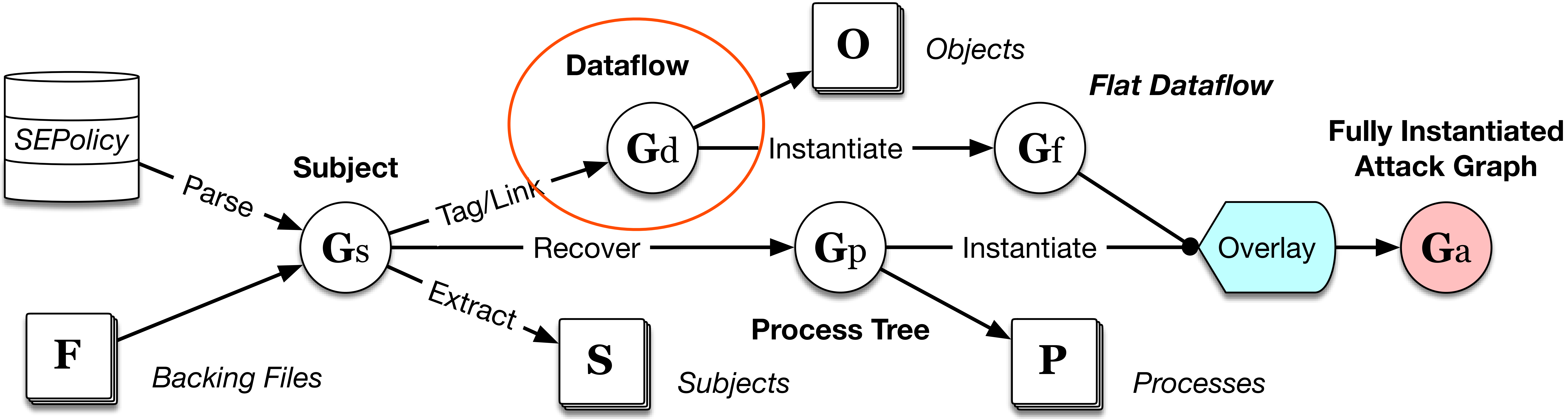
Building an Attack-Graph



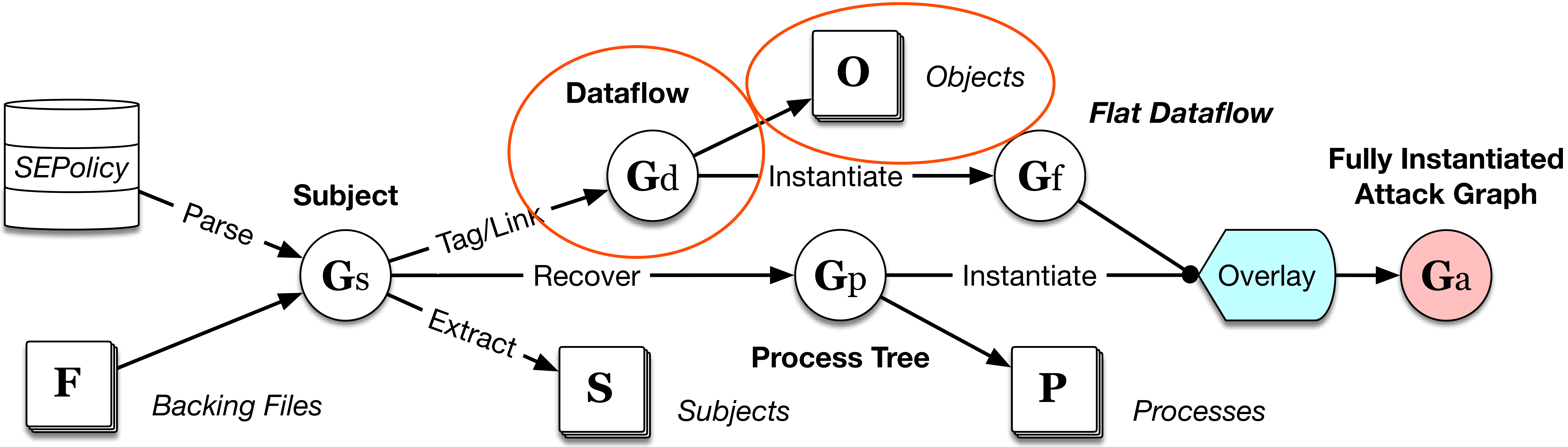
Building an Attack-Graph



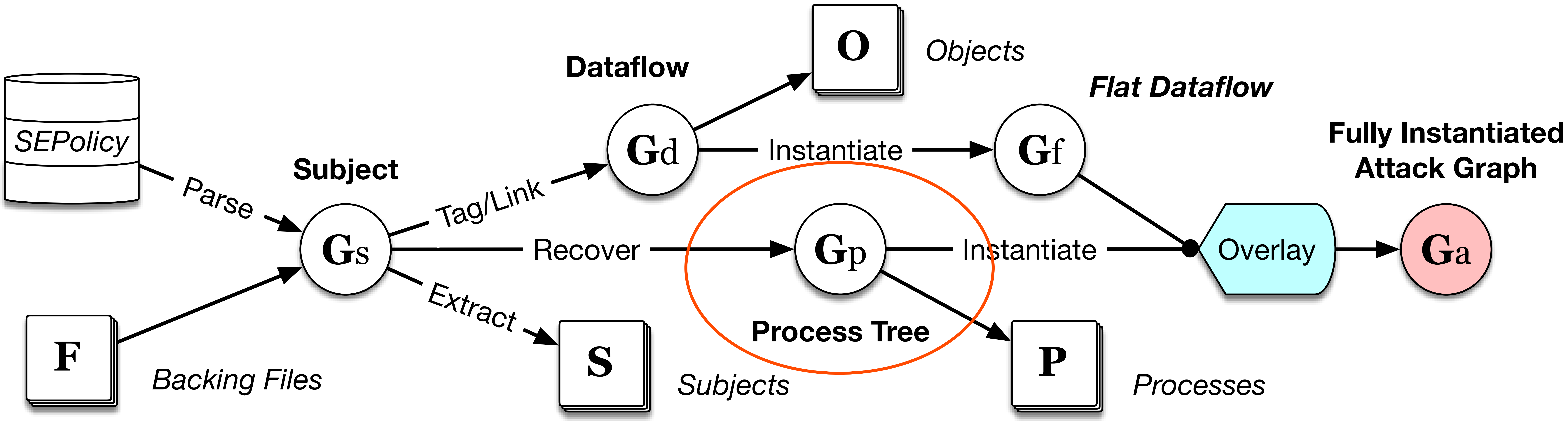
Building an Attack-Graph



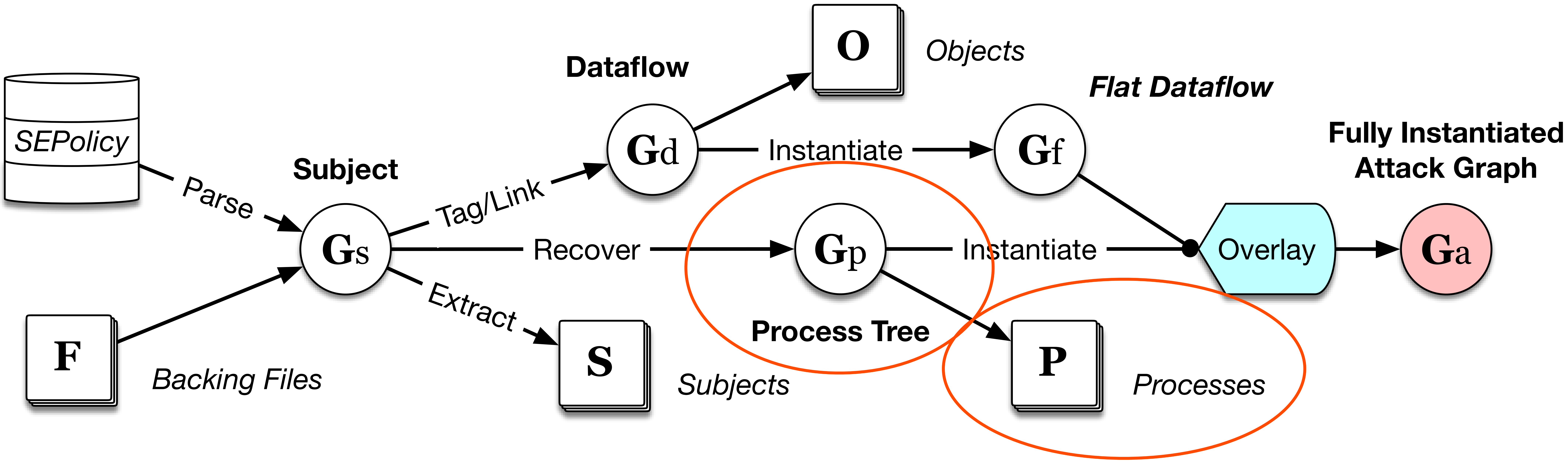
Building an Attack-Graph



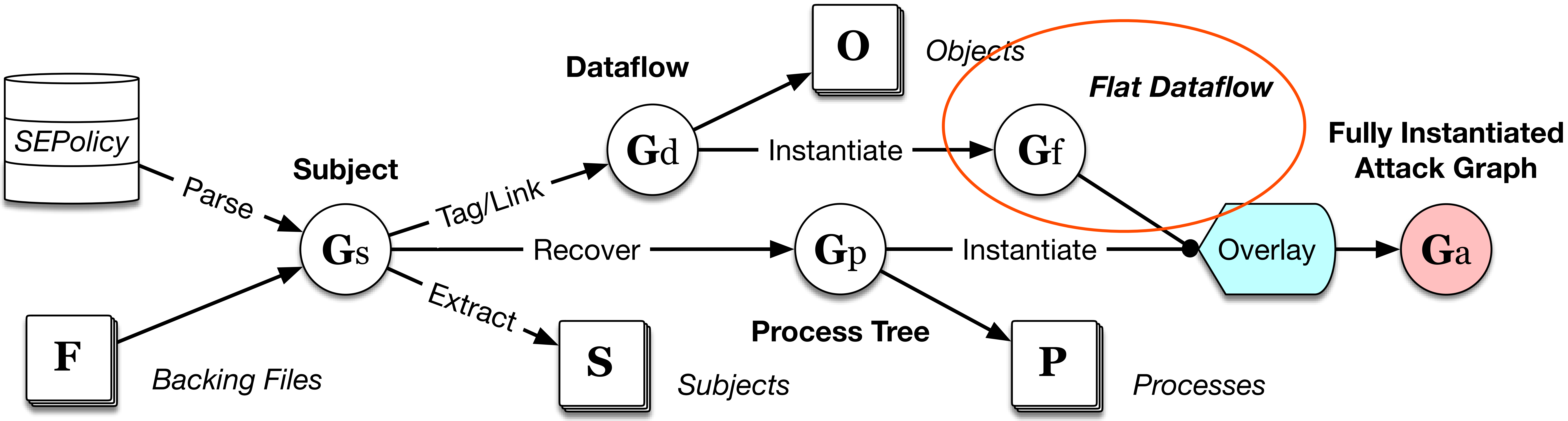
Building an Attack-Graph



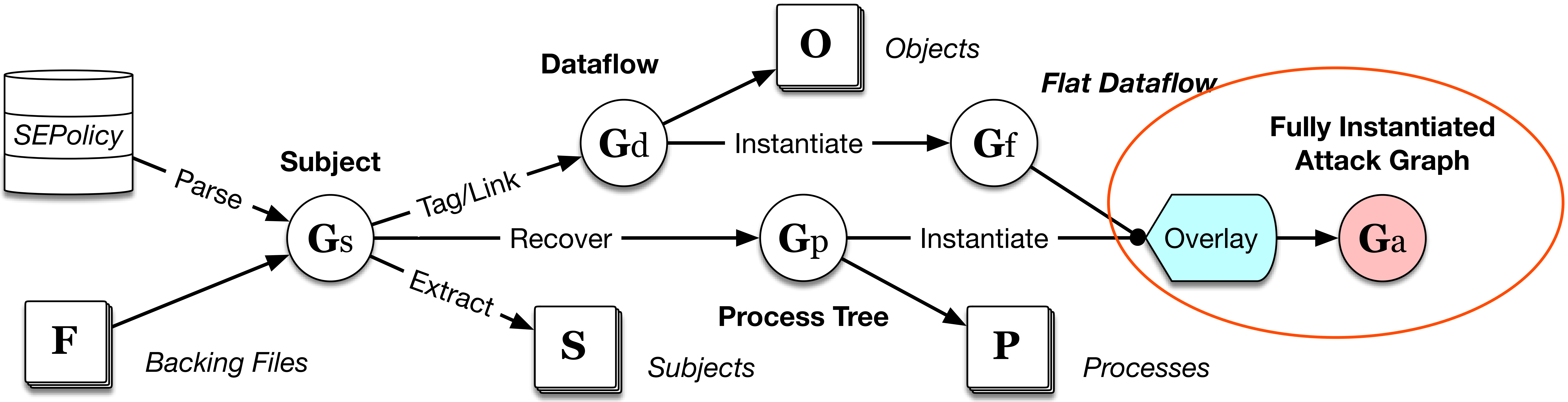
Building an Attack-Graph



