(In-)Security of Backend-as-a-Service

Siegfried Rasthofer (TU Darmstadt / CASED)
Steven Arzt (TU Darmstadt / CASED)
Robert Hahn (TU Darmstadt)
Max Kolhagen (TU Darmstadt)
Eric Bodden (Fraunhofer SIT / TU Darmstadt)

#Whoami

Siegfried Rasthofer

- 3rd year PhD-Student at TU Darmstadt
- Research interest in static-/dynamic code analyses
- AOSP exploits, App security vulnerabilities
- Talks at academic as well as industry conferences

Steven Arzt

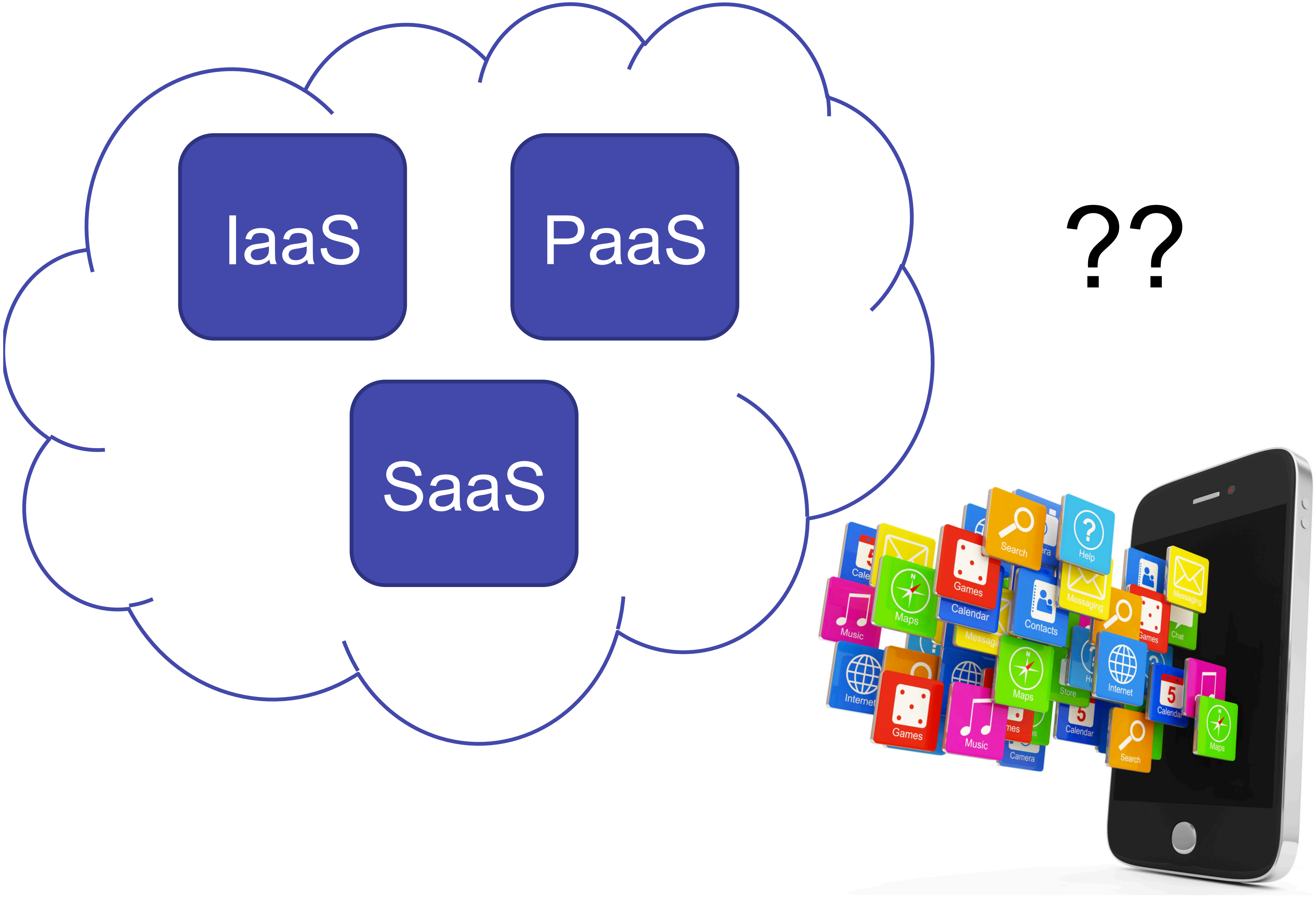
- 3rd year PhD-Student at TU Darmstadt
- Maintainer of the Soot and FlowDroid frameworks
- Works on static program analysis
- Likes to look for vulnerabilities

Access to 56 Mio non-public records...

Remote code execution...

Full VM control...

... with ease



IaaS

PaaS

SaaS

??

BaaS

Parse
The Cloud Application Platform
cocoafish
StackMob
kinvey
cloudmine
amazon
webservices™
BAASBOX
OPEN SOURCE BACKEND AS A SERVICE
mobeelizer

Security?

Parse

The Cloud Application Platform

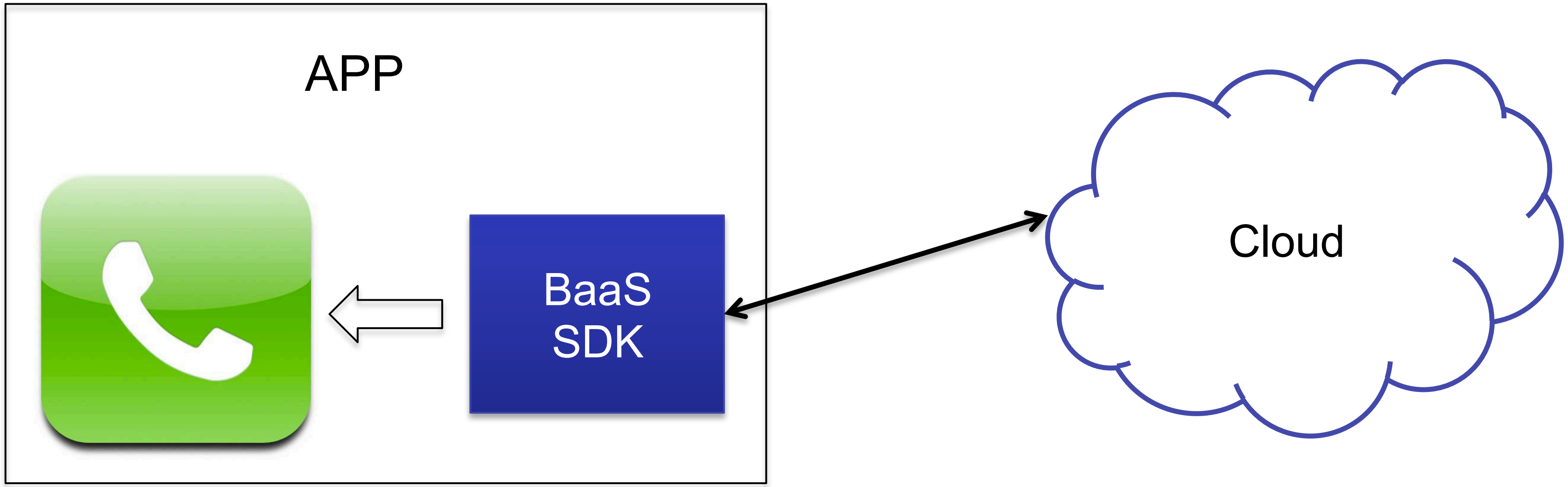
amazon

webservices™

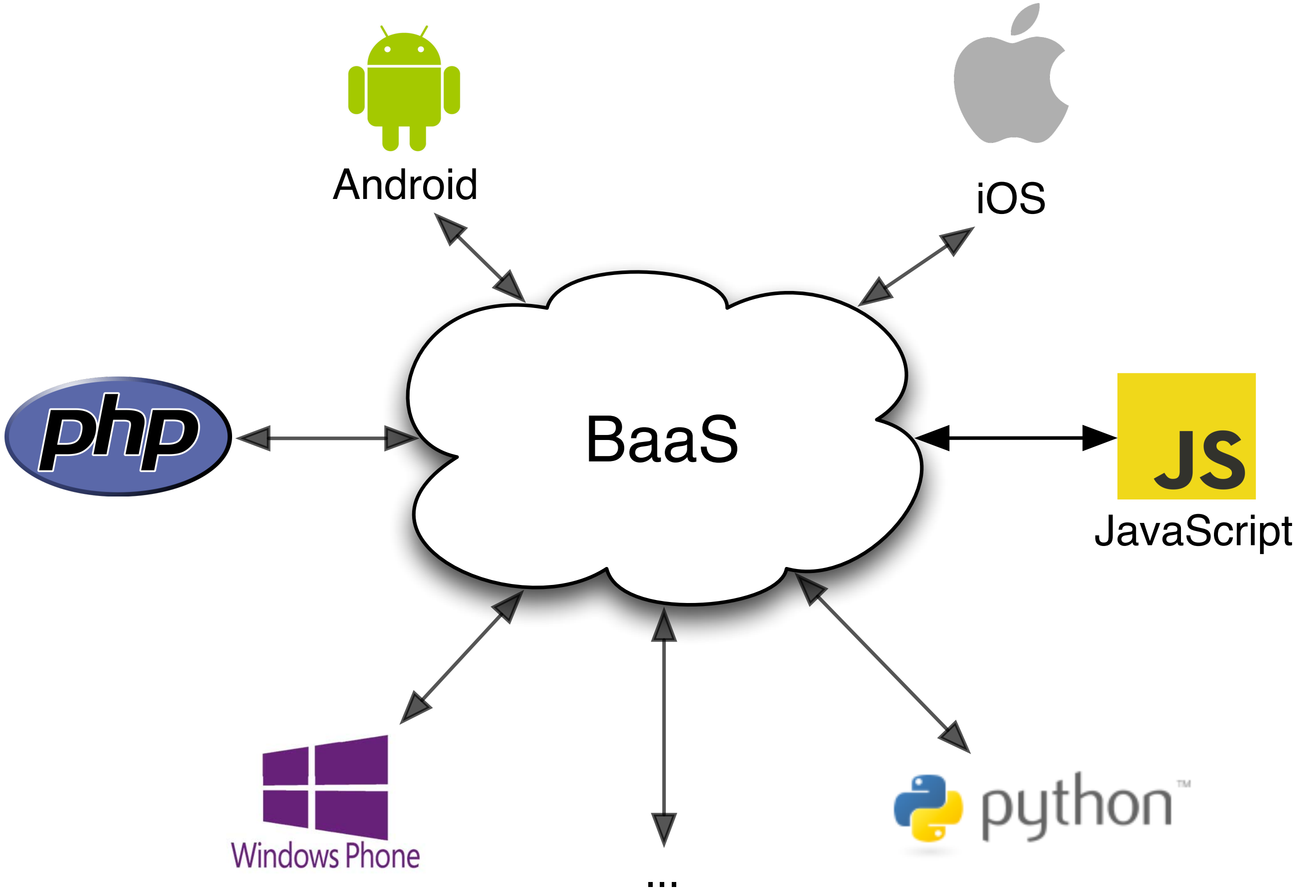
Agenda

- Introducing BaaS
- Security Analysis
- Findings
- Countermeasures
- The Wishlist
- Conclusion

