Automatically Uncovering Vulnerabilities in Clouds From Mobile Apps

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

Nov 7th, 2019

Mobile Apps Have Significantly Changed Our Lives

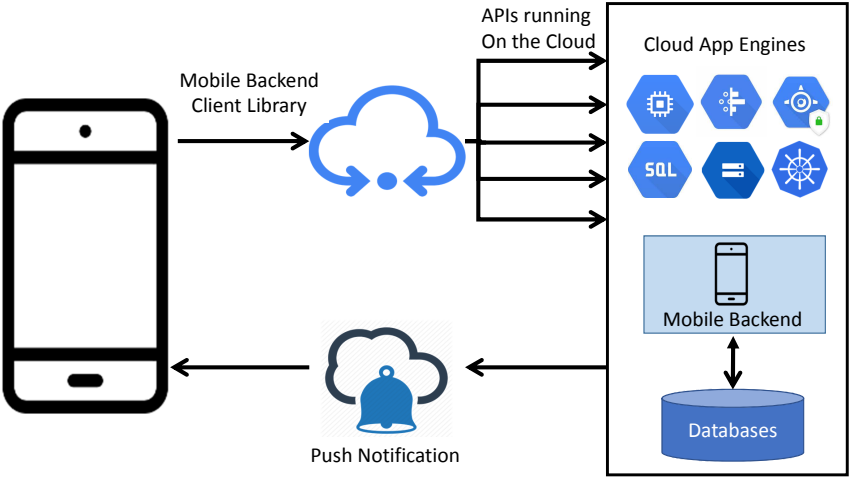
Source: cloudxtension.com

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. ...

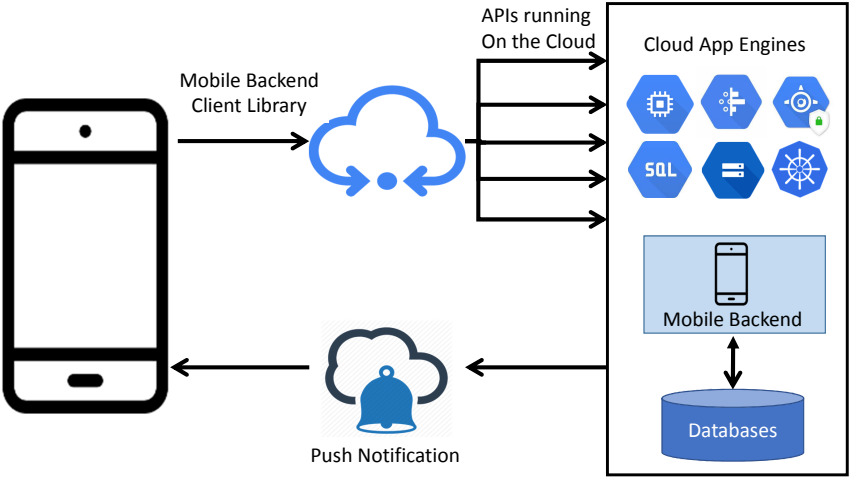
Mobile Apps, Web, and Clouds

Mobile Apps, Web, and Clouds

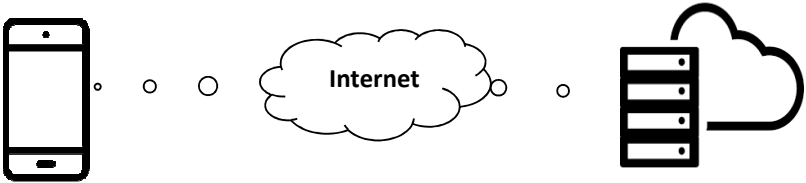
The Mobile Backend as a Service (mBaaS) Clouds



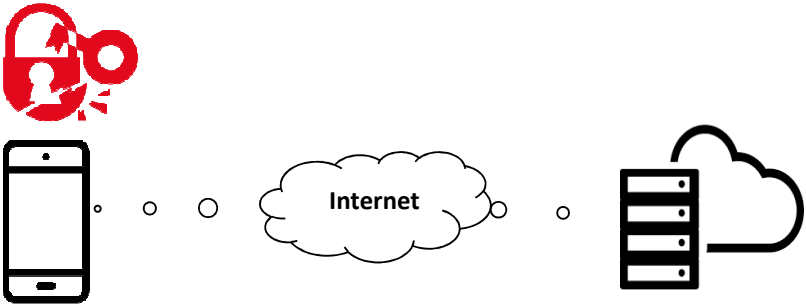
The Mobile Backend as a Service (mBaaS) Clouds



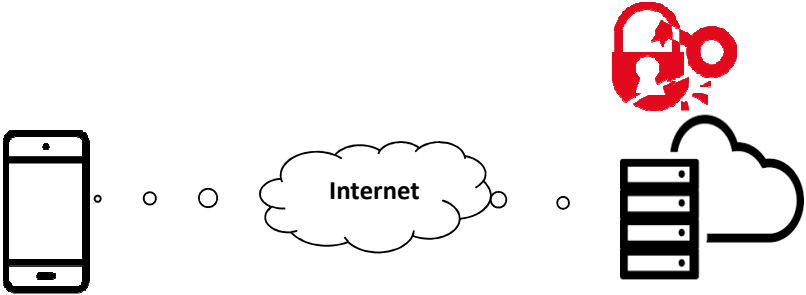
Our Focal Point: Uncovering Server Side Vulnerabilities



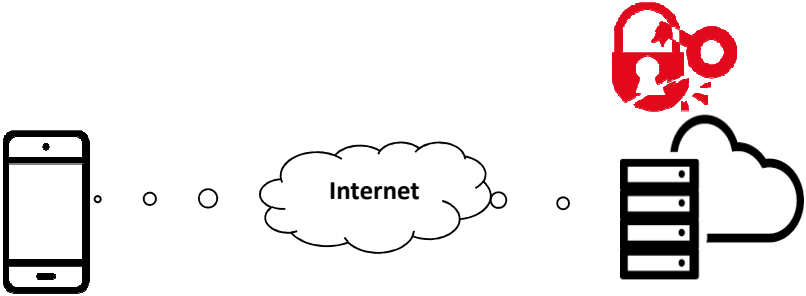
Our Focal Point: Uncovering Server Side Vulnerabilities



Our Focal Point: Uncovering Server Side Vulnerabilities

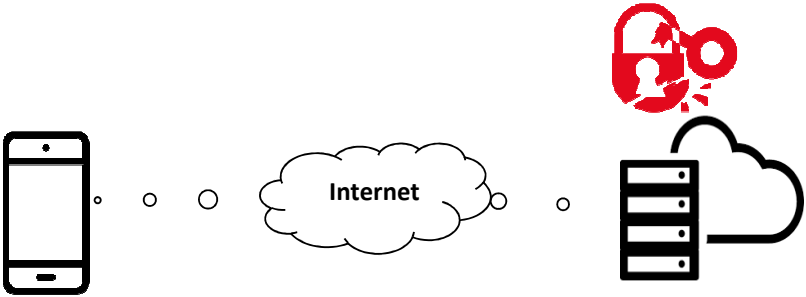


Our Focal Point: Uncovering Server Side Vulnerabilities



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

Our Focal Point: Uncovering Server Side Vulnerabilities



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: 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 I: 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 I: 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 I: 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