Building an Attack-Graph



Building an Attack-Graph



Processes Recovery

Processes Recovery

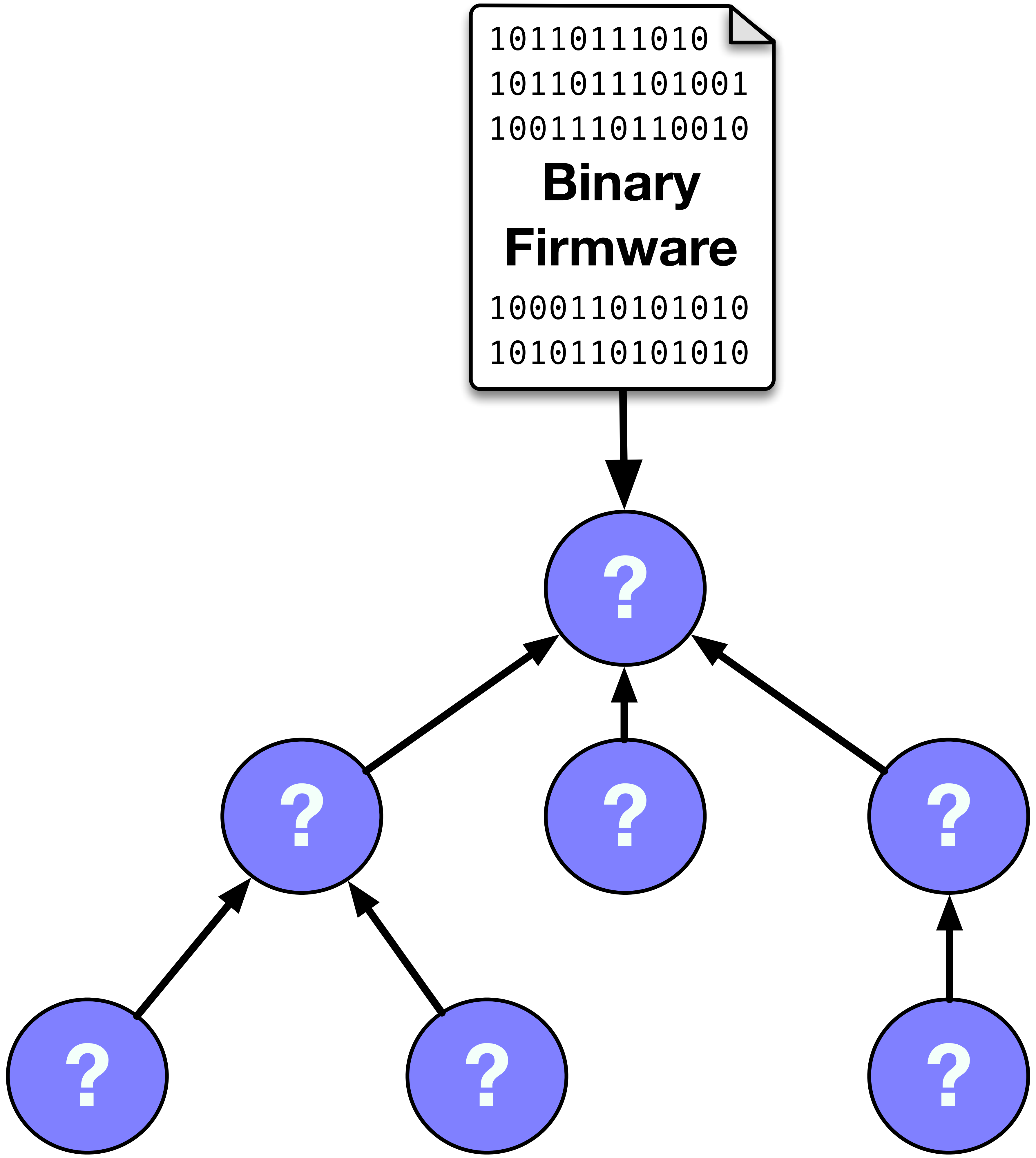
- We want to know what objects processes can access based upon the system policy

Processes Recovery

- We want to know what objects processes can access based upon the system policy
  - This is based upon their permissions (UID, GID, label, capabilities)

