# An App Centric Approach for Vulnerability Discovery in Mobile Platforms

**Dr. Zhiqiang Lin**
zlin@cse.ohio-state.edu
Joint work w/ Omar Alrawi, Mohamed Elsabagh, Ruian Duan, Ryan Johnson, Ranjita Kasturi, Brendan Saltaformaggio, Angelos Starvro, Yinqian Zhang, Qingchuan Zhao, Chaoshun Zuo

04/21/2020

# Outline


1. Introduction
2. FirmScope: Privilege-Escalation in Firmware
3. LeakScope: Leakage in Clouds
4. SkyWalker: n-day/0-day in Servers
5. Summary

# Outline


1. Introduction
2. FirmScope: Privilege-Escalation in Firmware
3. LeakScope: Leakage in Clouds
4. SkyWalker: n-day/0-day in Servers
5. Summary

# Mobile Apps Have Significantly Changed Our Lives

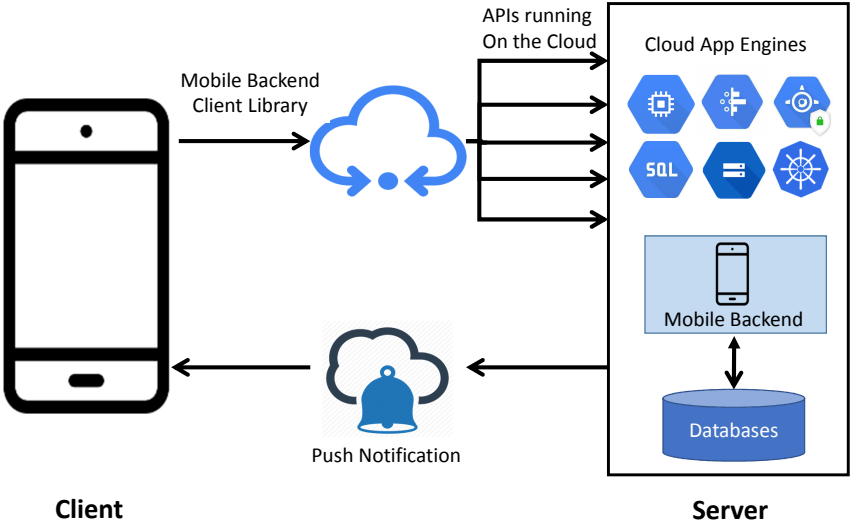
Source: cloudxtension.com

1. Weather, News
2. Maps, Search
3. Social, Communication
4. Games, Sports
5. Music, Media, Entertainment
6. Shopping, Retail
7. Health, Fitness
8. Food, Drink
9. ...

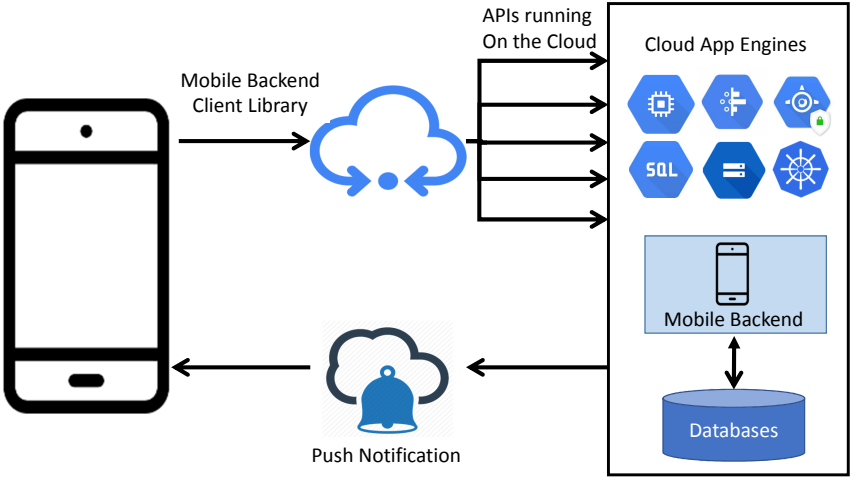
# Mobile Ecosystems: Apps, Web, and Clouds

# Mobile Ecosystems: Apps, Web, and Clouds

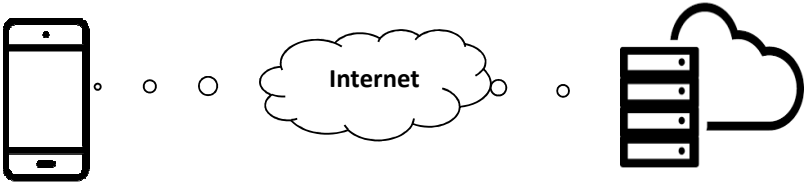
# The Mobile Backend as a Service (mBaaS) Clouds



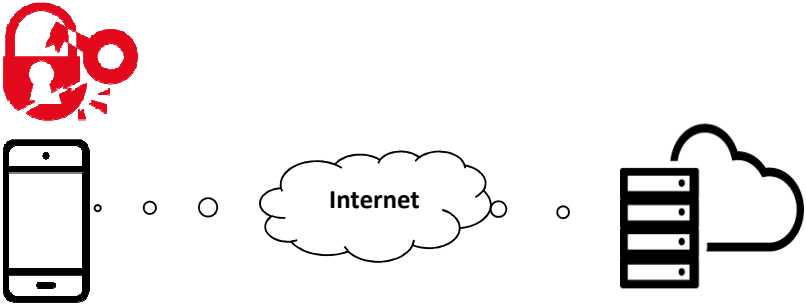
# The Mobile Backend as a Service (mBaaS) Clouds



# Our Focal Point I: Uncovering Vulnerabilities in the Phones

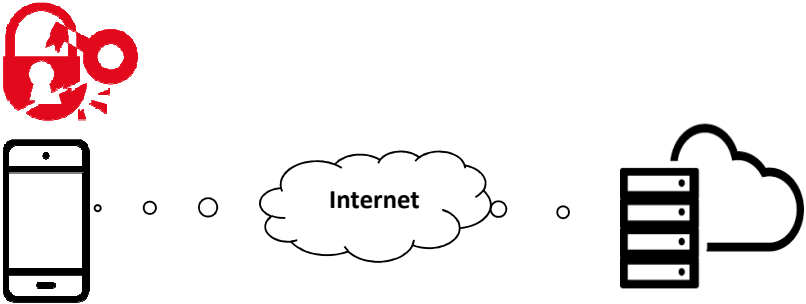


# Our Focal Point I: Uncovering Vulnerabilities in the Phones



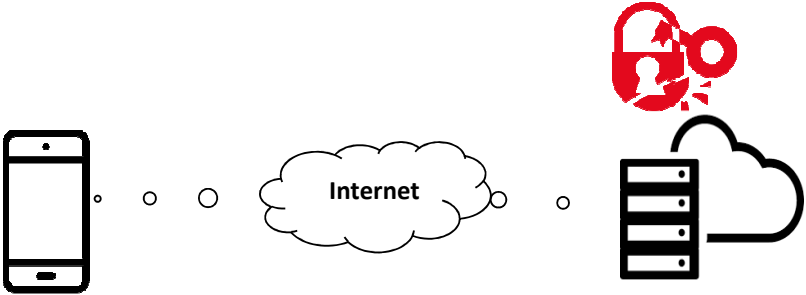

**Challenges**: Tons of existing work; we have to look for new problems.

# Our Focal Point I: Uncovering Vulnerabilities in the Phones

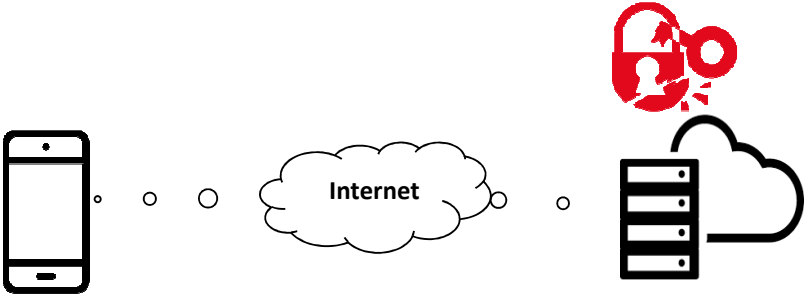



**Our Target**: focusing on **preinstalled apps**, since there is little work on this.

# Our Focal Point II: Uncovering Vulnerabilities in the Clouds

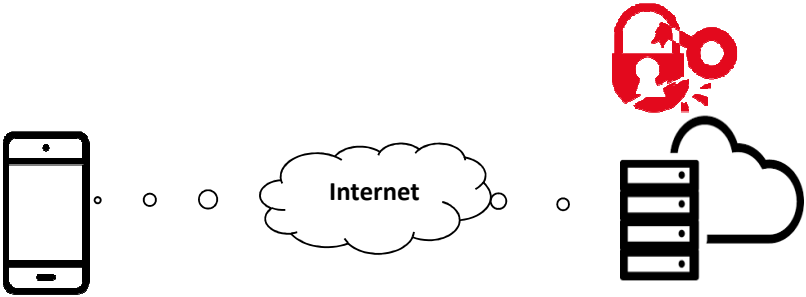


# Our Focal Point II: Uncovering Vulnerabilities in the Clouds




**Challenges**: No (source/binary) code of servers; Blackbox

# Our Focal Point II: Uncovering Vulnerabilities in the Clouds




**Challenges**: No (source/binary) code of servers; Blackbox

**Key Approach**: **Identifying Vulnerabilities in Remote Cloud Servers (i.e., Online Services) via Automated Analysis of Mobile Apps**

# Vulnerability I: Privilege Escalation in Preinstalled Apps

## The Pre-installed apps

- Framework apps
- Vendor apps
- Carrier apps

## Security Issue

- Preinstalled apps have the highest Privilege
- What if they are vulnerable?


