Deja Vu All Over Again: Access Control in Super Apps

Trent Jaeger, UC Riverside

October 14, 2024

ACM Workshop on Secure and Trustworthy Superapps (SaTS)

Super Apps

Emerging approach for deploying mobile system functionality

- “Everything app” that can “provide multiple services” to be an “all-encompassing self-contained commerce and communication platform”

Mini Apps

Key Feature of Super Apps Is That They Are a Platform for Third-Party Mini Apps

- “Mini apps are small (commonly 2-4 MB) apps that require a super app to run.” “The runtime of a mini app is a WebView in the super app, not the underlying operating system, which makes mini apps cross platform.”

PeppyOcean
Taxi Booking
Courier Delivery
Social Media
Food Delivery
Grocery Delivery
SUPER APP DEVELOPMENT
Bill Payment
Medicine Delivery
Call a Doctor
Shopping
Travel

Super App Vulnerabilities

But, there are growing pains in converting a popular app to a platform [1]

- ❑ **Unauthorized access**: Android requires applications to be granted a location permission (e.g., ACCESS FINE LOCATION) to access the Wi-Fi List via wx.getWifiList, but WeChat did not require this permission [2]
- ❑ **Risks to user privacy**: Mini apps may exploit ”hidden” APIs to access sensitive user data and sensors that can be used to spy on users. [3,4]
- ❑ **Mini app attacks on other mini apps**: Cross-Miniapp Request Forgery (CMRF), where a malicious mini app can send arbitrary payloads via cross-mini app channel. [5]

Access Control

These are all problems of insufficient access control enforcement

- ❑ Including…
- ❑ Ensure that we perform **access checks comprehensively** across all operations that may access sensitive data
- ❑ Figure out how to **protect user privacy** against a diverse set of attacks
- ❑ Determine whether access control enforces the **expected security guarantees** when running mini apps

Not Our “First Rodeo”

Access Control is a Fundamental Security Mechanism

- ❑ Goes back at least as far as the **Multics** system (1960s)
- ❑ **Reference monitor concept** describes how to build authorization systems (1972)
- ❑ **Mandatory access control** (Bell-La Padula and Biba) for strict enforcement (1976)
- ❑ Flurry of **access control improvements** in the 1990s and 2000s to combat malicious worm events
- ❑ **Android access control** (2000s and 2010s and 2020s)

Not My “First Rodeo” Either

Access Control Research Topics over 30 years

- ❑ Access control over mobile code (**Java applets** and **Tcl scripts**)
- ❑ **Microkernel** access control enforcement for L4
- ❑ **Linux Security Modules** for mandatory access control in Linux
- ❑ **Xen Security Modules** for mandatory access control for hypervisors
- ❑ Invent **access control policy analysis** to identify threats from weak policies
- ❑ Cloud and virtual machine system access control (**OpenStack**)
- ❑ **Android access control** over sensors and file systems

Do We Know How To Integrate Effective Access Control into Legacy Programs?

Shouldn’t You Have Solved This by Now?

- ❑ Not really

Do We Know How To Integrate Effective Access Control into Legacy Programs?

Shouldn’t You Have Solved This by Now?

- ❑ Not really
- ❑ **In this talk**: Discuss challenges and experiences in other contexts
- ❑ **And**: Impact on building authorization systems for super apps

Reference Monitor Concept

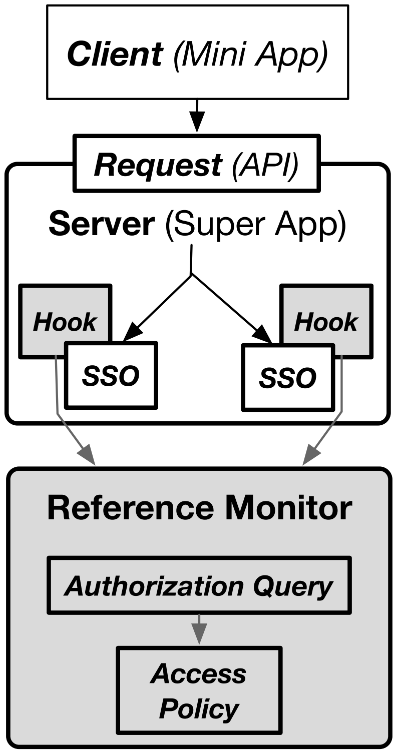
Need to Solve Three Major Problems

- **Complete mediation**: The reference validation must always be invoked.
  - **For all security-sensitive operations**
- **Tamperproof**: The reference validation mechanism must use trusted information to make access decisions.
  - **Access control must be enforced in a trusted way by reference monitor**
- **Verifiable**: The reference validation mechanism must be simple enough to be subject to analysis and tests, the completeness of which can be assured.
  - **Can we test the mechanism and the policy?**