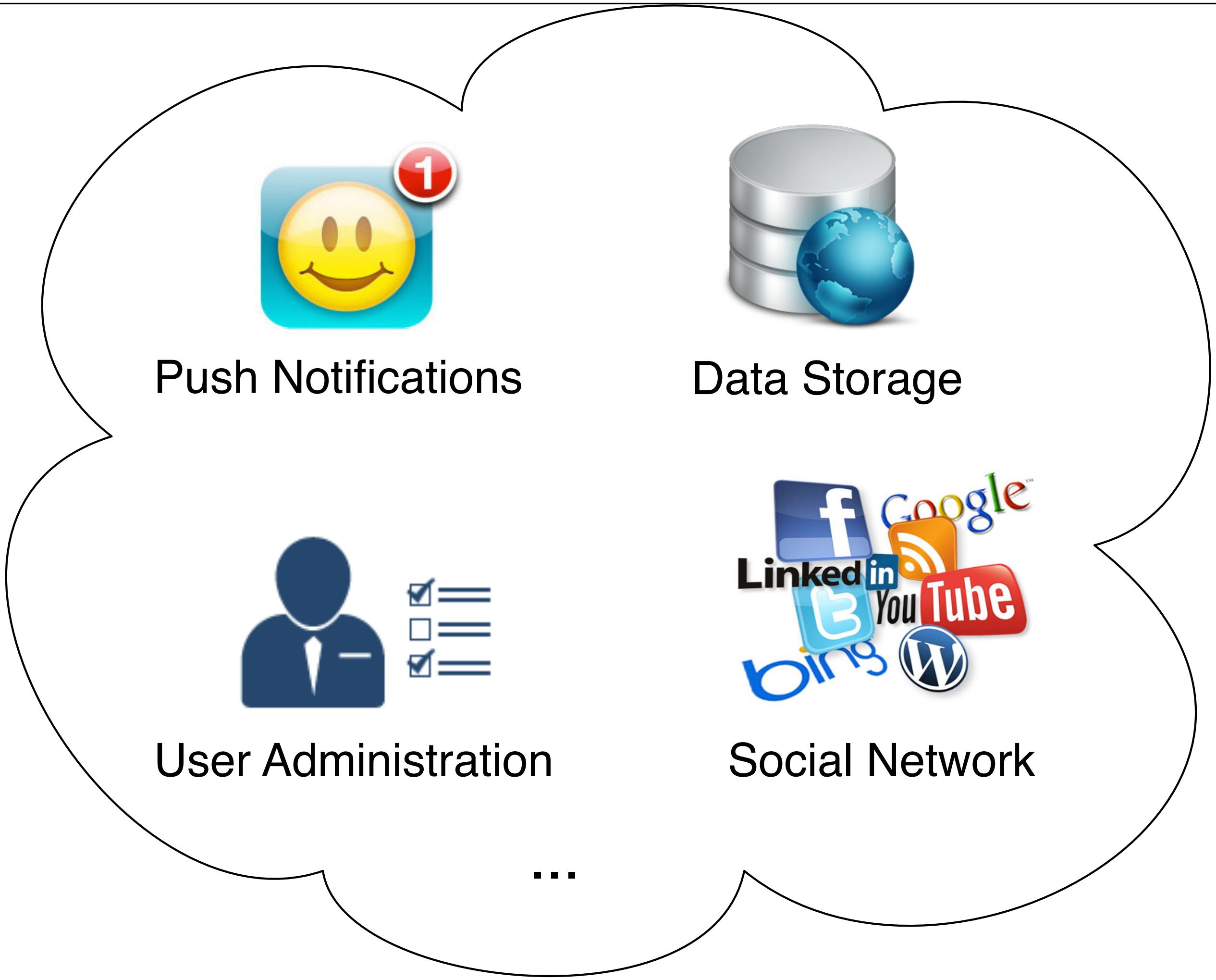
Backend-as-a-Service (1)



Backend-as-a-Service (2)



Backend-as-a-Service (3)



Push Notifications

Data Storage

User Administration

Social Network

…

BaaS
SDK

Amazon Tutorial

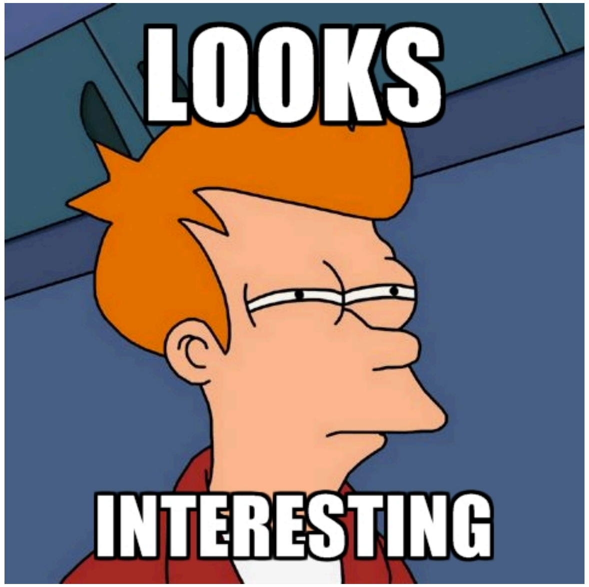
DB connection

```
AmazonS3Client s3Client = new AmazonS3Client(
        new BasicAWSCredentials(“ACCESS_KEY_ID“, “SECRET_KEY“) );
```

Amazon Tutorial

DB connection

```
AmazonS3Client s3Client = new AmazonS3Client(
        new BasicAWSCredentials(“ACCESS_KEY_ID“, “SECRET_KEY“) );
```



“When you access AWS programmatically, you use ***an access key to verify your identity and the identity of your applications. An access key consists of an access key ID and a secret access key.***

Anyone who has your access key has the same level of access to your AWS resources that you do.“

Source: http://docs.aws.amazon.com/

Amazon Tutorial

DB connection

```
AmazonS3Client s3Client = new AmazonS3Client(
        new BasicAWSCredentials(“ACCESS_KEY_ID“, “SECRET_KEY“) );
```

(username) (password)

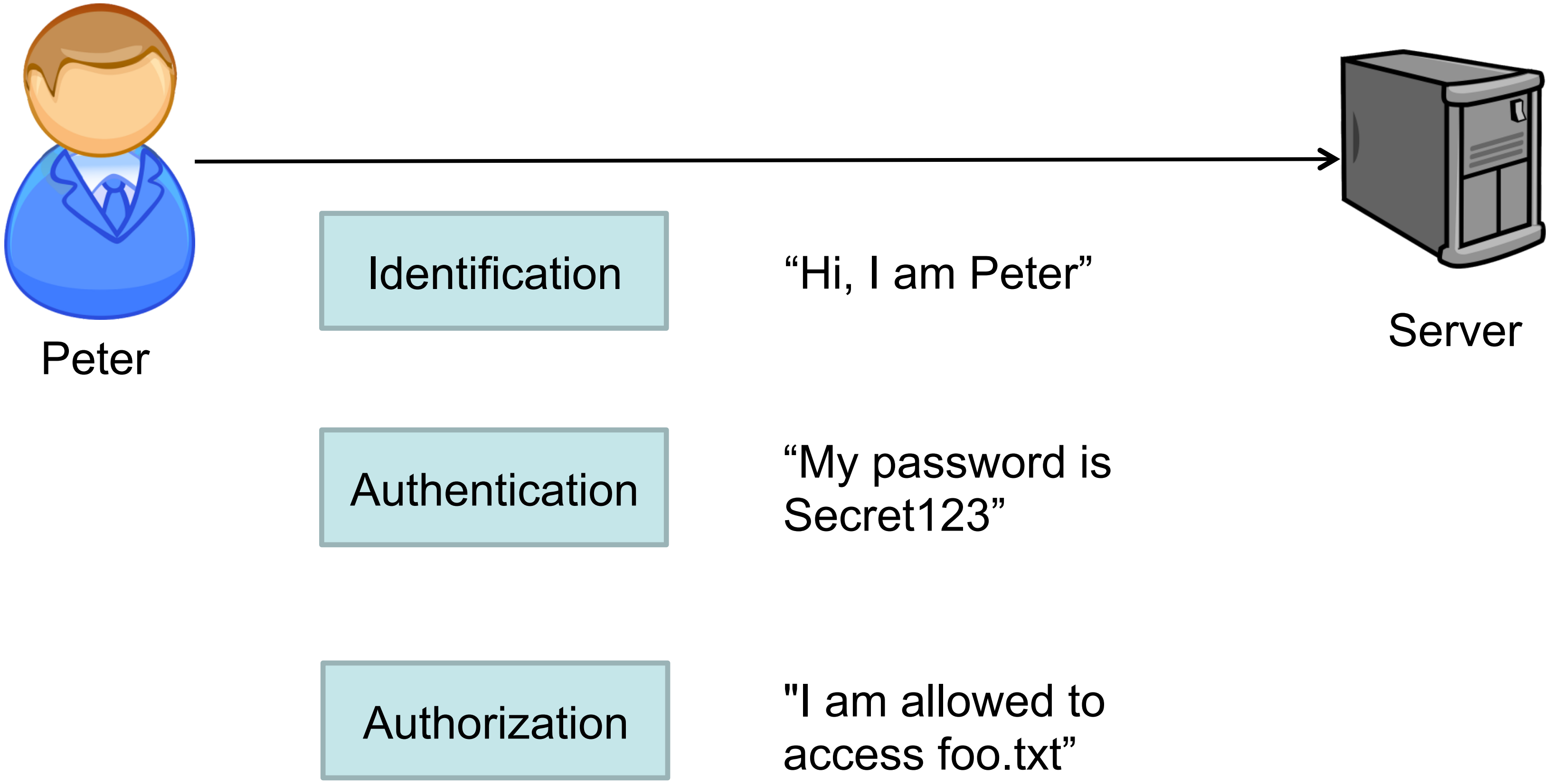
“…The AWS SDKs use your access keys to **sign requests** for you so that you don't have to handle the signing process…”

http://docs.aws.amazon.com/

“…Secret access keys are, as the name implies, secrets, like your **password**…”

Jim Scharf
Director, AWS Identity and Access Management

IT Security 101



Peter

Server

Identification

“Hi, I am Peter”

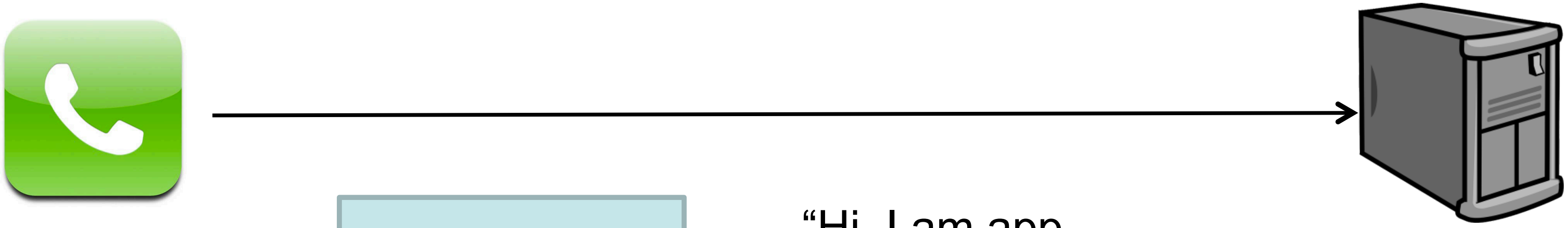
Authentication

“My password is Secret123”