Avast Threat Labs analyzed malware that has affected thousands of users around the world

# Vulnerability II: Cloud Data Leakage

CNN BUSINESS Markets Tech Media Success Perspectives Video

## Verizon data of 6 million users leaked online

by Selena Larson @selenalarson

July 12, 2017: 4:14 PM ET

Recommend 2

# Vulnerability II: Cloud Data Leakage

CNN BUSINESS Markets Tech Media Success Perspectives Video

## Verizon data of 6 million users leaked online

by Selena Larson @selenalarson

July 12, 2017: 4:14 PM ET

Recommend 2

CNN BUSINESS Markets Tech Media Success Perspectives Video

Cyber-Safe

## Pentagon exposed some of its data on Amazon server

by Selena Larson @selenalarson

November 17, 2017: 12:03 PM ET

Recommend 9

# Vulnerability II: Cloud Data Leakage

CNN BUSINESS Markets Tech Media Success Perspectives Video

Verizon data of C

by Sel

Biggest Cloud Security Breaches in 2018

admin | October 4th, 2018 | Cloud Security

Perspectives Video

Cyber-Safe

Pentagon exposed some of its data on Amazon server

by Selena Larson @selenalarson

November 17, 2017: 12:03 PM ET

Recommend 9

# Vulnerability II: Cloud Data Leakage

CNN BUSINESS Markets Tech Media Success Perspectives Video

Application Security, DDoS Mitigation , MAR 29 2018

**Cloud Security Concerns in 2018: Data Breaches, Security Misconfigurations, AI and Botnets**

Arun Balakrishnan

Amazon server

by Selena Larson @selenalarson

November 17, 2017: 12:03 PM ET

Recommend 9

# Vulnerability III: Traditional Web Vulnerabilities and Beyond

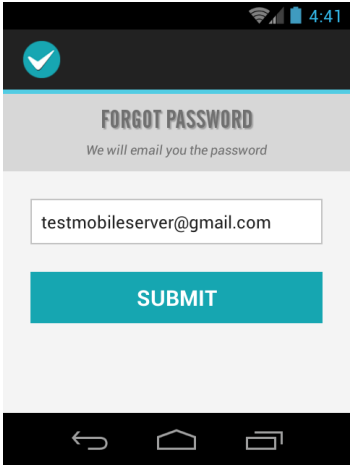
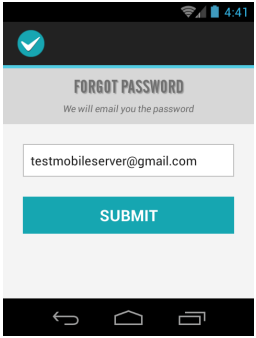



Figure: The password reset activity of ShopClues (between 10 million and 50 million installs).

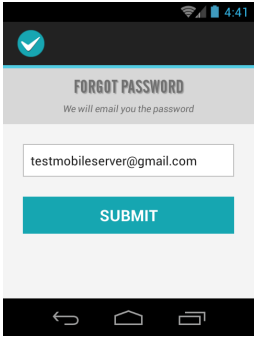
# Vulnerability III: Traditional Web Vulnerabilities and Beyond




```
PUT /api/v9/forgotpassword?key=d12121c70dda5edfgd1df6633fdb36c0
HTTP/1.1
Content-Type: application/json
Connection: close
User-Agent: Dalvik/1.6.0 (Linux; Android 4.2)
Host: sm.shopclues.com
Accept-Encoding: gzip
Content-Length: 73

{"user_email":"testmobileserver@gmail.com","key":"d12121c70dda5e
dfgd1df6633fdb36c0"}
```

# Vulnerability III: Traditional Web Vulnerabilities and Beyond




```
PUT /api/v9/forgotpassword?key=d12121c70dda5edfgd1df6633fdb36c0
HTTP/1.1
Content-Type: application/json
Connection: close
User-Agent: Dalvik/1.6.0 (Linux; Android 4.2)
Host: sm.shopclues.com
Accept-Encoding: gzip
Content-Length: 73

{"user_email":"testmobileserver@gmail.com","key":"d12121c70dda5e
dfgd1df6633fdb36c0"}
```

There was an SQL injection vulnerability at this password reset interface

# Our Efforts on **Vulnerability Discovery** w/ Mobile App Centric Analysis

1. **SMV-Hunter: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps**. In NDSS 2014
2. **AutoForge: Automatic Forgery of Cryptographically Consistent Messages to Identify Security Vulnerabilities in Mobile Services**. In NDSS 2016
3. **SmartGen: Exposing Server URLs of Mobile Apps With Selective Symbolic Execution**. In WWW 2017
4. **AuthScope: Towards Automatic Identification of Vulnerable Authorizations in Online Services**. In CCS 2017
5. **The Unexpected Danger of UX Features: A Case of Sensitive Data Leakage of Drivers in Ride-Hailing Services**. In NDSS 2019.
6. **LeakScope: Why Does Your Data Leak? Uncovering the Data Leakage in Cloud from Mobile Apps**. In Oakland 2019
7. **The Betrayal At Cloud City: An Empirical Analysis Of Cloud-Based Mobile Backends**, In USENIX Security 2019
8. **Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps**, In CCS 2019
9. **FirmScope: Automatic Uncovering of Privilege-Escalation Vulnerabilitiesin Pre-Installed Apps in Android Firmware**, In USENIX Security 2020

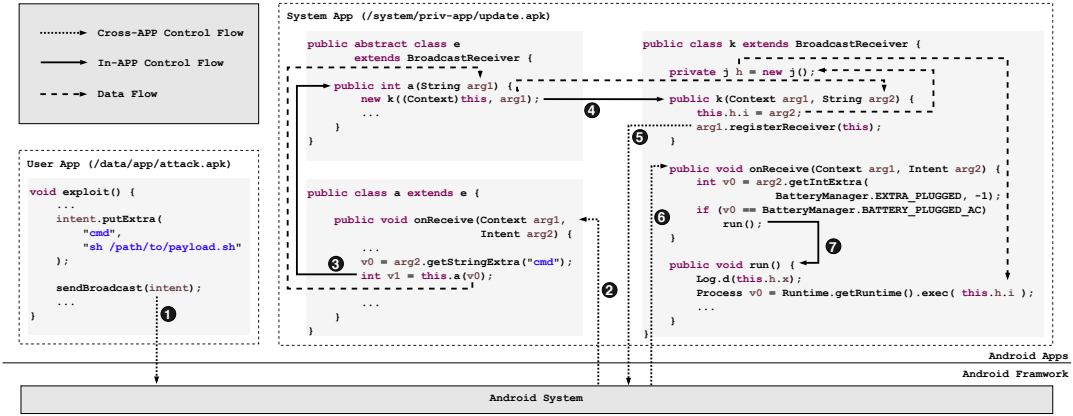
# Our Efforts on **Vulnerability Discovery** w/ Mobile App Centric Analysis

1. ~~**SMV-Hunter: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps**. In NDSS 2014~~
2. ~~**AutoForge: Automatic Forgery of Cryptographically Consistent Messages to Identify Security Vulnerabilities in Mobile Services**. In NDSS 2016~~
3. ~~**SmartGen: Exposing Server URLs of Mobile Apps With Selective Symbolic Execution**. In WWW 2017~~
4. ~~**AuthScope: Towards Automatic Identification of Vulnerable Authorizations in Online Services**. In CCS 2017~~
5. ~~**The Unexpected Danger of UX Features: A Case of Sensitive Data Leakage of Drivers in Ride-Hailing Services**. In NDSS 2019.~~
6. **LeakScope: Why Does Your Data Leak? Uncovering the Data Leakage in Cloud from Mobile Apps**. In Oakland 2019
7. **The Betrayal At Cloud City: An Empirical Analysis Of Cloud-Based Mobile Backends**, In USENIX Security 2019
8. ~~**Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps**, In CCS 2019~~
9. **FirmScope: Automatic Uncovering of Privilege-Escalation Vulnerabilitiesin Pre-Installed Apps in Android Firmware**, In USENIX Security 2020

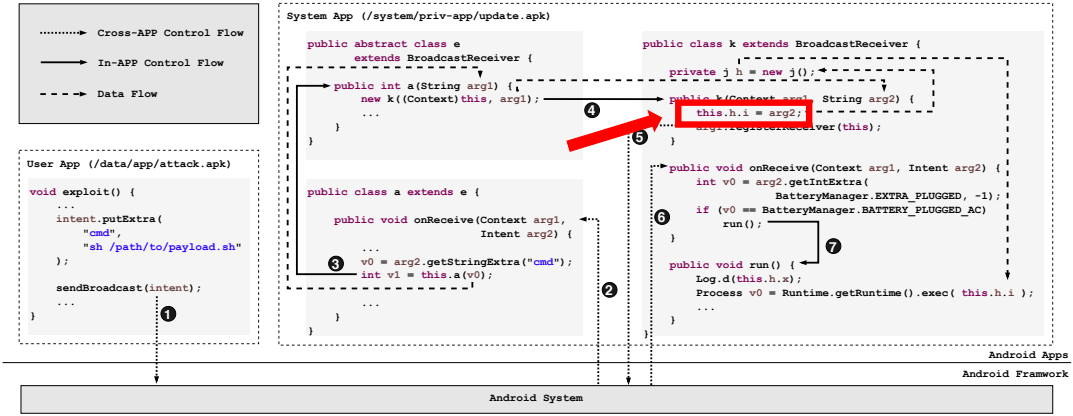
# Outline


1. Introduction
2. FirmScope: Privilege-Escalation in Firmware
3. LeakScope: Leakage in Clouds
4. SkyWalker: n-day/0-day in Servers
5. Summary

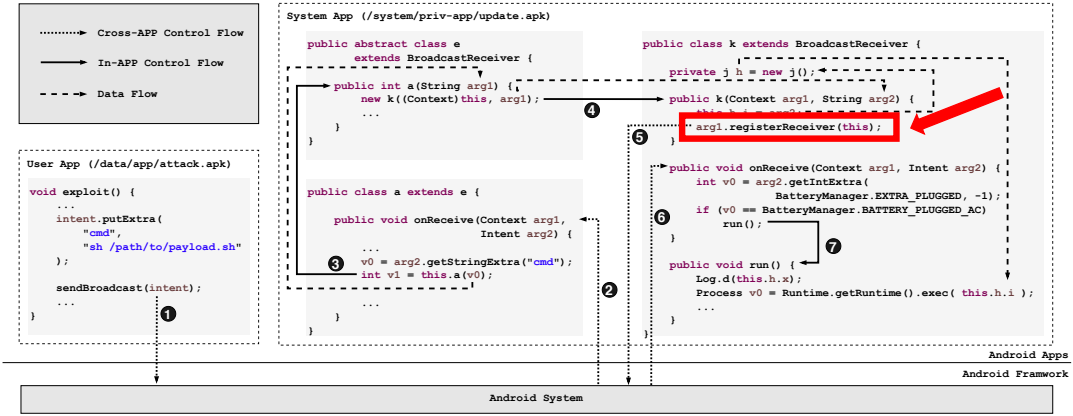
# Detecting Privilege Escalation in Preinstalled Apps: A Running Example



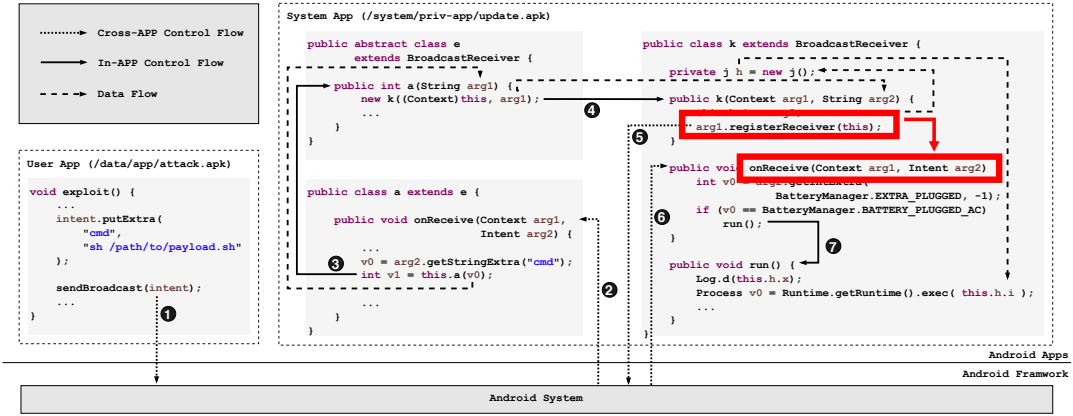
# Detecting Privilege Escalation in Preinstalled Apps: A Running Example



