

Android Security Threats

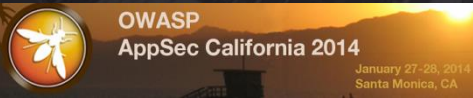
Alexey Chernin, Software Developer



Based on:



“OWASP AppSecUSA 2011: OWASP Mobile Top 10 Risks”
- Jack Mannino ([nVisium](#)), Zach Lanier ([Intrepidus Group](#)),
Mike Zusman ([Carve Systems](#))



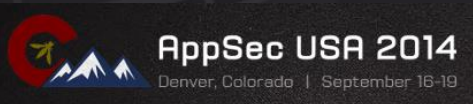
“OWASP Top 10 Mobile Risks: 2014 Reboot”
- Jack Mannino ([nVisium](#)), Jason Haddix ([HP Fortify](#))



“OWASP Mobile Top Ten 2014 - The New Lack of Binary Protection Category”
- Winston Bond ([Arxan Technologies](#))



“If 6,000 Mobile Malware Applications Could Talk! Ow, They Do, And A Lot!”
- Matias Madou, Daan Raman ([Nviso](#))



“Mobile Security Attacks: A Glimpse from the Trenches”
- Yair Amit, Adi Sharabani ([Skycure](#))

Based on:



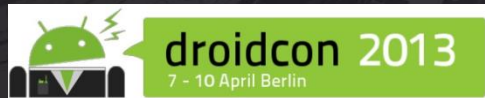
“Mobile OWASP TOP TEN”

- Leonid Igolnik ([Oracle](#))



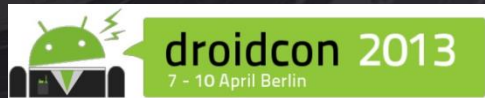
“Security under Android”

- Raimund Genes ([Trend Micro](#))



“Android Security Overview”

- Matthias Lange



“Security Testing Android with Mercury”

- Daniel Bradberry ([MWR Info Security](#))



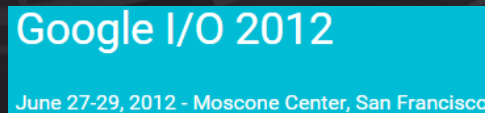
“When Security Gets in the Way: Pen Testing Apps that use Certificate Pinning”

- Alban Diquet, Justine Osborne



“Deep Dive into Android Security”

- Aleksandar Gargenta ([Marakana, Inc.](#))



“Security and Privacy in Android Apps”

- Jon Larimer, Kenny Root ([Android Team](#))

Captain Obvious Slide

Keep in mind:

- ✓ There is no silver bullet in security
- ✓ Always will be trade-off between convenience and security
- ✓ It's not if, but when: even all the possible approaches together will not give you a 100% guarantee of safety
- ✓ Security is not a feature
- ✓ Weak security is better than nothing



OWASP

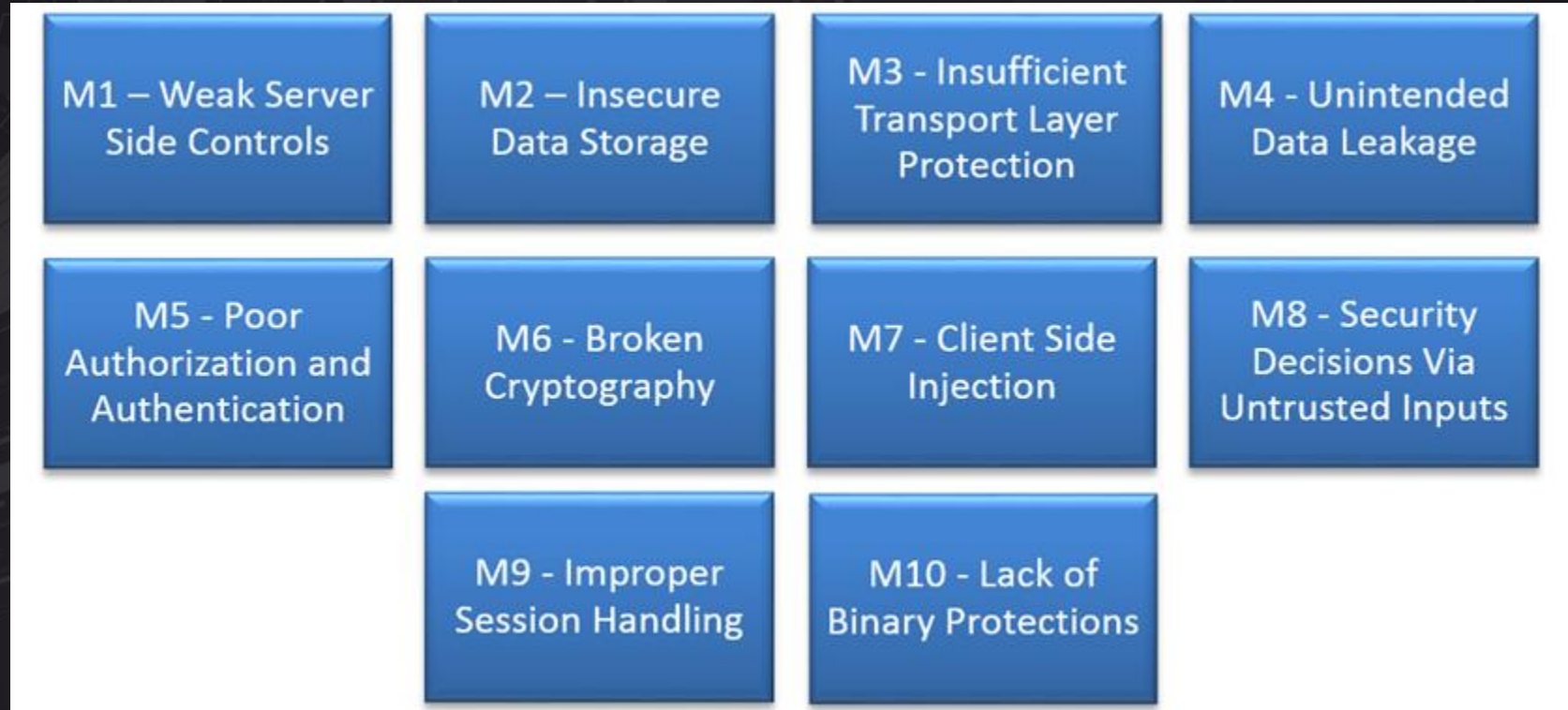


The **Open Web Application Security Project (OWASP)** is an online community dedicated to web application security. The OWASP community includes corporations, educational organizations and individuals from around the world. This community works to create freely-available articles, methodologies, documentation, tools, and technologies.

OWASP projects:

- **OWASP Top Ten:** The goal of the Top 10 project is to raise awareness about application security by identifying some of the most critical risks facing organizations.
- **OWASP Development Guide:** The Development Guide provides practical guidance and includes J2EE, ASP.NET, and PHP code samples.
- **OWASP Testing Guide:** The OWASP Testing Guide includes a "best practice" penetration testing framework that users can implement in their own organizations and a "low level" penetration testing guide that describes techniques for testing most common web application and web service security issues.

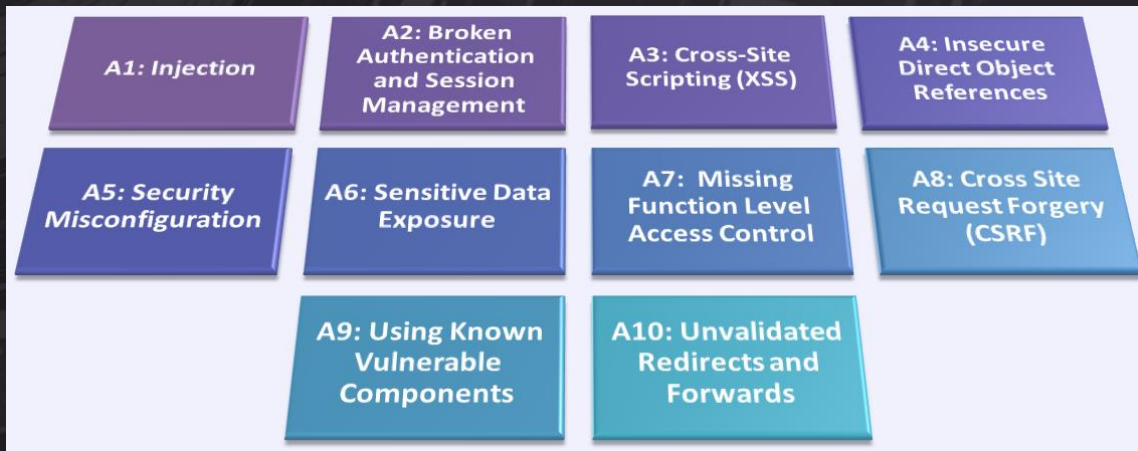
OWASP Mobile Top 10 Risks



https://www.owasp.org/index.php/OWASP_Mobile_Security_Project

OWASP Mobile Top 10 Risks: M1

The Ten Most Critical Web Application Security Risks



https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Cloud Top 10 Security Risks