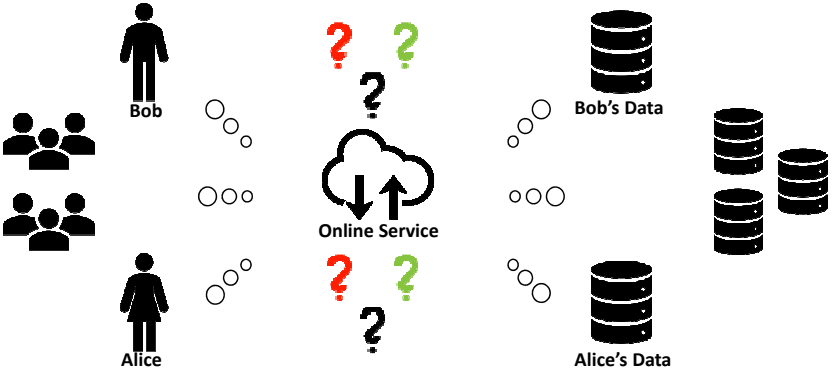
Amazon server

by Selena Larson @selenalarson

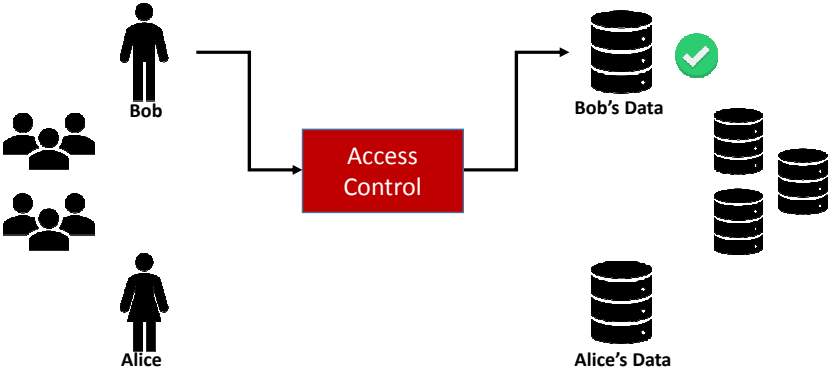
November 17, 2017: 12:03 PM ET

Recommend 9

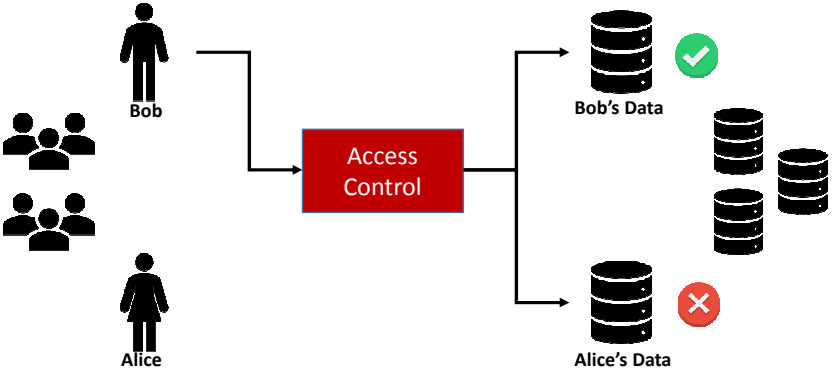
Vulnerability I: Data Leakage is Essentially an Access Control Problem



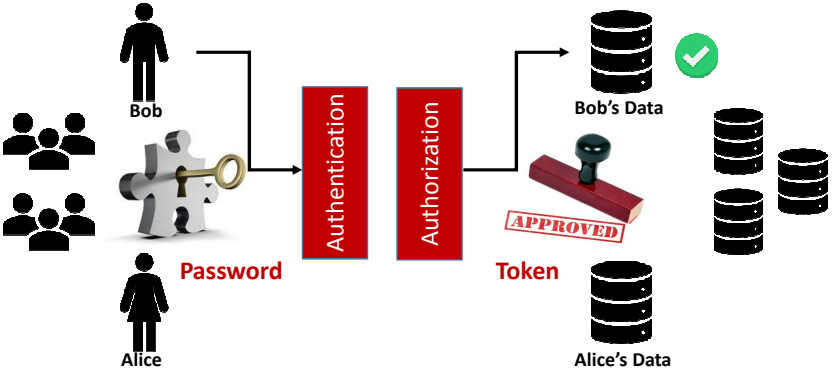
Vulnerability I: Data Leakage is Essentially an Access Control Problem



Vulnerability I: Data Leakage is Essentially an Access Control Problem



Vulnerability I: Data Leakage is Essentially an Access Control Problem



Vulnerability II: Traditional Web Vulnerabilities and Beyond

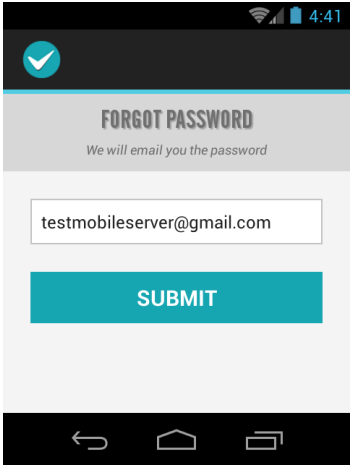
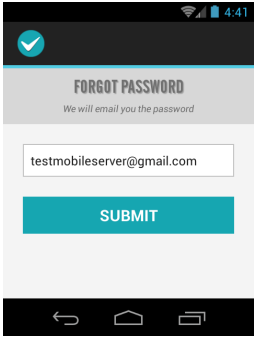


Figure: The password reset activity of ShopClues (between 10 million and 50 million installs).