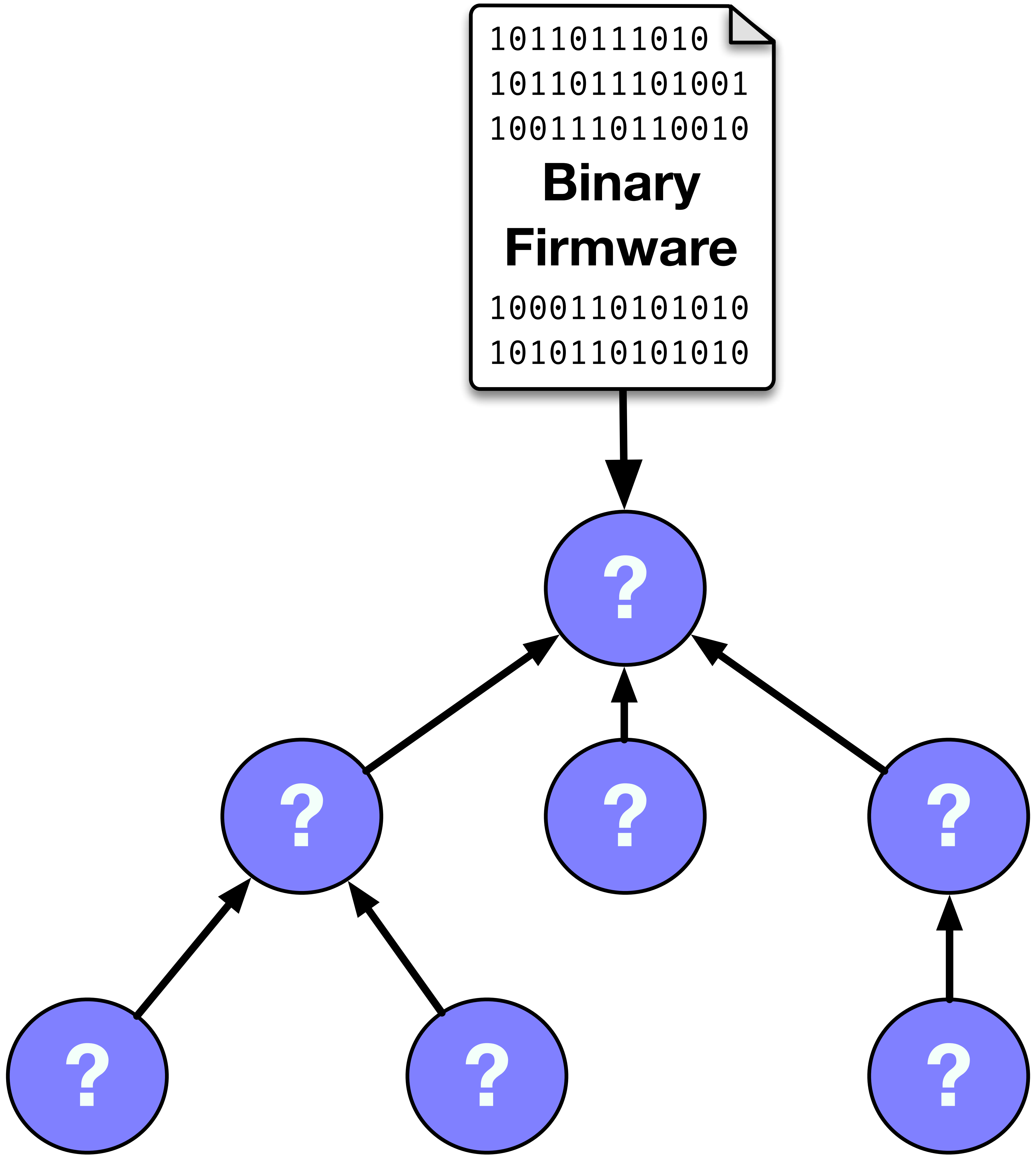
Processes Recovery

- We want to know what objects processes can access based upon the system policy
    - This is based upon their permissions (UID, GID, label, capabilities)
- ***We have no processes in static firmware!***

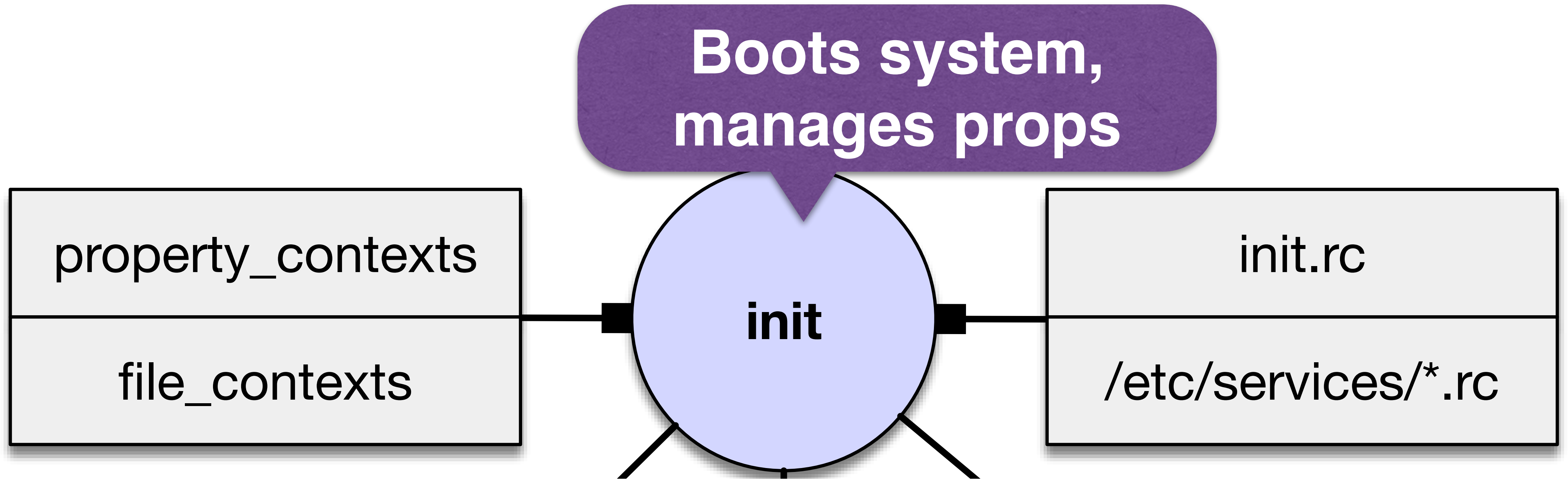


Processes Recovery

- We want to know what objects processes can access based upon the system policy
  - This is based upon their permissions (UID, GID, label, capabilities)
- ***We have no processes in static firmware!***
  - Can we recover processes and their credentials just from firmware?