Authorization

"I am allowed to access foo.txt”

App Authentication Model



App

Server

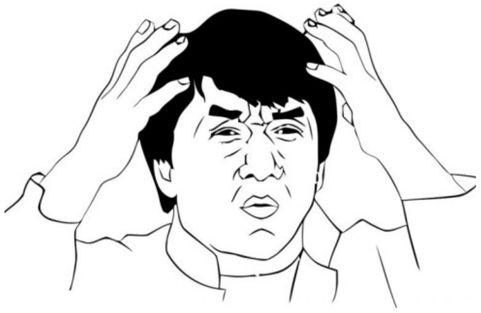
Identification

“Hi, I am app <Application ID>”

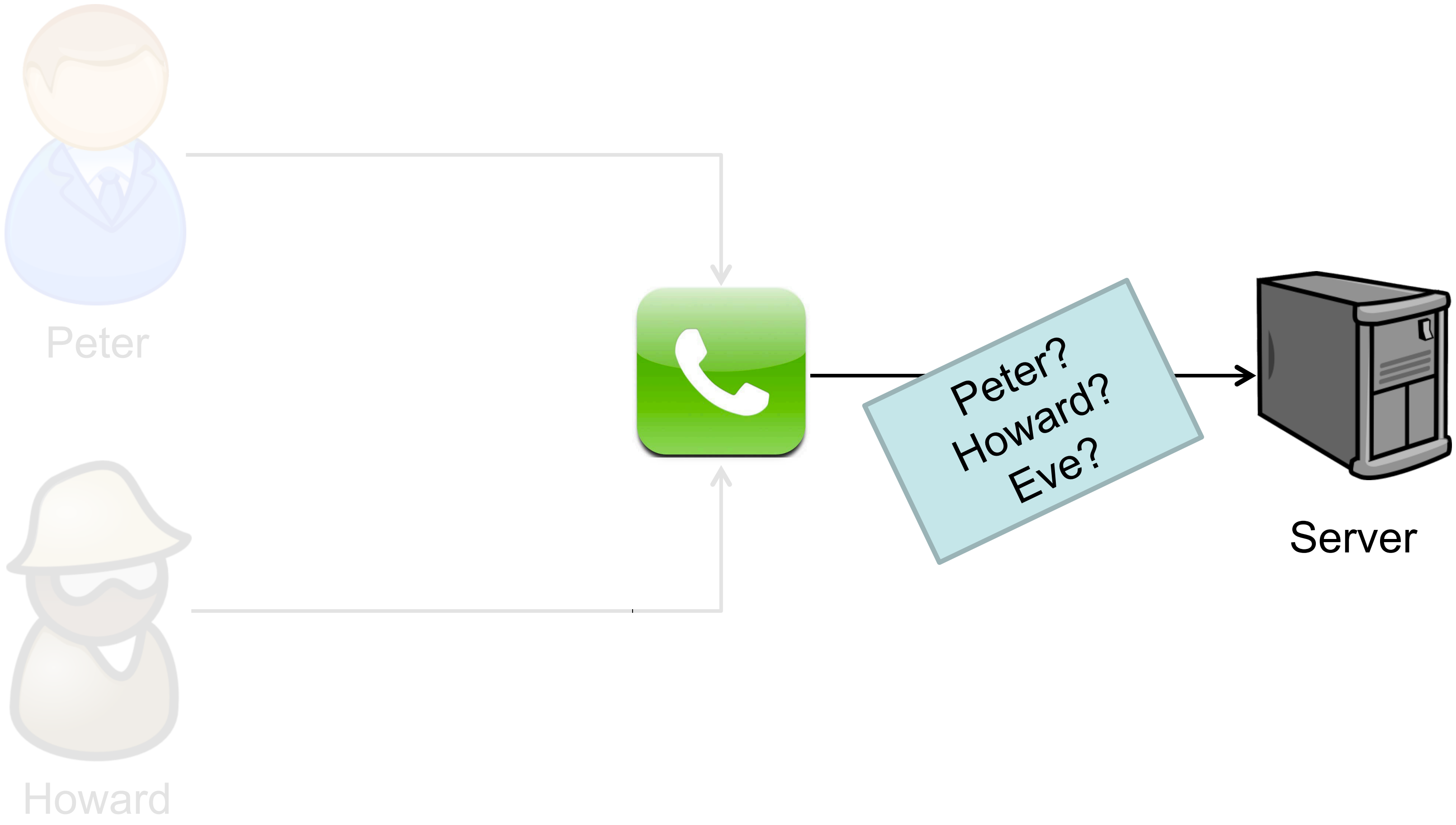
Authentication

“My <Secret Key> is in the app” ???

Identification ?? = Authentication



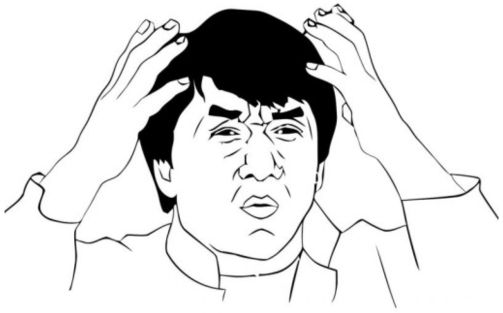
App Authentication Model



Developer Opinion

Q: [...]“***The App-Secret key should be kept private*** *- but when releasing the app they can be reversed by some guys. I want to know what is the best thing to* ***encrypt, obfuscate*** *or whatever to make this secure.*“[...]

(Source: stackoverflow.com)



NO!!!!

R: “*Few ideas, in my opinion only first one gives some guarantee:*

1. ***Keep your secrets on some server on internet****, and when needed just grab them and use.*
2. ***Put your secrets in jni code***
3. ***use obfuscator***

4. Put your secret key as ***last pixels of one of your image*** *in assets* “

(Source: stackoverflow.com)

Let’s go for it

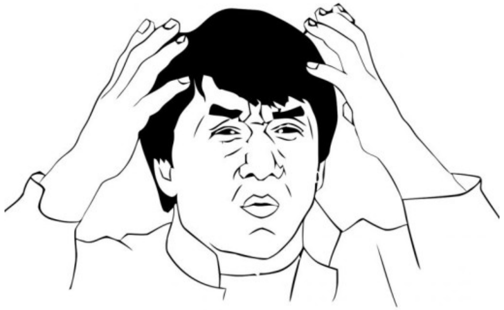
SECURITY ANALYSIS

Pre-Analysis (Parse example)

+ Code Inspect

```
public void onCreate()  {
  java.lang.String $S1, $S2;
  $S1 = “34lI1wgISkIUpTunWRAzXei20H3NAL7W6buKTe7e";
  $S2 = “pB7OlNi0jsEp3fpJfq9wvHBoOWgaOQCSW98BF7e3";
  staticinvoke <Parse: void initialize(Context, String, String)>(this, $S1, $S2);
 }
```

”User” Table

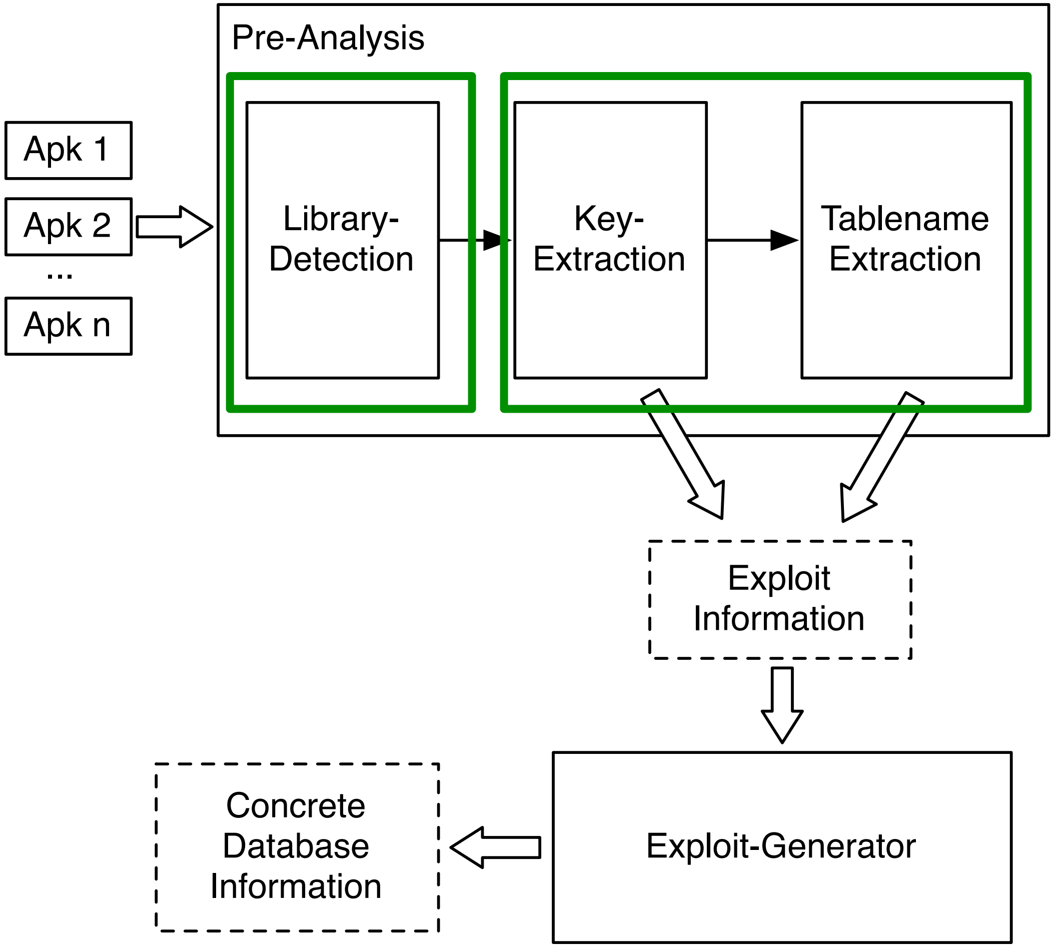


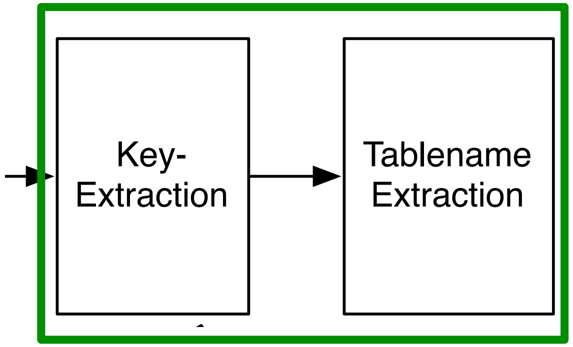
Pre-Analysis result:

- All records were accessible
- Few developers used obfuscation techniques (“security by obscurity”)

… let’s get ready for a mass-analysis

Mass Analysis





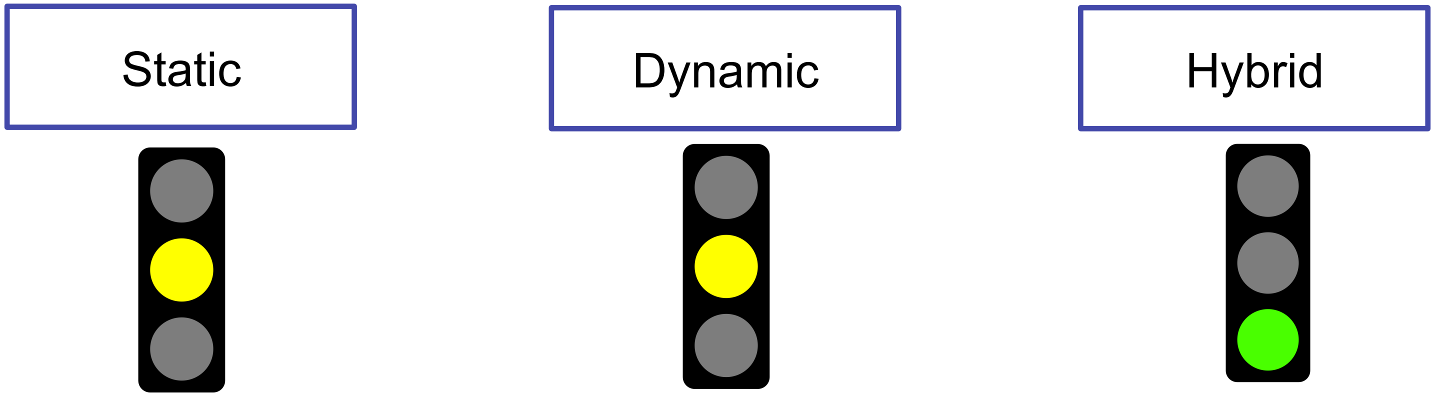
How can we extract **specific information (e.g. strings)** from Apks?