OWASP Mobile Top 10 Risks: M1

Factors that have lead to a proliferation of server-side vulnerabilities:

- Rush to market
- Lack of security knowledge because of the new-ness of the languages
- Easy access to frameworks that don't prioritize security
- Higher than average outsourced development
- Lower security budgets for mobile applications
- Assumption that the mobile OS takes full responsibility for security
- Weakness due to cross-platform development and compilation

OWASP Mobile Top 10 Risks: M2

M2: Insecure Data Storage

- **Shared Preferences:** Store private primitive data in key-value pairs
- **Internal Storage:** Store private data on the device memory
- **External Storage:** Store public data on the shared external storage
- **SQLite Databases:** Store structured data in a private database
- **Network Connection:** Store data on the web with your own network server

```
72  SharedPreferences preferences = PreferenceManager
73      .getDefaultSharedPreferences(this);
74  preferences.edit().putString(TAG, secret).apply();
```

✓ **Around 113 smartphones are stolen or lost each minute in the U.S**

Impact:



Confidentiality of data lost



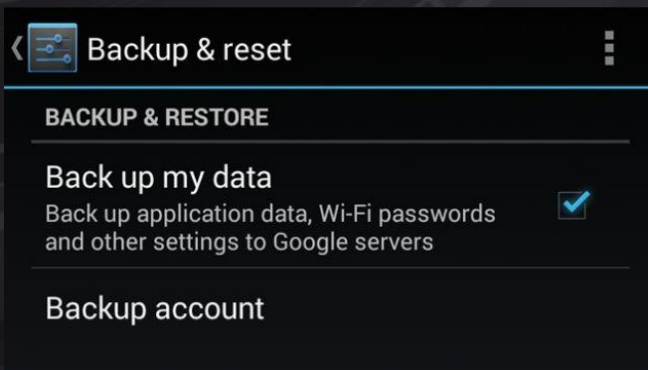
Credentials disclosed

OWASP Mobile Top 10 Risks: M2

Prevention Tips

- Store ONLY what is absolutely required
- Separate storage on the level of "secret"
- Never use public storage areas (SD card)
- Leverage secure containers and platform provided file encryption APIs
- Do not grant files world readable or world writeable permissions
- Define the data lifecycle
- Data kill switch

Google Knows the Wi-Fi Passwords of All Android Users



Lost device – change own Wi-Fi password and don't forget about friends (and maybe work)
Same for disclosed google account
Same for malware attack

Database Encryption

SQLCipher

<https://www.zetetic.net/sqlcipher/open-source/>



Benefits:

- Strong encryption (256-bit AES)
- Mature technology
- Maintained and supported by its developers and the open source community
- Supports virtually the same API as standard Android database functions
 - ✓ `getWritableDatabase("my_secure_key")`

Consequences:

- APK size will be increased substantially
- Database performance will decrease
- Secure management of the encryption key (or password) is required
- Use of strong encryption has legal considerations in some jurisdictions

<https://androidbycode.wordpress.com/2015/02/18/android-database-encryption-using-sqlcipher/>

Shared preferences encryption

Secure-preferences

<https://github.com/scottyab/secure-preferences>

Shared preference wrapper that encrypts the values of Shared Preferences using AES 128, CBC, and PKCS5 padding with integrity checking in the form of a SHA 256 hash. Each key is stored as a one way SHA 256 hash. Both keys and values are base64 encoded before storing into preferences xml file.

```
//Default SharedPreferences file:
SharedPreferences defaultPrefs = new SecurePreferences(context);

//Custom SharedPreferences file:
SharedPreferences customPrefs
    = new SecurePreferences(context, secretKey, "custom_prefs.xml");

//With user password:
SecurePreferences passwordPrefs
    = new SecurePreferences(this, "password", "pass_prefs.xml");
try {
    passwordPrefs.handlePasswordChange("password", this);
} catch (GeneralSecurityException e) {
    e.printStackTrace();
}
```


File encryption

Conceal

<https://github.com/facebook/conceal>

facebook

Conceal provides a set of Java APIs to perform cryptography on Android. It was designed to be able to encrypt large files on disk in a fast and memory efficient manner. The major target for this project is typical Android devices which run old Android versions, have low memory and slower processors.

```
// Creates a new Crypto object with default implementations of  
// a key chain as well as native library.  
Crypto crypto = new Crypto(  
    new SharedPrefsBackedKeyChain(context),  
    new SystemNativeCryptoLibrary());  
  
OutputStream fileStream = new BufferedOutputStream(new FileOutputStream(file));  
  
// Creates an output stream which encrypts the data as  
// it is written to it and writes it out to the file.  
OutputStream outputStream = crypto.getCipherOutputStream(fileStream, entity);  
  
// Write plaintext to it.  
outputStream.write(plainTextBytes);  
outputStream.close();
```

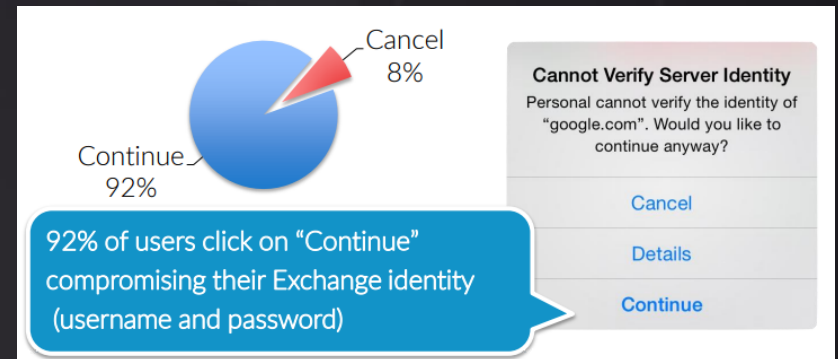
OWASP Mobile Top 10 Risks: M3

M3: Insufficient Transport Layer Protection

- Complete lack of encryption for transmitted data
- Weakly encrypted data in transit
- Strong encryption, but ignoring security warnings

10.1%

of scanned
networks
pose a threat



Impact:



Man-in-the-middle attacks



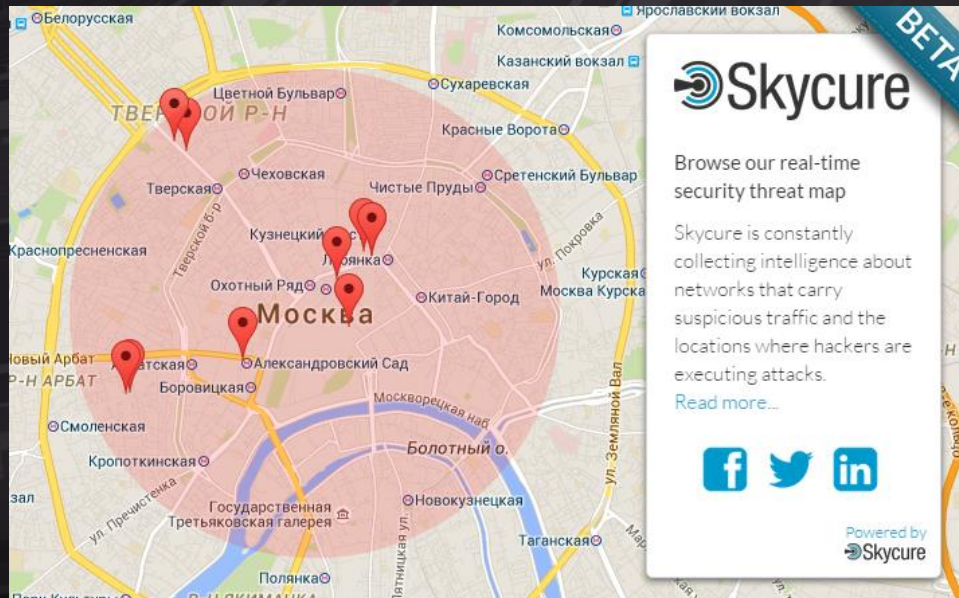
Packet sniffing

OWASP Mobile Top 10 Risks: M3

Prevention Tips

- Make sure app never falls back to HTTP
- Even the mobile network can be compromised – GSM can be broken or spoofed
- Ensure SSL certs are checked preventing Man in a Middle attack
- Implement certificate pinning:

https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning



<https://maps.skycure.com>

<https://play.google.com/store/apps/details?id=com.skycure.skycure>

Certificate Pinning

Two common ways to do SSL on Android:

1. Bundle keystore with app
2. Create TrustManager with keystore
3. Init SSLContext with TrustManager
4. Get SSLSocketFactory from SSLContext
5. Create HTTPSURLConnection and set to use SSLSocketFactory
5. Create new Scheme with SSLSocketFactory and register with SchemeRegistry

```
HttpsURLConnection urlConn = (HttpsURLConnection)url.openConnection();  
urlConn.setSSLSocketFactory(sslContext.getSocketFactory());
```

```
SSLSocketFactory sf = new SSLSocketFactory(pinningSSLContext);  
Scheme httpsScheme = new Scheme("https", 443, sf);  
SchemeRegistry schemeRegistry = new SchemeRegistry();  
schemeRegistry.register(httpsScheme);
```

Read more:

https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning

A standalone library project for certificate pinning on Android:

<https://github.com/moxie0/AndroidPinning>

Square Okhttp library: CertificatePinner

<http://square.github.io/okhttp/javadoc/com/squareup/okhttp/CertificatePinner.html>

OWASP Mobile Top 10 Risks: M4

M4: Unintended Data Leakage

- Mix of not disabling platform features and programmatic flaws.
- Sensitive data ends up in unintended places:
 - Web caches
 - Keystroke logging
 - Screenshots
 - Logs
 - Temp directories
- Understand what 3rd party libraries in your apps are doing with user data.



```
75 public String getSecret() {  
76     String secret = generateSecret();  
77     Log.d(TAG, secret); //TODO remove in production!!!  
78     persistSecret(secret);  
79     return secret;  
80 }
```

One more common case:

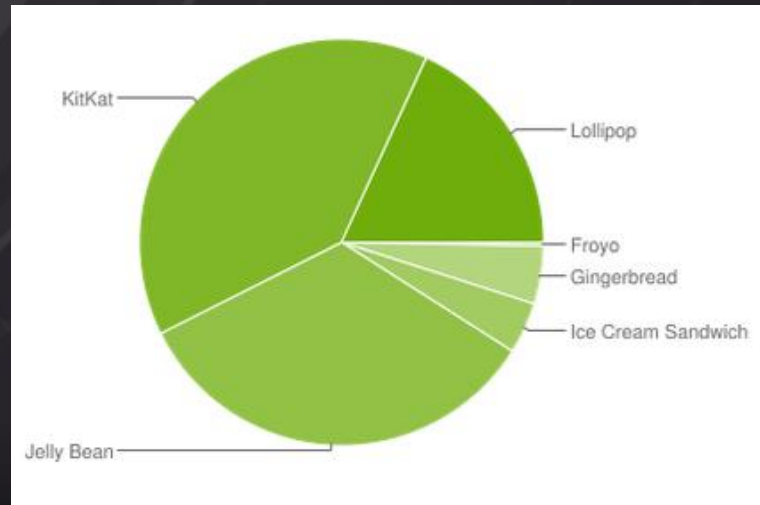
- Test server -> ignore SSL errors
- Production -> same

OWASP Mobile Top 10 Risks: M4

Prevention Tips

- Never log credentials or other sensitive data to system logs
- Remove sensitive data before screenshots are taken, disable keystroke logging per field, and utilize anti-caching directives for web content
- Debug your apps before releasing them to observe files created, written to, or modified in any way
- Carefully review any third party libraries you introduce and the data they consume
- Test your applications across as many platform versions as possible

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	4.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	4.1%
4.1.x	Jelly Bean	16	13.0%
4.2.x		17	15.9%
4.3		18	4.7%
4.4	KitKat	19	39.3%
5.0	Lollipop	21	15.5%
5.1		22	2.6%



August 3, 2015

OWASP Mobile Top 10 Risks: M5

M5: Poor Authorization and Authentication

- Relying on phone number
- IMEI, IMSI, UUID etc
- Small key space – can be brute forced
- Client side authorization instead of server
- Servers have to authenticate themselves to a client

```
public String generateSecret() {  
    //Considered unreliable because it can sometimes be null or can change  
    //Looks like: 9774d56d682e549c  
    String secret = Secure.getString(getContentResolver(), Secure.ANDROID_ID);  
    //Only for Android devices with phone use, requires READ_PHONE_STATE permission  
    TelephonyManager telephonyManager = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);  
    secret += telephonyManager.getDeviceId(); //Looks like: 359881030314356  
    //Only for Android devices with wifi use, requires ACCESS_WIFI_STATE permission  
    WifiManager wifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);  
    secret += wifiManager.getConnectionInfo().getMacAddress(); //Looks like: 00:11:22:33:44:55  
    //Only for Android devices with bluetooth use, requires BLUETOOTH permission  
    BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();  
    secret += bluetoothAdapter.getAddress(); //Looks like: 43:25:78:50:93:38  
    return secret;  
}
```

OWASP Mobile Top 10 Risks: M5

Prevention Tips

- Unique identifiers only as part of a multi-factor implementation
- Out-of-band doesn't work when it's all the same device
- Never use device ID or subscriber ID as sole authenticator
- Define & enforce password length, strength & uniqueness
- Differentiate client-side passcode vs. server authentication

```
89 public String generateSecret() { //Just for example!
90     String secret = "35" + //We make this look like a valid IMEI
91         Build.BOARD.length() % 10+ Build.BRAND.length() % 10
92         + Build.CPU_ABI.length() % 10 + Build.DEVICE.length() % 10
93         + Build.DISPLAY.length() % 10 + Build.HOST.length() % 10
94         + Build.ID.length() % 10 + Build.MANUFACTURER.length() % 10
95         + Build.MODEL.length() % 10 + Build.PRODUCT.length() % 10
96         + Build.TAGS.length() % 10 + Build.TYPE.length() % 10
97         + Build.USER.length() % 10 ; //13 digits
98     return mixWithSomething(secret);
99 }
```

✓ `UUID.randomUUID().toString()`

OWASP Mobile Top 10 Risks: M6

M6: Broken Cryptography

- Two primary categories:
 - Broken implementations using strong crypto libraries
 - Custom, easily defeated crypto implementations
- Encoding != encryption
- Obfuscation != encryption
- Serialization != encryption

Prevention Tips

- Storing the key with the encrypted data negates everything
- Leverage battle-tested crypto libraries vice writing your own
- Take advantage of what your platform already provides:

<http://developer.android.com/training/articles/security-tips.html>

<https://source.android.com/devices/tech/security/index.html>

OWASP Mobile Top 10 Risks: M7

M7: Client Side Injection

- Apps using browser libraries:
 - Pure web apps
 - Hybrid web/native apps
- Some familiar faces:
 - XSS and HTML Injection
 - SQL Injection
- New and exciting twists:
 - Abusing phone dialer + SMS
 - Abusing in-app payments

```
public void deleteCaseValue(String paramString) {  
    SQLiteDatabase localSQLiteDatabase = this.mDb;  
    String str = "DELETE FROM case_values WHERE _id = "  
        + paramString;  
    localSQLiteDatabase.execSQL(str);  
}
```