Reference Monitor Architecture

Consist of Three Main Components

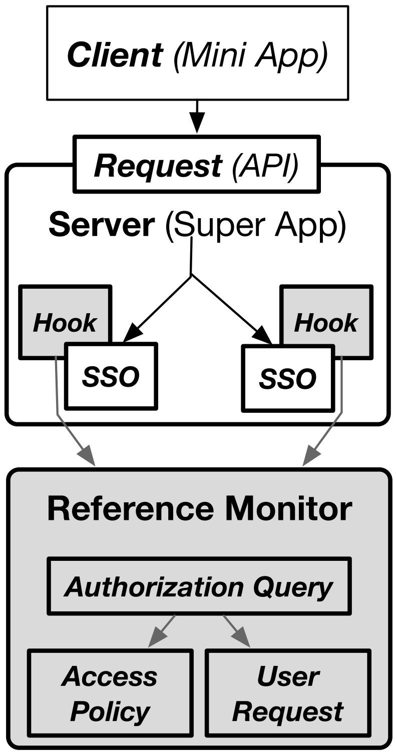
- **Authorization Hooks**: Calls to a reference monitor are placed in the program, which aim for complete mediation of all security-sensitive operations
- **Authorization Query Processing**: Ask a trusted authority (relative to the access request being authorized) to make an access control decision
- **Access Control Policy**: The policy provided by the trusted authority, which determines the security guarantees enforced by the Authorization System.



Adding a Reference Monitor to a Legacy Program is Hard

Need to Solve Three Major Problems

- Determine when to perform an access control check
  - Place **authorization hooks** that invoke a **reference monitor** **(complete mediation)**
- Find the right authority for each access control decision for each authorization query (**tamperproof**)
  - **Program**, **User**, **System**, **Cloud** may be responsible
- Determine whether the access control enforcement achieves the **intended security goals** (**verifiable**)



Challenges in Authorization Hook Placement

Misplaced Authorization Hooks Lead to Vulnerabilities

- ❑ **Missing**: A security-sensitive operation lacks any authorization hook
- ❑ **Inconsistent**: The same security-sensitive operation may be run under different sets of permissions
- ❑ **Overpermission**: A hook may grant access to multiple security-sensitive operations, be we may only want some subjects to perform a subset
- ❑ **TOCTTOU**: The target (object) of an operation may change between the access check (at the hook placement) and the time of the operation

Challenges in Authorization Hook Placement

Ideally, we would know all security-sensitive operations and place hooks to mediate their execution

- Programmers do not identify security-sensitive operations
  - May lead to **missing hooks**
- To keep policies as simple as possible, programmers aim minimize the number of authorization hooks (i.e., each one needs a policy)
  - May lead to **inconsistent checks** or **overpermissioning**
- The program may allow the relationships among objects to change concurrently to a security-sensitive operation
  - May lead to **TOCTTOU**

TOCTTOU Vulnerability in Linux [6]

```
/* from fs/fcntl.c */
long sys_fcntl(unsigned int fd, ...
   struct file * filp;
   ...
   filp = fget(fd);
   ...
   err = security ops->file ops ->fcntl(filp, cmd, arg);  // hook
   ...
   err = do fcntl(fd, cmd, arg, ...);  // leads to SSO
        // leads to extraction of filp from fd again
        // but the file associated with a fd can be changed concurrently
                   // TOCTTOU!!!
```

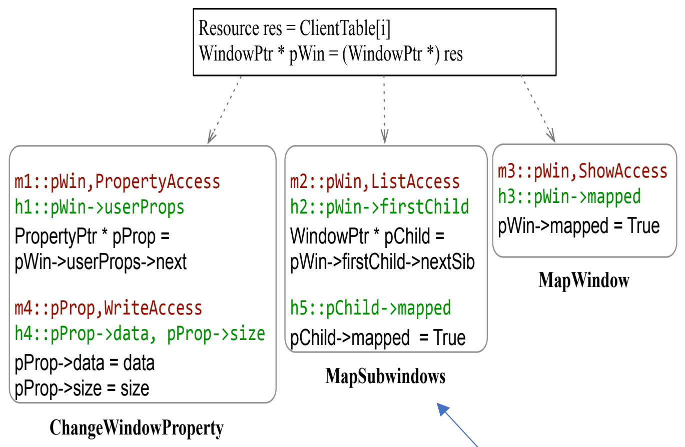
Overpermissioning? [7]

```
if (rc == BadMatch)
   {/* Op 1*/
       pProp->name = property;
       pProp->format = format;
       pProp->data = data;
       pProp->size = len;
   }
else
   {/* Op2*/
       if (mode == REPLACE)
           { /*Op2.1*/
               pProp->data = data;
               pProp->size = len;
               pProp->format = format;
           }
       else if (mode == APPEND)
           {/* Op 2.2 */
               pProp->data = data;
               pProp->size += len;
           }
   }
```