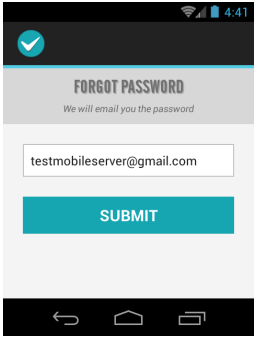
Vulnerability II: Traditional Web Vulnerabilities and Beyond




```
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 II: 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 Analysis

1. **SMV-Hunter: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps**. In NDSS 2014
2. **You Shouldn't Collect My Secrets: Thwarting Sensitive Keystroke Leakage in Mobile IME Apps**. In USENIX Security 2015
3. **AutoForge: Automatic Forgery of Cryptographically Consistent Messages to Identify Security Vulnerabilities in Mobile Services**. In NDSS 2016
4. **SmartGen: Exposing Server URLs of Mobile Apps With Selective Symbolic Execution**. In WWW 2017
5. **AuthScope: Towards Automatic Identification of Vulnerable Authorizations in Online Services**. In CCS 2017
6. **The Unexpected Danger of UX Features: A Case of Sensitive Data Leakage of Drivers in Ride-Hailing Services**. In NDSS 2019.
7. **LeakScope: Why Does Your Data Leak? Uncovering the Data Leakage in Cloud from Mobile Apps**. In Oakland 2019
8. **The Betrayal At Cloud City: An Empirical Analysis Of Cloud-Based Mobile Backends**, In USENIX Security 2019
9. **Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps**, In CCS 2019

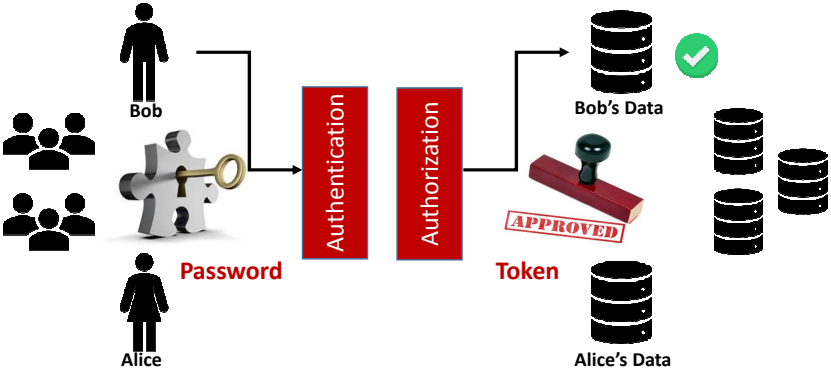
Our Efforts on **Vulnerability Discovery** w/ Mobile App Analysis

1. **SMV-Hunter: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps**. In NDSS 2014
2. **You Shouldn't Collect My Secrets: Thwarting Sensitive Keystroke Leakage in Mobile IME Apps**. In USENIX Security 2015
3. **AutoForge: Automatic Forgery of Cryptographically Consistent Messages to Identify Security Vulnerabilities in Mobile Services**. In NDSS 2016
4. **SmartGen: Exposing Server URLs of Mobile Apps With Selective Symbolic Execution**. In WWW 2017
5. **AuthScope: Towards Automatic Identification of Vulnerable Authorizations in Online Services**. In CCS 2017
6. **The Unexpected Danger of UX Features: A Case of Sensitive Data Leakage of Drivers in Ride-Hailing Services**. In NDSS 2019.
7. **LeakScope: Why Does Your Data Leak? Uncovering the Data Leakage in Cloud from Mobile Apps**. In Oakland 2019
8. **The Betrayal At Cloud City: An Empirical Analysis Of Cloud-Based Mobile Backends**, In USENIX Security 2019
9. **Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps**, In CCS 2019

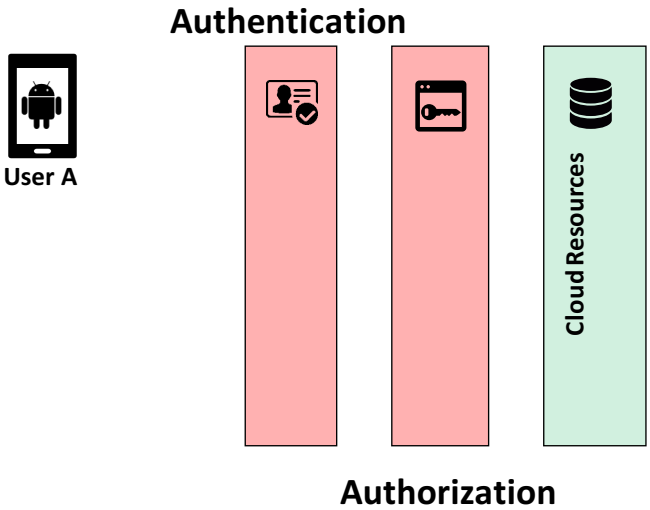
Our Efforts on **Vulnerability Discovery** w/ Mobile App Analysis

1. ~~**SMV-Hunter: Large Scale, Automated Detection of SSL/TLS Man-in-the-Middle Vulnerabilities in Android Apps**. In NDSS 2014~~