APK Information Extraction

```
$S1 = “34lI1wgISkIUpTunWRAzXei20H3NAL7W6buKTe7e";
$S2 = “pB7OlNi0jsEp3fpJfq9wvHBoOWgaOQCSW98BF7e3";
staticinvoke <Parse: void initialize(Context, String, String)>(this, $S1, $S2);
```

1. API Identification

2. Information Extraction:

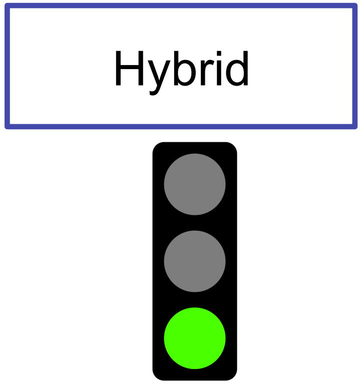


Static

Dynamic

Hybrid

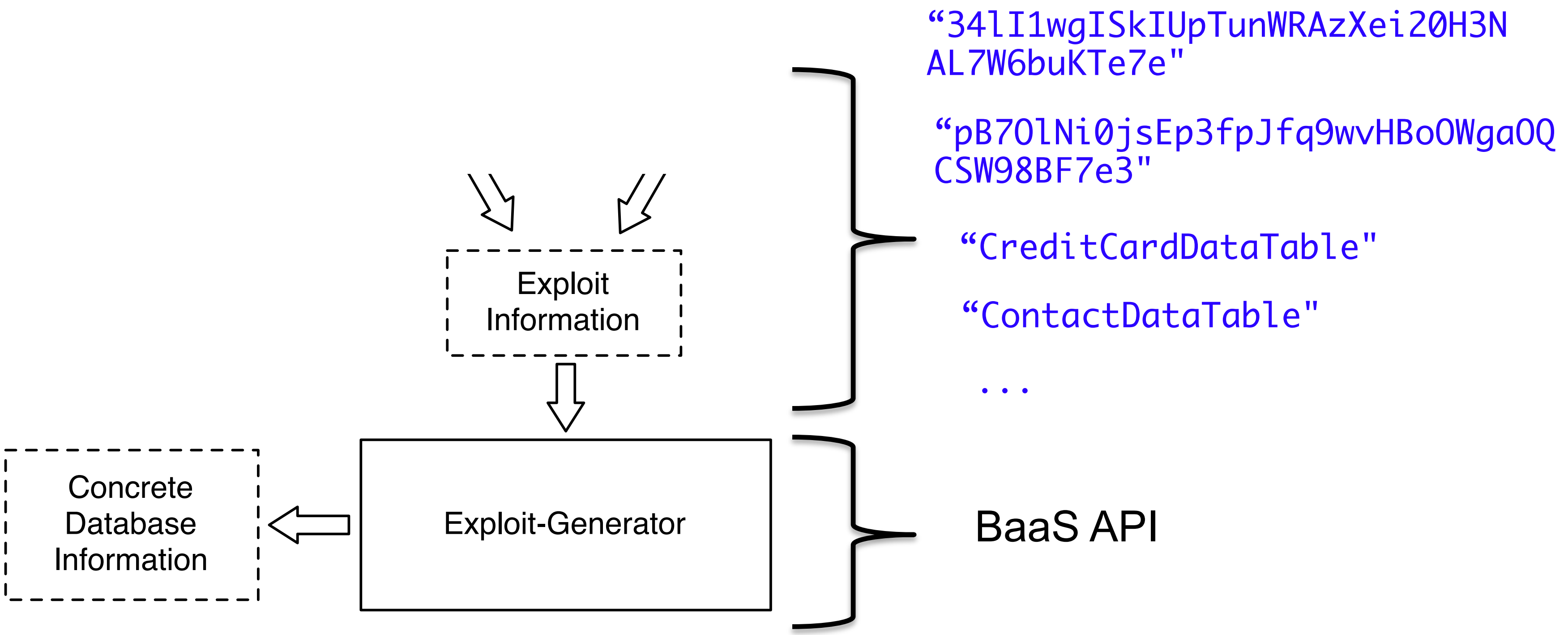
HARVESTER (Hybrid Data Extraction)



Hybrid

Harvesting Runtime Data in Android Applications for Identifying Malware and Enhancing Code Analysis
Siegfried Rasthofer, Steven Arzt, Marc Miltenberger, Eric Bodden
Technical Report, February 2015.

Data Access



“34lI1wgISkIUpTunWRAzXei20H3N
AL7W6buKTe7e"

“pB7OlNi0jsEp3fpJfq9wvHBoOWgaOQ
CSW98BF7e3"

“CreditCardDataTable"

“ContactDataTable"

...

Exploit Information

Exploit-Generator

Concrete Database Information

BaaS API

So … how bad is it?

OUR FINDINGS

Findings Parse

Precise car accident Information

Pictures

User-centric location data

Birthday Information

Contact data

Phone numbers

Valid email addresses

Facebook Information
- User‘s friends
- User‘s blocked friends

Purchase data

Findings Parse (2)

Intercepted SMS messages

C&C tasks

Leaked data

C&C commands

We know what you did this summer: Android Banking Trojan exposing its sins in the cloud
Siegfried Rasthofer, Eric Bodden, Carlos Castillo, Alex Hinchliffe
VirusBulletin 2015, AVAR 2015

Responsible Disclosure Process – Parse (Facebook)

	Full access to 100 tables Partial access to 32 tables (~56 Mio Data)
28th April 2015	Contacted Facebook with a few samples
5th Mai 2015	Facebook verified it and asked for more data
18th Mai 2015	We provided all information
20th Mai 2015	Facebook forwarded everything to Parse (we assume they contacted the developers)
12th Nov 2015	Full access to 99 tables Partial access to 33 tables

Findings Amazon (3)

Server Backups

Baby Growth Data

Photos

Findings Amazon (4)