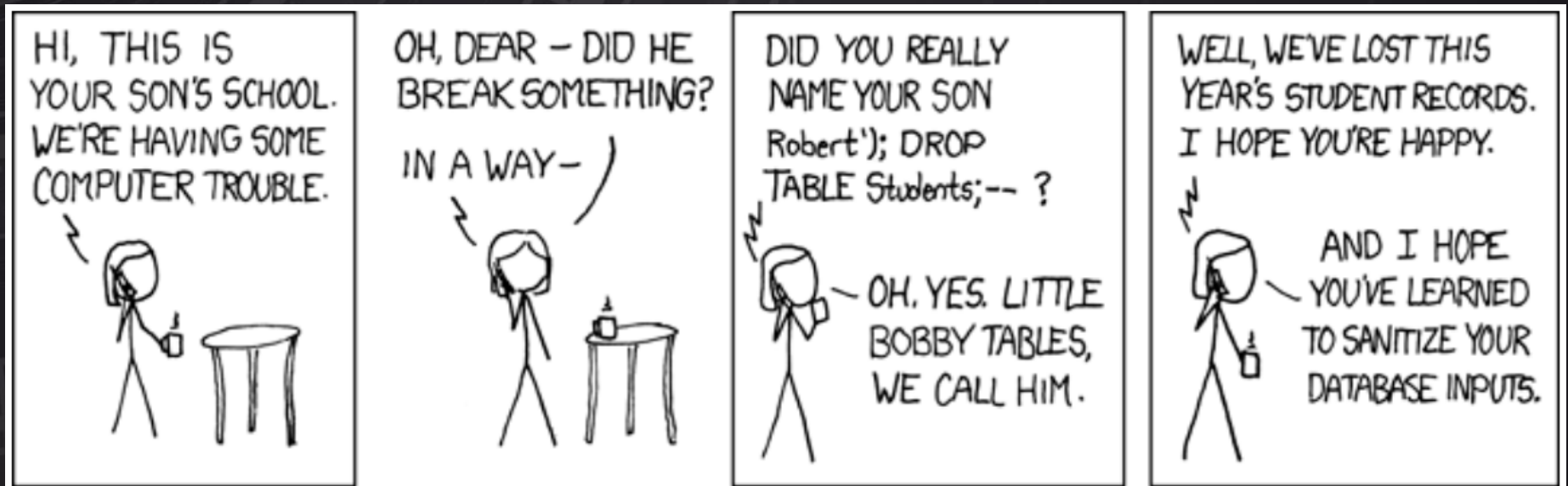
```
@Override  
public void onCreate(Bundle savedInstanceState) {  
  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.demo);  
    context = this.getApplicationContext();  
    webView = (WebView) findViewById(R.id.demoWebView);  
    webView.getSettings().setJavaScriptEnabled(true);  
    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;  
}
```

```
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);  
    }  
}
```

OWASP Mobile Top 10 Risks: M7

Prevention Tips

- Sanitize or escape untrusted data before rendering or executing it
- Use prepared statements for database calls... concatenation is still bad, and always will be bad
- Minimize the sensitive native capabilities tied to hybrid web functionality



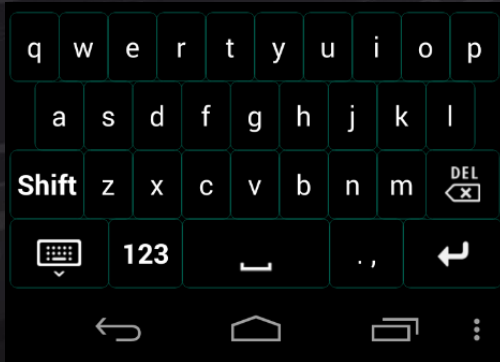
OWASP Mobile Top 10 Risks: M8

M8: Security Decisions Via Untrusted Inputs

- Reliance on files, settings, network resources or other inputs which may be modified.

Prevention Tips

- Check caller's permissions at input boundaries
- Prompt the user for additional authorization before allowing
- Where permission checks cannot be performed, ensure additional steps required to launch sensitive actions



- ✓ Custom keyboard with shuffle keys for EditText and WebView

OWASP Mobile Top 10 Risks: M8

Your mobile application can accept data from all kinds of sources. In most cases this will be an Inter Process Communication (IPC) mechanism. In general try and adhere to the following IPC design patterns:

- If there is a business requirement for IPC communication, the mobile application should restrict access to a white-list of trusted applications
- Sensitive actions which are triggered through IPC entry points should require user interaction before performing the action
- All input received from IPC entry points must undergo stringent input validation in order to prevent input driven attacks
- Do not pass any sensitive information through IPC mechanisms, as it may be susceptible to being read by third party applications under certain scenarios

OWASP Mobile Top 10 Risks: M9

M9: Improper Session Handling

- Lack of Adequate Timeout Protection, mobile app sessions are generally MUCH longer because of convenience and usability
- Failure to Invalidate Sessions on the Backend
- Failure to Properly Rotate Cookies
- Insecure Token Creation, bad idea = using a device identifier as a session token

Prevention Tips

- Don't be afraid to make users re-authenticate every so often
 - 15 minutes for high security applications
 - 30 minutes for medium security applications
 - 1 hour for low security applications
- Ensure that tokens can be revoked quickly in the event of a lost/stolen device
- Utilize high entropy, tested token generation resources

OWASP Mobile Top 10 Risks: M10

M10: Lack of Binary Protections

- Software in untrusted environments is exposed to reverse-engineering, analysis, modification, and exploitation by attackers
- Attackers can directly access the binary and compromise its integrity with various tools and techniques
- Attackers may cause brand, revenue, or IP loss through reverse-engineering

Impact:

- Compromise (disable, circumvent) of security controls, e.g., authentication, encryption, license management / checking, DRM, root detection
- Exposure of sensitive application information, e.g., keys, certificates, credentials, metadata
- Tampering with critical business logic, control flows, and program operations
- Insertion of malware or exploits in the application and repackaging
- Exposure of application internals (logic, vulnerabilities) via reverse-engineering
- IP theft (e.g., proprietary algorithms) via reverse-engineering
- Piracy and unauthorized distribution

Alternative App Stores

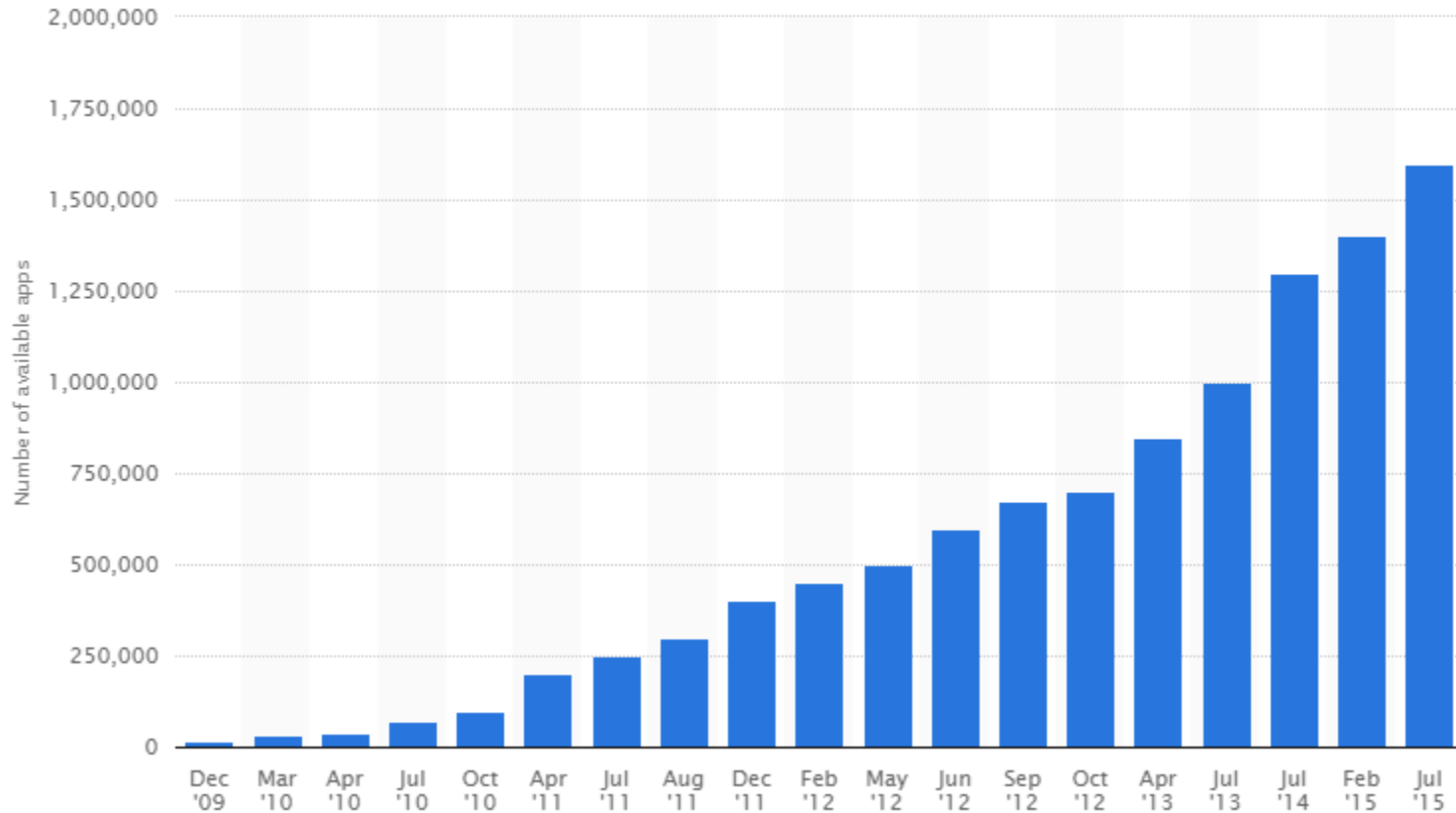
The list of most popular alternative app stores:

- SlideMe
- Amazon Appstore
- GetJar
- Opera Mobile Store
- AppsLib
- AppBrain
- Anzhi
- Aptoide
- Yandex
- Mobango
- AppsZoom
- Baidu
- Samsung Apps
- 1Mobile
- AppChina
- AndroidPit
- Soc.io Mall
- PandaApp
- Tencent App Gem
- Wandoujia
- HiAPK
- SK T-Store
- D.cn Games Center



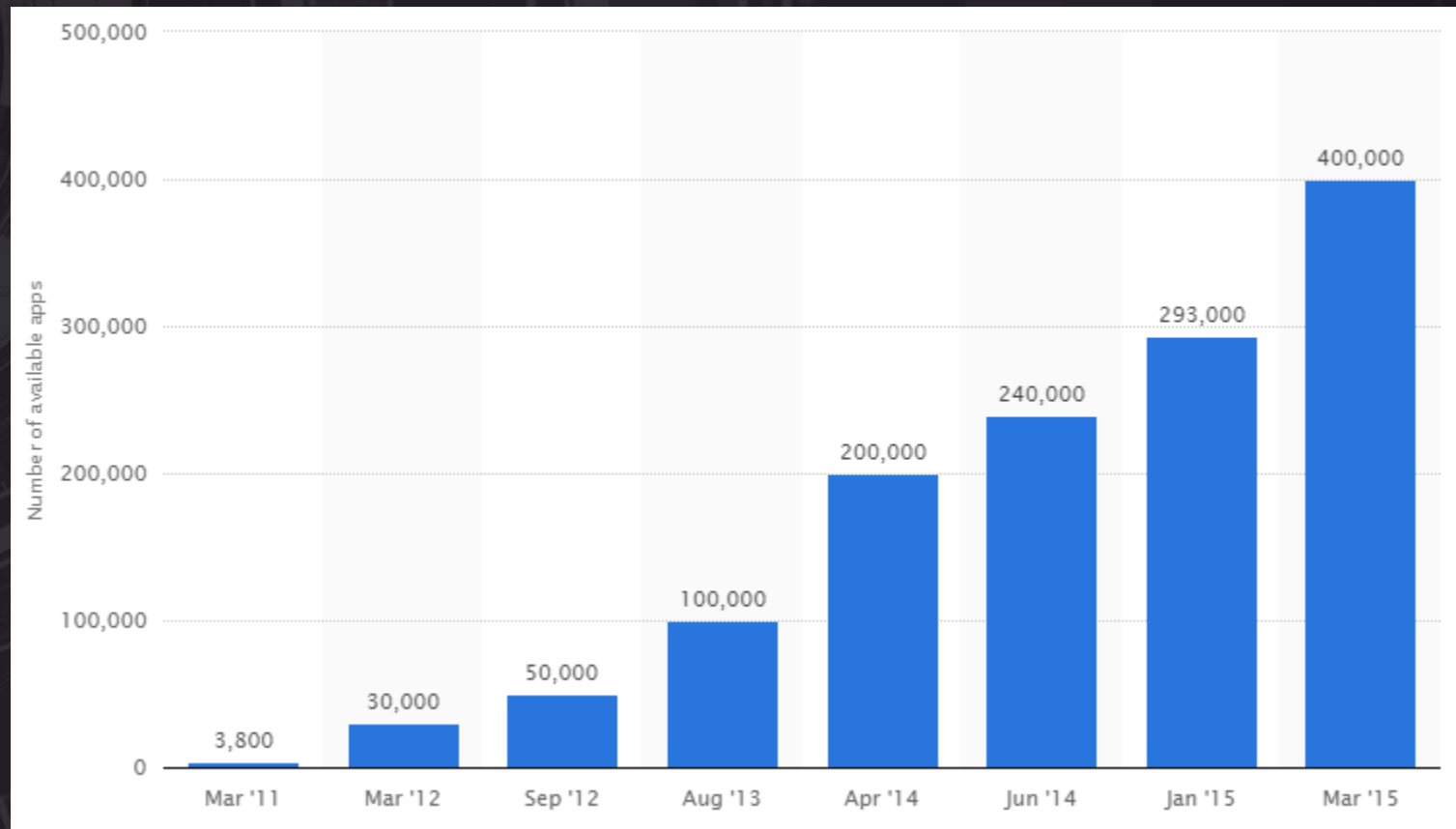
<http://www.mobyaffiliates.com/blog/a-list-of-alternative-app-stores-for-distributing-your-app-or-mobile-game/>

Google Play Store



Number of available applications in the Google Play Store
from December 2009 to July 2015 (www.statista.com)

Amazon Appstore



Number of available apps in the Amazon Appstore
from March 2011 to January 2015 (www.statista.com)

Android App Stores in China

- China has the world's largest Android population with 270 million active users. However, Google Play is only accessible by about 30% of them, and third-party app stores are thus used by 70% of them for daily Android apps (applications) discovery.
- The trustworthy level of almost all the top Android app stores in China is fairly low.

“Which Android App Store Can be Trusted in China?”

- Yi Ying Ng, Hucheng Zhou, Zhiyuan Ji, Huan Luo, Yuan Dong

<http://research.microsoft.com/pubs/230934/PID3226063.pdf>



"WHAT IT TAKES TO WIN In the Chinese App Market"

- inmobi.com, 2013

http://info.inmobi.com/rs/inmobi/images/What_it_Takes_Win_Chinese_App_Market.pdf

Google Play LVL

A licensing service that lets you enforce licensing policies for applications that you publish on Google Play. With Google Play Licensing, your application can query Google Play at run time to obtain the licensing status for the current user, then allow or disallow further use as appropriate.

<http://developer.android.com/google/play/licensing/index.html>

Adding license verification with the LVL involves these tasks:

- Adding the licensing permission your application's manifest.
- Implementing a Policy — you can choose one of the full implementations provided in the LVL or create your own.
- Implementing an Obfuscator, if your Policy will cache any license response data.
- Adding code to check the license in your application's main Activity.

`com.android.vending.licensing`

- AESObfuscator
- LicenseChecker
- LicenseCheckerCallback
- ServerManagedPolicy

Google Play LVL

```
// Google Play LVL Sample
public class MainActivity extends Activity {
    private static final String BASE64_PUBLIC_KEY = "PUBLIC KEY";
    private static final byte[] SALT = new byte[] {...};
    private LicenseCheckerCallback mLicenseCheckerCallback;
    private LicenseChecker mChecker;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        // Getting the ID of a device
        String deviceId= Secure.getString(getContentResolver(),Secure.ANDROID_ID);
        // Initializing the license checker classes
        mLicenseCheckerCallback = new MyLicenseCheckerCallback();
        mChecker = new LicenseChecker(this, new ServerManagedPolicy(this,
            new AESObfuscator(SALT, getPackageName(), deviceId)),
            BASE64_PUBLIC_KEY);
        // Checking the license
        mChecker.checkAccess(mLicenseCheckerCallback); // ← уязвимая часть
    }
}
```


Hacking Techniques

Facilities to analysis and modification:

- Apktool <http://ibotpeaches.github.io/Apktool>
- Dex2Jar <https://github.com/pxb1988/dex2jar>
- JD-GUI <http://jd.benow.ca>

Automatic facilities:

AntiLVL http://androidcracking.blogspot.ru/p/antilvl_01.html



Okay, prepare your apps

AntiLVL

- Created to automatic hacking:
 - Mechanisms of licensing
 - Checks certificates
 - Mechanisms determination debugging mode
 - Control integrity
- Supports:
 - Google Play LVL
 - Amazon DRM
 - Verizon DRM
- It is possible to configure to work with new / other mechanisms of protection and licensing

How it works:

- Finds and replaces calls Licensing API, Certificate API, check Debug API on its realization
- Can modify DateTime API, to shutdown checks based on binding to system time
- Proxies reflection calls

DexProtector



<https://dexprotector.com>

```
Object localObject = ha.kkbab(ha.pnbab(this), R.MainActivity.1.access$300("植讞變  
مم ㄱ 字 三 가"));
```

```
ha.mlmbo(this, ha.ilbab(this, null), 4140);
```