2. ~~**You Shouldn't Collect My Secrets: Thwarting Sensitive Keystroke Leakage in Mobile IME Apps**. In USENIX Security 2015~~
3. ~~**AutoForge: Automatic Forgery of Cryptographically Consistent Messages to Identify Security Vulnerabilities in Mobile Services**. In NDSS 2016~~
4. **SmartGen: Exposing Server URLs of Mobile Apps With Selective Symbolic Execution**. In WWW 2017
5. **AuthScope: Towards Automatic Identification of Vulnerable Authorizations in Online Services**. In CCS 2017
6. ~~**The Unexpected Danger of UX Features: A Case of Sensitive Data Leakage of Drivers in Ride-Hailing Services**. In NDSS 2019.~~
7. **LeakScope: Why Does Your Data Leak? Uncovering the Data Leakage in Cloud from Mobile Apps**. In Oakland 2019
8. **The Betrayal At Cloud City: An Empirical Analysis Of Cloud-Based Mobile Backends**, In USENIX Security 2019
9. ~~**Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps**, In CCS 2019~~

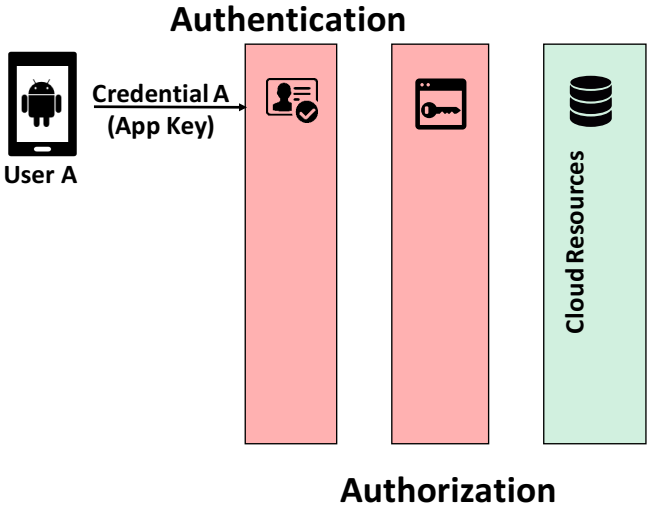
Vulnerability I: Data Leakage is Essentially an Access Control Problem



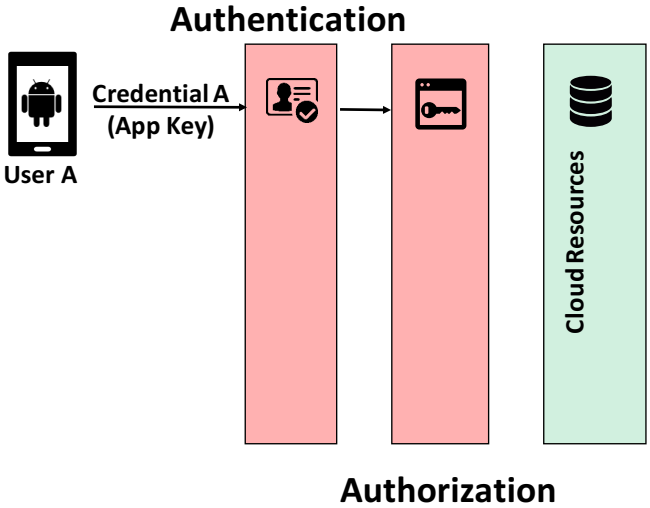
How Do Mobile Apps and mBaaS Cloud Communicate



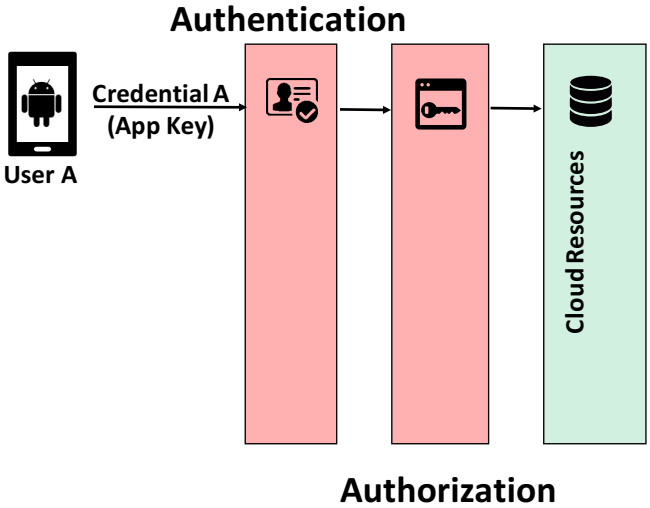
How Do Mobile Apps and mBaaS Cloud Communicate



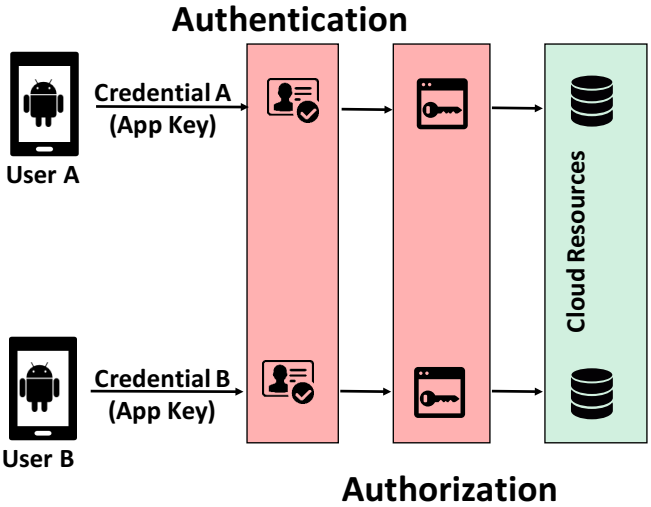
How Do Mobile Apps and mBaaS Cloud Communicate



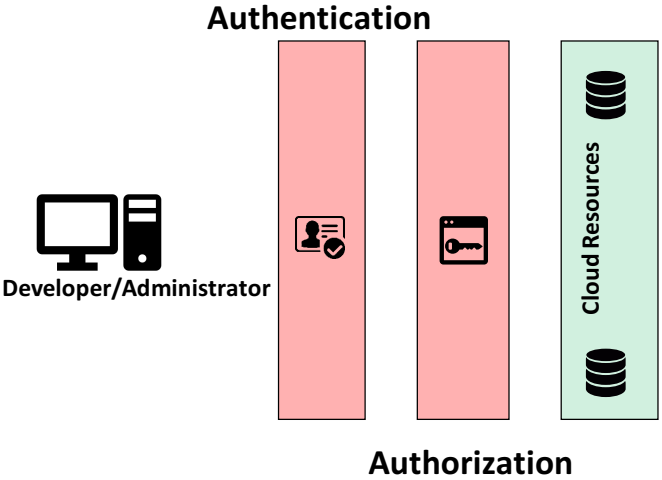
How Do Mobile Apps and mBaaS Cloud Communicate



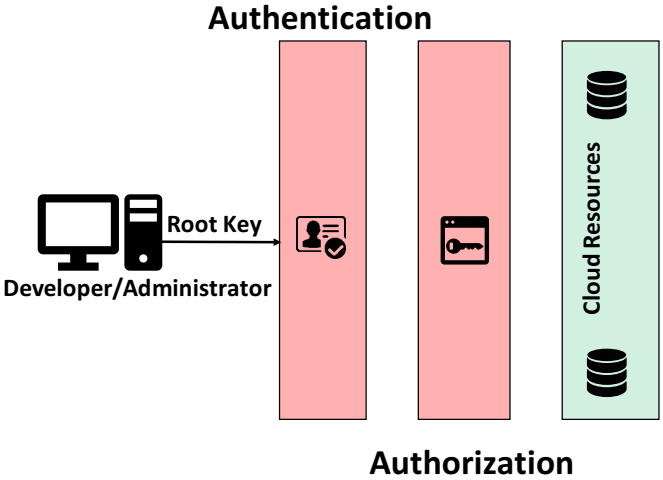
How Do Mobile Apps and mBaaS Cloud Communicate



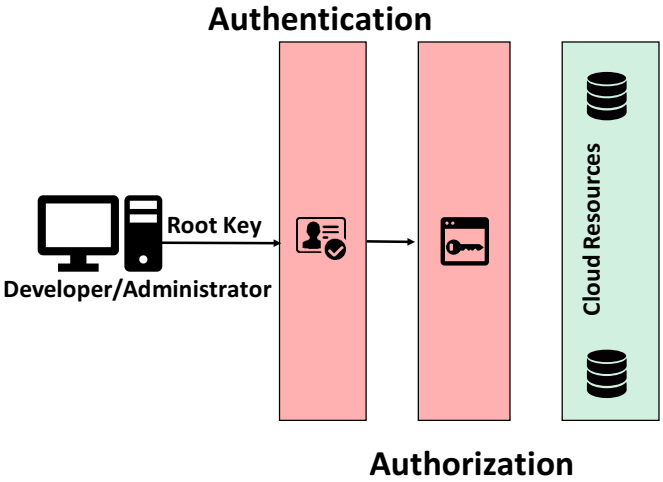
How Do Mobile Apps and mBaaS Cloud Communicate



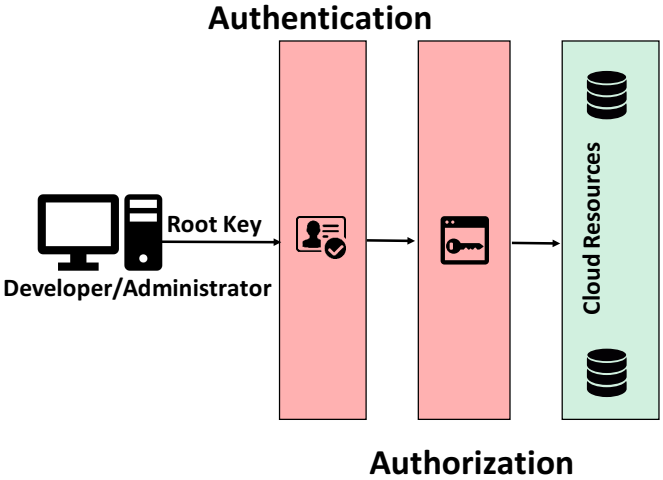
How Do Mobile Apps and mBaaS Cloud Communicate



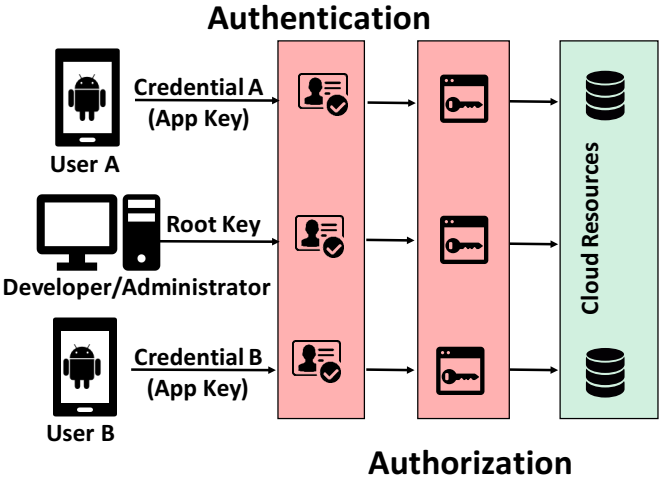
How Do Mobile Apps and mBaaS Cloud Communicate



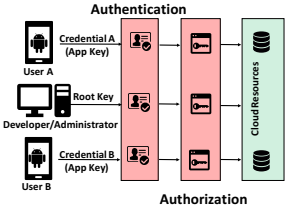
How Do Mobile Apps and mBaaS Cloud Communicate



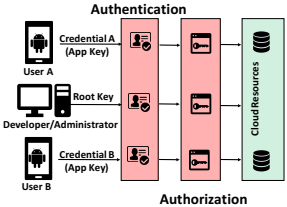
How Do Mobile Apps and mBaaS Cloud Communicate



Our Discovery of Why Cloud Data was Leaked

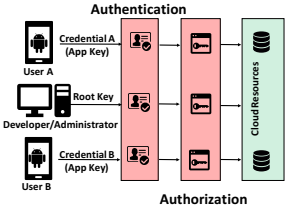


Our Discovery of Why Cloud Data was Leaked



AWS Firebase Azure

Our Discovery of Why Cloud Data was Leaked



AWS Firebase Azure

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= Default**Listen**SharedAccessSignature; SharedAccessKey=*
	Full Access Key	Endpoint=sb://*.servicebus.windows.net/; SharedAccessKeyName= Default**Full**SharedAccessSignature; 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

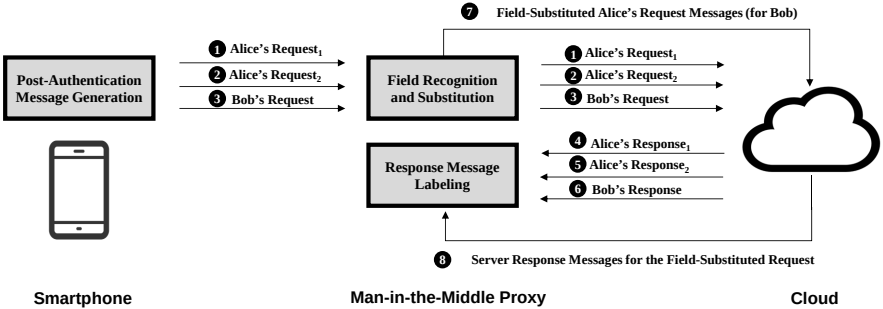
```
GET /api/v1//users/21691/notifications?in_app_token=e67315b35aa3
8d4ac8cac3cd9c7f88ae7f576d373f HTTP/1.1
Host: api.w****.com
Connection: close
```