# Detecting Privilege Escalation in Preinstalled Apps: A Running Example



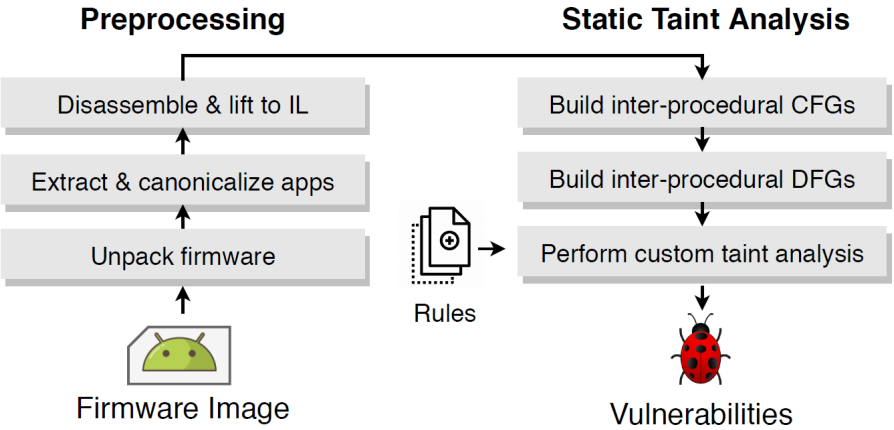
# Detecting Privilege Escalation in Preinstalled Apps: A Running Example



# Challenges

1. Tracking Flows Through Class Fields
2. Full vs. Partial Object Sensitivity
3. Handling the Android Runtime Framework APIs
4. Handling Asynchronous Callbacks
5. Handling Inter-component Communication

# FirmScope


Preprocessing
Static Taint Analysis
Disassemble & lift to IL
Extract & canonicalize apps
Unpack firmware
Firmware Image
Rules
Build inter-procedural CFGs
Build inter-procedural DFGs
Perform custom taint analysis
Vulnerabilities

# Dataset: 2,017 firmware form more than 100 Android vendors

| Vendor | # Firmware | # Apps | v4 | v5 | v6 | v7 | v8 | v9 |
|---|---|---|---|---|---|---|---|---|
| Alcatel | 31 | 4,390 | 15 | 3 | 9 | 3 | 1 | 0 |
| Alps | 48 | 9,557 | 15 | 7 | 22 | 3 | 1 | 0 |
| ASUS | 93 | 17,944 | 16 | 24 | 21 | 19 | 13 | 0 |
| BLU | 132 | 16,355 | 32 | 17 | 58 | 20 | 5 | 0 |
| Coolpad | 29 | 3,429 | 12 | 7 | 3 | 1 | 6 | 0 |
| Doogee | 25 | 3,310 | 3 | 3 | 10 | 9 | 0 | 0 |
| Elephone | 23 | 2,840 | 4 | 10 | 5 | 3 | 1 | 0 |
| Google | 372 | 54,057 | 0 | 1 | 0 | 175 | 142 | 54 |
| HTC | 39 | 9,361 | 15 | 11 | 11 | 2 | 0 | 0 |
| Huawei | 63 | 9,143 | 19 | 21 | 19 | 3 | 1 | 0 |
| Infinix | 29 | 4,476 | 0 | 0 | 8 | 8 | 13 | 0 |
| Lenovo | 82 | 9,209 | 52 | 14 | 12 | 3 | 1 | 0 |
| Motorola | 65 | 11,101 | 5 | 17 | 13 | 19 | 11 | 0 |
| Panasonic | 21 | 2,963 | 6 | 3 | 5 | 5 | 2 | 0 |
| Samsung | 219 | 61,457 | 9 | 1 | 65 | 71 | 55 | 18 |
| TCL | 33 | 5,309 | 6 | 6 | 16 | 4 | 1 | 0 |
| Tecno | 55 | 8,057 | 21 | 6 | 8 | 8 | 12 | 0 |
| XBO | 72 | 8,264 | 24 | 35 | 13 | 0 | 0 | 0 |
| Xiaomi | 102 | 21,331 | 11 | 10 | 36 | 14 | 20 | 11 |
| ZTE | 73 | 10,557 | 12 | 13 | 24 | 24 | 0 | 0 |
| Other | 411 | 58,232 | 126 | 82 | 83 | 55 | 65 | 0 |
| Total | 2,017 | 331,342 | 403 | 291 | 441 | 449 | 350 | 83 |
| | | | 19% | 14% | 22% | 23% | 17% | 4% |

# Vulnerabilities: 3,483 vulnerabilities (147 confirmed CVEs))

| Vulnerability | # Total | # Unique | %Firmware |
|---|---|---|---|
| Command Injection | 1,420 | 211 | 41% |
| Wireless Settings Modification | 901 | 212 | 26% |
| SMS Injection | 232 | 63 | 7% |
| Screen Recording | 207 | 63 | 6% |
| Factory Reset | 169 | 48 | 5% |
| System Properties Modification | 160 | 54 | 5% |
| App (Un)Installation | 153 | 54 | 5% |
| Full Logcat Leakage | 110 | 85 | 4% |
| Microphone Audio Recording | 61 | 38 | 2% |
| AT Command Injection | 55 | 17 | 2% |
| Code Injection | 15 | 5 | 1% |
| Total | 3,483 | 850 | 77% |

# Vulnerabilities: 3,483 vulnerabilities (147 confirmed CVEs))

| Vulnerability | # Total | # Unique | %Firmware |
|---|---|---|---|
| Command Injection | 1,420 | 211 | 41% |
| Wireless Settings Modification | 901 | 212 | 26% |
| SMS Injection | 232 | 63 | 7% |
| Screen Recording | 207 | 63 | 6% |
| Factory Reset | 169 | 48 | 5% |
| System Properties Modification | 160 | 54 | 5% |
| App (Un)Installation | 153 | 54 | 5% |
| Full Logcat Leakage | 110 | 85 | 4% |
| Microphone Audio Recording | 61 | 38 | 2% |
| AT Command Injection | 55 | 17 | 2% |
| Code Injection | 15 | 5 | 1% |
| Total | 3,483 | 850 | 77% |

1. Command Injection
2. Wireless Settings Modification
3. SMS Injection
4. Screen Recording
5. Factory Reset
6. System Properties Modification
7. App (Un)Installation
8. Full Logcat Leakage
9. Microphone Audio Recording
10. AT Command Injection
11. Code Injection

# Vulnerabilities: 3,483 vulnerabilities (147 confirmed CVEs))

| Vendor | # Total Vuln. per firmware | # Unique Vulnerabilities | Command Injection | Wireless Settings Modification | SMS Injection | Screen Recording | Factory Reset | System Properties Modification | App (Un)Installation | Full Logcat Leakage | Microphone Audio Recording | AT Command Injection | Code Injection |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Alcatel | 1.3 | 23 | 15 | 0 | 4 | 1 | 0 | 0 | 0 | 0 | 3 | 0 | 0 |
| Alps | 1.1 | 21 | 20 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ASUS | 3.7 | 132 | 41 | 53 | 5 | 0 | 11 | 17 | 2 | 2 | 0 | 1 | 0 |
| BLU | 2.2 | 63 | 43 | 5 | 7 | 4 | 0 | 0 | 1 | 0 | 1 | 2 | 0 |
| Coolpad | 3 | 54 | 22 | 4 | 2 | 3 | 0 | 7 | 6 | 1 | 1 | 7 | 1 |
| Doogee | 3.3 | 48 | 26 | 2 | 6 | 9 | 1 | 0 | 2 | 0 | 0 | 2 | 0 |
| Elephone | 2.7 | 36 | 26 | 1 | 0 | 2 | 0 | 2 | 3 | 0 | 0 | 1 | 1 |
| Google | 0.6 | 21 | 0 | 3 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| HTC | 1.5 | 27 | 4 | 15 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Huawei | 1.2 | 22 | 2 | 12 | 0 | 0 | 0 | 0 | 0 | 6 | 1 | 1 | 0 |
| Infinix | 0.6 | 8 | 2 | 0 | 2 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| Lenovo | 1.2 | 44 | 21 | 4 | 2 | 1 | 1 | 1 | 0 | 11 | 1 | 2 | 0 |
| Motorola | 0.6 | 24 | 0 | 7 | 10 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 |
| Panasonic | 2.3 | 34 | 27 | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 0 | 3 | 0 |
| Samsung | 3.3 | 178 | 16 | 50 | 1 | 15 | 29 | 0 | 30 | 37 | 0 | 0 | 0 |
| TCL | 1.4 | 33 | 20 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 9 | 2 | 0 |
| Tecno | 1.2 | 28 | 17 | 0 | 3 | 2 | 0 | 2 | 0 | 0 | 0 | 2 | 2 |
| XBO | 2.2 | 37 | 29 | 0 | 2 | 3 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| Xiaomi | 2.2 | 118 | 46 | 27 | 4 | 1 | 0 | 5 | 2 | 19 | 13 | 1 | 0 |
| ZTE | 0.6 | 23 | 11 | 0 | 3 | 0 | 1 | 6 | 0 | 0 | 0 | 2 | 0 |
| Other | 1.7 | 239 | 82 | 50 | 20 | 23 | 6 | 20 | 15 | 8 | 11 | 2 | 2 |

1. Command Injection
2. Wireless Settings Modification
3. SMS Injection
4. Screen Recording
5. Factory Reset
6. System Properties Modification
7. App (Un)Installation
8. Full Logcat Leakage
9. Microphone Audio Recording
10. AT Command Injection
11. Code Injection

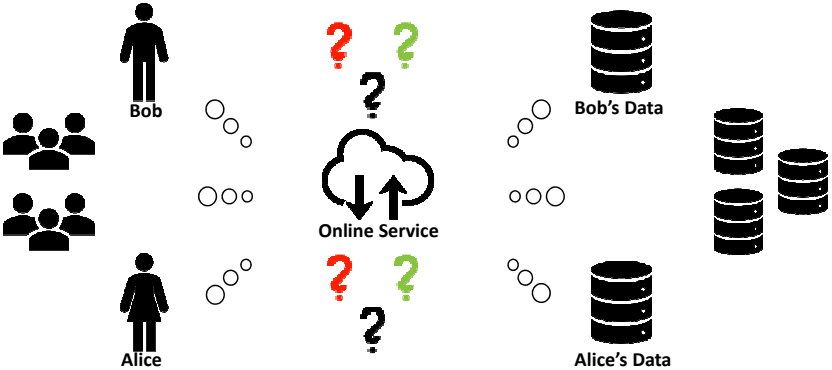
# FirmScope vs. Other Frameworks

| Benchmarks | FlowDroid | | Amandroid | | DroidSafe | | FIRMSCOPE | |
|---|---|---|---|---|---|---|---|---|
| | FP | FN | FP | FN | FP | FN | FP | FN |
| Aliasing | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AndroidSpecific | 0 | 2 | 0 | 2 | 0 | 1 | 0 | 1 |
| ArraysAndLists | 4 | 0 | 4 | 2 | 4 | 0 | 3 | 0 |
| Callbacks | 2 | 1 | 7 | 9 | 4 | 0 | 1 | 1 |
| EmulatorDetection | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| FieldAndObjectSensitivity | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| GeneralJava | 4 | 4 | 4 | 5 | 2 | 1 | 3 | 2 |
| InterComponentComm. | 0 | 8 | 2 | 0 | 0 | 1 | 0 | 4 |
| Lifecycle | 0 | 9 | 3 | 10 | 11 | 8 | 0 | 3 |
| Reflection | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 |
| Threading | 0 | 0 | 0 | 0 | 5 | 4 | 0 | 0 |
| Total (lower is better) | 10 | 24 | 20 | 30 | 28 | 15 | 7 | 11 |