Private Messages

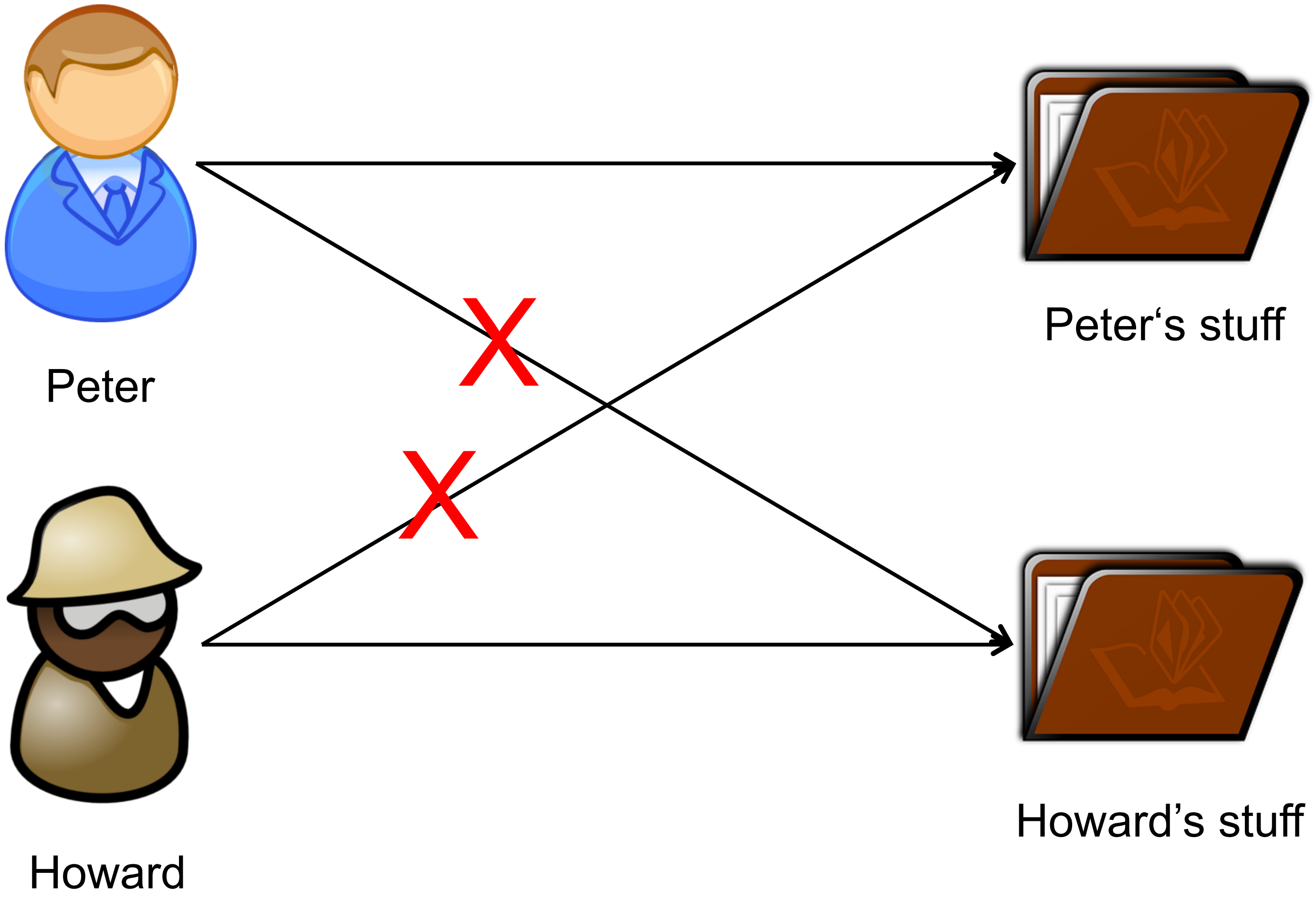
Lottery Data

Web Page Content

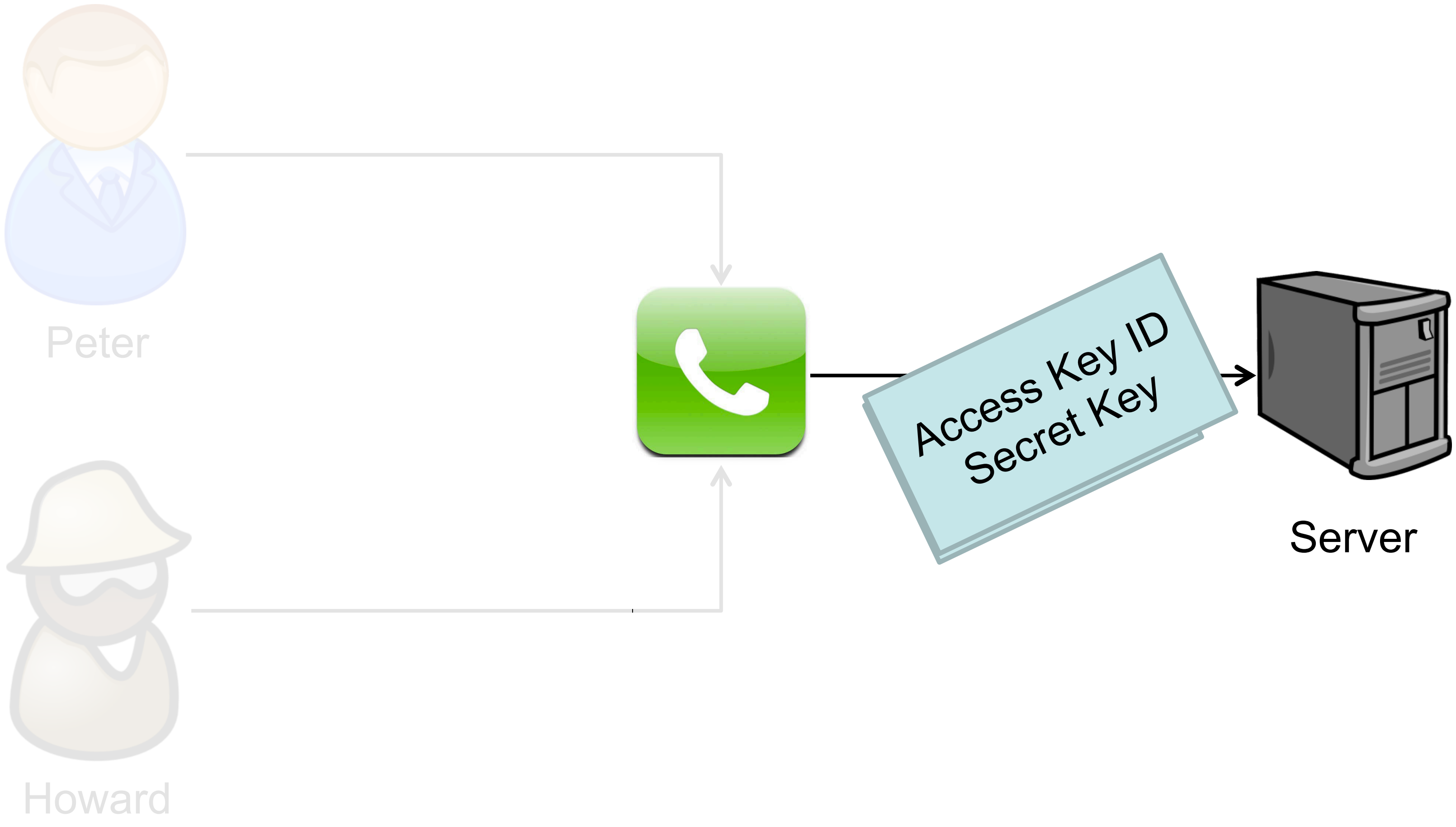
How can we get it right?