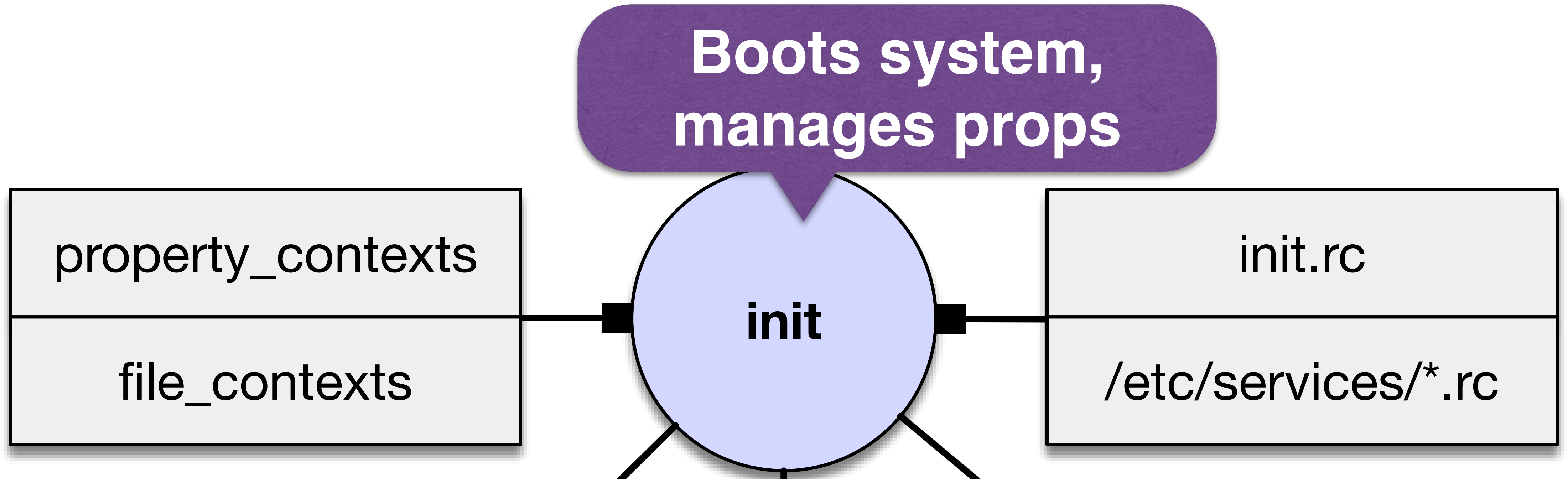


Emulating Android's Boot



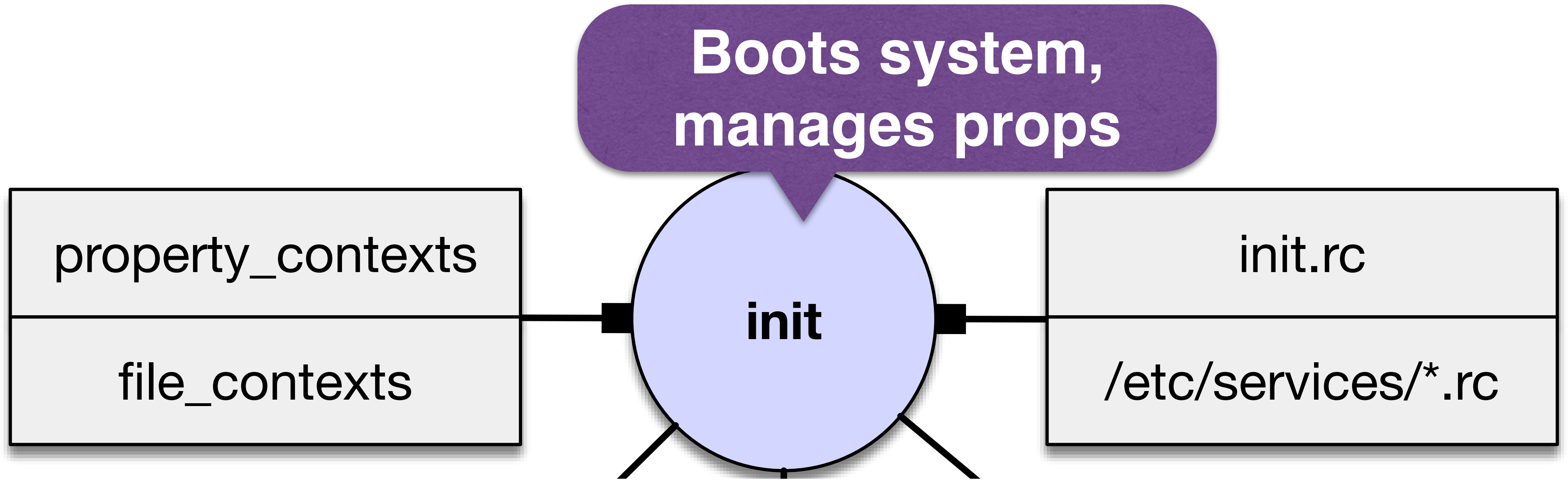
Emulating Android's Boot

- Android's boot process is well-specified by the platform



Emulating Android's Boot

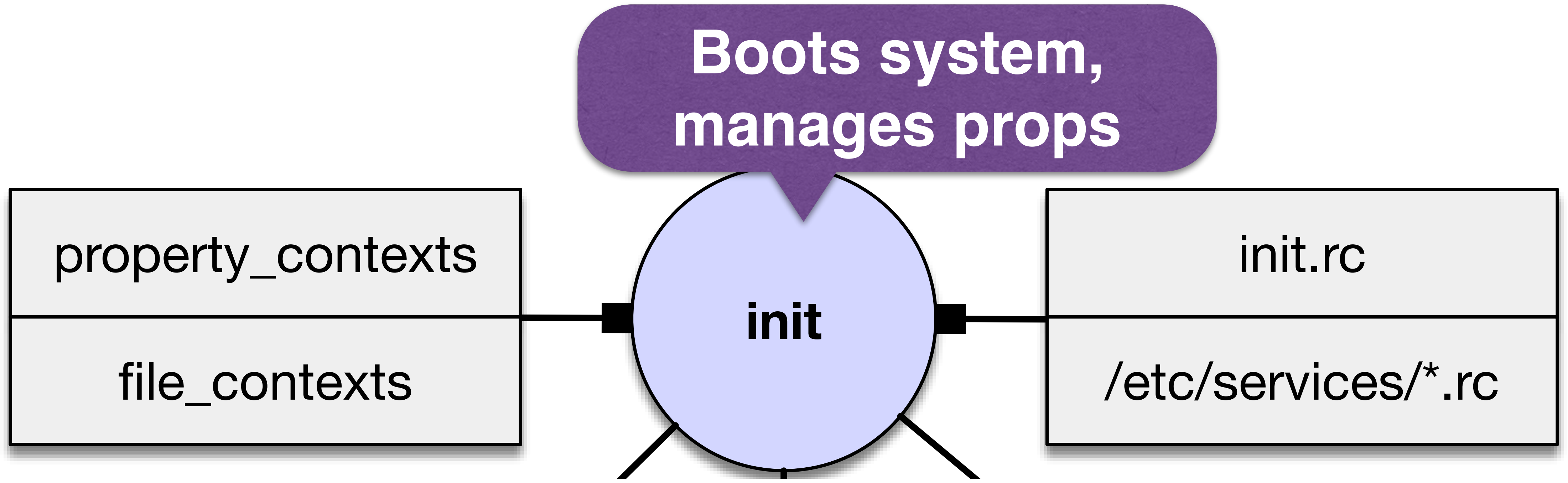
- Android's boot process is well-specified by the platform
- `Init.rc` files are loaded describing services, or native daemons



Boots system, manages props

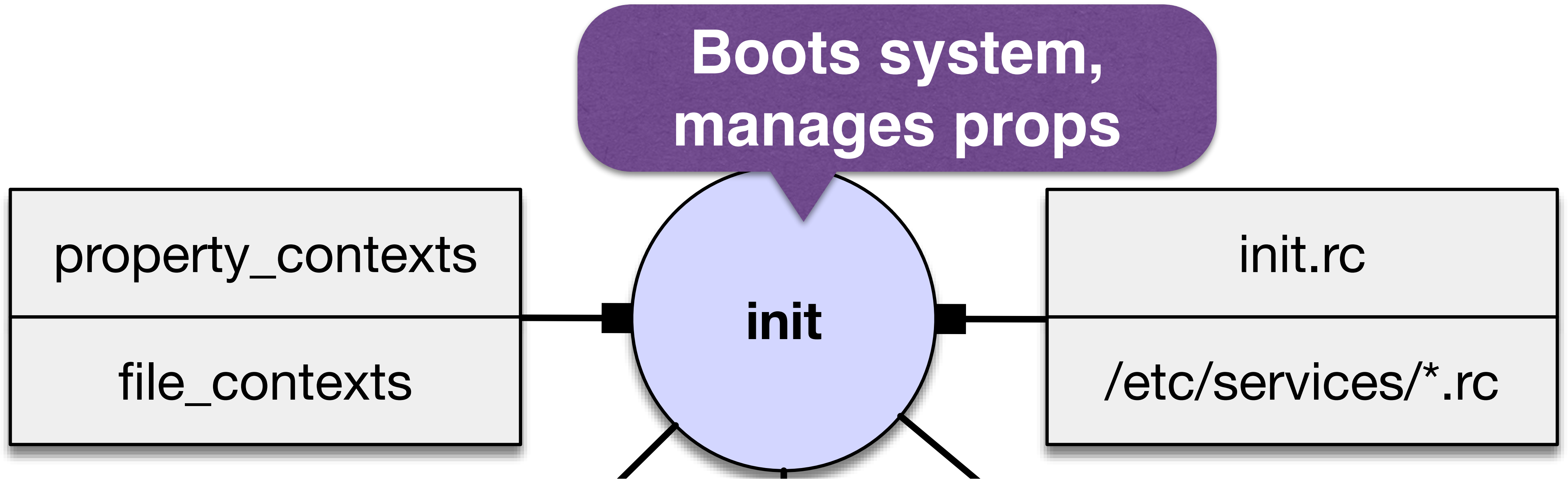
Emulating Android’s Boot

- Android’s boot process is well-specified by the platform
- `Init.rc` files are loaded describing services, or native daemons
- Explicit credential assignment for services



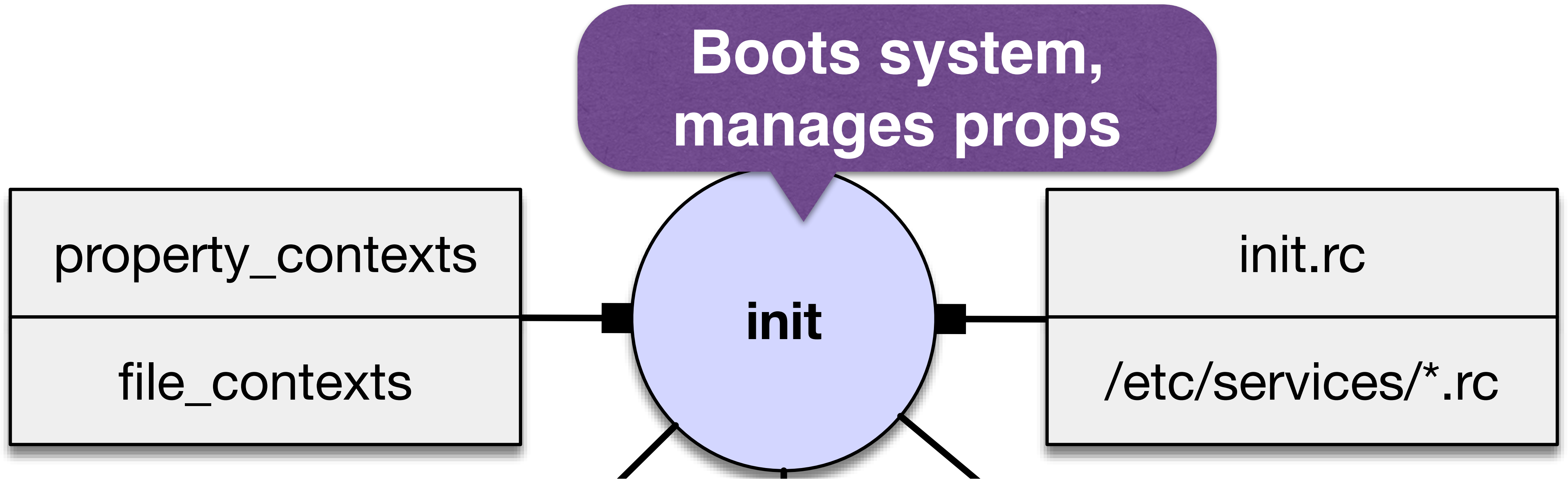
Emulating Android's Boot

- Android's boot process is well-specified by the platform
- `Init.rc` files are loaded describing services, or native daemons
- Explicit credential assignment for services
- Allows the capture of boot-time changes to the filesystem