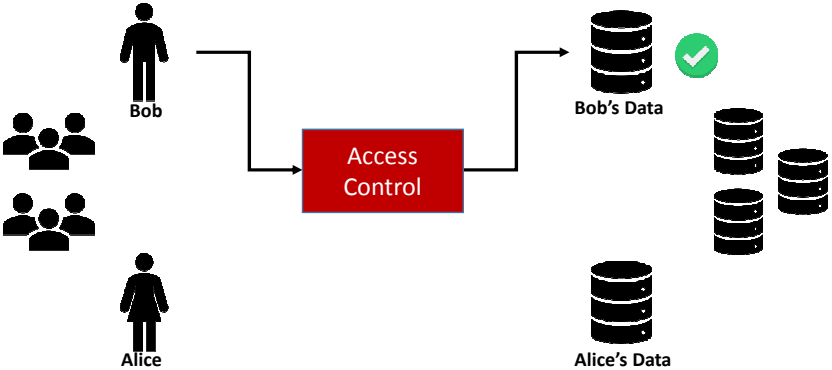
# Outline


1. Introduction
2. FirmScope: Privilege-Escalation in Firmware
3. LeakScope: Leakage in Clouds
4. SkyWalker: n-day/0-day in Servers
5. Summary

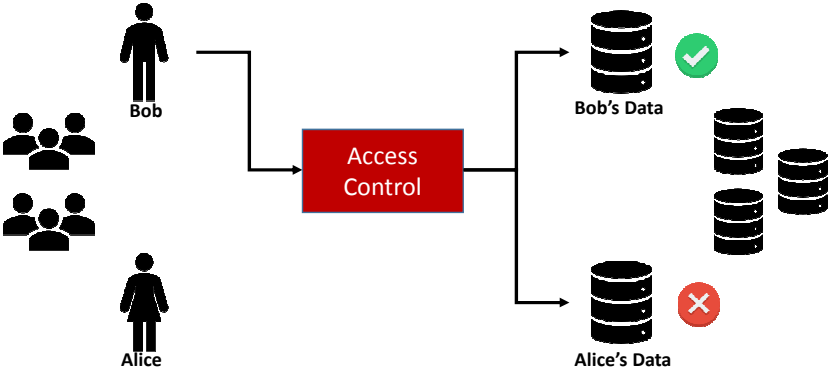
# Vulnerability II: Data Leakage is Essentially an Access Control Problem



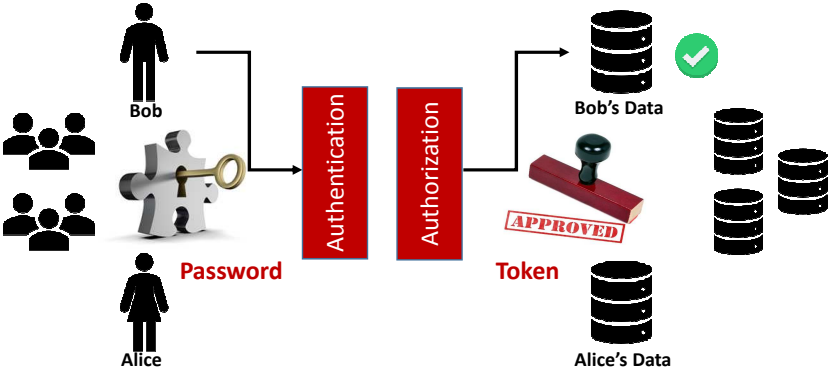
# Vulnerability II: Data Leakage is Essentially an Access Control Problem



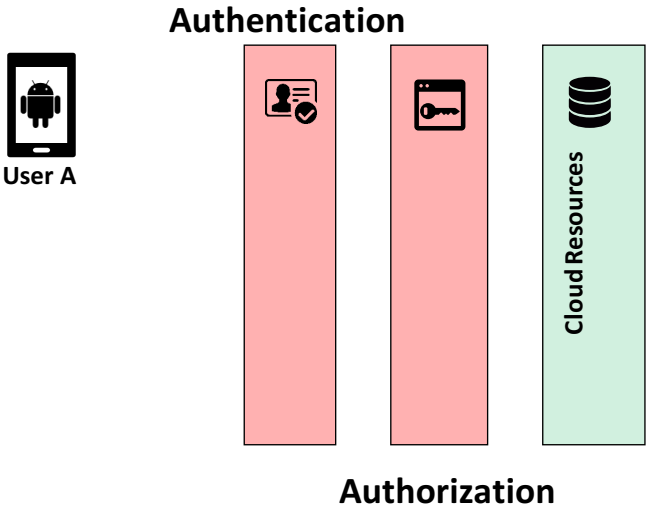
# Vulnerability II: Data Leakage is Essentially an Access Control Problem



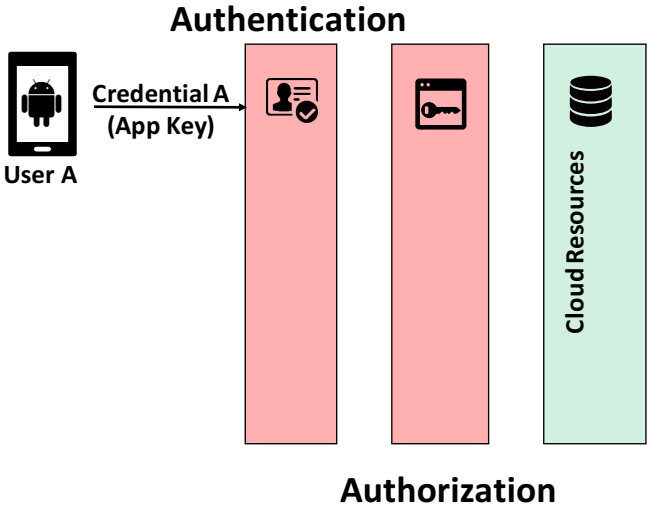
# Vulnerability II: Data Leakage is Essentially an Access Control Problem



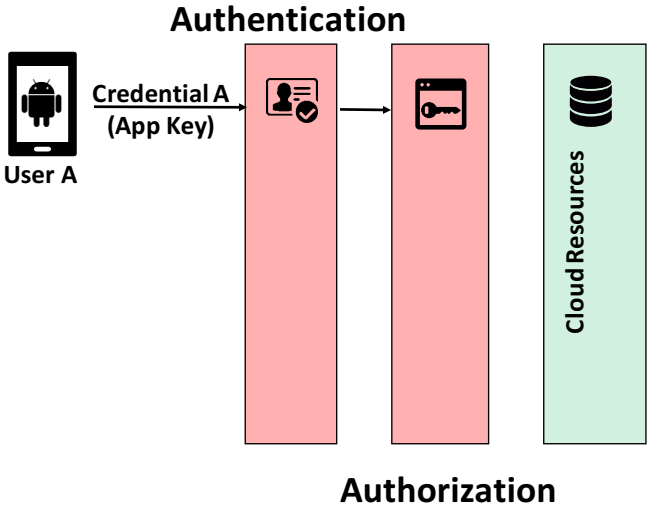
# How Do Mobile Apps and mBaaS Cloud Communicate



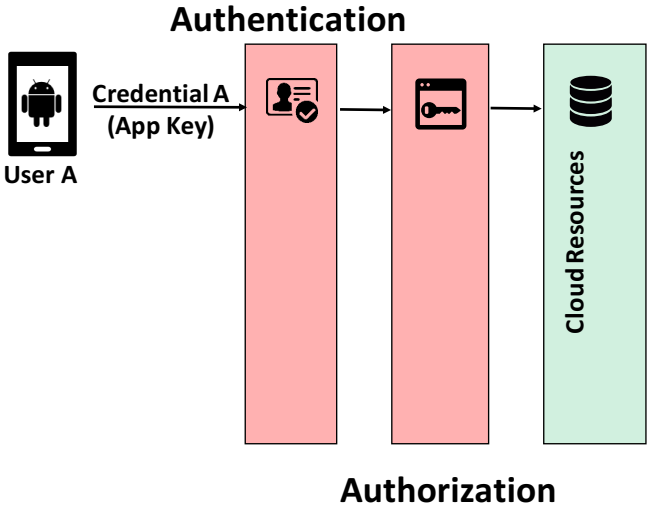
# How Do Mobile Apps and mBaaS Cloud Communicate



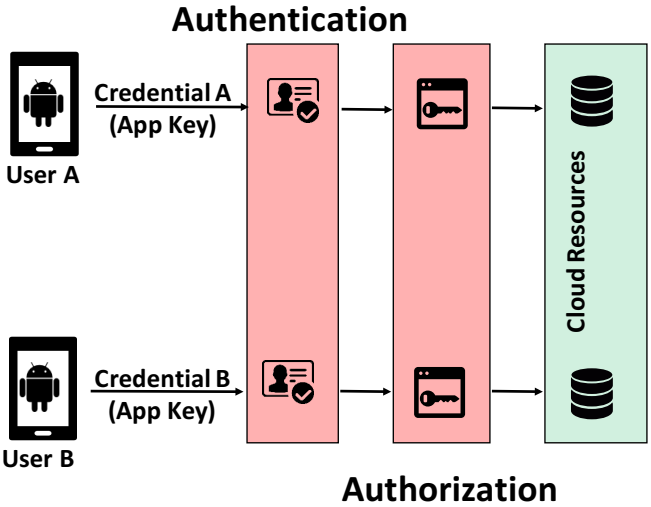
# How Do Mobile Apps and mBaaS Cloud Communicate