[illegible]

```
ha.ikcab(this);
```

Comparison Obfuscators

	Proguard	DexGuard (Saikoa)	DexProtector (Licel)
Переименование	Да	Да	Нет
Оптимизация кода	Да	Да	Да
Шифрование кода	Нет	Частично	Полностью (весь classes.dex)
Шифрование ресурсов	Нет	Да	Да
Проверка целостности	Нет	Да	Да
Защита API (скрытие)	Нет	Частично (через Reflection)	Полностью
Защита рекламы и встроенных покупок	Нет	Да	Да
Защита лицензии	Нет	Да	Да
Привязка к магазинам приложений	Нет	Нет	Да
Тест на AntiLVL	Не пройден	Пройден	Пройден
Цена	бесплатно	\$624	\$500

ProGuard

ProGuard is not about obfuscation, but optimization and shrinking

OWASP Mobile Top 10 Risks: M10

Prevention Tips

- Private API keys are called that for a reason...keep them off of the client
- Keep proprietary and sensitive business logic on the server
- Almost never a legitimate reason to hardcode a password (if there is, you have other problems)
- Implement adequate algorithms for:
 - Root detection
 - Checksum controls
 - Certificate pinning controls
 - Debugger detection controls
- Protect these algorithms from:
 - Reverse-engineering
 - Unauthorized code modification

https://www.owasp.org/index.php/OWASP_Reverse_Engineering_and_Code_Modification_Prevention_Project

Your mobile app must be able to:

- Prevent an adversary from reverse-engineering sensitive parts of your app
- Detect at runtime that code modification has occurred
- React appropriately at runtime to integrity violations

Device Fingerprint

- **Location**
- **Android ID**
- **Locale**
- **Time**
- **OS version**
- **Sim serial number**
- **Wi-Fi IP Address**
- **IMEI**
- **Device model**
- **Screen size**
- **Wi-Fi MAC address**



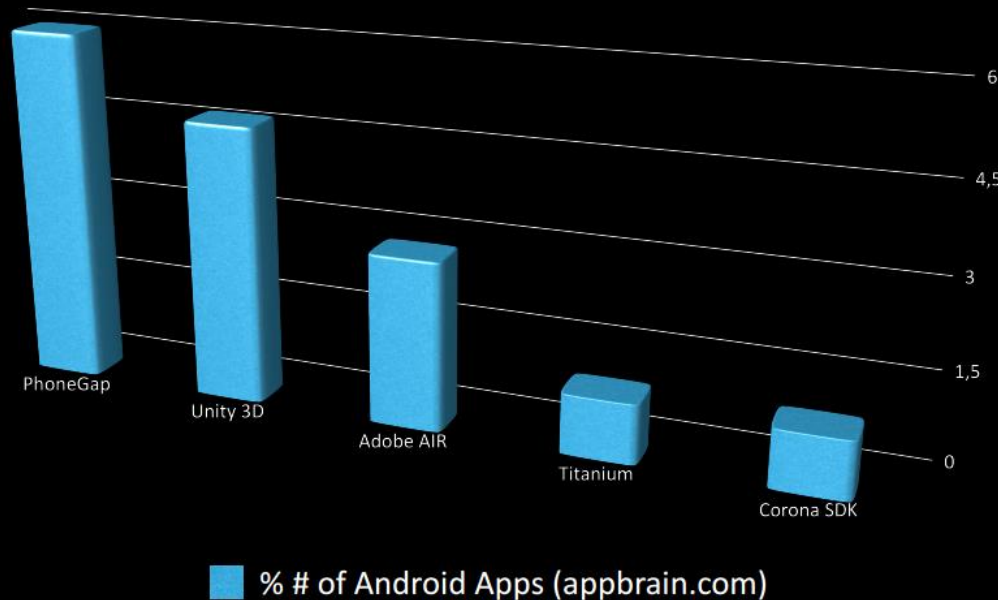
Accumulate Collected Data Checks

```
AuthStateManager am = AuthStateManager.getInstance();  
  
am.updateInternalState(AnomalyDetector.testBuildKeys());  
  
am.updateInternalState(AnomalyDetector.listApplications());  
  
am.updateInternalState(AnomalyDetector.listSuShells());  
  
am.updateInternalState(AnomalyDetector.calcFirmwareChecksum());  
  
am.updateInternalState(AnomalyDetector.getWifiState());
```

The level of paranoia

Cross Platform Frameworks

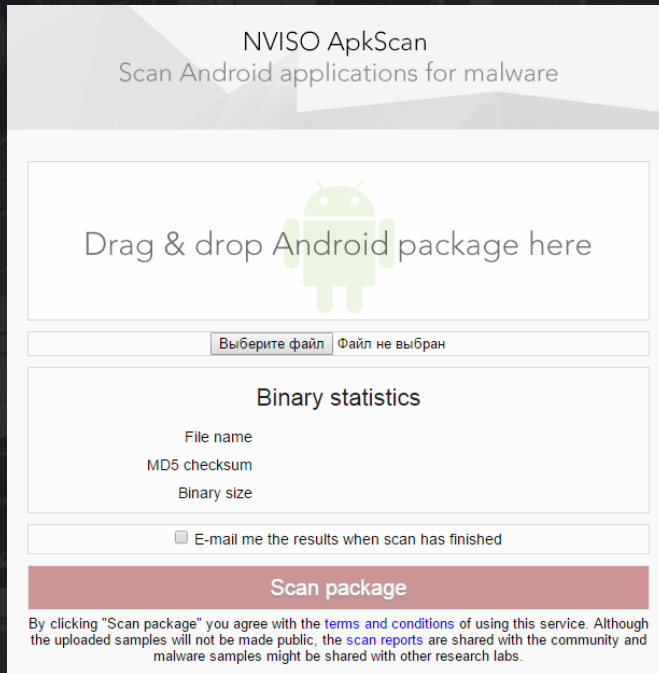
What cross platform frameworks are out there?



“The Nightmare Behind the Cross-Platform Apps Dream”
- Marco Grassi, Sebastián Guerrero Selma (NowSecure)

<https://www.nowsecure.com/blog/2015/04/13/the-nightmare-behind-the-cross-platform-apps-dream/>

APK Analysis



Online malware analysis:

- <https://mars.trendmicro.com>
- <https://apkscan.nviso.be>
- <http://anubis.iseclab.org>
- <http://www.apk-analyzer.net>

Androguard – tool for reverse engineering of applications (goodwares, malwares):

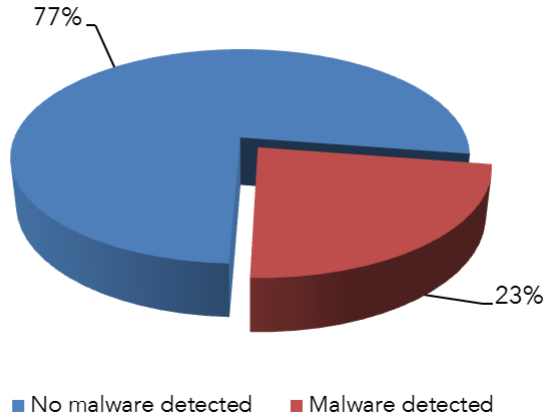
- Disassemble/Decompilation/Modification of DEX/ODEX/APK format
- Measure the efficiency of obfuscators
- Etc..

drozer (formerly **Mercury**) is the leading security testing framework for Android. drozer allows you to search for security vulnerabilities in apps and devices by assuming the role of an app and interacting with the Dalvik VM, other apps' IPC endpoints and the underlying OS.