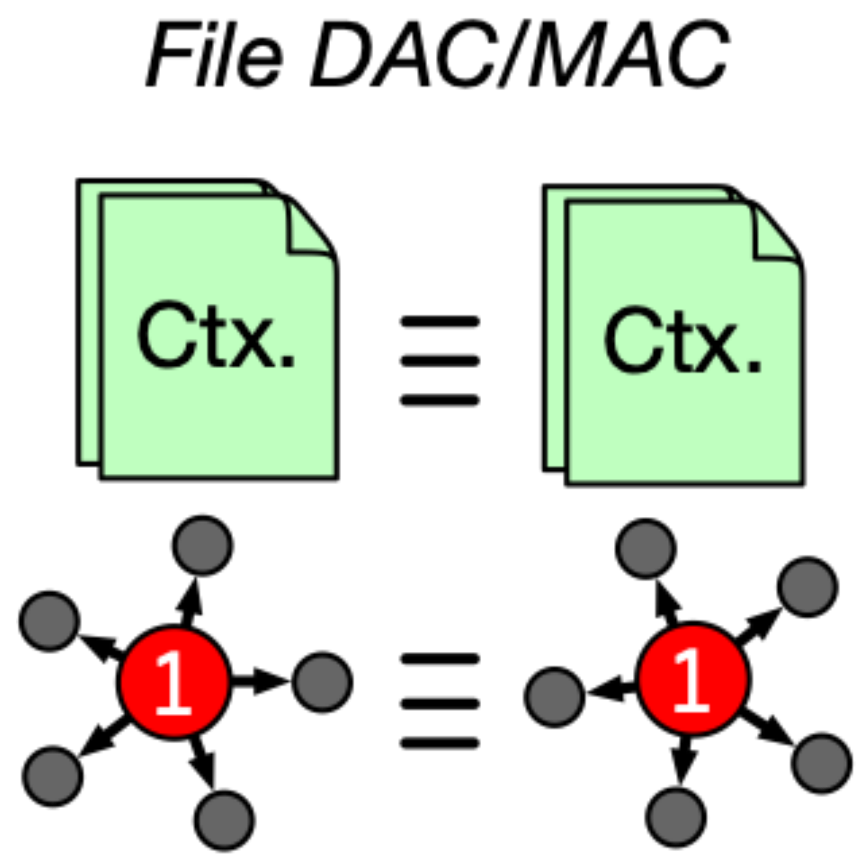
Emulating Android's Boot

- Android's boot process is well-specified by the platform
- `Init.rc` files are loaded describing services, or native daemons
- Explicit credential assignment for services
- Allows the capture of boot-time changes to the filesystem
- Without incorporating this, cross-vendor analysis doesn't scale and accuracy suffers

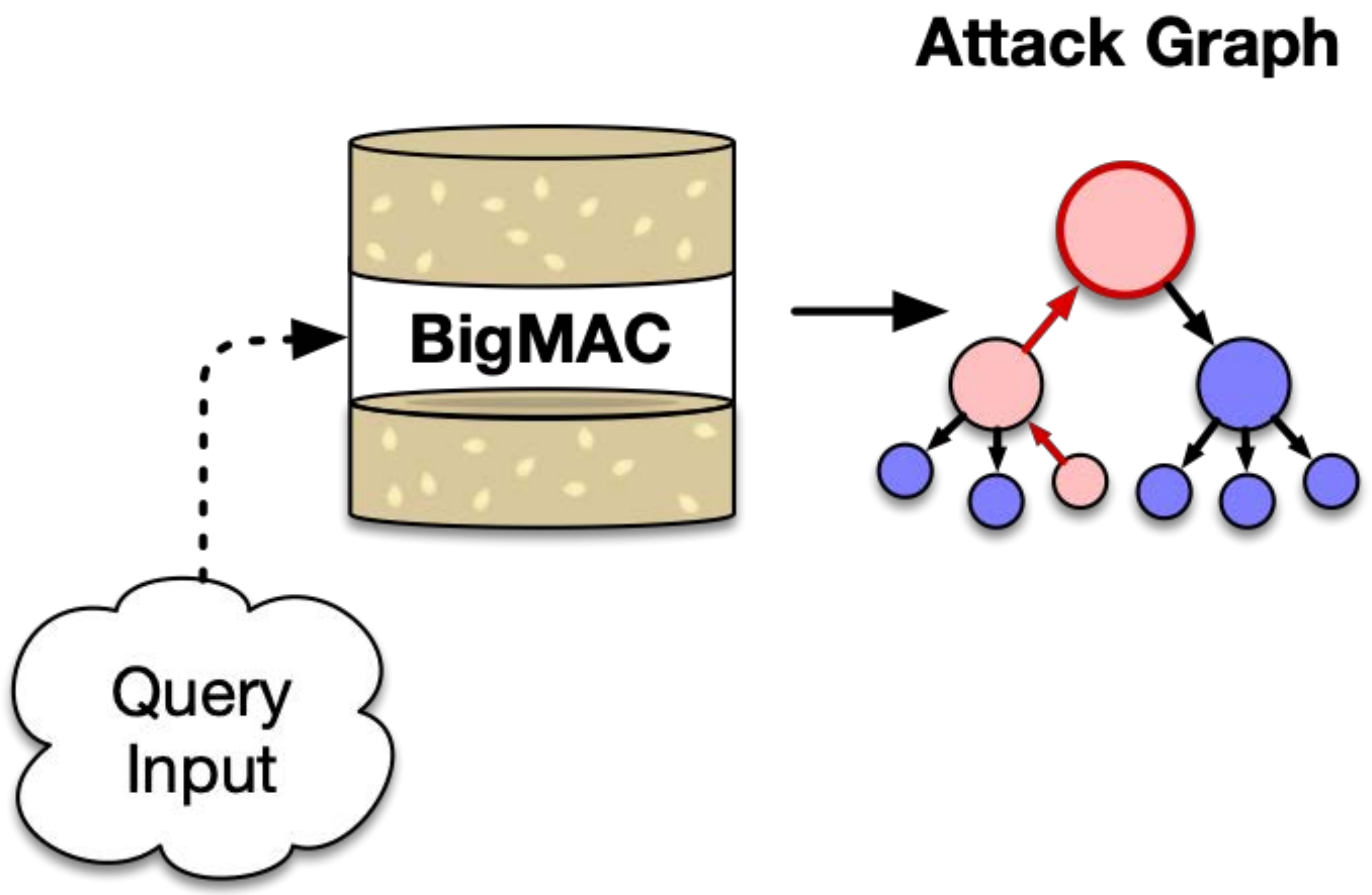


Evaluation of BigMAC

- **Ground Truth Evaluation**
  - How does BigMAC recovery compare to extracting security policies from a running device?



- **Attack Surface Case Studies**
  - Evaluation of our Prolog query engine to discover attack paths from and to critical Android components



Ground-truth Evaluation (Files)

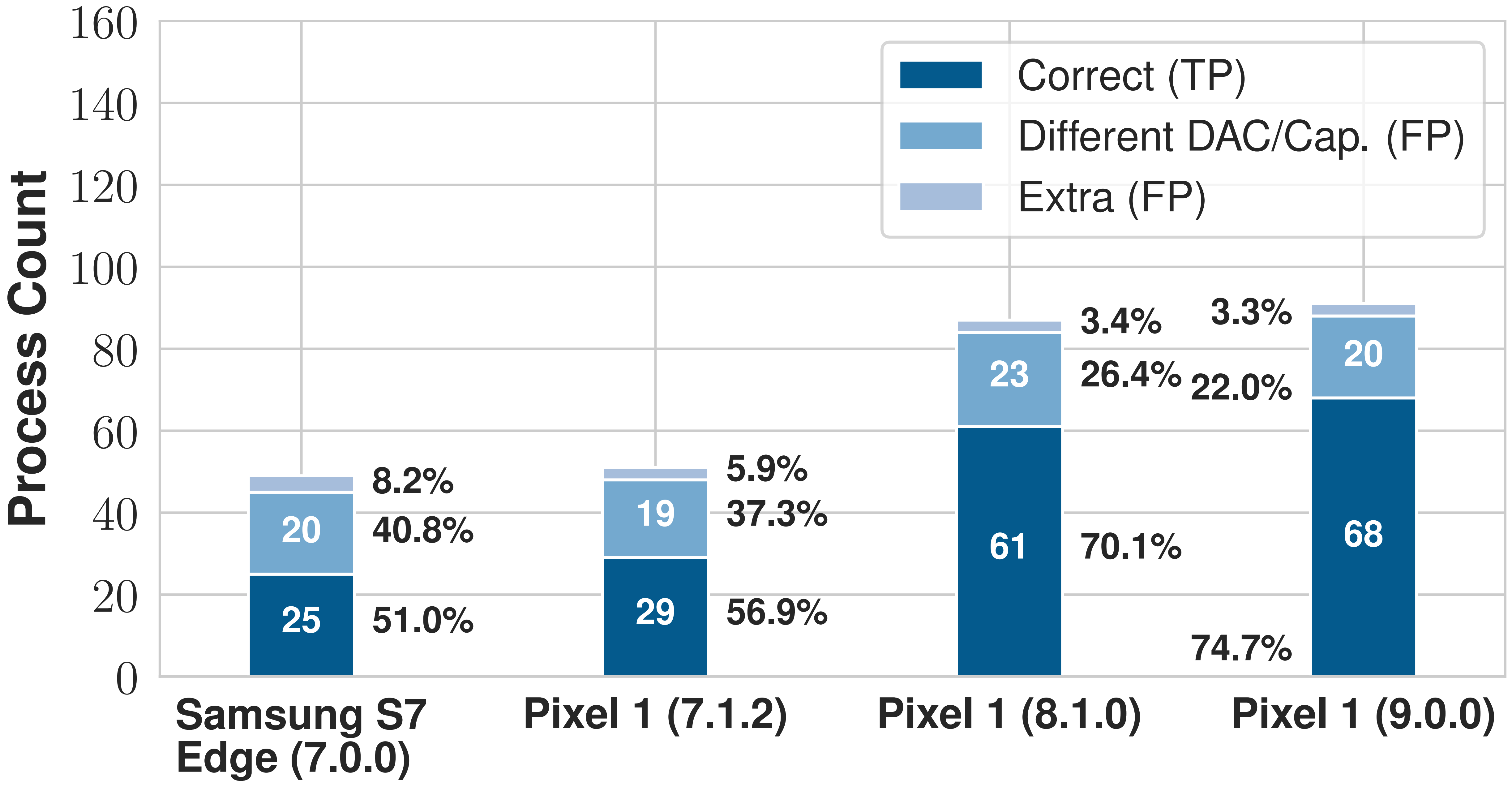
	Samsung S7 Edge (7.0.0)			Pixel 1 (7.1.2)		
	Path	Count	%Files	Path	Count	%Files
Correct Files	*/system*	5,233	93.1%	*/system*	2,301	67.6%
	/data	115	2.0%	*/vendor*	630	18.5%
	/dev	40	0.7%	*/data*	115	3.4%
Different DAC/MAC	*/dev*	46	0.8%	*/dev*	28	0.8%
	/mnt	7	0.1%	*/sbin*	5	0.1%
	/system	5	0.1%	*/mnt*	2	0.1%
Extra Files	*/dev*	73	1.3%	*/dev*	167	4.9%
	/system	6	0.1%	*/cache*	4	0.1%
	/acct	1	0.0%	*/acct*	1	0.0%
	Total:	5,621	100%	**Total:**	3,405	100%
	DAC/MAC Correct:		98.7%	**DAC/MAC Correct:**		98.6%