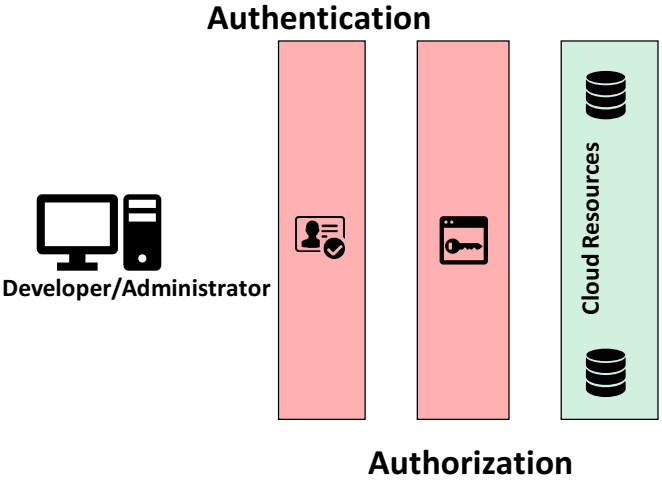
# How Do Mobile Apps and mBaaS Cloud Communicate



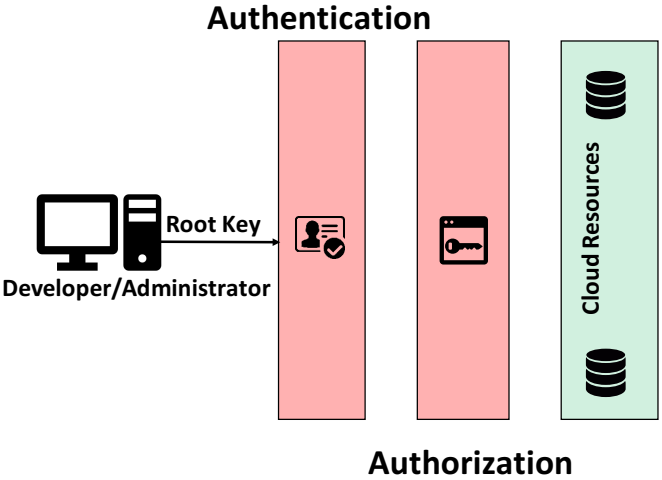
# How Do Mobile Apps and mBaaS Cloud Communicate



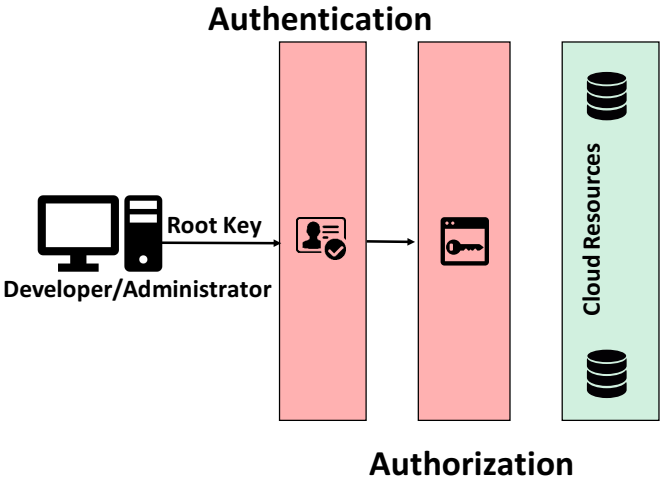
# How Do Mobile Apps and mBaaS Cloud Communicate



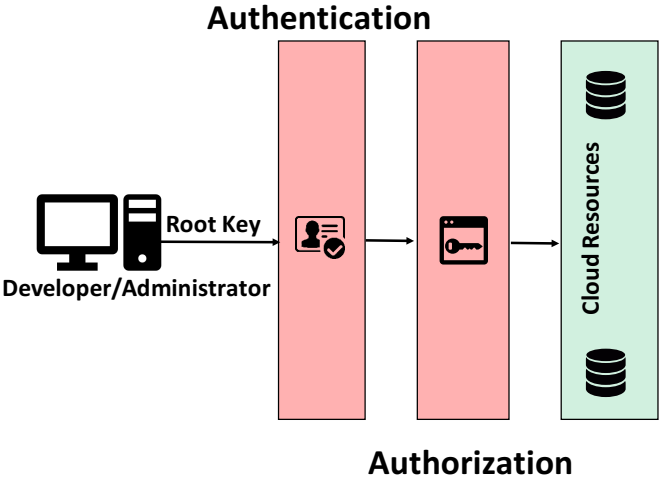
# How Do Mobile Apps and mBaaS Cloud Communicate



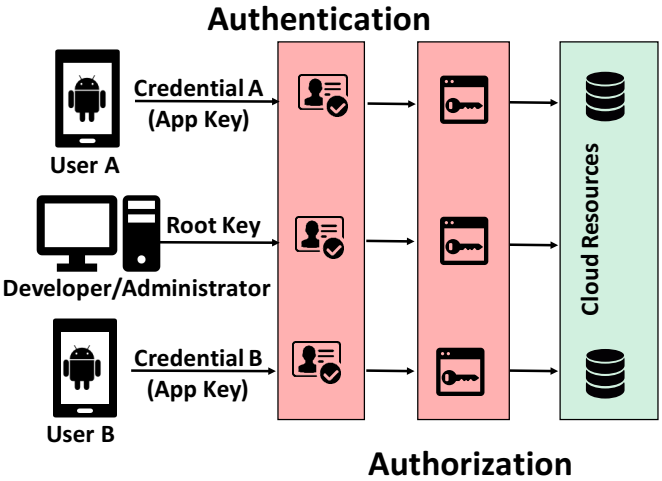
# How Do Mobile Apps and mBaaS Cloud Communicate



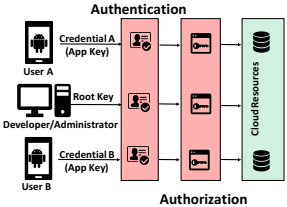
# How Do Mobile Apps and mBaaS Cloud Communicate



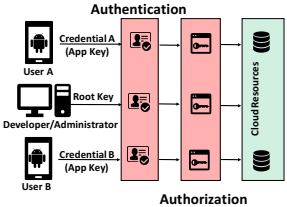
# How Do Mobile Apps and mBaaS Cloud Communicate



# Our Discovery of Why Cloud Data was Leaked

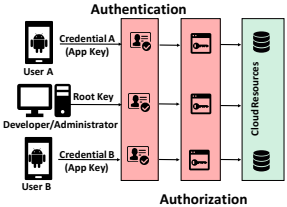


# Our Discovery of Why Cloud Data was Leaked




AWS Firebase Azure

# Our Discovery of Why Cloud Data was Leaked




## The Root Causes of the Cloud Data Leakage

1. Misuse of Various Keys in **Authentication**
   - Microsoft Azure Storage
   - Microsoft Azure Notification Hubs
   - Amazon AWS
2. Misconfiguration of User Permissions in **Authorization**
   - Google Firebase
   - Amazon AWS

# Misuse of Root Keys in Microsoft Azure

| Service | Key Type | Example |
|---|---|---|
| Azure Storage | Account Key | DefaultEndpointsProtocol=https; AccountName=*;AccountKey=* |
| | SAS | https://*.blob.core.windows.net/* ?sv=*&st=*&se=*&sr=b& sp=rw&sip=*&spr=https&sig=* |
| Notification Hub | Listening Key | Endpoint=sb://*.servicebus.windows.net/; SharedAccessKeyName= DefaultListenSharedAccessSignature; SharedAccessKey=* |
| | Full Access Key | Endpoint=sb://*.servicebus.windows.net/; SharedAccessKeyName= DefaultFullSharedAccessSignature; SharedAccessKey=* |

Table: Examples of the Keys Used in Mobile App Development with Microsoft Azure Cloud. We use symbol * to anonymize those sensitive data in the keys.

# Misconfiguration of User Permissions in Google Firebase

```
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

Figure: A Correct Firebase Authorization Rule

# Misconfiguration of User Permissions in Google Firebase

```
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

Figure: A Correct Firebase Authorization Rule

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

**(a)**

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

**(b)**

Figure: Two Misconfigured Firebase Authorization Rules

# How to Identify a Token vs the Token: A Running Example

```
GET /api/v1//users/21690/notifications?in_app_token=e67315b35aa3
8d4ac8cac3cd9c7f88ae7f576d373f HTTP/1.1
Host: api.w****.com
Connection: close
```

```
HTTP/1.1 200 OK
Cache-Control: max-age=0, private, must-revalidate
Content-Type: application/json
ETag: W/"5319d96924bb6d0a761b5f13b248919c"
Server: nginx/1.6.2
X-Request-Id: 5775d45e-cc3b-4665-8bc6-c2c7a2c9180d
X-Runtime: 0.027840
Content-Length: 191
Connection: Close

[{"id":433222,"sender":null,"dog":null,"notification_type":15,"n
otification_text":"Welcome to w****.","object_id":21690,"is_seen
":true,"is_read":true,"created_at":"2017-01-28T23:54:59.831Z"}]
```

**Alice's first request and response message after login**

# How to Identify a Token vs the Token: A Running Example

```
GET /api/v1//users/21691/notifications?in_app_token=fb153b7d8c0a
0c6ac841d7bfbd9446de627c642858 HTTP/1.1
Host: api.w****.com
Connection: close
```

```
HTTP/1.1 200 OK
Cache-Control: max-age=0, private, must-revalidate
Content-Type: application/json
ETag: W/"6ee365b32e7f3e145d5c74778ea243cd"
Server: nginx/1.6.2
X-Request-Id: 4970cafb-9438-4a70-96e0-ca2f789f0d5d
X-Runtime: 0.022889
Content-Length: 192
Connection: Close

[{"id":433227,"sender":null,"dog":null,"notification_type":15,"n
otification_text":"Welcome to w****.","object_id":21691,"is_seen
":true,"is_read":false,"created_at":"2017-01-28T23:56:40.533Z"}]
```

**Bob's first request and response message after login**

# How to Identify a Token vs the Token: A Running Example

```
GET /api/v1//users/21690/notifications?in_app_token=e67315b35aa3
8d4ac8cac3cd9c7f88ae7f576d373f HTTP/1.1
Host: api.w****.com
Connection: close
```

**Alice's first request message after login**

```
GET /api/v1//users/21691/notifications?in_app_token=fb153b7d8c0a
0c6ac841d7bfbd9446de627c642858 HTTP/1.1
Host: api.w****.com
Connection: close
```

**Bob's first request message after login**

# How to Identify a Token vs the Token: A Running Example

```
GET /api/v1//users/21690/notifications?in_app_token=e67315b35aa3
8d4ac8cac3cd9c7f88ae7f576d373f HTTP/1.1
Host: api.w****.com
Connection: close
```

```
GET /api/v1//users/21691/notifications?in_app_token=fb153b7d8c0a
0c6ac841d7bfbd9446de627c642858 HTTP/1.1
Host: api.w****.com
Connection: close
```

# How to Identify a Token vs the Token: A Running Example

```
GET /api/v1//users/21690/notifications?in_app_token=e67315b35aa3
8d4ac8cac3cd9c7f88ae7f576d373f HTTP/1.1
Host: api.w****.com
Connection: close
```

```
GET /api/v1//users/21691/notifications?in_app_token=fb153b7d8c0a
0c6ac841d7bfbd9446de627c642858 HTTP/1.1
Host: api.w****.com
Connection: close
```