Request determines which branch is taken

Each branch has different field accesses

Impact of Policy on Hook Placement [8]



Use **constraints on policies** (e.g., all fields of an object are accessible if one field is accessible) to **generate a minimal* hook placement automatically**

*** Minimal relative to constraints**

Not everyone who can modify pWin can access a particular pProp – **two hooks**

Everyone who can modify pWin can access the pChild – **one hook**

Lessons in Authorization Hook Placement

- Goal is to **minimize the number of authorization hooks** necessary to enforce the intended access control policies
  - The specific access control policies are not known in advance
  - But, the hook placement defines the “operations” to be authorized
- **Identifying security-sensitive operations is hard**
  - Lots of program objects and accesses
  - Even just using the control-flow paths dependent on client requests is more fine-grained than actual hook placements [7]
- Can **produce authorization hook placements automatically**
  - But, need more insights to elide unnecessary hooks for policy [8]
  - And need to account systematically for TOCTTOU problems

Finding the Right Authority for Controlling Access

As more types of resources are managed, this complicates finding the authority

- **Resources**: Originally, access control was applied primarily to files, but now covers many specialized system objects (Android intents and sensors) and application objects (in databases, webservers, browsers).
  - **Even more complex user data is managed by super apps**
  - **As well as cloud-stored data**
- **Authority**: Originally, operating systems made all access control decisions, but Android delegates many access control decisions to users
  - **Users have to make even finer access decisions for super apps**
  - **Super apps have to make many decisions for themselves and the OS**

User-Driven Access Control Challenges

Mobile systems normalized user-driven access control [9]

- ❑ **Install Time**: Users authorize permissions for apps up front
- ❑ **First Use**: Users authorize permissions on first use

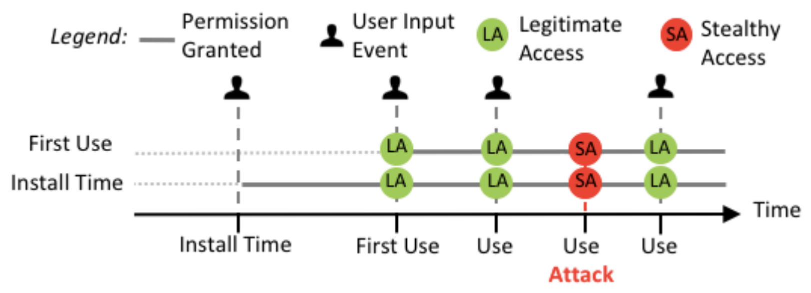


Figure 1: In mobile platforms, once the system authorizes an application to perform a operation, the application may perform that operation at any time, enabling adversaries to stealthily access privacy-sensitive sensors, e.g., record speech using the microphone, at any time.

User-Driven Access Control Challenges

User interface attacks [10]

- Four types of attacks against UDAC

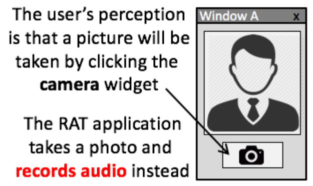

The user's perception is that a picture will be taken by clicking the **camera** widget
The RAT application takes a photo and **records audio** instead
Window A

Figure 2: The user's perception of the operation that is going to be performed differs from the actual operation requested by the application, which abuses a previous granted permission.

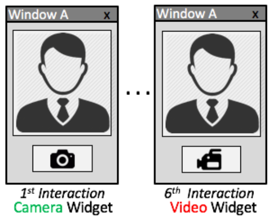


Figure 4: A malicious application keeps the windowing display context but switches the widget to trick users who have made several similar selections to grant the malicious application also access to the microphone mistakenly.

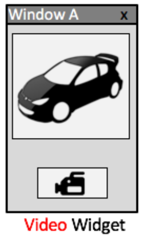


Figure 3: A photo capturing application presents a video camera widget, instead of a camera widget, to trick the user into also granting access to the microphone. The windowing display context surrounding the widget shows a camera preview for photo capturing.