COUNTERMEASURES

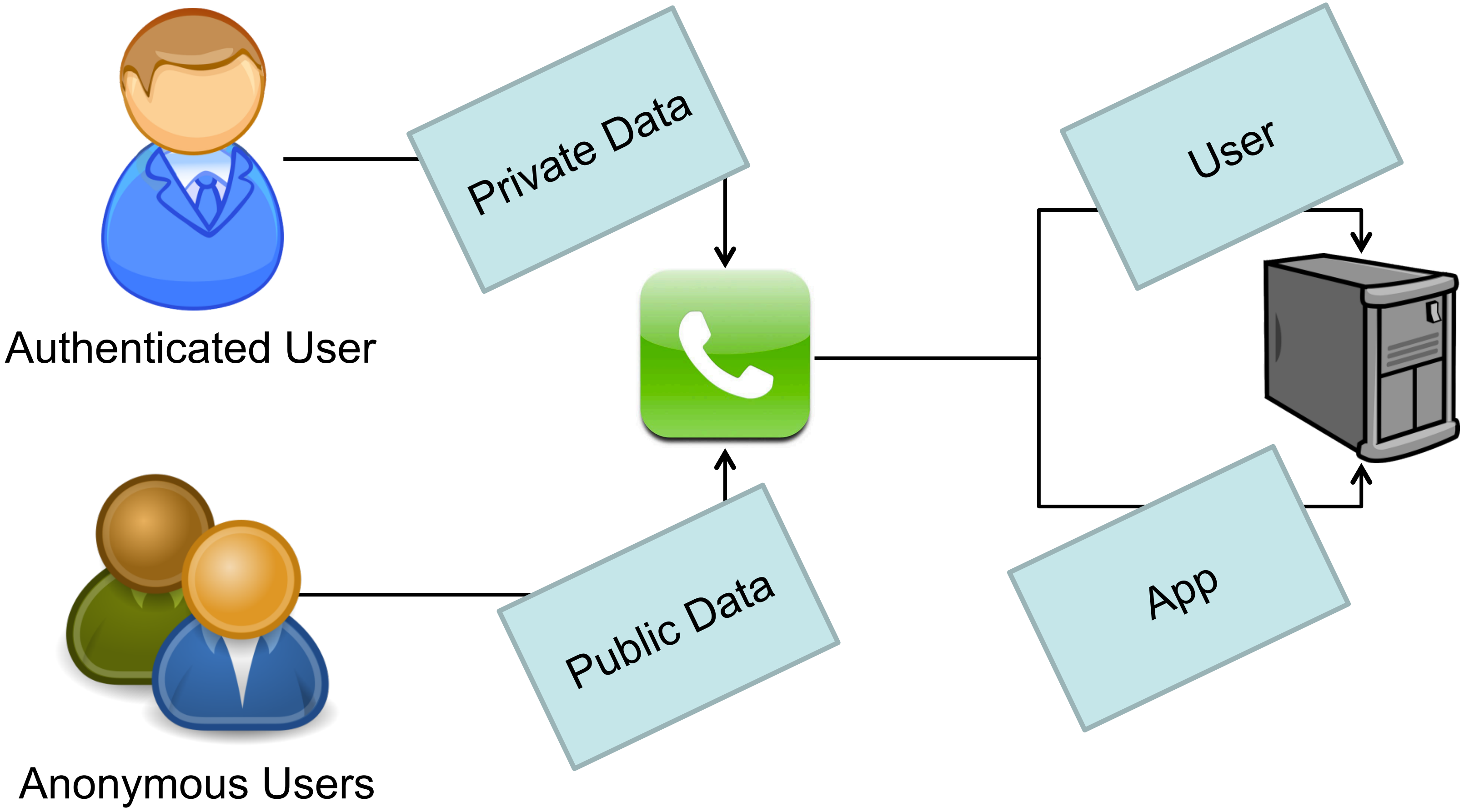
IT Security 101: ACLs



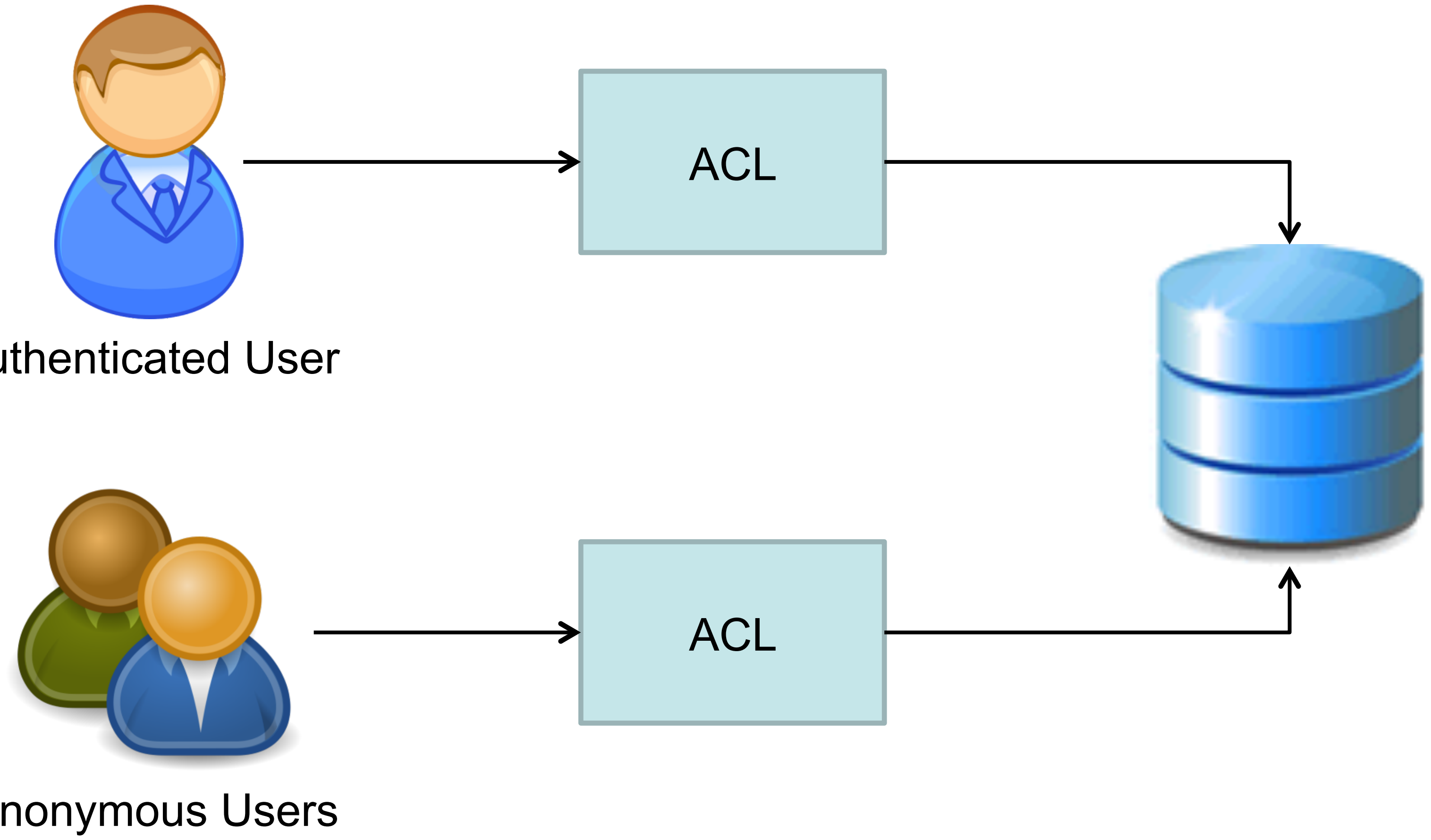
Recall: App Authentication Model



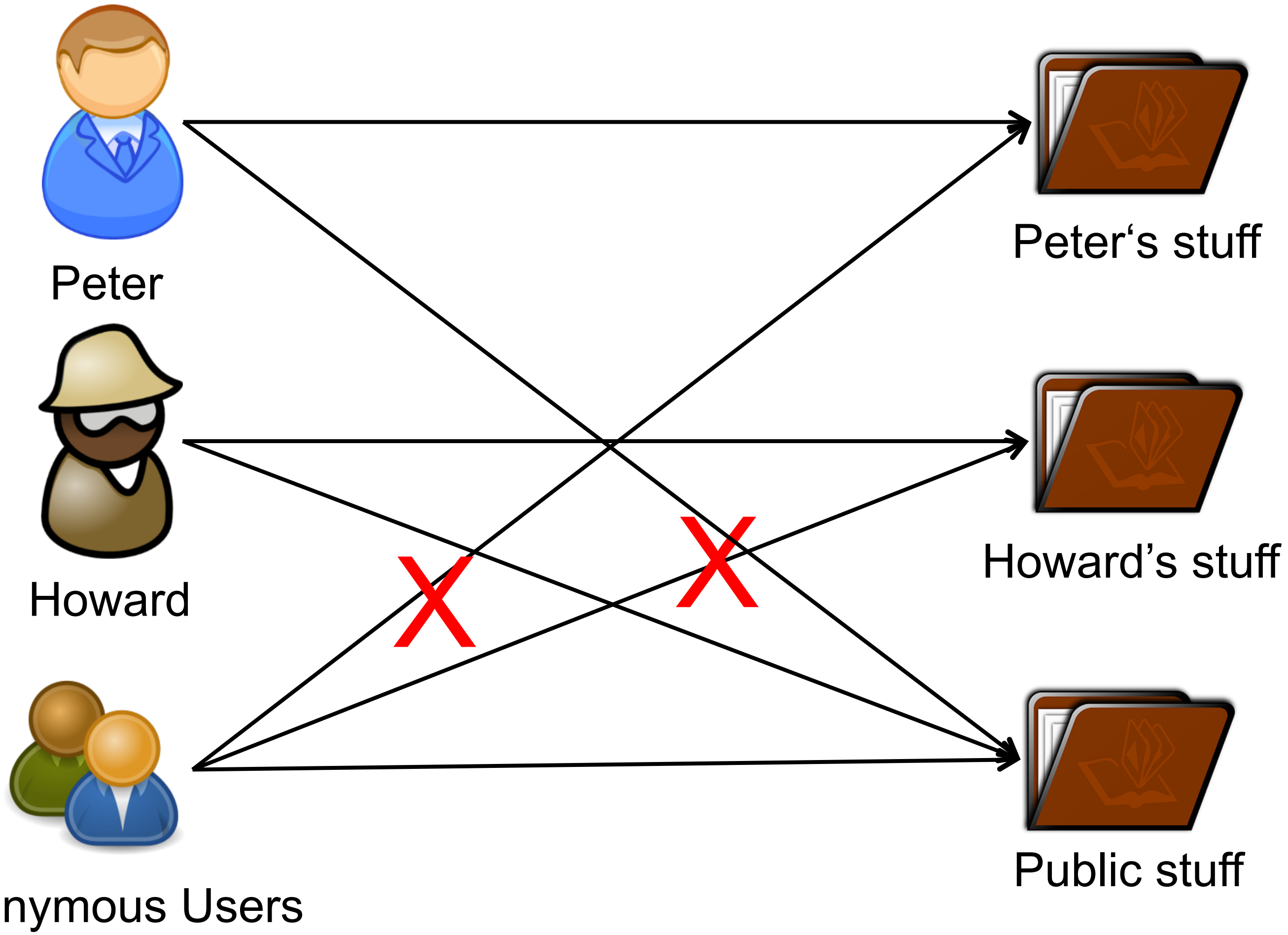
Two BaaS Usage Scenarios



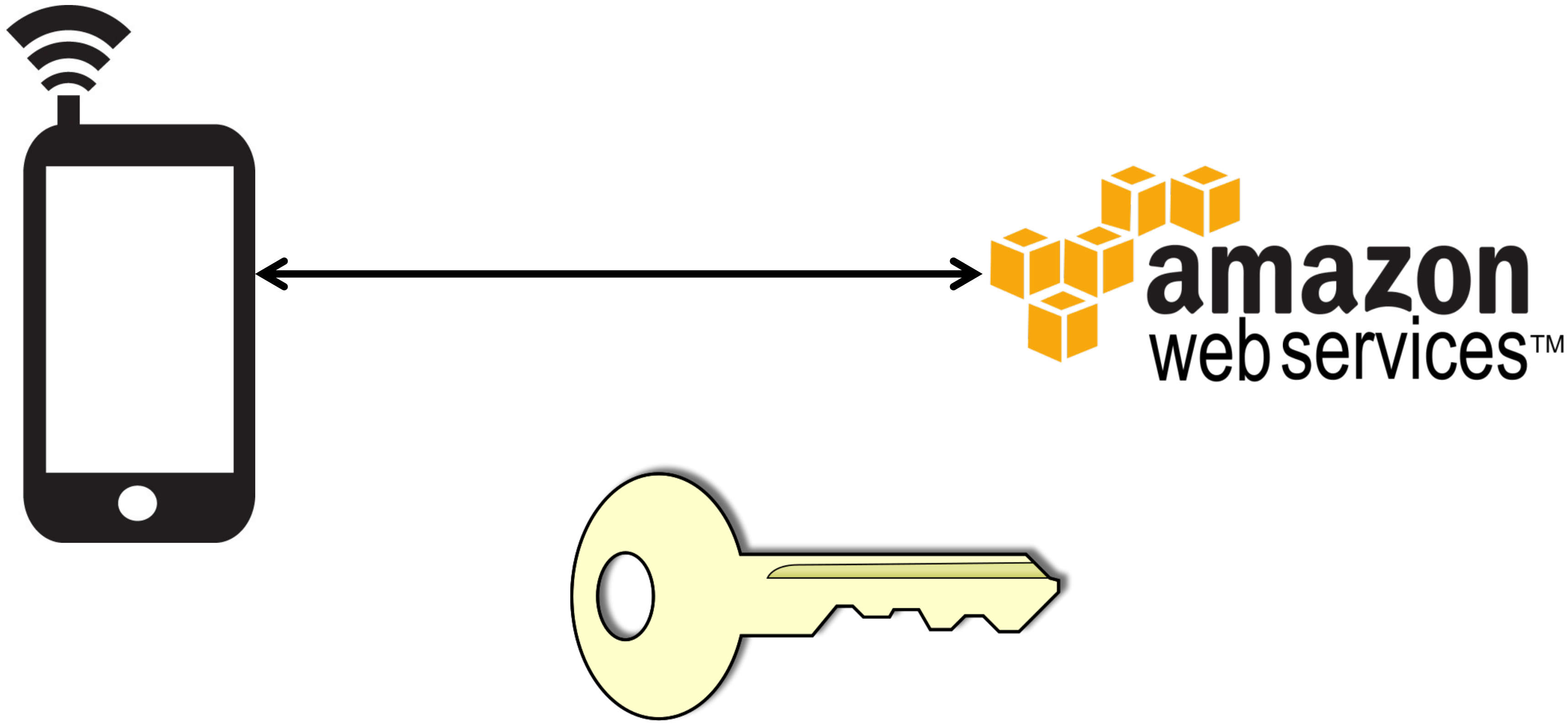
Two BaaS Usage Scenarios



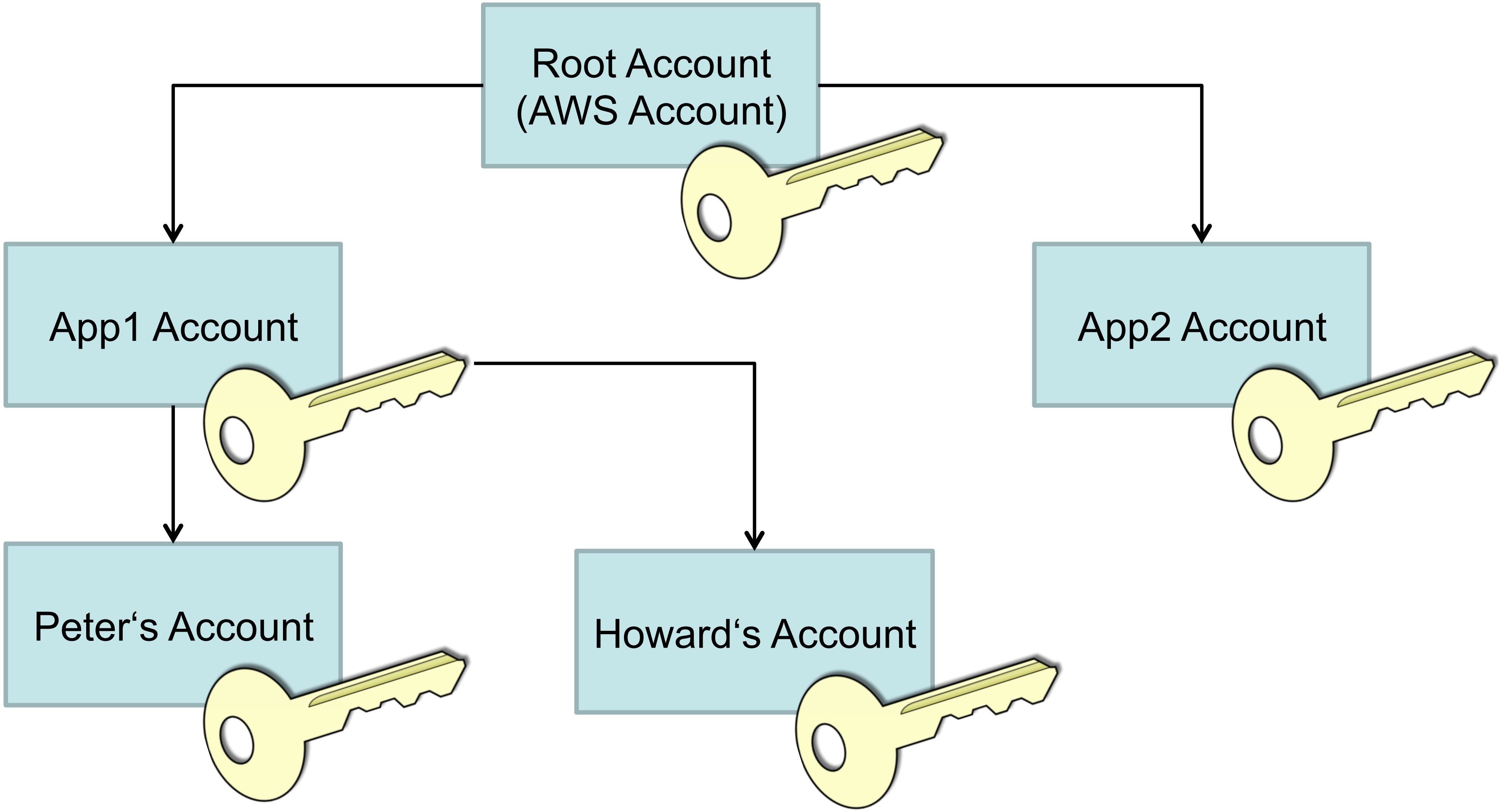
ACLs in The App Security Model



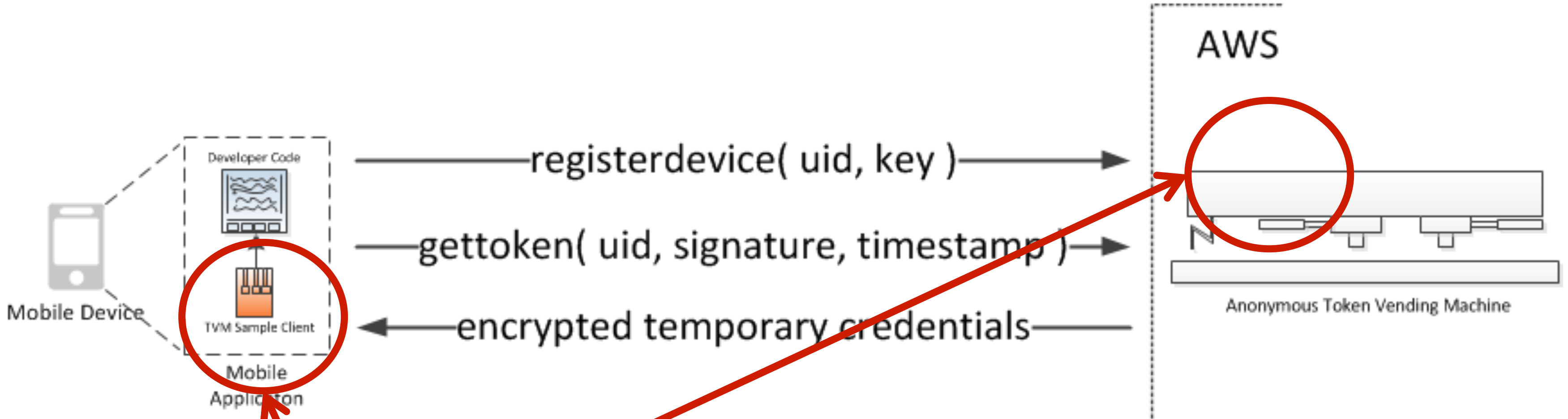
Amazon Key Hierarchy (1)



Amazon Key Hierarchy (2)



Amazon Token Vending Machine (1)


AWS
Developer Code
TVM Sample Client
Mobile Device
Mobile Application
registerdevice(uid, key)
gettoken(uid, signature, timestamp)
encrypted temporary credentials
Anonymous Token Vending Machine

Interaction between Anonymous TVM and Mobile Client Application

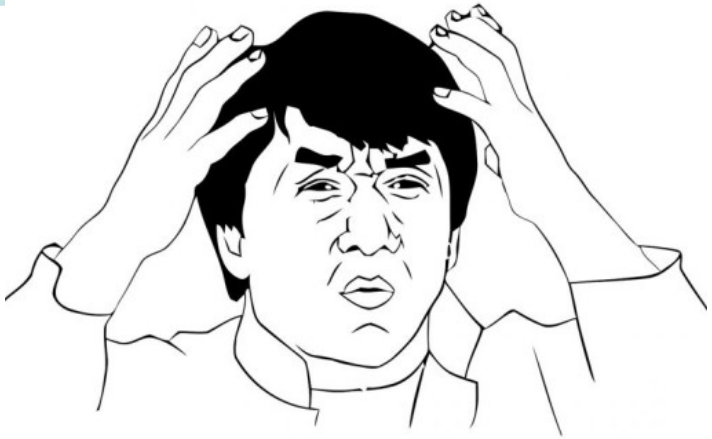
Sample available, final implementation is on you

Needs hosting. Tomcat, Elasticbeanstalk anyone?

Amazon Token Vending Machine (2)

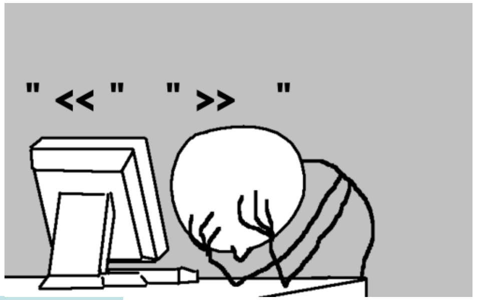
Although you will need to use your AWS account credentials to deploy the TVM, we recommend that you do not run the TVM under your AWS account. Instead, create an IAM user and configure the TVM to use the credentials of this IAM user, which we will call the *TVM user*.