# How to Identify a Token vs the Token: A Running Example

```
GET /api/v1//users/21691/notifications?in_app_token=e67315b35aa3
8d4ac8cac3cd9c7f88ae7f576d373f HTTP/1.1
Host: api.w****.com
Connection: close
```

# How to Identify a Token vs the Token: A Running Example

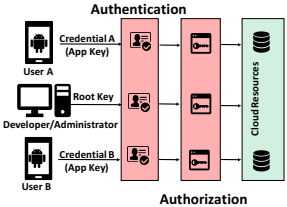
```
GET /api/v1//users/21691/notifications?in_app_token=e67315b35aa3
8d4ac8cac3cd9c7f88ae7f576d373f HTTP/1.1
Host: api.w****.com
Connection: close
```

```
HTTP/1.1 200 OK
Cache-Control: max-age=0, private, must-revalidate
Content-Type: application/json
ETag: W/"6ee365b32e7f3e145d5c74778ea243cd"
Server: nginx/1.6.2
X-Request-Id: 4970cafb-9438-4a70-96e0-ca2f789f0d5d
X-Runtime: 0.022889
Content-Length: 192
Connection: Close

[{"id":433227,"sender":null,"dog":null,"notification_type":15,"n
otification_text":"Welcome to w****.","object_id":21691,"is_seen
":true,"is_read":false,"created_at":"2017-01-28T23:56:40.533Z"}]
```

**Alice reads Bob's notifications**

# Problem Statement: How to automatically detect the cloud leakage at scale

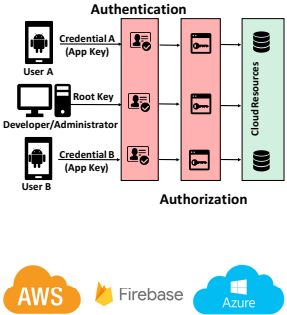


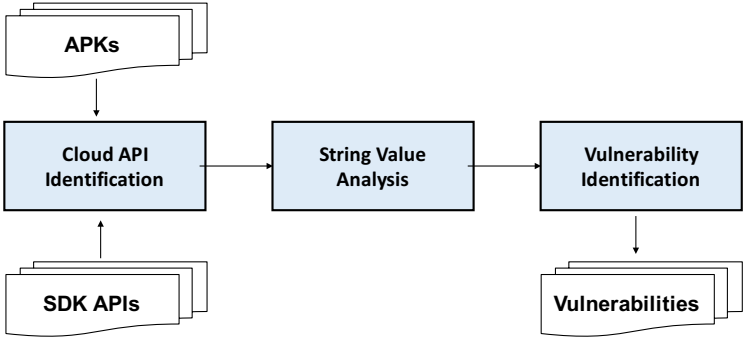

AWS Firebase Azure

## Challenges

1. How to systematically identify various keys used by mobile apps (**Cloud APIs**)
2. How to identify the relevant key strings that are used by mobile apps (**String Analysis**)
3. How to design an obfuscation-resilient approach to identify cloud APIs and key strings of our interest (**Obufscation-Resilient**)
4. How to determine the vulnerablity without leaking sensitive information in the cloud (**Vulnerability Confirmation**)

# Our LeakScope [Oakland'19]



# Distributions of the Testing Apps

| | Total | | Non-Obfuscated | | Obfuscated | |
|---|---|---|---|---|---|---|
| | #Apps | % | #Apps | % | #Apps | % |
| w/ Cloud API | 107,081 | - | 85,357 | 79.71 | 21,724 | 20.29 |
| w/ AWS only | 4,799 | 4.48 | 4,548 | 5.33 | 251 | 1.16 |
| w/ Azure only | 899 | 0.84 | 720 | 0.84 | 179 | 0.82 |
| w/ Firebase only | 99,186 | 92.63 | 78,475 | 91.94 | 20,711 | 95.34 |
| w/ AWS & Azure | 3 | 0.00 | 2 | 0.00 | 1 | 0.00 |
| w/ AWS & Firebase | 1,973 | 1.84 | 1,427 | 1.67 | 546 | 2.51 |
| w/ Azure & Firebase | 210 | 0.20 | 174 | 0.20 | 36 | 0.17 |
| w/ Three Services | 11 | 0.01 | 11 | 0.01 | 0 | 0.00 |

# Distributions of the Testing Apps

| | Total | | Non-Obfuscated | | Obfuscated | |
|---|---|---|---|---|---|---|
| | #Apps | % | #Apps | % | #Apps | % |
| w/ Cloud API | 107,081 | - | 85,357 | 79.71 | 21,724 | 20.29 |
| w/ AWS only | 4,799 | 4.48 | 4,548 | 5.33 | 251 | 1.16 |
| w/ Azure only | 899 | 0.84 | 720 | 0.84 | 179 | 0.82 |
| w/ Firebase only | 99,186 | 92.63 | 78,475 | 91.94 | 20,711 | 95.34 |
| w/ AWS & Azure | 3 | 0.00 | 2 | 0.00 | 1 | 0.00 |
| w/ AWS & Firebase | 1,973 | 1.84 | 1,427 | 1.67 | 546 | 2.51 |
| w/ Azure & Firebase | 210 | 0.20 | 174 | 0.20 | 36 | 0.17 |
| w/ Three Services | 11 | 0.01 | 11 | 0.01 | 0 | 0.00 |

# Statistics of The Detected Vulnerabilities

| | The Root Cause | Non-Obfuscated #Apps | Non-Obfuscated % | Obfuscated #Apps | Obfuscated % |
|---|---|---|---|---|---|
| Azure | Account Key Misuse | 85 | 9.37 | 18 | 8.33 |
| | Full Access Key Misuse | 101 | 11.14 | 12 | 5.56 |
| AWS | Root key Misuse | 477 | 7.97 | 92 | 11.53 |
| | "Open" S3 Storage | 916 | 15.30 | 195 | 24.44 |
| Firebase | "Open" Database | 5,166 | 6.45 | 1,214 | 5.70 |
| | No Permission Check | 6,855 | 8.56 | 2,168 | 10.18 |

# Statistics of The Detected Vulnerabilities

| | The Root Cause | Non-Obfuscated #Apps | Non-Obfuscated % | Obfuscated #Apps | Obfuscated % |
|---|---|---|---|---|---|
| Azure | Account Key Misuse | 85 | 9.37 | 18 | 8.33 |
| | Full Access Key Misuse | 101 | 11.14 | 12 | 5.56 |
| AWS | Root key Misuse | 477 | 7.97 | 92 | 11.53 |
| | "Open" S3 Storage | 916 | 15.30 | 195 | 24.44 |
| Firebase | "Open" Database | 5,166 | 6.45 | 1,214 | 5.70 |
| | No Permission Check | 6,855 | 8.56 | 2,168 | 10.18 |

# Statistics of The Detected Vulnerabilities

| | # Non-Vulnerable Apps | | | | # Vulnerable Apps | | | |
|---|---|---|---|---|---|---|---|---|
| **#Downloads** | **Azure** | **AWS** | **Firebase** | **Obfuscated%** | **Azure** | **AWS** | **Firebase** | **Obfuscated%** |
| $1,000,000,000 - 5,000,000,000$ | 0 | 0 | 1 | 100.00 | 0 | 0 | 0 | 0.00 |
| $500,000,000 - 1,000,000,000$ | 0 | 0 | 3 | 66.67 | 0 | 0 | 0 | 0.00 |
| $100,000,000 - 500,000,000$ | 0 | 1 | 35 | 58.33 | 0 | 1 | 9 | 50.00 |
| $50,000,000 - 100,000,000$ | 0 | 4 | 67 | 45.07 | 0 | 2 | 12 | 71.43 |
| $10,000,000 - 50,000,000$ | 2 | 35 | 480 | 47.78 | 1 | 4 | 75 | 50.00 |
| $5,000,000 - 10,000,000$ | 3 | 32 | 467 | 37.85 | 1 | 6 | 66 | 38.36 |
| $1,000,000 - 5,000,000$ | 16 | 136 | 2,405 | 32.15 | 2 | 21 | 369 | 30.10 |
| $500,000 - 1,000,000$ | 10 | 105 | 1,823 | 29.36 | 1 | 29 | 260 | 28.28 |
| $100,000 - 500,000$ | 65 | 356 | 6,987 | 26.01 | 14 | 66 | 1,026 | 26.13 |
| $50,000 - 100,000$ | 42 | 249 | 4,608 | 25.52 | 11 | 50 | 695 | 25.13 |
| $10,000 - 50,000$ | 167 | 679 | 12,868 | 24.85 | 21 | 174 | 1,862 | 21.88 |
| $5,000 - 10,000$ | 82 | 369 | 6,090 | 24.05 | 11 | 100 | 770 | 23.61 |
| $1,000 - 5,000$ | 272 | 976 | 15,920 | 21.42 | 40 | 248 | 1,977 | 20.66 |
| $0 - 1,000$ | 464 | 3,844 | 49,626 | 15.92 | 111 | 754 | 6,402 | 20.30 |

Table: The Number of Apps that Have Used the Cloud APIs in Each of The Accumulated Download Category.

# Statistics of The Detected Vulnerabilities

| | # Non-Vulnerable Apps | | | | # Vulnerable Apps | | | |
|---|---|---|---|---|---|---|---|---|
| **#Downloads** | **Azure** | **AWS** | **Firebase** | **Obfuscated%** | **Azure** | **AWS** | **Firebase** | **Obfuscated%** |
| $1,000,000,000 - 5,000,000,000$ | 0 | 0 | 1 | 100.00 | 0 | 0 | 0 | 0.00 |
| $500,000,000 - 1,000,000,000$ | 0 | 0 | 3 | 66.67 | 0 | 0 | 0 | 0.00 |
| $100,000,000 - 500,000,000$ | 0 | 1 | 35 | 58.33 | 0 | 1 | 9 | 50.00 |
| $50,000,000 - 100,000,000$ | 0 | 4 | 67 | 45.07 | 0 | 2 | 12 | 71.43 |
| $10,000,000 - 50,000,000$ | 2 | 35 | 480 | 47.78 | 1 | 4 | 75 | 50.00 |
| $5,000,000 - 10,000,000$ | 3 | 32 | 467 | 37.85 | 1 | 6 | 66 | 38.36 |
| $1,000,000 - 5,000,000$ | 16 | 136 | 2,405 | 32.15 | 2 | 21 | 369 | 30.10 |
| $500,000 - 1,000,000$ | 10 | 105 | 1,823 | 29.36 | 1 | 29 | 260 | 28.28 |
| $100,000 - 500,000$ | 65 | 356 | 6,987 | 26.01 | 14 | 66 | 1,026 | 26.13 |
| $50,000 - 100,000$ | 42 | 249 | 4,608 | 25.52 | 11 | 50 | 695 | 25.13 |
| $10,000 - 50,000$ | 167 | 679 | 12,868 | 24.85 | 21 | 174 | 1,862 | 21.88 |
| $5,000 - 10,000$ | 82 | 369 | 6,090 | 24.05 | 11 | 100 | 770 | 23.61 |
| $1,000 - 5,000$ | 272 | 976 | 15,920 | 21.42 | 40 | 248 | 1,977 | 20.66 |
| $0 - 1,000$ | 464 | 3,844 | 49,626 | 15.92 | 111 | 754 | 6,402 | 20.30 |

Table: The Number of Apps that Have Used the Cloud APIs in Each of The Accumulated Download Category.