Ground-truth Evaluation (Files)

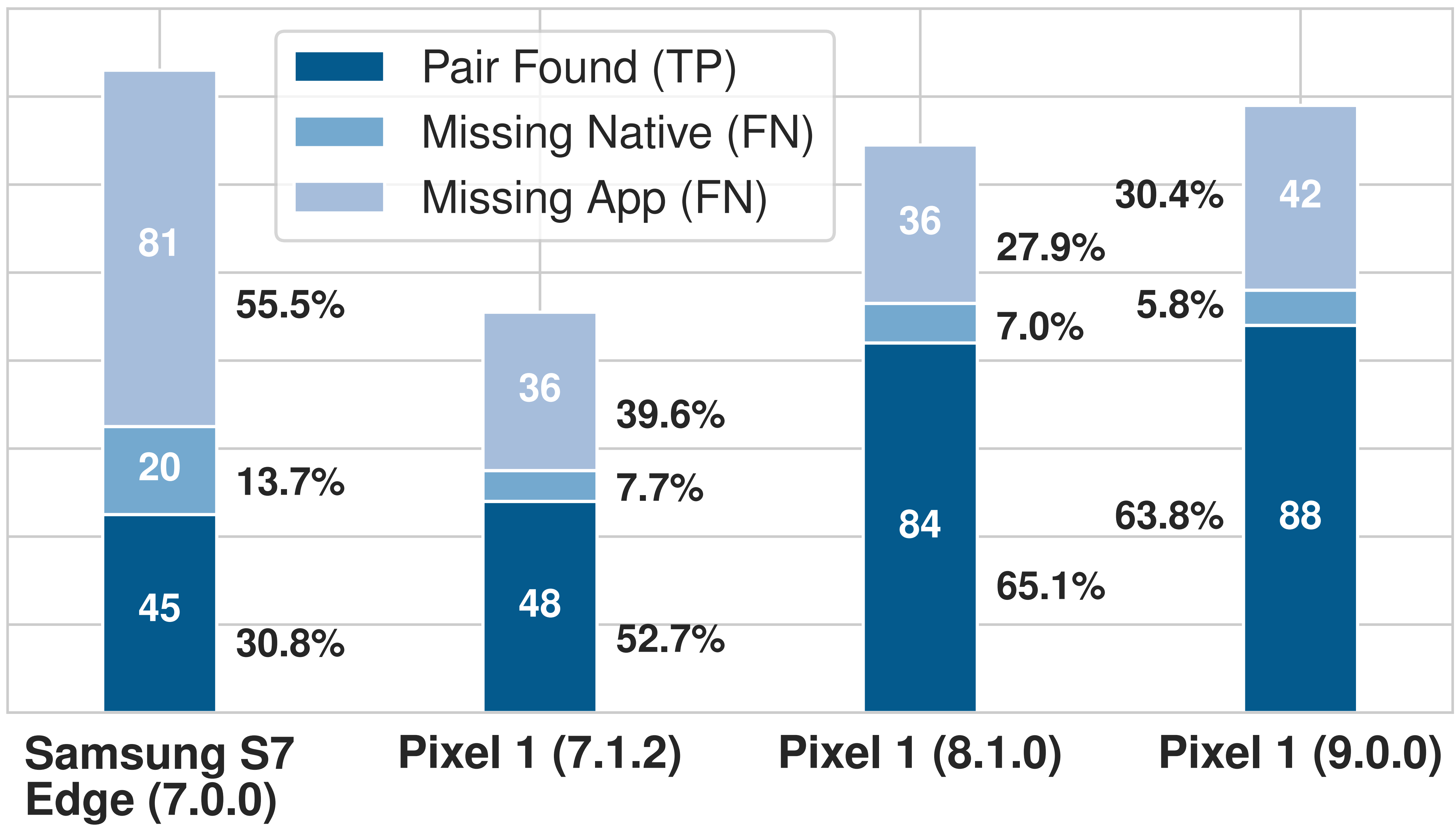
	Samsung S7 Edge (7.0.0)			Pixel 1 (7.1.2)		
	Path	Count	%Files	Path	Count	%Files
Correct Files	*/system*	5,233	93.1%	*/system*	2,301	67.6%
	/data	115	2.0%	*/vendor*	630	18.5%
	/dev	40	0.7%	*/data*	115	3.4%
Different DAC/MAC	*/dev*	46	0.8%	*/dev*	28	0.8%
	/mnt	7	0.1%	*/sbin*	5	0.1%
	/system	5	0.1%	*/mnt*	2	0.1%
Extra Files	*/dev*	73	1.3%	*/dev*	167	4.9%
	/system	6	0.1%	*/cache*	4	0.1%
	/acct	1	0.0%	*/acct*	1	0.0%
	Total:	5,621	100%	**Total:**	3,405	100%
	DAC/MAC Correct:		98.7%	**DAC/MAC Correct:**		98.6%

Our recovered file metadata is 98% accurate to an equivalent running device.

Ground-truth Evaluation (Proc.)

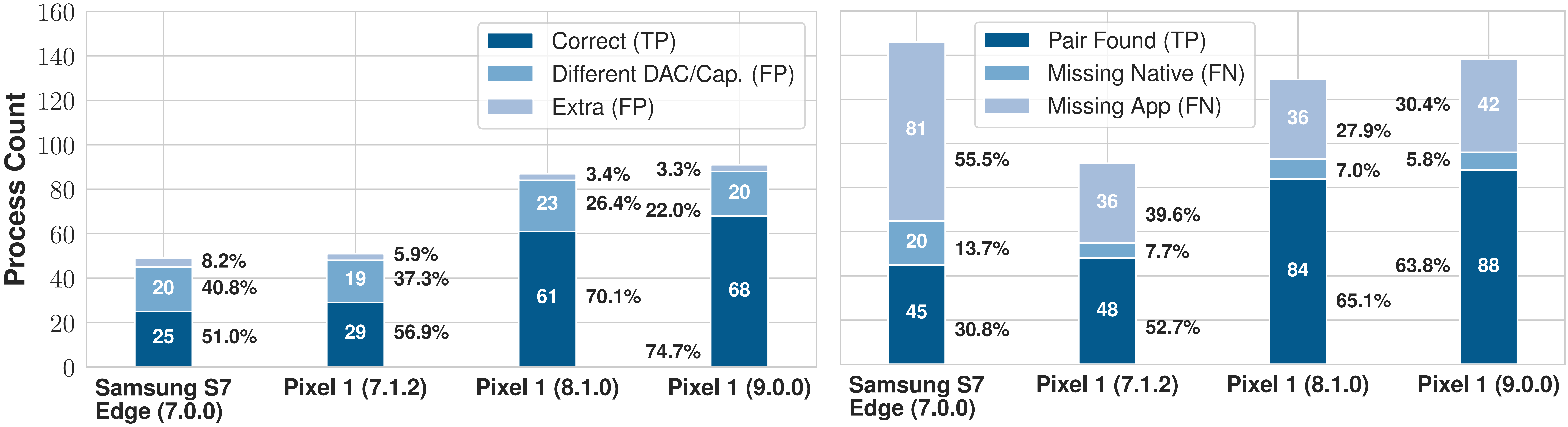


(a) Processes Recovered by BIGMAC



(b) Actual device processes

Ground-truth Evaluation (Proc.)



