True Security®

Mobile Application Security Services

Safe Frontier

True Security®

About Safe Frontier

Expert mobility management and security company focusing on:

- Cutting edge enterprise mobility management
- Mobile application management and security solutions
- Telecom services enabling solutions

True Security®

About Safe Frontier

Our customers:

- Businesses and agencies
- Managed service providers
- Telecom companies
- Mobile application developers
- Systems integrators

There are two main categories of mobile app risks

Malicious Functionality

Activity sniffing and data theft (Trojan)
Unauthorized network connectivity
Unauthorized access to paid-for resources (dialing, SMS, etc.)
User interface impersonation
System modification
Logic bomb

Vulnerabilities

Sensitive data leakage (side channel or involuntary)
Unsafe sensitive data storage
Unsafe sensitive data transmission
Hardcoded authentication credentials

What makes mobile application security different?

Different threat model

New vulnerabilities related to client side (for example Android), such as:

“one application can exploit other application installed on the device”

Different skills for penetration testers and different tools required:

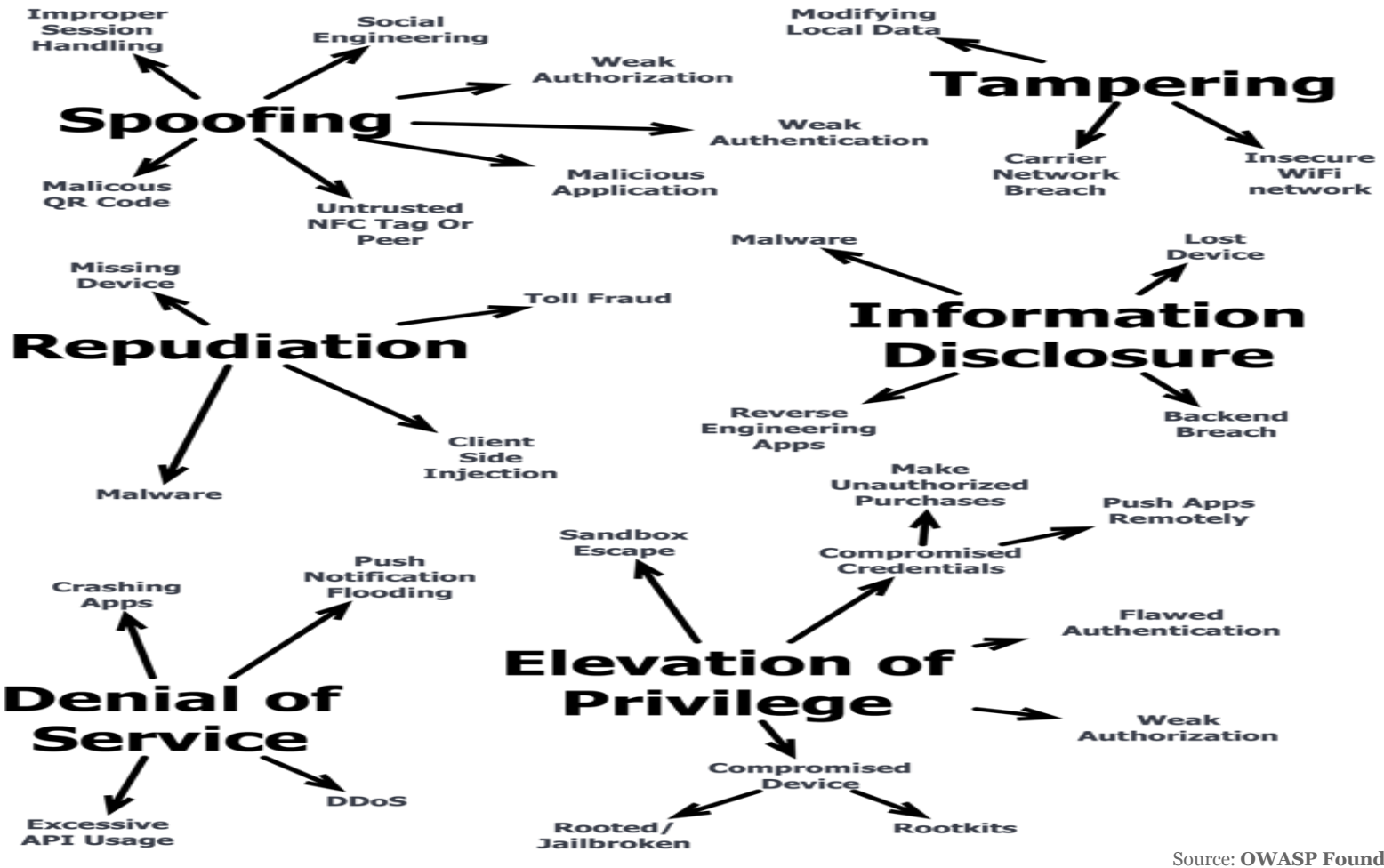
- Lack of mature security tools
- Knowledge
- Complex static analysis
- Dynamic analysis