```
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"}]
```

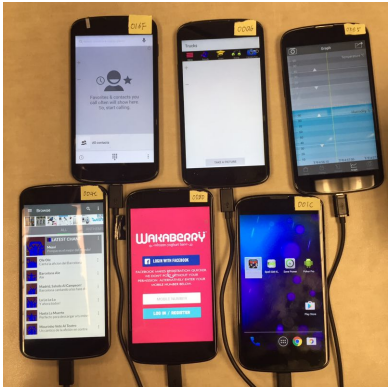
Alice reads Bob's notifications

Introducing AuthScope [CCS'17]



Experiment Setup and Overall Result

- Top 10% Apps (200,000) of Google Play Store (March 2017)
- 33,950 contains Facebook library
- 4,838 Facebook Login



Experiment Setup and Overall Result

Item	**Value**
Σ # Apps	$4,838$
Σ Time to perform the test (hours)	562.4
Σ # Request messages	$3,220,886$
Σ # HTTP Messages	$178,539$
Σ Size of the messages (G-bytes)	59.2
Σ Time of activity exploration before authentication (hours)	169.9
Σ # Explored activities after authentication	$20,704$
Σ # Mutated fields	$57,736$
Σ # Vulnerable apps	306

Table: Overall Experimental Result.

Experiment Setup and Overall Result

Item	Value
Σ # Apps	$4,838$
Σ Time to perform the test (hours)	562.4
Σ # Request messages	$3,220,886$
Σ # HTTP Messages	$178,539$
Σ Size of the messages (G-bytes)	59.2
Σ Time of activity exploration before authentication (hours)	169.9
Σ # Explored activities after authentication	$20,704$
Σ # Mutated fields	$57,736$
Σ # Vulnerable apps	306

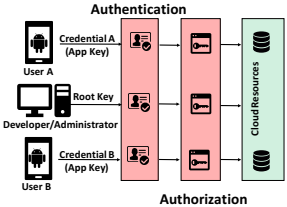
Table: Overall Experimental Result.

Experiment Setup and Overall Result

Item	**Value**
Σ # Apps	$4,838$
Σ Time to perform the test (hours)	562.4
Σ # Request messages	$3,220,886$
Σ # HTTP Messages	$178,539$
Σ Size of the messages (G-bytes)	59.2
Σ Time of activity exploration before authentication (hours)	169.9
Σ # Explored activities after authentication	$20,704$
Σ # Mutated fields	$57,736$
Σ # Vulnerable apps	306

Table: Overall Experimental Result.

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

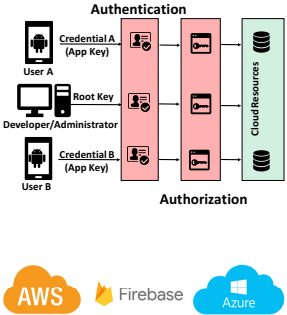


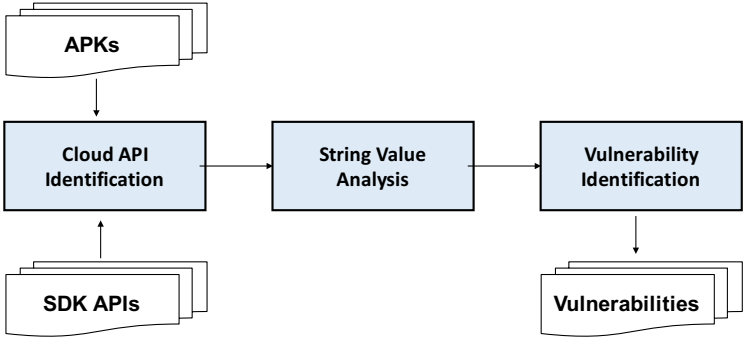
AWS Firebase Azure

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**)

Introducing 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

master (#65)

Browse files

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

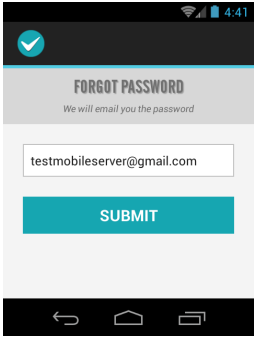
```
@@ -24,6 +24,13 @@
 24  24      * MODIFY THIS!
 25  25      *
 26  26      * Stores the storage connection string.
     27  +   * Only use Shared Key authentication (Account Key) for testing purposes!
     28  +   * Your account name and account key, which give full read/write access to the associated 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."*