So, we have S3, TVM, IAM, Elastic Beanstalk



Amazon Token Vending Machine (3)

- What if I want ACLs?
- Identity TVM samples do exist, but…



You would need to modify the provided samples in order to implement these user-specific policy objects. For more information about policy objects, see the Identity and Access Management (IAM) documentation

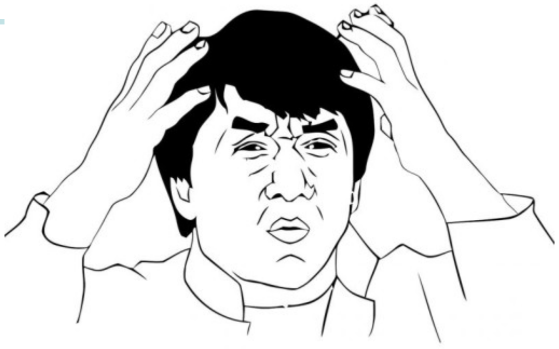
Amazon Cognito (1)

- Provides Identity Management
  - Real users
  - Anonymous identities
- Rather New Service
  - Not commonly used yet

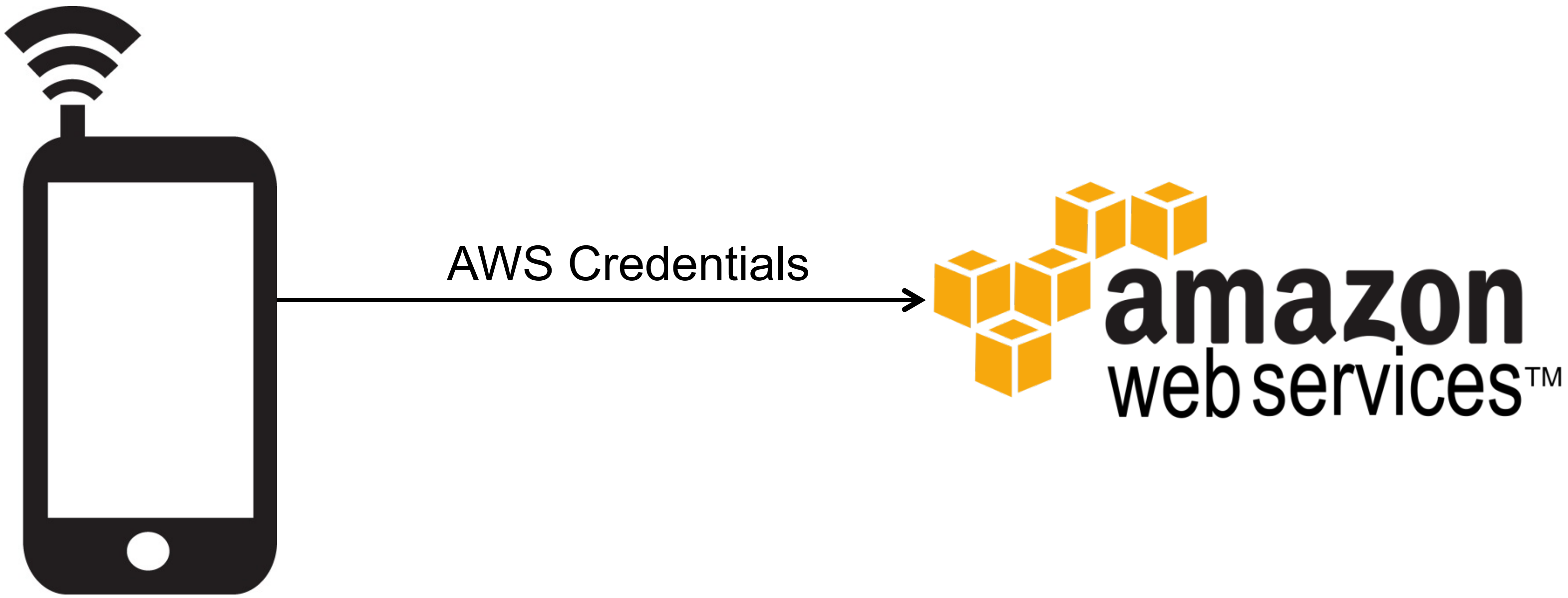
So … let‘s try to use it

Amazon Cognito (2)

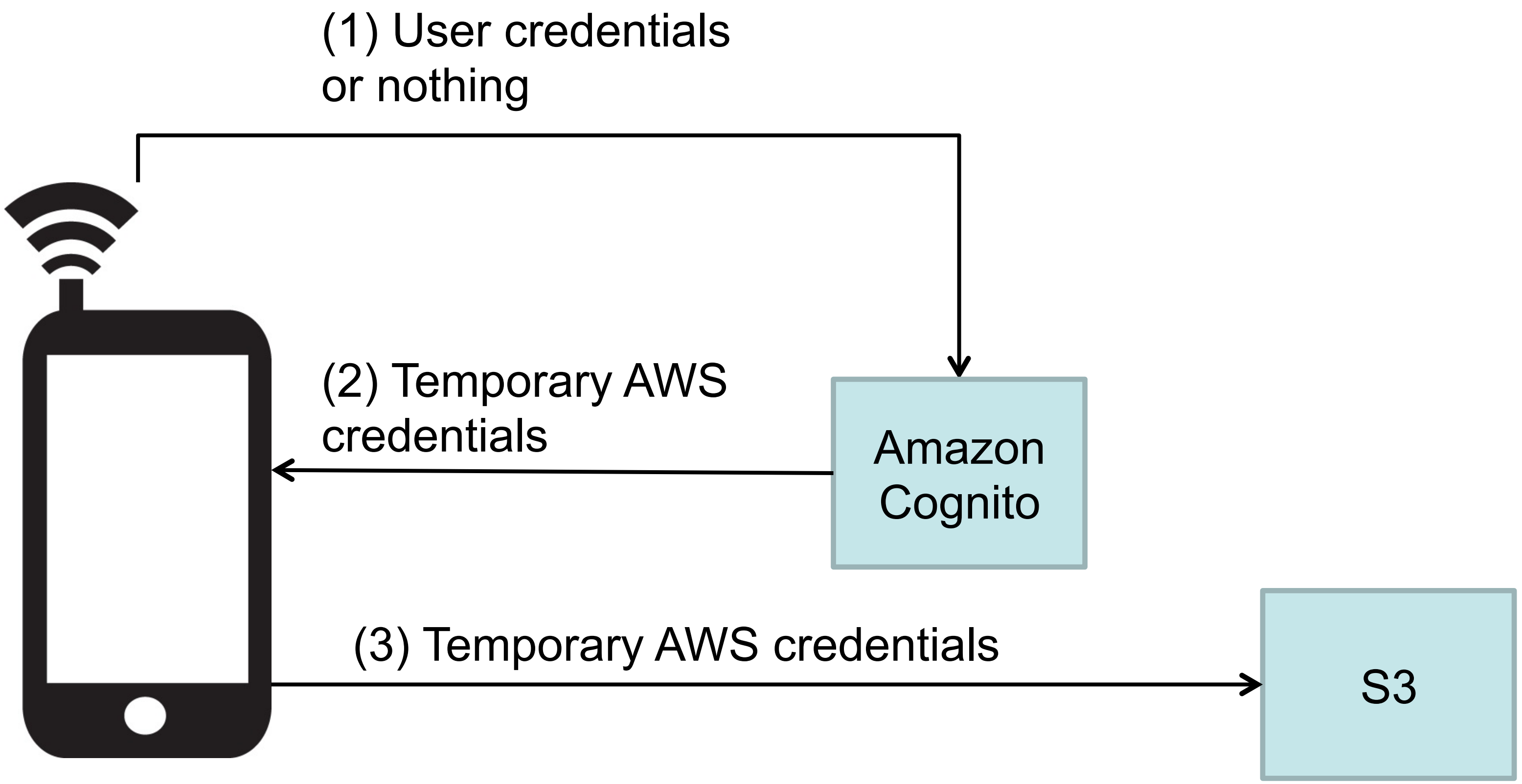
> **Note:** If you created your identity pool before February 2015, you will need to reassociate your roles with your identity pool in order to use this constructor without the roles as parameters. To do so, open the Amazon Cognito Console, select your identity pool, click **Edit Identity Pool**, specify your authenticated and unauthenticated roles, and save the changes.