# Engaging with the Cloud Providers

**Disclosed** all the vulnerabilities we have identified. Cloud providers futher notified the app developers.

1. **Microsoft** immediately corrected the wrong documentation
2. **Google** plans to provide more user-friendly SDKs when configuring user permissions in authorization.
3. **Amazon** added new permission checks with its S3 storage in November 2017 (two weeks before we disclosed our details to them)

# Engaging with the Cloud Providers

## Disclaimer on the use of account key

Browse files

master (#65)

seguler committed on Dec 21, 2017 1 parent 191f088 commit d90c3a49312e77c2cc911c8f55a37be9947454e4

Showing 1 changed file with 7 additions and 0 deletions. Unified Split

7 microsoft-azure-storage-samples/src/com/microsoft/azure/storage/samples/MainActivity.java View

```
@@ -24,6 +24,13 @@
 24  24      * MODIFY THIS!
 25  25      *
 26  26      * Stores the storage connection string.
     27  +   * Only use Shared Key authentication (Account Key) for testing purposes!
     28  +   * Your account name and account key, which give full read/write access to the associated Storage account,
     29  +   * will be distributed to every person that downloads your app.
     30  +   * This is not a good practice as you risk having your key compromised by untrusted clients.
     31  +   * Please consult following documents to understand and use Shared Access Signatures instead.
     32  +   * https://docs.microsoft.com/en-us/rest/api/storageservices/delegating-access-with-a-shared-access-signature
     33  +   * https://docs.microsoft.com/en-us/azure/storage/common/storage-dotnet-shared-access-signature-part-1
 27  34      */
 28  35     public static final String storageConnectionString = "DefaultEndpointsProtocol=https;"
 29  36             + "AccountName=[MY_ACCOUNT_NAME];"
```

# Engaging with the Cloud Providers

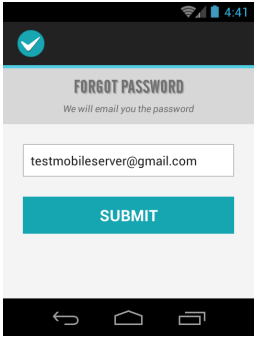
## Google's Update

1. *"The big additions on our side are tools for local emulation and writing tests against the database products including their security rules, which we expect to have a marked improvement on the ability of customers to test and validate security rules."*
2. *"Additionally, we have alerting for customers (sent every few weeks) for anyone using the Realtime Database or Cloud Firestore with open rules. "*
3. *"We're exploring more options, but those are a start."*

# Outline


1. Introduction
2. FirmScope: Privilege-Escalation in Firmware
3. LeakScope: Leakage in Clouds
4. SkyWalker: n-day/0-day in Servers
5. Summary

# Generating the **IPs/URLs** of Servers for Vulnerability Discovery



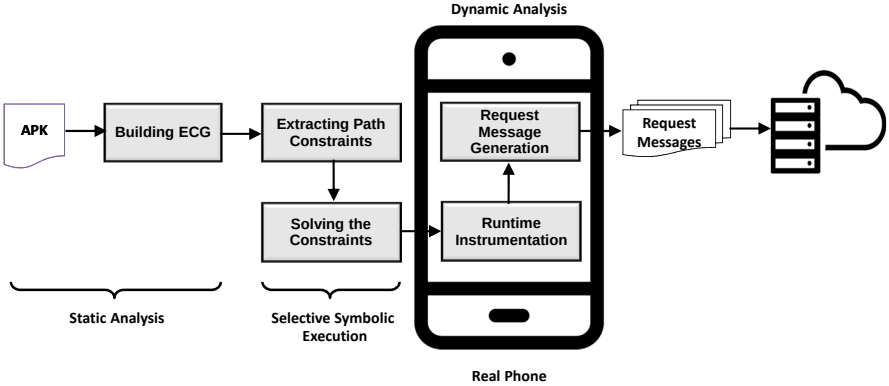

```
1  package com.shopclues;
2
3  class y implements View$OnClickListener {
4      EditText b;
5      ...
6      public void onClick(View arg5) {
7          String v0 = this.b.getText().toString().trim();
8          if(v0.equalsIgnoreCase("")) {
9              Toast.makeText(this.a, "Email Id should not be
               empty", 1).show();
10         }
11         else if(!al.a(v0)) {
12             Toast.makeText(this.a, "The email entered is not
               a valid email", 1).show();
13         }
14         else if(al.b(this.a)) {
15             this.a.c = new ac(this.a, v0);
16             this.a.c.execute(new Void[0]);
17         }
18         else {
19             Toast.makeText(this.a, "Please check your
               internet connection", 1).show();
20         }
21     }
22 }
```

# Generating the **IPs/URLs** of Servers for Vulnerability Discovery

## Various Constraints in Mobile Apps

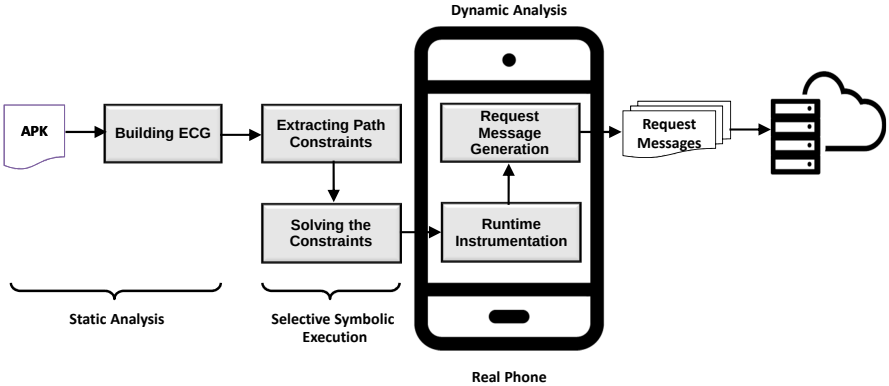
1. Two text-box's inputs need to be equivalent
2. The "age" needs to be greater than 18
3. A "zip code" needs to be a five digit sequence
4. A "phone number" needs to be a phone number
5. A file name extension needs to be some type (e.g., `jpg`)
6. ...

# Our SmartGen [WWW'17]



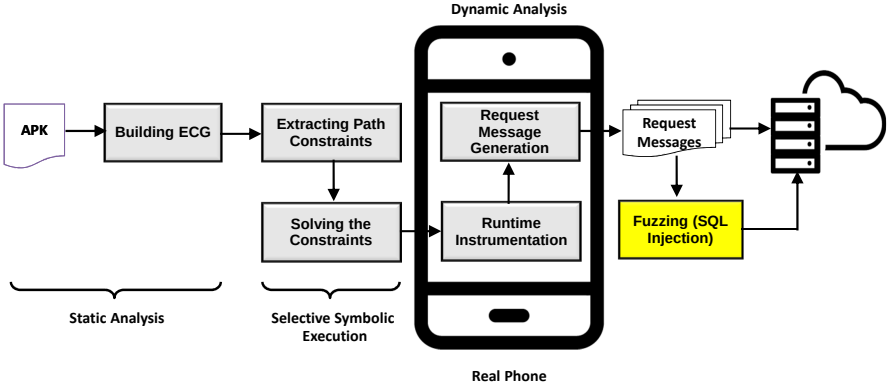

- Automated
- Systematic
- Scalable

# Our SmartGen [WWW'17]



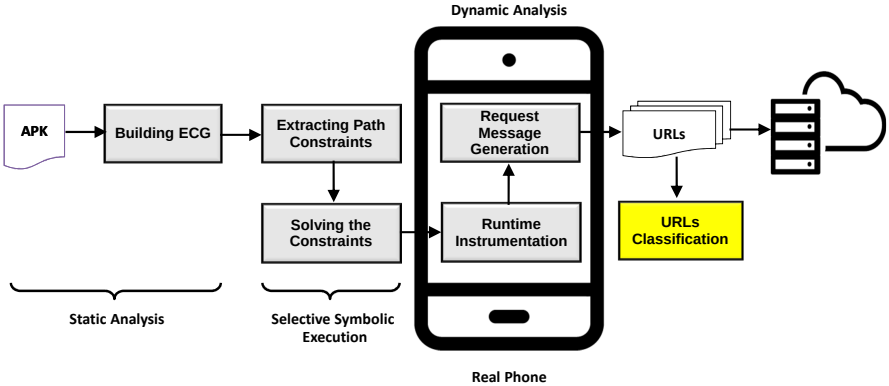

- Static analysis
- Selective symbolic execution
- Dynamic analysis

# SQL Injection Fuzzing



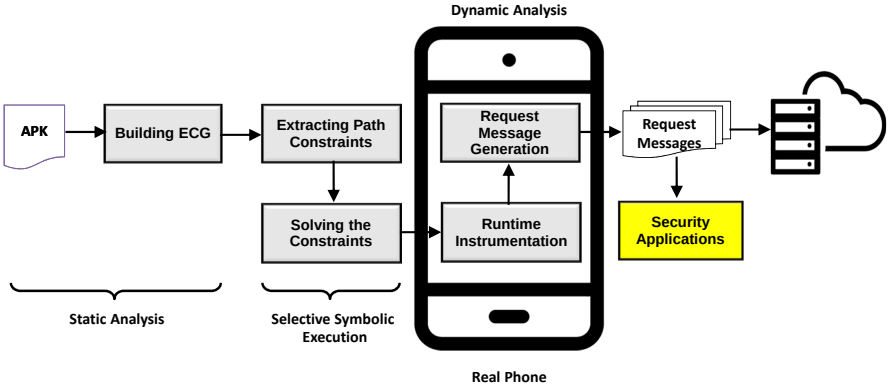

- "SELECT PG_SLEEP(5);", "SELECT PG_SLEEP(10);"
- "';WAITFOR DELAY '0:0:5'--"
- ";SELECT COUNT(*) FROM SYSIBM.SYSTABLES"

# Malicious URL Detection [WWW'17]




- Malware sites
- Compromised sites
- `VirusTotal` provides services for these detections

# Other Security Vulnerabilities




- **n-day vulnerabilities** by fingerprinting servers (e.g., Hypervisors, OS, Networking daemons, and libraries such as vulnerable SSL/TLS, Apache, and PHP)
- **0-day vulnerabilities**, e.g., Cross Site Scripting (XSS), XML External Entity (XXE) Vulnerabilities, in addition to SQLi

# Our SkyWalker [USENIX Security 2019]

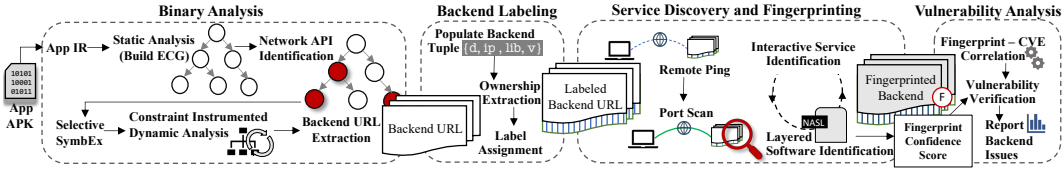



Figure: SkyWalker Overview. Phase 1 (Binary Analysis) extracts backend URLs through a dynamic binary instrumentation technique. Phase 2 labels backends into first-party, third-party, and hybrid based on IP and public available information. Phase 3 discovers and fingerprints the backend services to collect cloud layer information. Phase 4 (vulnerability analysis) uses the fingerprints and correlates them with public vulnerabilities to identify vulnerable backends.