Mobile Threat Model

Portable nature of devices and type of apps

Mobile apps are connected to web services
Mobile apps are updated frequently
Mobile apps are acquired from an app store
Mobile apps interact with device sensors (microphone, camera, location)
Mobile devices can be employee liable and access organization network
Mobile devices have access to payment capabilities
Mobile devices are network enabled
Mobile devices have higher probability of being lost or stolen

Mobile Threat Model



Source: **OWASP Foundation**

What are the vulnerabilities we're after?

Just to name a few…

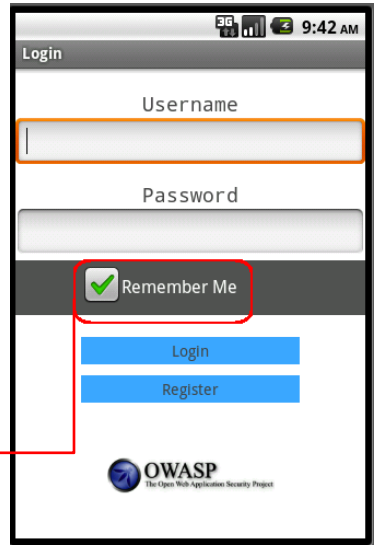
- Backdoors, malicious code, compromising activity
- Insecure data storage
- Weak server side controls
- Insufficient transport layer protection
- Malicious input from another app
- Poor authorization and authentication
- Insecure file system permissions
- Unauthorized access to paid-for resources
- Unprotected content provider
- Weak or broken cryptography
- Leaking confidential information to logs

Insecure Data Storage

Generally a result of:

Sensitive data left unprotected
Applies to locally stored data + cloud synced
Not encrypting data
Caching data not intended for long-term storage
Weak or global permissions
Not leveraging platform best-practices

- Confidentiality of data lost
- Credentials disclosed
- Privacy violations
- Non-compliance



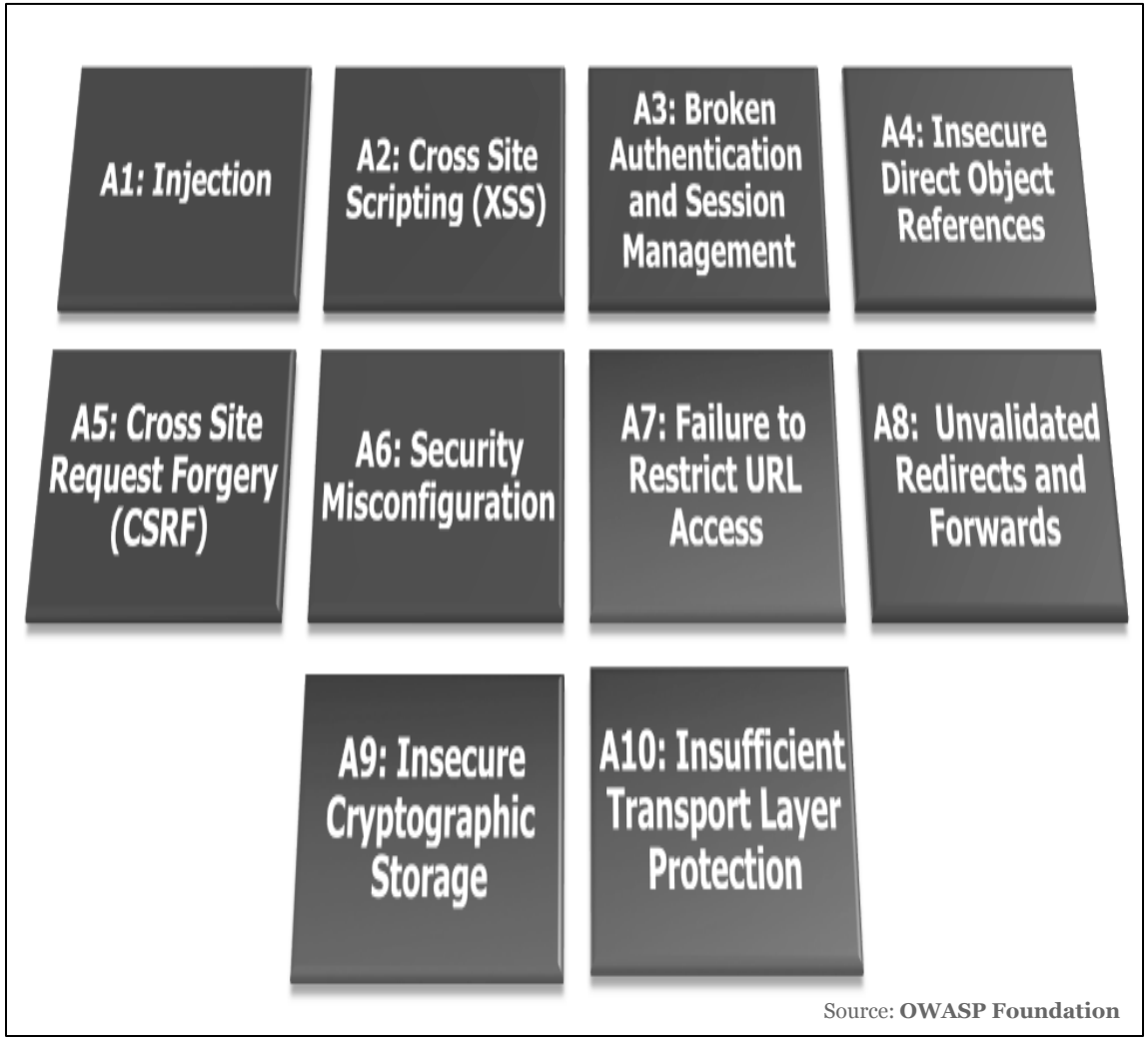

```
public void saveCredentials(String userName, String password) {

    SharedPreferences credentials = this.getSharedPreferences(
            "credentials", MODE_WORLD_READABLE);
    SharedPreferences.Editor editor = credentials.edit();
    editor.putString("username", userName);
    editor.putString("password", password);
    editor.putBoolean("remember", true);
    editor.commit();
}
```