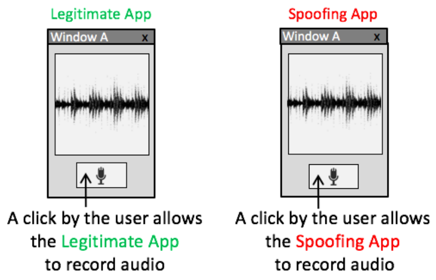
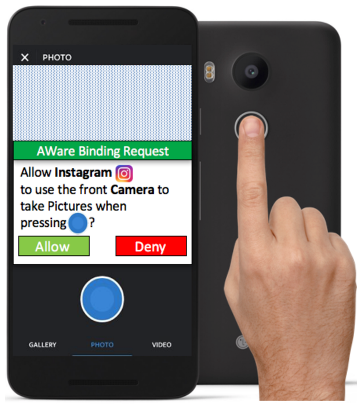

Legitimate App
Window A
A click by the user allows the Legitimate App to record audio
Spoofing App
Window A
A click by the user allows the Spoofing App to record audio

Figure 5: The user may mistakenly authorize access to the microphone to a RAT application spoofing the graphical aspect of a well-known legitimate application.

Restricting Permission Use in User-Driven Access Control

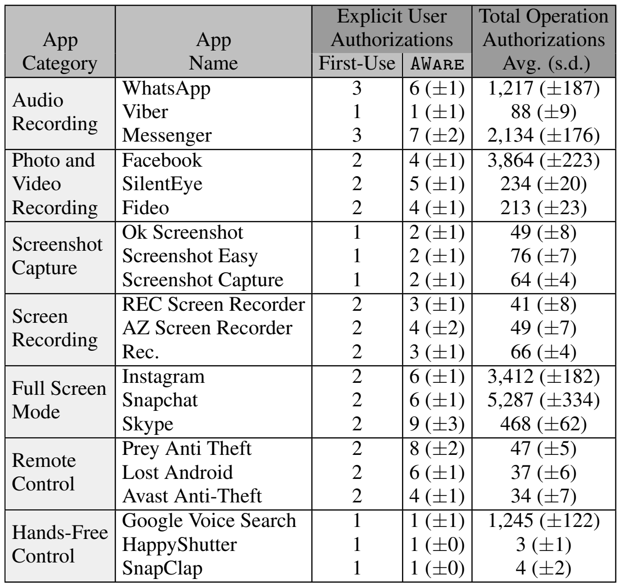
AWare System: Associating permissions with limited contexts does not result in too many access control decisions for users [10]

- ❑ Bind the GUI state upon user input with the permissions to the sensors



Allow **Instagram** to use the front **Camera** to take Pictures when pressing ?

Figure 8: AWARE *Binding Request* prompted to the user on the mobile platform's screen at *Operation Binding* creation. The app's identity is proved by the name and the graphical mark. For better security, in mobile platforms equipped with a fingerprint scanner, AWARE recognizes the device owner's fingerprint as the only authorized input for creating a new *Operation Binding*.



App Category	App Name	Explicit User Authorizations First-Use	Explicit User Authorizations AWARE	Total Operation Authorizations Avg. (s.d.)
Audio Recording	WhatsApp	3	6 (±1)	1,217 (±187)
	Viber	1	1 (±1)	88 (±9)
	Messenger	3	7 (±2)	2,134 (±176)
Photo and Video Recording	Facebook	2	4 (±1)	3,864 (±223)
	SilentEye	2	5 (±1)	234 (±20)
	Fideo	2	4 (±1)	213 (±23)
Screenshot Capture	Ok Screenshot	1	2 (±1)	49 (±8)
	Screenshot Easy	1	2 (±1)	76 (±7)
	Screenshot Capture	1	2 (±1)	64 (±4)
Screen Recording	REC Screen Recorder	2	3 (±1)	41 (±8)
	AZ Screen Recorder	2	4 (±2)	49 (±7)
	Rec.	2	3 (±1)	66 (±4)
Full Screen Mode	Instagram	2	6 (±1)	3,412 (±182)
	Snapchat	2	6 (±1)	5,287 (±334)
	Skype	2	9 (±3)	468 (±62)
Remote Control	Prey Anti Theft	2	8 (±2)	47 (±5)
	Lost Android	2	6 (±1)	37 (±6)
	Avast Anti-Theft	2	4 (±1)	34 (±7)
Hands-Free Control	Google Voice Search	1	1 (±1)	1,245 (±122)
	HappyShutter	1	1 (±0)	3 (±1)
	SnapClap	1	1 (±0)	4 (±2)