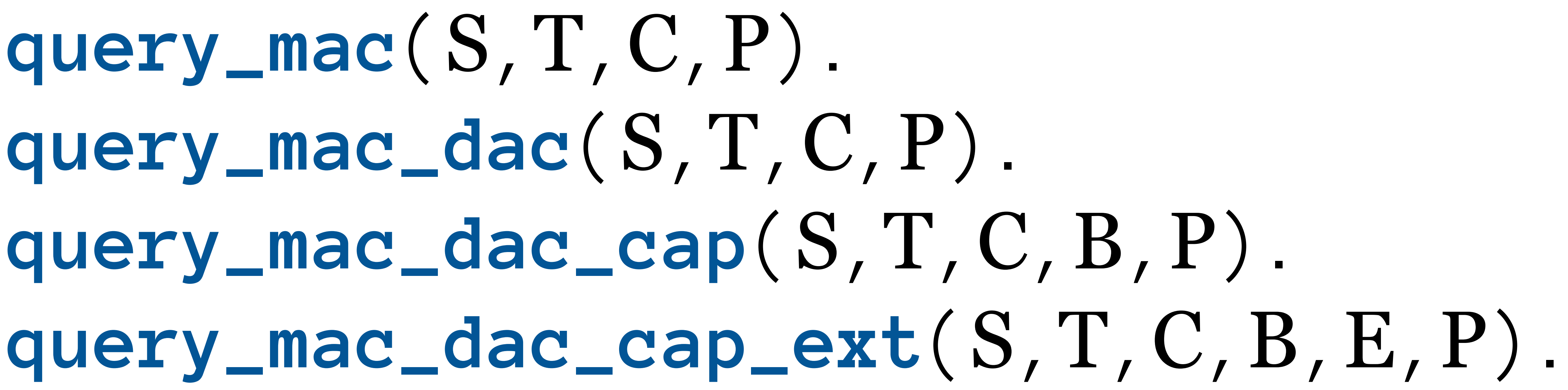
(a) Processes Recovered by BIGMAC

(b) Actual device processes

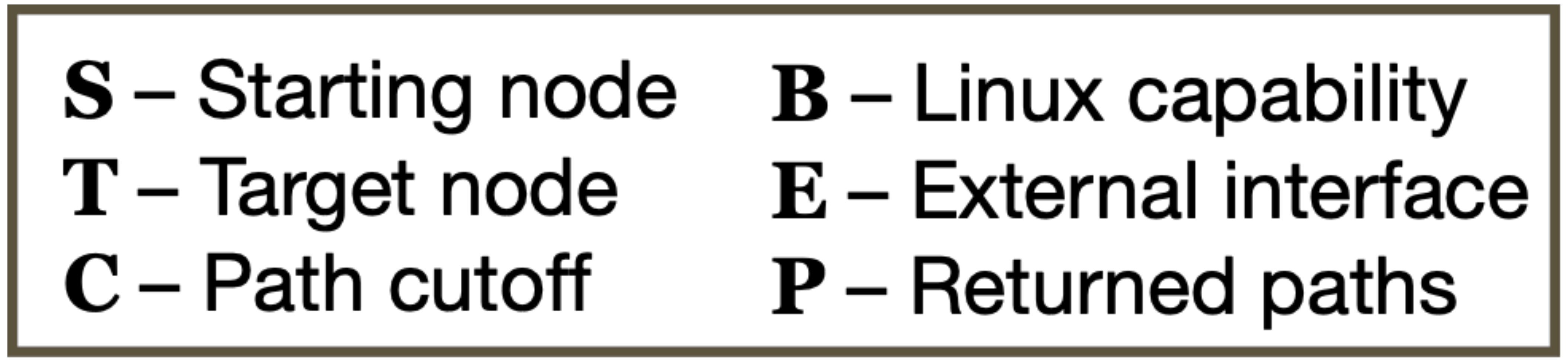
Of the paired processes, we achieve, at best, 74.7% accuracy of process credentials

Prolog Query Interface

We developed a Prolog query engine to find attack-paths with MAC, DAC, CAP, and external attack surface filtering



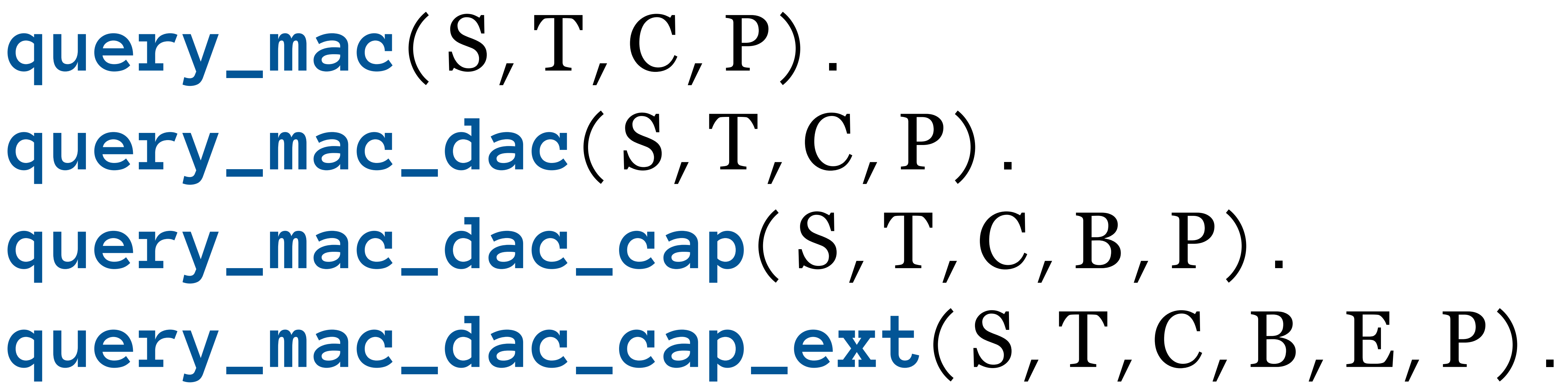
```
query_mac(S,T,C,P).
query_mac_dac(S,T,C,P).
query_mac_dac_cap(S,T,C,B,P).
query_mac_dac_cap_ext(S,T,C,B,E,P).
```



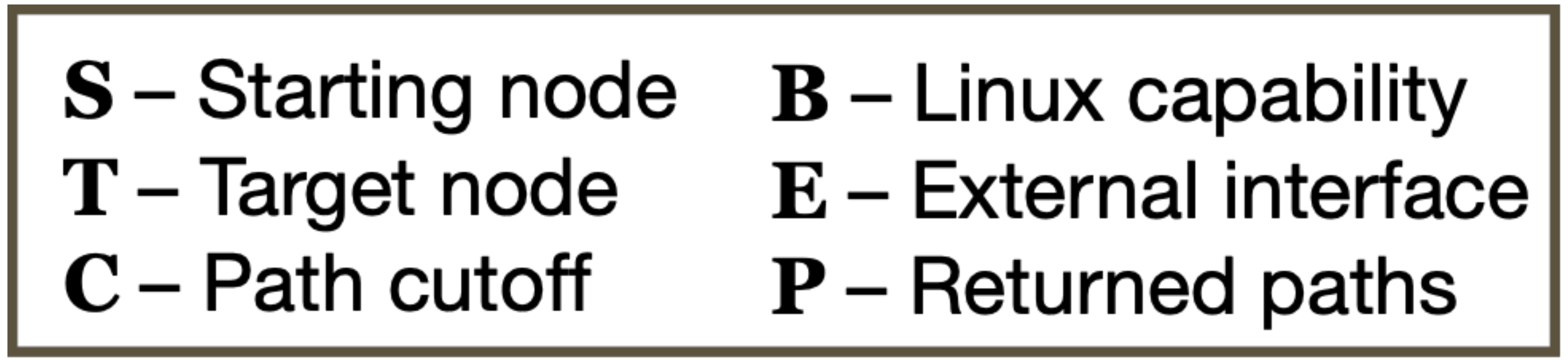
S	– Starting node	**B**	– Linux capability
T	– Target node	**E**	– External interface
C	– Path cutoff	**P**	– Returned paths

Prolog Query Interface

We developed a Prolog query engine to find attack-paths with MAC, DAC, CAP, and external attack surface filtering



```
query_mac(S,T,C,P).
query_mac_dac(S,T,C,P).
query_mac_dac_cap(S,T,C,B,P).
query_mac_dac_cap_ext(S,T,C,B,E,P).
```



S – Starting node
T – Target node
C – Path cutoff
B – Linux capability
E – External interface
P – Returned paths

As a case study, we ran queries against a 1.3 million edge Samsung S8+ and a ~2 million edge LG G7 image

Layered Path Reduction

```
query_mac(untrusted_app,mediaserver,4,P).
query_mac_dac(untrusted_app,mediaserver,4,P).
```

#Paths	Time (s)
102,915	22.48
5,146	518.62

Layered Path Reduction

```
query_mac(untrusted_app,mediaserver,4,P).
query_mac_dac(untrusted_app,mediaserver,4,P).
```

#Paths	Time (s)
102,915	22.48
5,146	518.62

Each additional layer reduces the number of possible paths.