Very Bad

Convenient!

Source: **OWASP Foundation**

Weak Server Side Controls



Insufficient Transport Layer Protection

Generally a result of:

Complete lack of encryption for transmitted data
No server side authentication
Weakly encrypted data in transit
Strong encryption, but ignoring security warnings, such as certificate errors or falling back to plain text after failures

- Man-in-the-middle attacks
- Tampering with data in transit
- Confidentiality of data lost

Client Side Injection

Apps using browser libraries
XSS, SQL and HTML Injection
Abusing phone dialer + SMS
Abusing in-app payments

- Device compromise
- Paid-for resources fraud
- Privilege escalation

Access to paid-for resources

```
public class SmsJSInterface implements Cloneable {

    Context mContext;

    public SmsJSInterface(Context context) {

        mContext = context;
    }

    public void sendSMS(String phoneNumber, String message) {

        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message, null, null);
    }
```

Garden-variety cross-site scripting (XSS)

```
@Override
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.demo);
    context = this.getApplicationContext();
    webView = (WebView) findViewById(R.id.demoWebView);
    webView.getSettings().setJavaScriptEnabled(true);
    webView.addJavascriptInterface(new SmsJSInterface(this),
            "smsJSInterface");
    GetSomeInfo getInfo = new GetSomeInfo();
    getInfo.execute(null, null);
}

public String generateHTML(String untrustedData) {

    return "<b>Check this out!</b><br>" + untrustedData;
}
```

Source: **OWASP Foundation**

Poor Authorization & Authentication

Mobile and architecture problem
Relying on immutable, potentially compromised values (IMEI, IMSI, UUID)
Hardware identifiers persist across data wipes and factory resets
Contextual information is not foolproof

- Privilege escalation
- Unauthorized access

```
if (dao.isDevicePermanentlyAuthorized(deviceID)) {
    int newSessionToken = LoginUtils.generateSessionToken();
    dao.openConnection();
    dao.updateAuthorizedDeviceSession(deviceID,
            sessionToken, LoginUtils.getTimeMilliseconds());
    bean.setSessionToken(newSessionToken);
    bean.setUserName(dao.getUserName(sessionToken));
    bean.setAccountNumber(dao.getAccountNumber(sessionToken));
    bean.setSuccess(true);
    return bean;
}
```

Source: **OWASP Foundation**

Improper Session Handling

Generally a result of:

Mobile app sessions are generally MUCH longer (convenience and usability)
Apps maintain sessions via (HTTP cookies, OAuth tokens, SSO authentication services)
Using a device identifier as a session token is a bad idea