Lessons in User-Driven Access Control

- Figuring out **which authority should govern which permissions** is done in an ad hoc manner at present
  - But, this is a critically important decision
- Goal is to **minimize the number of authorization queries/decisions** necessary for users to enforce the intended access control policies
  - But, permissions may be abused by untrusted apps
- **We can restrict the context in which permissions may be used**
  - But, this makes the system more complex
  - Super apps will have much more complex decisions over user privacy

The Future – Impact on Super App Access Control

Super App Access Control

Is at least as complex as Android OEM systems – probably a lot more complex

- ❑ Same threat model as Android – untrusted apps
- ❑ More complex challenges for **authorization hook placement** due to need to control access to: user data, platform resources, cloud resources, etc.
- ❑ **User authority for access control decisions** in super apps could be much more common, as super apps manage a wider variety of user data
- ❑ **Validating intended security guarantees** among users, mini apps, and super apps will be more complex unless very simple policies are employed. Will there be super app OEMs?

Super App Access Control

Based on lessons from deploying access control in the past

- ❑ We need to **determine the security goals** we really want to enforce
- ❑ We should consider how much **authority we will place on users** to achieve those security goals
- ❑ Only then can be start to **build automated tools to vet super app** access control vulnerabilities proactively
- ❑ And even then, producing authorization systems that enforce those goals correctly and effectively – possibly even **generating reference monitors automatically!**

Conclusions

Experiences in access control and impact on super app systems

- Integrating access control into a legacy app of any kind correctly is difficult
- We do have a fair amount of experience to build upon, but manual approaches to deploying authorization systems lead to vulnerabilities
- And super app systems appear to be more challenging, particularly with respect to managing access to user data
- We have argued that making the intended security guarantees for access control explicit can guide authorization hook placement and policy analysis
  - But more research needs to be done to automate sufficiently

Questions

- [1] Y. Yang, C. Wang, Y. Zhang, Z. Lin. SoK: Decoding the Super App Enigma: The Security Mechanisms, Threats, and Trade-offs in OS-alike Apps. arXiv, CoRR abs/2306.07495, June 2023.
- [2] H. Lu, L. Xing, Y. Xiao, Y. Zhang, X. Liao, X. Wang, and X. Wang. Demystifying resource management risks in emerging mobile app-in-app ecosystems. In Proceedings of the 2020 ACM Conference on Computer and Communications Security, 2020.
- [3] W. Chao, Y. Zhang, and Z. Lin. One size does not fit all: Uncovering and exploiting cross platform discrepant APIs in WeChat. In Proceedings of the 31st USENIX Security Symposium, 2023.
- [4] W. Chao, Y. Zhang, and Z. Lin. Uncovering and exploiting hidden APIs in mobile super apps. In Proceedings of the 2023 ACM Conference on Computer and Communications Security, 2023.
- [5] Y. Yang, Y. Zhang, and Z. Lin. Cross miniapp request forgery: Root causes, attacks, and vulnerability detection. In Proceedings of the 2022 ACM Conference on Computer and Communications Security, 2022.
- [6] X. Zhang, A. Edwards, T. Jaeger. Using CQUAL for Static Analysis of Authorization Hook Placement. In Proceedings of the 11th USENIX Security Symposium, August 2002.

Questions

- [7] D. Muthukumaran, T. Jaeger, V. Ganapathy. Leveraging "Choice" to Automate Authorization Hook Placement. In Proceedings of the 19th ACM Conference on Computer and Communications Security, October 2012.
- [8] D. Muthukumaran, N. Talele, T. Jaeger, G. Tan. Producing Hook Placements to Enforce Expected Access Control Policies. In Proceedings of the 2015 International Symposium on Engineering Secure Software and Systems (ESSoS), March 2015.
- [9] F. Roesner, et. al. User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems. In Proceedings of the 2012 IEEE Symposium on Security and Privacy, 2012.
- [10] G. Petracca, A-A. Reineh, Y. Sun, J. Grossklags, T. Jaeger. AWare: Preventing Abuse of Privacy-Sensitive Sensors via Operation Bindings. In Proceedings of the 26th USENIX Security Symposium, August 2017.
- [11] Y-T. Lee, et al. PolyScope: Multi-Policy Access Control Analysis to Compute Authorized Attack Operations in Android Systems. In Proceedings of the 30th USENIX Security Symposium, August 2021.
- [12] Y-T. Lee, et al. PolyScope: Multi-Policy Access Control Analysis to Triage Android Scoped Storage. IEEE Transactions on Dependable and Secure Computing, Early Access, 2023.