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



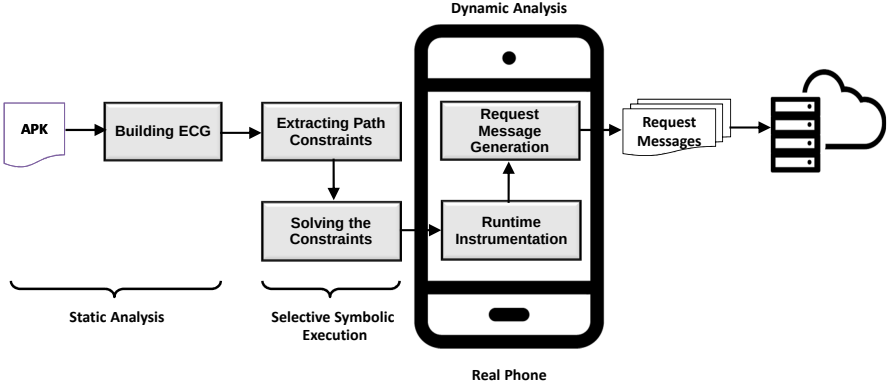

```
1 package com.shopclues;
2
3 class y implements View$OnClickListener {
4     EditText b;
5     ...
6     public void onClick(View arg5) {
7         String v0 = this.b.getText().toString().trim();
8         if(v0.equalsIgnoreCase("")) {
9             Toast.makeText(this.a, "Email Id should not be
              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

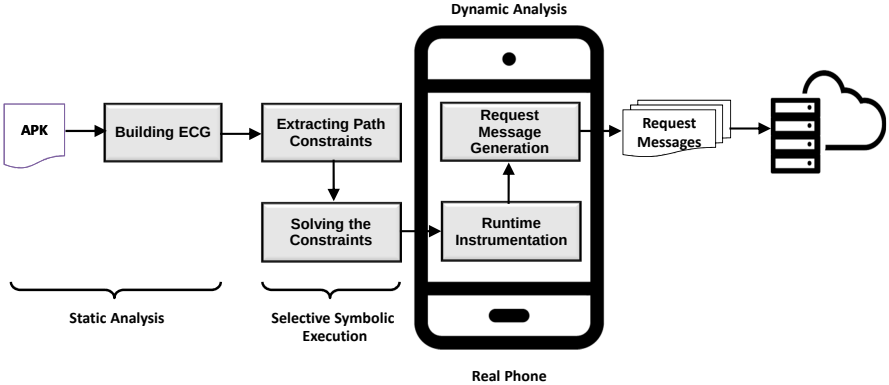
1. Two text-box's inputs need to be equivalent
2. The "age" needs to be greater than 18
3. A "zip code" needs to be a five digit sequence
4. A "phone number" needs to be a phone number
5. A file name extension needs to be some type (e.g., `jpg`)
6. ...

Introducing SmartGen [WWW'17]



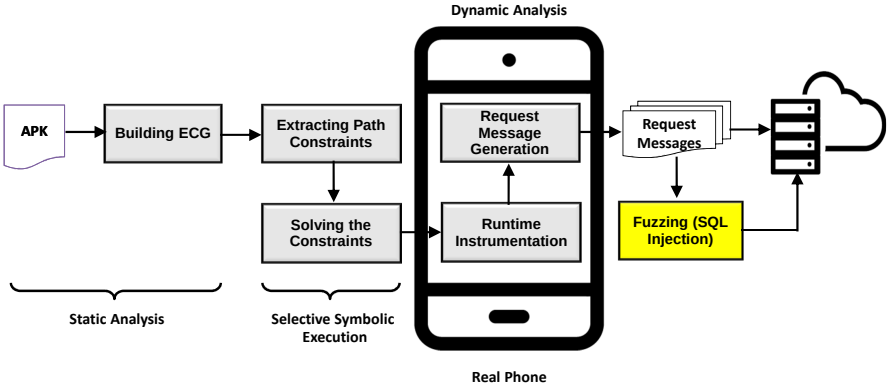
- Automated
- Systematic
- Scalable

Introducing SmartGen [WWW'17]



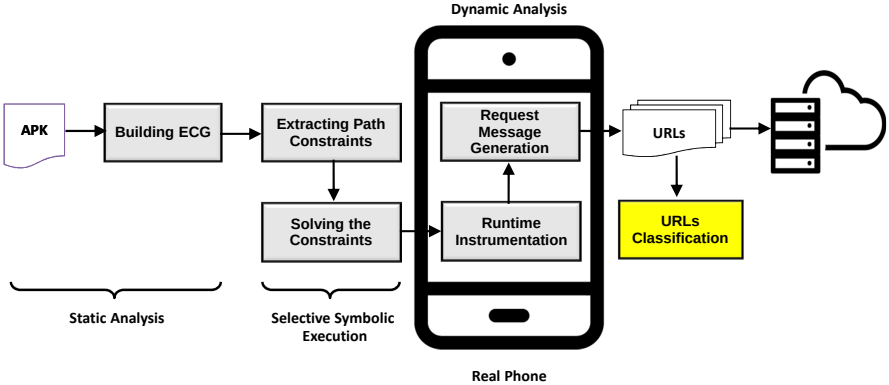
- Static analysis
- Selective symbolic execution
- Dynamic analysis

SQL Injection Fuzzing



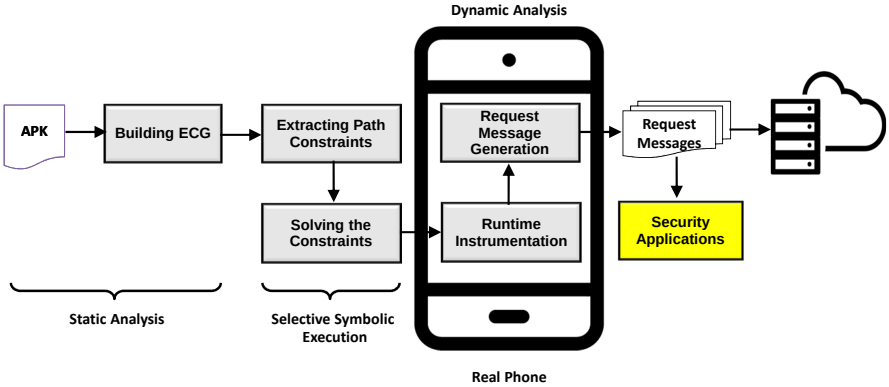
- "SELECT PG_SLEEP(5);", "SELECT PG_SLEEP(10);"
- "';WAITFOR DELAY '0:0:5'--"
- ";SELECT COUNT(*) FROM SYSIBM.SYSTABLES"

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

Introducing SkyWalker [USENIX Security 2019]

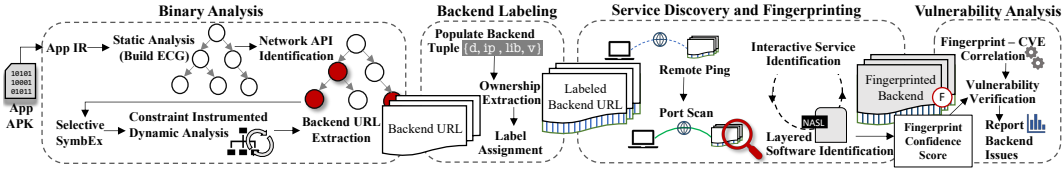


Figure: SkyWalker Overview. Phase 1 (Binary Analysis) extracts backend URLs through a dynamic binary instrumentation technique. Phase 2 labels backends into first-party, third-party, and hybrid based on IP and public available information. Phase 3 discovers and fingerprints the backend services to collect 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 [Sec18a], 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.

Recommendations for Mobile App Developers (4D)

1. First, developers should **delegate** as much of the backend functionality to reputable third-party backends and minimize the number of features and functions their backend needs to support.
2. Second, developers should **dedicate** personnel to manage and maintain their backends including the routine maintenance of OS, SS and CS, and timely fixes of known vulnerabilities affecting their cloud backends and mobile apps using patching tools [Sec18b, AK, DBJ+].
3. Third, developers should **develop** an audit plan and a mitigation plan and be familiar with it to execute during an incident or vulnerability disclosure.
4. Finally, developers should utilize **defense** tools like web app firewalls (WAF), DDoS mitigation, and crawler/scanner blockers to protect from internet scanners, DDoS threats, and web app attacks (SQLi, authentication bypass, etc.).

Related Work

1. **Protocol Reverse Engineering**. A large body of research focusing on protocol reverse engineering [Bed, MLK+06, CKW07, CS07, WMKK08, LJXZ08, MWKK09, CPKS09]
2. **Dynamic Analysis**. Monkey [mon17] automatically executes and randomly navigates an app. AppsPlayground [RCE13] and SMV-Hunter [SSG+14] more intelligent. A3E [AN13], a targeted exploration of mobile apps. DynoDroid [MTN13] instruments the Android framework and uses adb to monitor UI interaction and generate UI events.

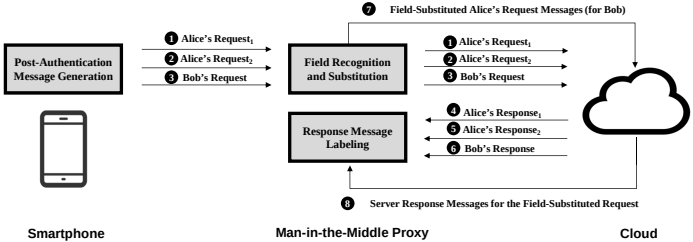
Related Work

1. **Vulnerability Discovery w/ Mobile Apps**:
   - Client Side: TaintDroid [EGC+10], PiOS [EKKV11], CHEX [LLW+12], SMV-Hunter [SSG+14].
   - Server Side: AutoForge [ZWWL16], SmartGen [ZL17], AuthScope [ZZL17], and LeakScope [ZLZ19].
2. **Misconfiguration Vulnerability Identification**: FIREMAN [YMS+06], ConfErr [KUC08], ConfAid [AF10], SPEX [XZH+13].

Related Work

1. **Measurement Studies**
   - Durumeric et al. [DKBH] conducted a comprehensive internet-wide study of the HTTPS certificate ecosystem. Later, Durumeric et al. [DLK+] carried out a similar internet-wide study for the Heartbleed vulnerability [hea].
   - Perez-Botero et al. [PBSL] presented an in-depth study characterizing hypervisor vulnerabilities in cloud services.
   - Zuo et al. [ZL17] proposed a system to identify mobile app URLs and examine their reputation with public blacklists to detect malicious apps.

AuthScope [CCS'17]