# Implementation

1. For the **binary analysis** tool implementation, we relied on Soot [soo], SmartGen [ZL17], FlowDroid [ARF+], Z3-str [ZZG], and Xposed [xpo] with custom code written in Java ($7,000$ lines of code) and Python ($900$ lines of code).
2. For our **backend labeling** implementation, we relied on Team Cymru IP-to-ASN [Cym18], MaxMind Geolocation [Max18], Alexa ranking [ale], ipcat list [Gal19], and Domaintools WHOIS [Dom18] with custom code written in Python ($480$ lines of code).
3. For **fingerprinting**, we relied on the Nessus scanner and commercial plugins [Sec18], sqlmap [sql], and Acunetix [Acu18]. We used Nessus plugins and custom Python code (1010 lines of code) to perform the vulnerability analysis.

# Overall Experimental Results

| **Category** | **# Mobile Apps** | **Vulnerabilties** | | | | |
|---|---|---|---|---|---|---|
| | | **#** $OS$ | **#** $SS$ | **#** $AS$ | **#** $CS$ | **Total** |
| Books & Reference | 332 | 15 | 49 | 55 | 71 | 190 |
| Business | 145 | 5 | 22 | 10 | 37 | 74 |
| Entertainment | 1,177 | 36 | 108 | 158 | 170 | 472 |
| Games | 1,283 | 34 | 81 | 147 | 106 | 368 |
| Lifestyle | 363 | 20 | 50 | 79 | 72 | 221 |
| Misc | 199 | 6 | 21 | 45 | 46 | 118 |
| Tools | 792 | 19 | 84 | 184 | 115 | 402 |
| Video & Audio | 689 | 24 | 46 | 89 | 98 | 257 |
| **Total** | 4,980 | 121 | 356 | 655 | 506 | 1,638 |

Table: An overview of the vulnerable mobile apps per category along with the *raw* counts of vulnerabilities: Operating System ($OS$), Software Services ($SS$), Application Software ($AS$), Communication Services ($CS$)

# Overall Experimental Results

| Category | # Mobile Apps | Vulnerabilties | | | | |
|---|---|---|---|---|---|---|
| | | # $OS$ | # $SS$ | # $AS$ | # $CS$ | Total |
| Books & Reference | 332 | 15 | 49 | 55 | 71 | 190 |
| Business | 145 | 5 | 22 | 10 | 37 | 74 |
| Entertainment | 1,177 | 36 | 108 | 158 | 170 | 472 |
| Games | 1,283 | 34 | 81 | 147 | 106 | 368 |
| Lifestyle | 363 | 20 | 50 | 79 | 72 | 221 |
| Misc | 199 | 6 | 21 | 45 | 46 | 118 |
| Tools | 792 | 19 | 84 | 184 | 115 | 402 |
| Video & Audio | 689 | 24 | 46 | 89 | 98 | 257 |
| **Total** | 4,980 | 121 | 356 | 655 | 506 | 1,638 |

Table: An overview of the vulnerable mobile apps per category along with the *raw* counts of vulnerabilities: Operating System ($OS$), Software Services ($SS$), Application Software ($AS$), Communication Services ($CS$)

# Other Statistics

| Comp. | Vulnerability (Top 3) | #Apps |
|---|---|---|
| $OS$ | Expired Lifecycle for Linux OS (various) | 124 |
| | Windows Server RCE (MS15-034) | 64 |
| | Expired Lifecycle for Windows Server | 9 |
| $SS$ | Vulnerable PHP Version | 357 |
| | Expired Lifecycle for Web Server (various) | 181 |
| | Vulnerable Apache Version | 76 |
| $AS$ | XSS (various) | 262 |
| | SQLi (various) | 160 |
| | XXE (various) | 86 |
| $CS$ | Support for Vulnerable SSL Version 2 and 3 | 997 |
| | OpenSSH Bypass (CVE-2015-5600) | 16 |
| | Vulnerable OpenSSL (various) | 15 |

Table: The top three vulnerabilities found per cloud layer along with the number of affected mobile apps.

# Other Statistics

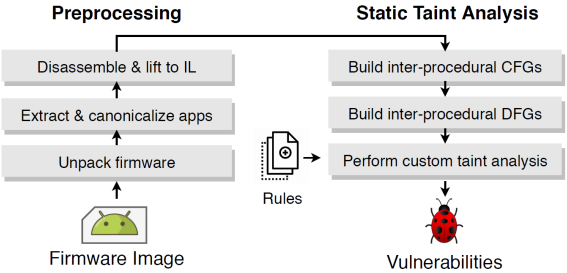
| # Installs | # Apps | # SQLi | # XSS | # XXE |
|---|---|---|---|---|
| 1B | 5 | 0 | 0 | 0 |
| 500M | 11 | 0 | 0 | 0 |
| 100M | 116 | 0 | 3 | 1 |
| 50M | 131 | 4 | 10 | 5 |
| 10M | 1,049 | 25 | 85 | 15 |
| 5M | 1,047 | 54 | 89 | 9 |
| 1M | 2,621 | 132 | 316 | 17 |

Table: The number of *0-day* vulnerabilities found per install category.

# Outline


1. Introduction
2. FirmScope: Privilege-Escalation in Firmware
3. LeakScope: Leakage in Clouds
4. SkyWalker: n-day/0-day in Servers
5. Summary

# FirmScope [USENIX Security 2020]



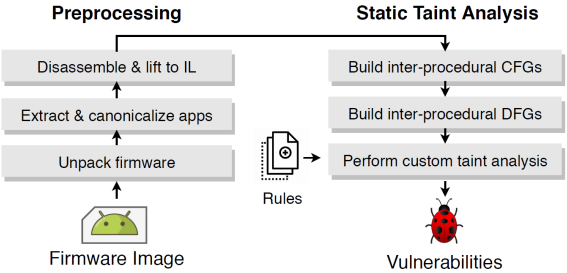

## FirmScope

- A **fully automated** tool to detect privildge escalation vulnerabilities
- Can also detect **PII leakage**

## Experimental Result w/ 2K firmware

- 331,342 pre-installed apps
- 3,483 vulnerabilities
- 147 confirmed CVEs

# FirmScope [USENIX Security 2020]




## FirmScope

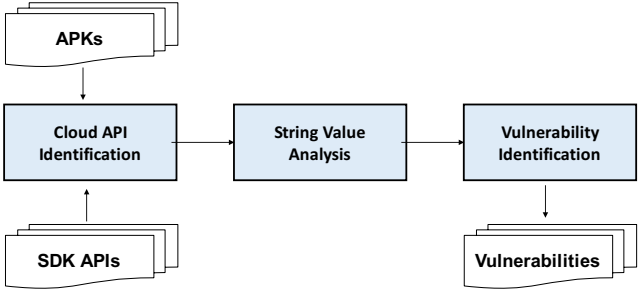
- A **fully automated** tool to detect privildge escalation vulnerabilities
- Can also detect **PII leakage**

## Experimental Result w/ 2K firmware

- 331,342 pre-installed apps
- 3,483 vulnerabilities
- 147 confirmed CVEs

Alternatives: `github.com/secure-software-engineering/FlowDroid`, `github.com/MIT-PAC/droidsafe-src`

# LeakScope [Oakland'19]



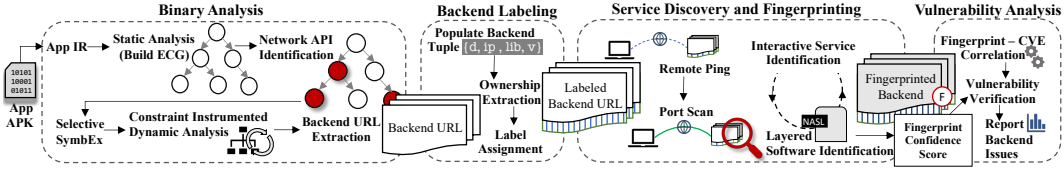

## LeakScope

- A static analysis to identify server side data leakage vulnerabilities
- It performs cloud API identification, string value analysis to identify the vulnerabilities

## Experimental Result w/ 100K apps

- 15,098 apps' cloud servers are vulnerable
- 200 Azure, 1,600 AWS, 13,200 Firebase
- Source code is available at `github.com/OSUSecLab/LeakScope`

# SkyWalker [USENIX Security 2019]




## SkyWalker

- A **fully automated** mobile app backend analysis framework
- Can be used to **vet various security vulnerabilities in mobile backends**

## Experimental Result w/ $4,980$ apps

- 983 N-day and 655 0-day spanning across the software layers (OS, software services, communication, and web apps) of cloud backends
- `https://MobileBackend.vet`

# Thank You

## An App Centric Approach for Vulnerability Discovery in Mobile Platforms

**Dr. Zhiqiang Lin**
zlin@cse.ohio-state.edu
Joint work w/ Omar Alrawi, Mohamed Elsabagh, Ruian Duan, Ryan Johnson, Ranjita Kasturi, Brendan Saltaformaggio, Angelos Starvro, Yinqian Zhang, Qingchuan Zhao, Chaoshun Zuo

04/21/2020

- Acunetix, *Audit Your Web Security with Acunetix Vulnerability Scanner*, https://www.acunetix.com/vulnerability-scanner/, 2018.
- *Alexa - actionable analytics for the web*, http://www.alexa.com/.
- Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel, *Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps*.
- Team Cymru, *IP TO ASN MAPPING*, http://www.team-cymru.com/IP-ASN-mapping.html, 2018.
- DomainTools, *About Us*, https://www.domaintools.com/company/, 2018.
- Nick Galbreath, *Categorization of IP Addresses*, https://github.com/client9/ipcat, 2019.
- MaxMind, *About MaxMind*, https://www.maxmind.com/en/company, 2018.

- Tenable Security, *Nessus Professional*, https://www.tenable.com/products/nessus/nessus-professional, 2018.
- *A framework for analyzing and transforming java and android apps*, https://sable.github.io/soot/.
- *Automatic sql injection and database takeover tool*, http://sqlmap.org/.
- *Xposed module repository*, http://repo.xposed.info/.
- Chaoshun Zuo and Zhiqiang Lin, *Smartgen: Exposing server urls of mobile apps with selective symbolic execution*, Proceedings of the 26th World Wide Web Conference (Perth, Australia), April 2017.
- Yunhui Zheng, Xiangyu Zhang, and Vijay Ganesh, *Z3-str: a z3-based string solver for web application analysis*.