- Privilege escalation
- Unauthorized access
- Circumvent licensing and payment

Security Decisions Via Untrusted Inputs

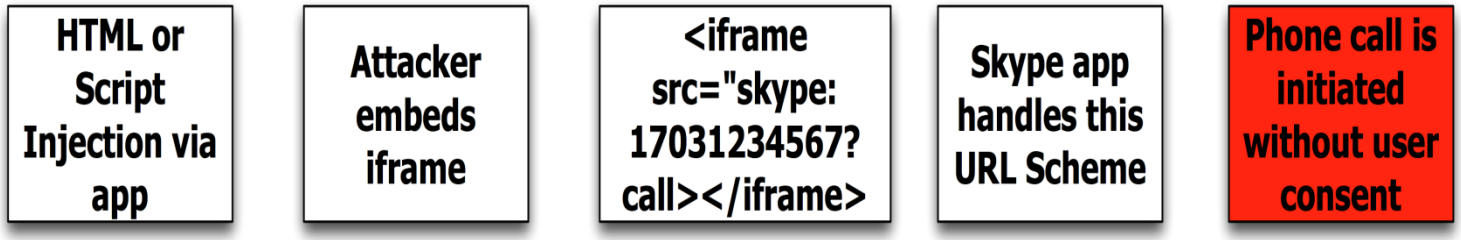
Can be leveraged to bypass permissions and security models

Similar but different depending on platform (iOS- abusing URL Schemes, Android- abusing Intents)

Several attack vectors (malicious apps, client side injection)

- Consuming paid resources
- Data exfiltration
- Privilege escalation

Skype iOS URL Scheme Handling Issue



Source: **OWASP Foundation**

Improper Session Handling

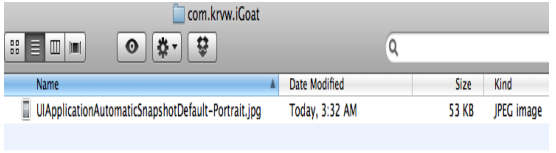
Not disabling platform features and programmatic flaws

Sensitive data ends up in unintended places (web caches, keystroke logging, screenshots (ie- iOS backgrounding), logs (system, crash), temp directories)

Understanding of third-party libraries activities with user data in your app (ie- ad networks, analytics)

- Data retained indefinitely
- Privacy violations

Screenshots



Logging

```
try {
    userInfo = client.validateCredentials(userName, password);
    if (userInfo.get("success").equals("true"))
        launchHome(v);
    else {
        Log.w("Failed login", userName + " " + password);
    }

} catch (Exception e) {
    Log.w("Failed login", userName + " " + password);
}
```

Source: **OWASP Foundation**

Broken Cryptography

Generally a result of:

Broken implementations using strong crypto libraries
Custom, easily defeated crypto implementations
Encoding != encryption
Obfuscation != encryption
Serialization != encryption

- Confidentiality of data lost
- Privilege escalation
- Circumvent business logic

```
ldc literal_876:"QlVtT0JoVmY2N2E="
invokestatic byte[] decode( java.lang.String ) // Base 64
invokespecial_lib java.lang.String.<init> // pc=2
astore 8

private final byte[]
com.picuploader.BizProcess.SendRequest.routine_12998
   (com.picuploader.BizProcess.SendRequest, byte[], byte[]
);
 {
  enter
  new_lib net.rim.device.api.crypto.TripleDESKey
```

Source: **OWASP Foundation**

Sensitive Information Disclosure

Insecure data Storage is different from sensitive information disclosure (embedded/hardcoded)

Apps can be reverse engineered with relative ease

Code obfuscation raises the bar, but doesn't eliminate the risk

Commonly found "treasures" (API keys, passwords, sensitive business logic)

- Credentials disclosed
- Intellectual property exposed

```
if (rememberMe)
    saveCredentials(userName, password);
//our secret backdoor account
if (userName.equals("all_powerful")
        && password.equals("iamsosmart"))
    launchAdminHome(v);
```

```
public static final double SECRET_SAUCE_FORMULA = (1.2344 * 4.35 - 4 + 1.442) * 2.221;
```

Source: **OWASP Foundation**

Reality Check

- 51% of organizations experienced data loss from employee use of insecure mobile devices, within the past year.
- 59% experienced an increase in malware infections as a result of insecure mobile devices in the workplace.
- 59% report that employees circumvent or disengage mobile security features, such as passwords and key locks, on corporate and personal mobile devices.

True Security®

Don't know the security of your mobile apps?

Submit your mobile app to Safe Frontier for security assessment today!

http://safefrontier.com/content/mobile-application-security-services

Safe Frontier provides security assessment of Google's Android and Apple iOS applications.

True Security®

Safe Frontier

<SAFEBIZAPP I> certification signifies that the software has been independently assessed and it meets or exceeds the certification criteria.

<SAFEBIZAPP II> certification is recommended for critical applications. It signifies that the software has been independently assessed and it meets or exceeded the certification criteria.

To learn more, follow the link and then navigate to the Methodology Tab:
http://safefrontier.com/content/mobile-application-security-services

<SAFEGOVAPP I> certification is recommended for agencies and government contractors. It signifies that the software has been independently assessed and it meets or exceeded the certification criteria.

Q/A

safefrontier.com
sales@safefrontier.com

Thank you