MAC to MAC+DAC has a 20x reduction in the number of paths to be considered.

Process Strength

```
query_mac_dac(init,_,1,P).
query_mac_dac(system_server,_,1,P).
query_mac_dac(lpm,_,1,P).

query_mac_dac(init,_,1,P).
query_mac_dac(system_server,_,1,P).
query_mac_dac(hal_usb,_,1,P).
```

Image	Process	# Writable	# IPC
Samsung S8+	`init`	2,066	296
	`system_server`	1,398	458
	`lpm`	634	8
LG G7	`init`	1,233	418
	`system_server`	573	368
	`hal_usb_default`	508	19

Process Strength

```
query_mac_dac(init,_,1,P).
query_mac_dac(system_server,_,1,P).
query_mac_dac(lpm,_,1,P).

query_mac_dac(init,_,1,P).
query_mac_dac(system_server,_,1,P).
query_mac_dac(hal_usb,_,1,P).
```

Image	Process	# Writable	# IPC
Samsung S8+	`init`	2,066	296
	`system_server`	1,398	458
	`lpm`	634	8
LG G7	`init`	1,233	418
	`system_server`	573	368
	`hal_usb_default`	508	19

Some of the most powerful processes (`system_server`) on Android deal with some of the most untrusted data.

Process Strength

```
query_mac_dac(init,_,1,P).
query_mac_dac(system_server,_,1,P).
query_mac_dac(lpm,_,1,P).

query_mac_dac(init,_,1,P).
query_mac_dac(system_server,_,1,P).
query_mac_dac(hal_usb,_,1,P).
```

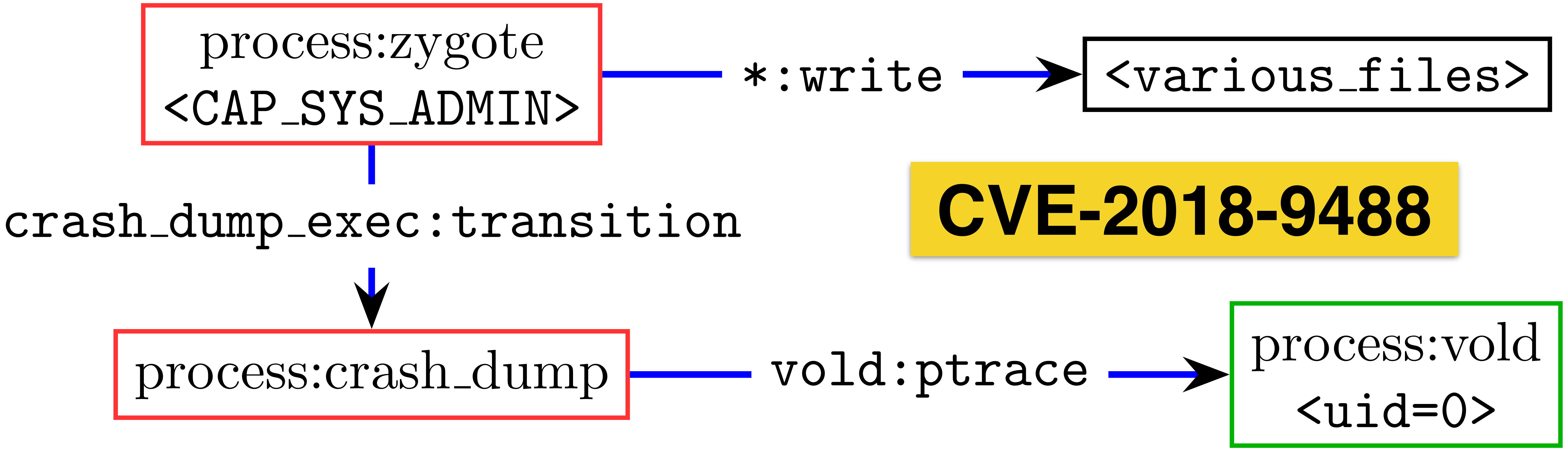
Image	Process	# Writable	# IPC
Samsung S8+	`init`	2,066	296
	`system_server`	1,398	458
	`lpm`	634	8
LG G7	`init`	1,233	418
	`system_server`	573	368
	`hal_usb_default`	508	19

Some of the most powerful processes (`system_server`) on Android deal with some of the most untrusted data.

`system_server` **should be refactored into smaller, less privileged processes, similar to mediaserver**

Privilege Escalation Analysis

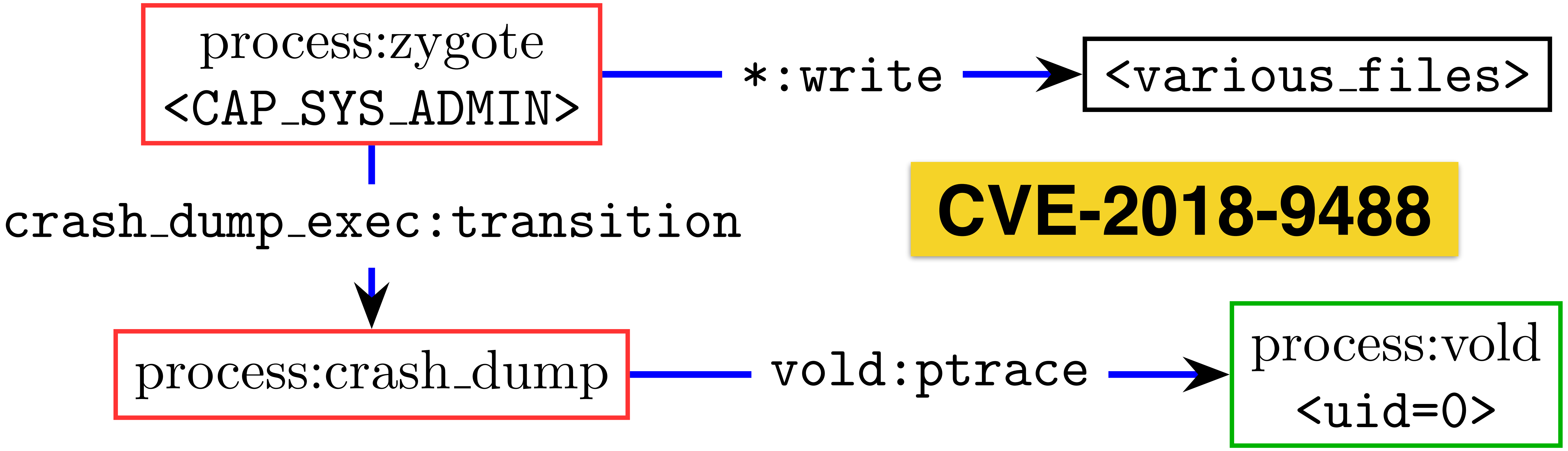
```
#1 query_mac_dac(zygote,vold,3,P).
```



CVE-2018-9488

Privilege Escalation Analysis

```
#1 query_mac_dac(zygote,vold,3,P).
```



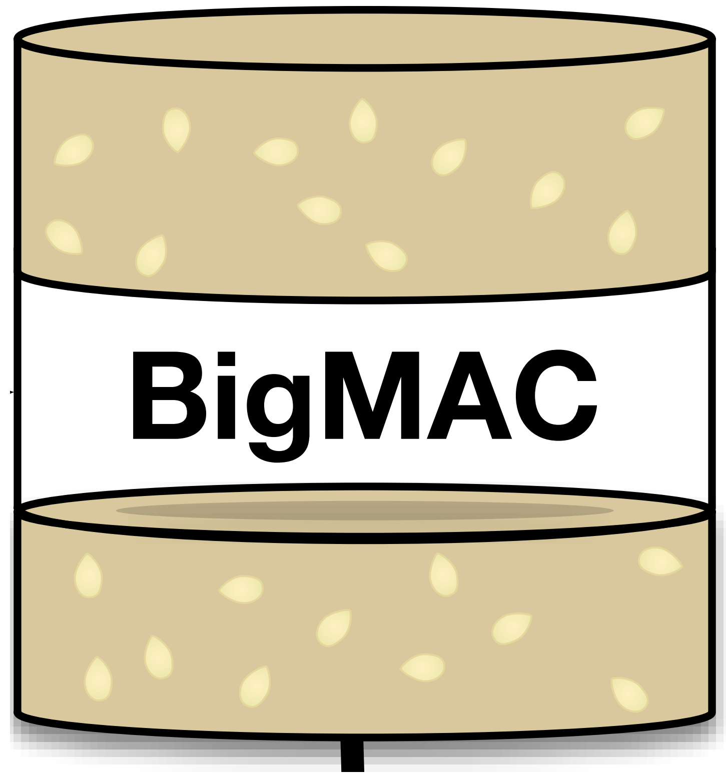
CVE-2018-9488

```
#2 query_mac_dac_cap(_,crash_dump,1,CAP_SYS_ADMIN,P).
```

22 additional processes beyond zygote could escalate

Conclusion

- We create **BigMAC**, one of the most fine-grained policy analysis frameworks for Android devices, and recover a running system's security state from static firmware
- **BigMAC** surpasses previous MAC-only policy analysis approaches through its layered path reduction, improving analysis results and discarding impossible runtime paths
- We highlight **BigMACs** ability to investigate escalation paths and examine the strength of processes



https://github.com/fics/BigMAC

https://hernan.de/z

@Digital_Cold