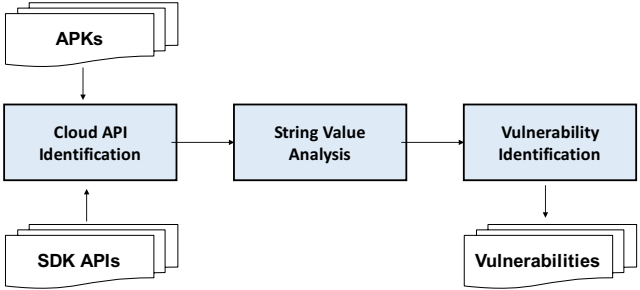
AuthScope

- A **fully automated** automated system to analyze mobile apps, and reveal post-authentication messages
- Can be used to **test various post authentication vulnerabilities**

Experimental Result w/ 5K apps

- Each app has large number of (e.g., 1,000,000+) installs
- There were close to 300 apps whos servers contain vulnerable authorization services

LeakScope [Oakland'19]



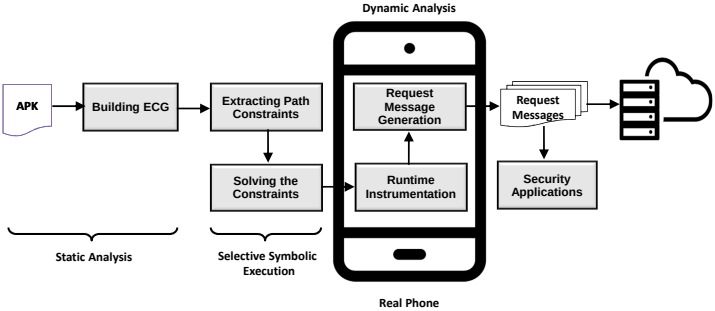
LeakScope

- 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
- Responsible disclosures were made to the cloud providers

SmartGen [WWW'17]



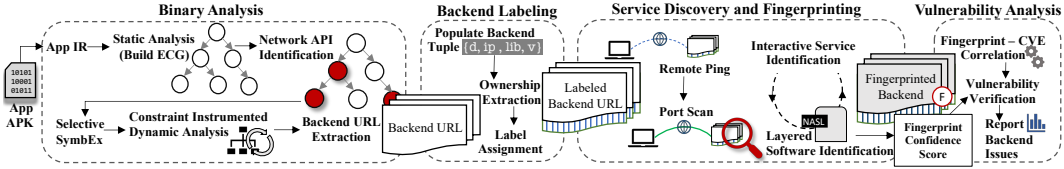
SmartGen

- A **fully automated** mobile app execution framework via **symbolic execution**
- Can be used to **test various security vulnerabilities in mobile systems**

Experimental Result w/ $5,000$ apps

- 340 SQL Injection Vulnerabilities
- These apps actually talk to $2,071$ phishing sites, $3,722$ malware sites, and $3,228$ malicious sites

SkyWalker [USENIX Security 2019]



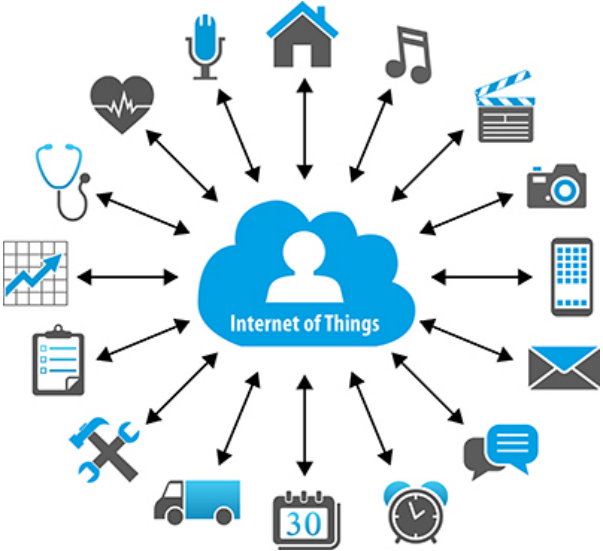
SkyWalker

- 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`

Furure Works: Security in IoT (e.g., BLEScope [CCS'19])



Furure Works: Security in IoT (e.g., BLEScope [CCS'19])

Furure Works: Security in IoT (e.g., BLEScope [CCS'19])

Thank You

Automatically Uncovering Vulnerabilities in Clouds From Mobile Apps

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

Nov 7th, 2019

References I

Acunetix, *Audit Your Web Security with Acunetix Vulnerability Scanner*, https://www.acunetix.com/vulnerability-scanner/, 2018.

Mona Attariyan and Jason Flinn, *Automating configuration troubleshooting with dynamic information flow analysis*, Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (Vancouver, BC, Canada), OSDI'10, 2010, pp. 237–250.

Jeff Arnold and M Frans Kaashoek, *Ksplice: Automatic rebootless kernel updates*.

Alexa - actionable analytics for the web, http://www.alexa.com/.

Tanzirul Azim and Iulian Neamtiu, *Targeted and depth-first exploration for systematic testing of android apps*, Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications (New York, NY, USA), OOPSLA '13, ACM, 2013, pp. 641–660.

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*.

Marshall Beddoe, *The protocol informatics project*, http://www.4tphi.net/~awalters/PI/PI.html.

Weidong Cui, Jayanthkumar Kannan, and Helen J. Wang, *Discoverer: Automatic protocol reverse engineering from network traces*, Proceedings of the 16th USENIX Security Symposium (Security'07) (Boston, MA), August 2007.

Juan Caballero, Pongsin Poosankam, Christian Kreibich, and Dawn Song, *Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering*, Proceedings of the 16th ACM Conference on Computer and and Communications Security (CCS'09) (Chicago, Illinois, USA), 2009, pp. 621–634.

References II

Juan Caballero and Dawn Song, *Polyglot: Automatic extraction of protocol format using dynamic binary analysis*, Proceedings of the 14th ACM Conference on Computer and and Communications Security (CCS'07) (Alexandria, Virginia, USA), 2007, pp. 317–329.

Team Cymru, *IP TO ASN MAPPING*, `http://www.team-cymru.com/IP-ASN-mapping.html`, 2018.

Ruian Duan, Ashish Bijlani, Yang Ji, Omar Alrawi, Yiyuan Xiong, Moses Ike, Brendan Saltaformaggio, and Wenke Lee, *Automating patching of vulnerable open-source software versions in application binaries*.

Zakir Durumeric, James Kasten, Michael Bailey, and J Alex Halderman, *Analysis of the https certificate ecosystem*.

Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, et al., *The matter of heartbleed*.

DomainTools, *About Us*, `https://www.domaintools.com/company/`, 2018.

W. Enck, P. Gilbert, B.G. Chun, L.P. Cox, J. Jung, P. McDaniel, and A.N. Sheth, *TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones*, OSDI, 2010.

M. Egele, C. Kruegel, E. Kirda, and G. Vigna, *Pios: Detecting privacy leaks in ios applications*, NDSS, 2011.

Nick Galbreath, *Categorization of IP Addresses*, `https://github.com/client9/ipcat`, 2019.

Heartbleed, `http://heartbleed.com/`.

References III