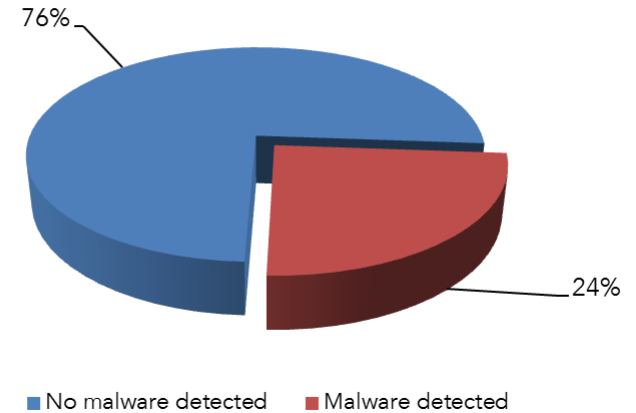


NVISO ApkScan Report 2013

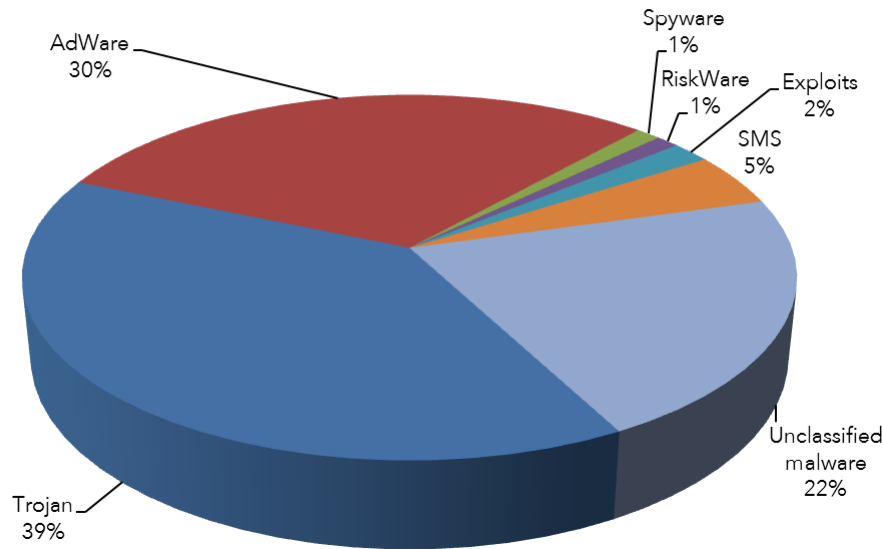
Anti-Virus scan results for user-submitted samples



Anti-Virus scan results for samples downloaded from app stores



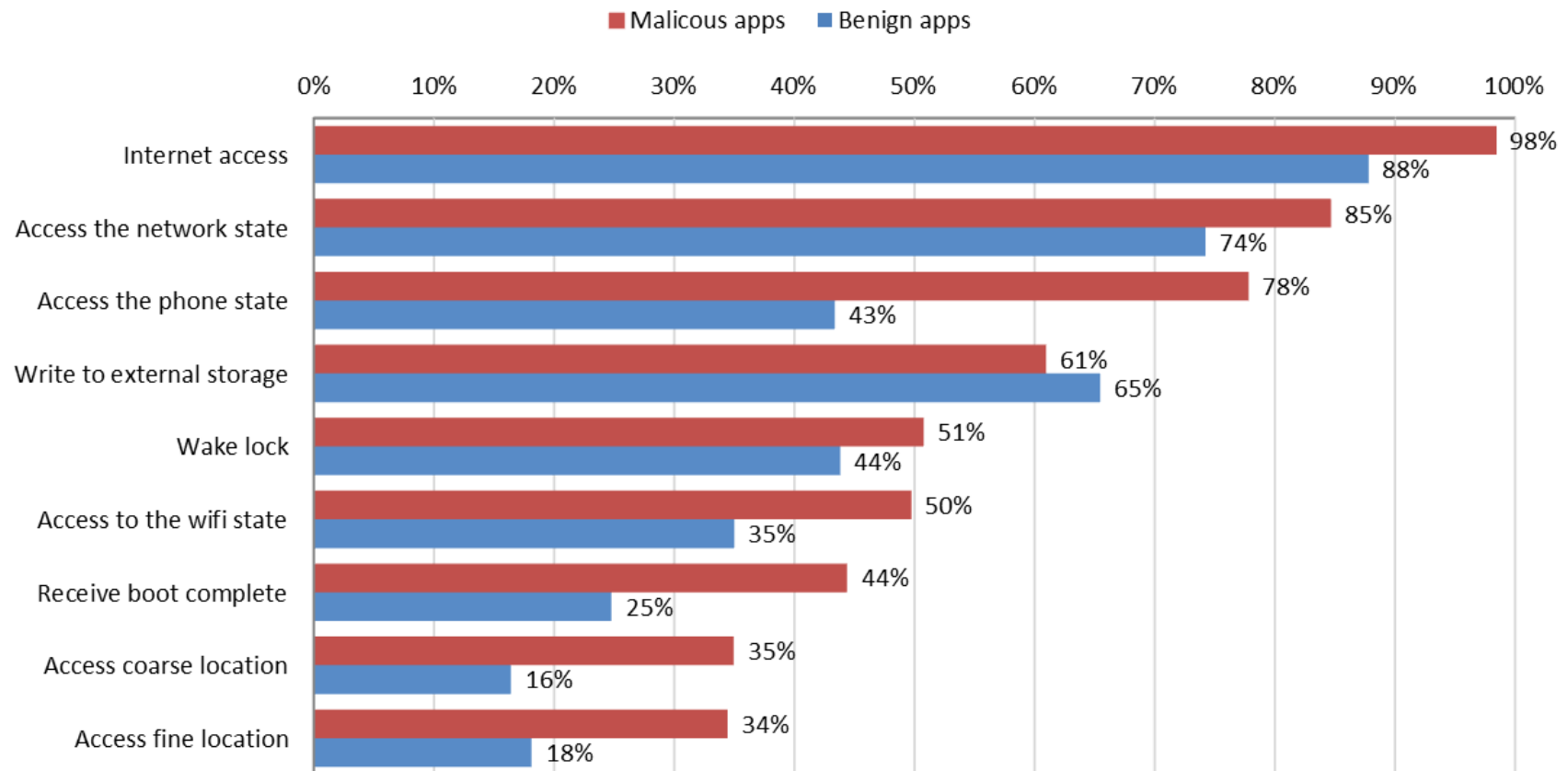
Detected types of Android malware



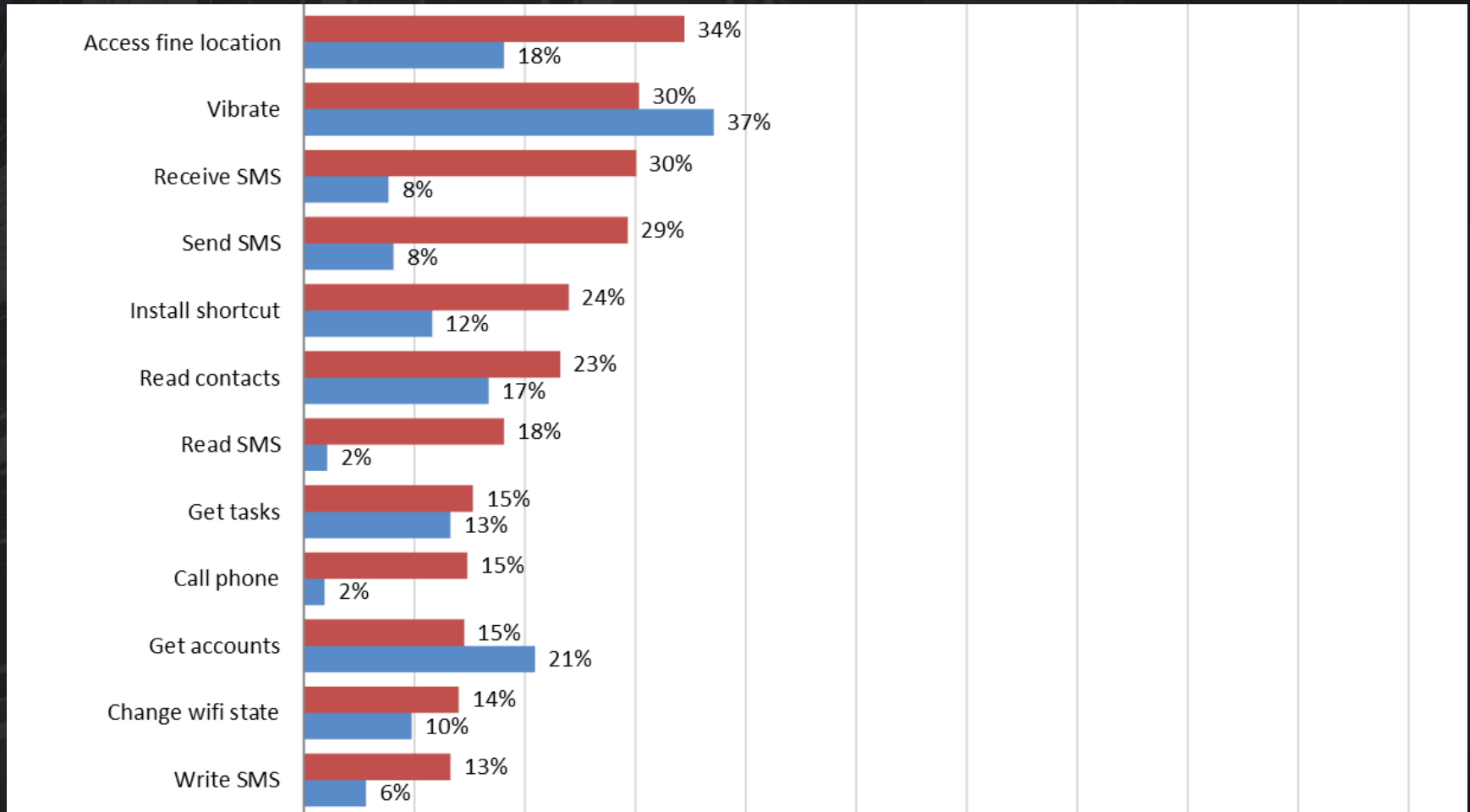
<https://apkscan.nviso.be/reports/NVISO%20ApkScan%20-%20Android%20malware%20report%20-%202013.pdf>