Amazon Cognito (5)



Amazon Cognito (6)



(1) User credentials or nothing

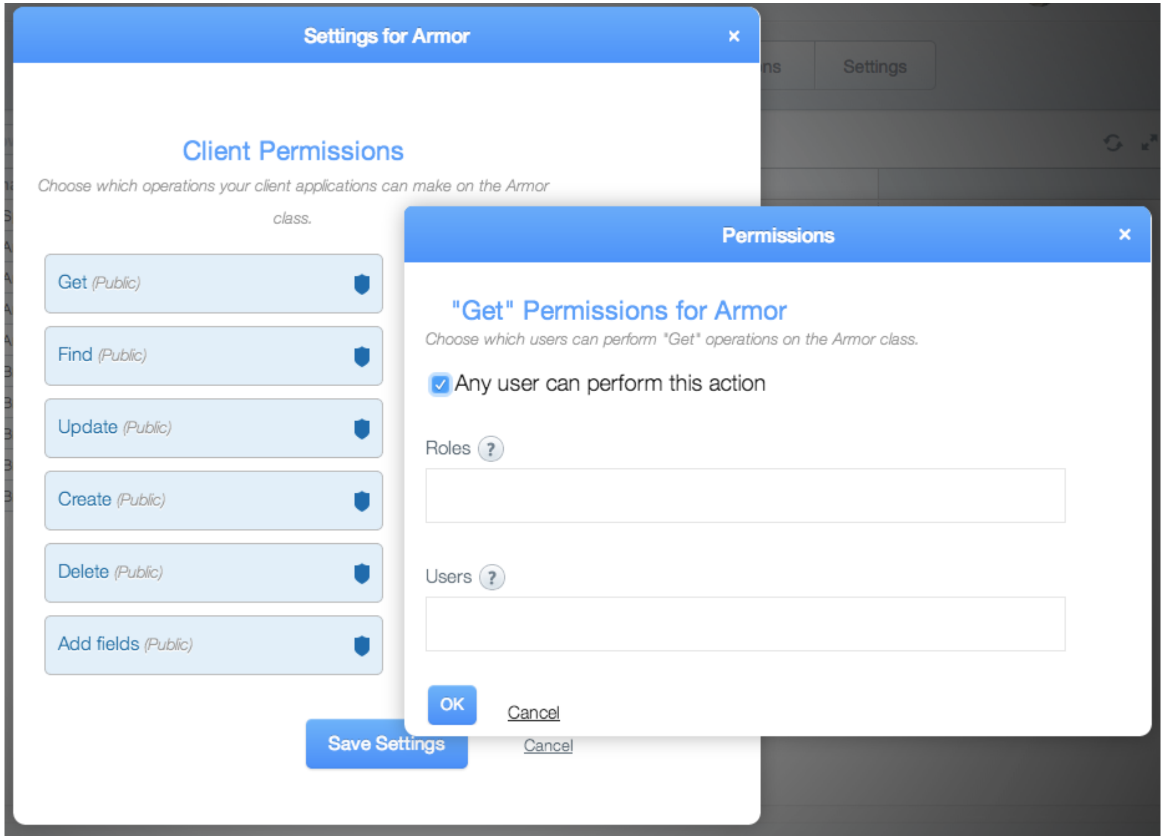
(2) Temporary AWS credentials

Amazon Cognito

(3) Temporary AWS credentials

S3

Parse.com ACLs (1)



Settings for Armor

Client Permissions

Choose which operations your client applications can make on the Armor class.

Get (Public)

Find (Public)

Update (Public)

Create (Public)

Delete (Public)

Add fields (Public)

Permissions

"Get" Permissions for Armor

Choose which users can perform "Get" operations on the Armor class.

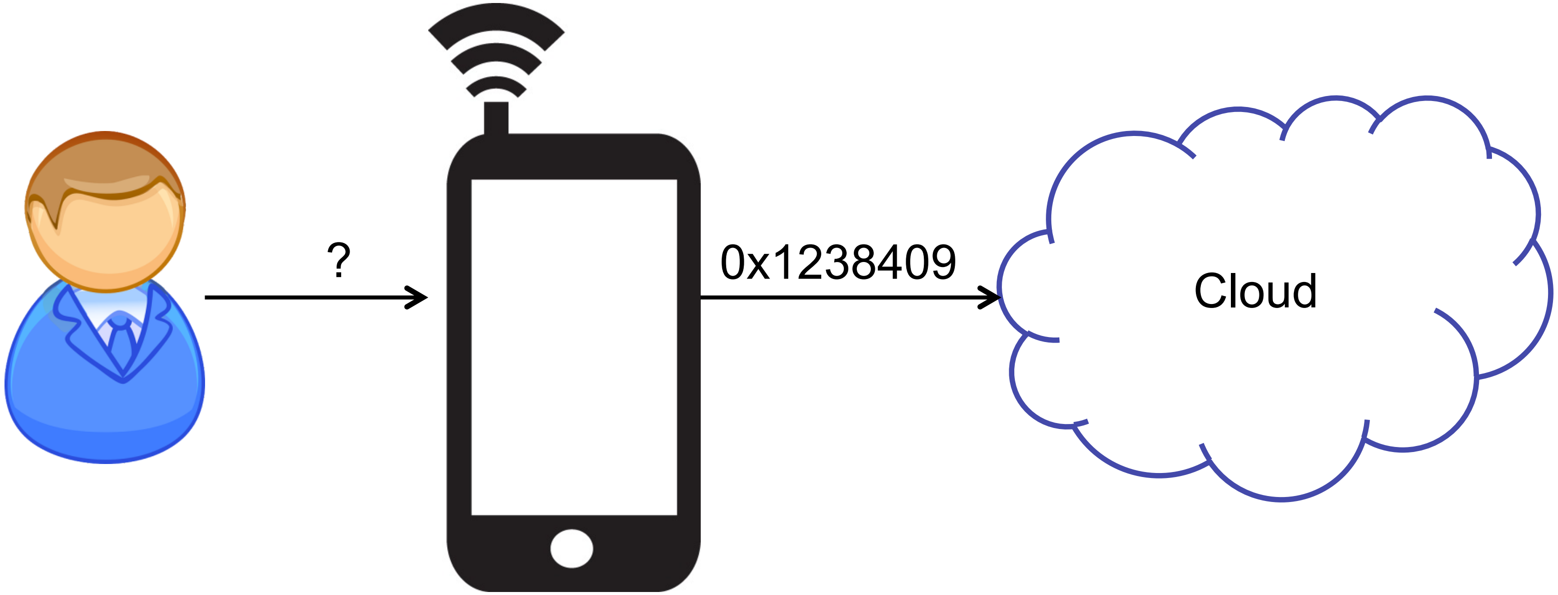
☑ Any user can perform this action

Roles ?

Users ?

Source: http://blog.parse.com/learn/engineering/parse-security-ii-class-hysteria/

Parse.com ACLs (2)



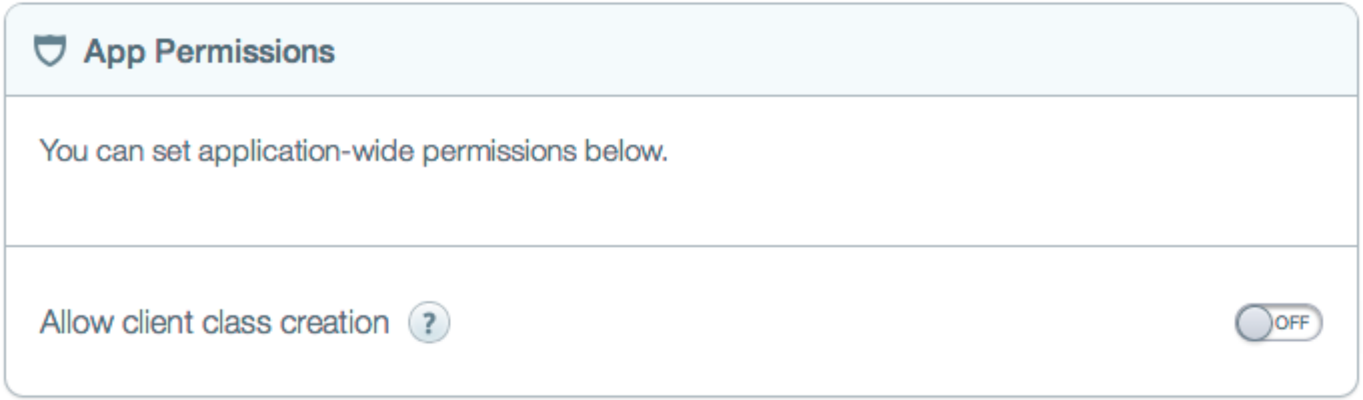
http://blog.parse.com/announcements/protect-user-data-with-new-parse-features/

Parse.com ACLs (3)

Anonymous users are special, however, in that once logged out, the user cannot be recovered – a new user will need to be created, and the original user (and its associated data) will be orphaned.

Double-check your cloud storage space!

Parse.com Global Settings



App Permissions

You can set application-wide permissions below.

Allow client class creation ? OFF

Get this wrong and offer free disk space to anyone!

Source: http://blog.parse.com/learn/engineering/parse-security-ii-class-hysteria/

What now?

THE WISHLIST

What shall change?

Improved Documentation

Checks and Alerts

Legal Framework

Takeaway Messages

- Security in the cloud doesn‘t come for free
- Attacks are free, effortless, and simple
- Mitigation techniques exist
  - ➢ People must care about them
  - ➢ Secure your apps now – we’re there!

Siegfried Rasthofer
Secure Software Engineering Group
Email: *siegfried.rasthofer@cased.de*
Twitter: @CodeInspect

Steven Arzt
Secure Software Engineering Group
Email: *steven.arzt@cased.de*

Blog: http://sse-blog.ec-spride.de
Website: http://sse.ec-spride.de