Lorenzo Keller, Prasang Upadhyaya, and George Candea, *Conferr: A tool for assessing resilience to human configuration errors*, Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on, IEEE, 2008, pp. 157–166.

Zhiqiang Lin, Xuxian Jiang, Dongyan Xu, and Xiangyu Zhang, *Automatic protocol format reverse engineering through context-aware monitored execution*, Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08) (San Diego, CA), February 2008.

Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang, *Chex: statically vetting android apps for component hijacking vulnerabilities*, Proceedings of the 2012 ACM conference on Computer and communications security, ACM, 2012, pp. 229–240.

MaxMind, *About MaxMind*, `https://www.maxmind.com/en/company`, 2018.

Justin Ma, Kirill Levchenko, Christian Kreibich, Stefan Savage, and Geoffrey M. Voelker, *Unexpected means of protocol inference*, Proceedings of the 6th ACM SIGCOMM on Internet measurement (IMC'06) (Rio de Janeriro, Brazil), ACM Press, 2006, pp. 313–326.

Ui/application exerciser monkey, `https://developer.android.com/tools/help/monkey.html`, 2017.

Aravind Machiry, Rohan Tahiliani, and Mayur Naik, *Dynodroid: An input generation system for android apps*, Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM, 2013, pp. 224–234.

Paolo Milani Comparetti, Gilbert Wondracek, Christopher Kruegel, and Engin Kirda, *Prospex: Protocol Specification Extraction*, IEEE Symposium on Security & Privacy (Oakland, CA), 2009, pp. 110–125.

Diego Perez-Botero, Jakub Szefer, and Ruby B. Lee, *Characterizing hypervisor vulnerabilities in cloud computing servers*.

References IV

Vaibhav Rastogi, Yan Chen, and William Enck, *Appsplayground: Automatic security analysis of smartphone applications*, Proceedings of the Third ACM Conference on Data and Application Security and Privacy (New York, NY, USA), CODASPY '13, ACM, 2013, pp. 209–220.

Tenable Security, *Nessus Professional*, https://www.tenable.com/products/nessus/nessus-professional, 2018.

SecurityFTW - cs-suite, *Cloud Security Suite - One stop tool for auditing the security posture of AWS/GCP/Azure infrastructure.*, https://github.com/SecurityFTW/cs-suite, 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/.

David Sounthiraraj, Justin Sahs, Garrett Greenwood, Zhiqiang Lin, and Latifur Khan, *Smv-hunter: Large scale, automated detection of ssl/tls man-in-the-middle vulnerabilities in android apps*, Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'14) (San Diego, CA), February 2014.

Gilbert Wondracek, Paolo Milani, Christopher Kruegel, and Engin Kirda, *Automatic network protocol analysis*, Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08) (San Diego, CA), February 2008.

Xposed module repository, http://repo.xposed.info/.

Tianyin Xu, Jiaqi Zhang, Peng Huang, Jing Zheng, Tianwei Sheng, Ding Yuan, Yuanyuan Zhou, and Shankar Pasupathy, *Do not blame users for misconfigurations*, Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (Farminton, Pennsylvania), SOSP '13, 2013, pp. 244–259.

References V

Lihua Yuan, Jianning Mai, Zhendong Su, Hao Chen, Chen-Nee Chuah, and Prasant Mohapatra, *Fireman: A toolkit for firewall modeling and analysis*, Proceedings of the 2006 IEEE Symposium on Security and Privacy, SP'06, 2006, pp. 199–213.

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.

Chaoshun Zuo, Zhiqiang Lin, and Yinqian Zhang, *Why does your data leak? uncovering the data leakage in cloud from mobile apps*, Proceedings of the 2019 IEEE Symposium on Security and Privacy (San Francisco, CA), May 2019.

Chaoshun Zuo, Wubing Wang, Rui Wang, and Zhiqiang Lin, *Automatic forgery of cryptographically consistent messages to identify security vulnerabilities in mobile services*, Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS'16) (San Diego, CA), February 2016.

Yunhui Zheng, Xiangyu Zhang, and Vijay Ganesh, *Z3-str: a z3-based string solver for web application analysis*.

Chaoshun Zuo, Qingchuan Zhao, and Zhiqiang Lin, *Authscope: Towards automatic discovery of vulnerable authorizations in online services*, Proceedings of the 24th ACM Conference on Computer and Communications Security (CCS'17) (Dallas, TX), November 2017.