NVISO ApkScan Report 2013

Permissions most often requested by Android applications

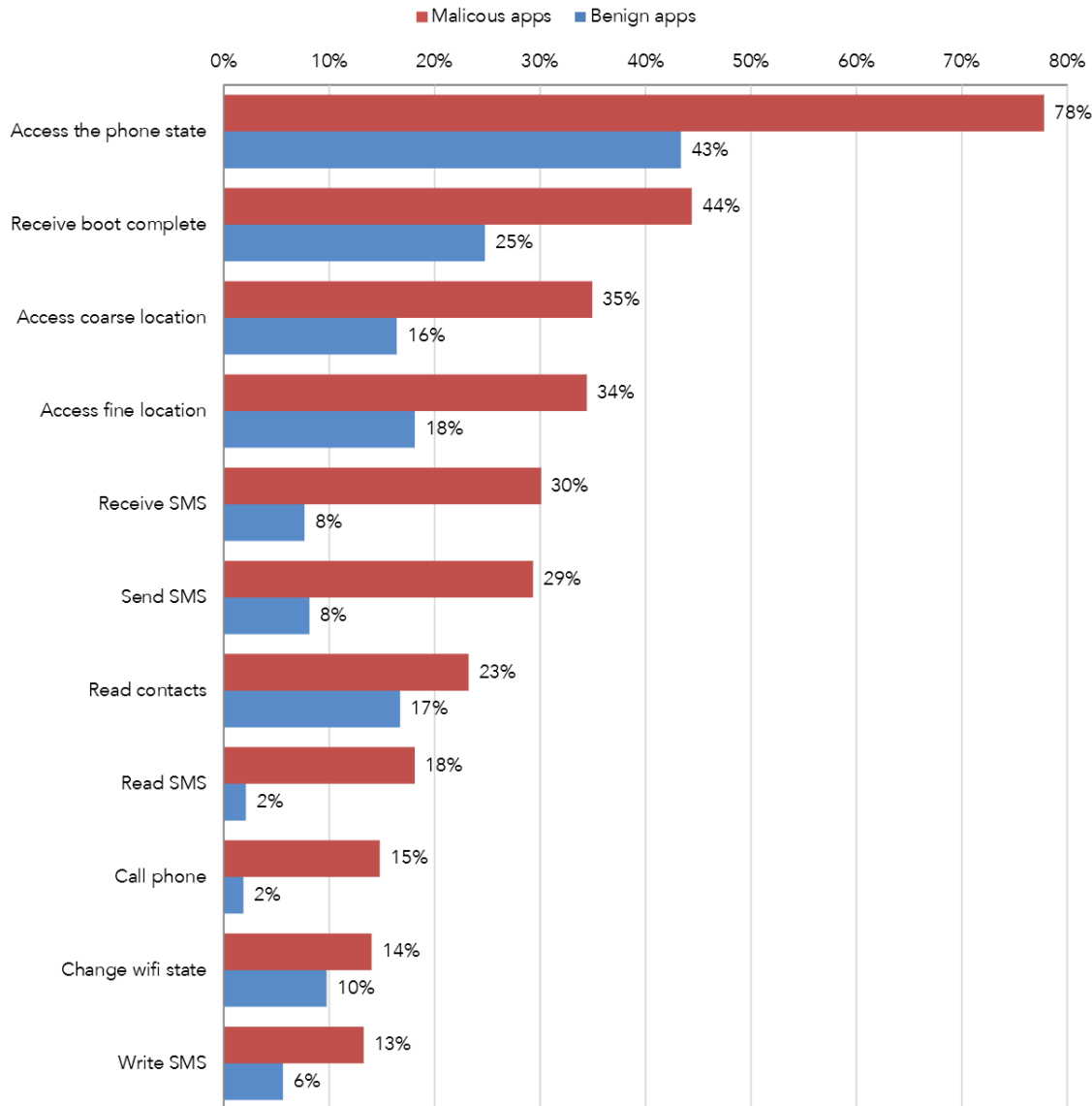


NVISO ApkScan Report 2013



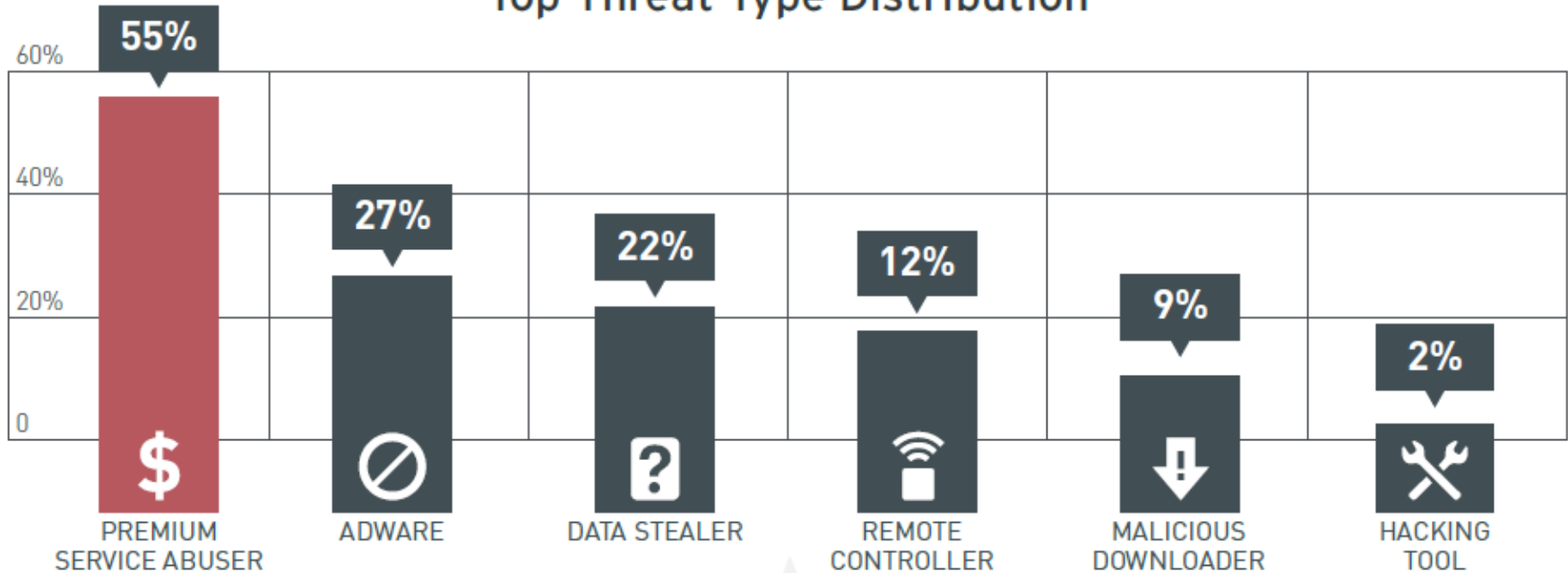
NVISO ApkScan Report 2013

Suspicious permissions requested by Android applications



TrendLabs 2Q 2013 Security Roundup

Top Threat Type Distribution



Like last quarter, premium service abusers comprised more than half of the mobile threats this quarter though the number of mobile adware also increased to regain the top 2 post.

<http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt-2q-2013-trendlabs-security-roundup.pdf>

TrendLabs 2Q 2013 Security Roundup

Countries with the Highest Malicious Android App Download Volumes

1	★ United Arab Emirates	13.79%
2	↓ Myanmar (Burma)	5.05%
3	★ Vietnam	4.94%
4	★ Mexico	4.23%
5	↓ Russia	4.17%
6	↓ India	3.74%
7	★ China	3.57%
8	★ Venezuela	3.11%
9	↓ Malaysia	2.97%
10	★ Singapore	2.84%



★ NEW ENTRY

↑ MOVED UP

↓ MOVED DOWN

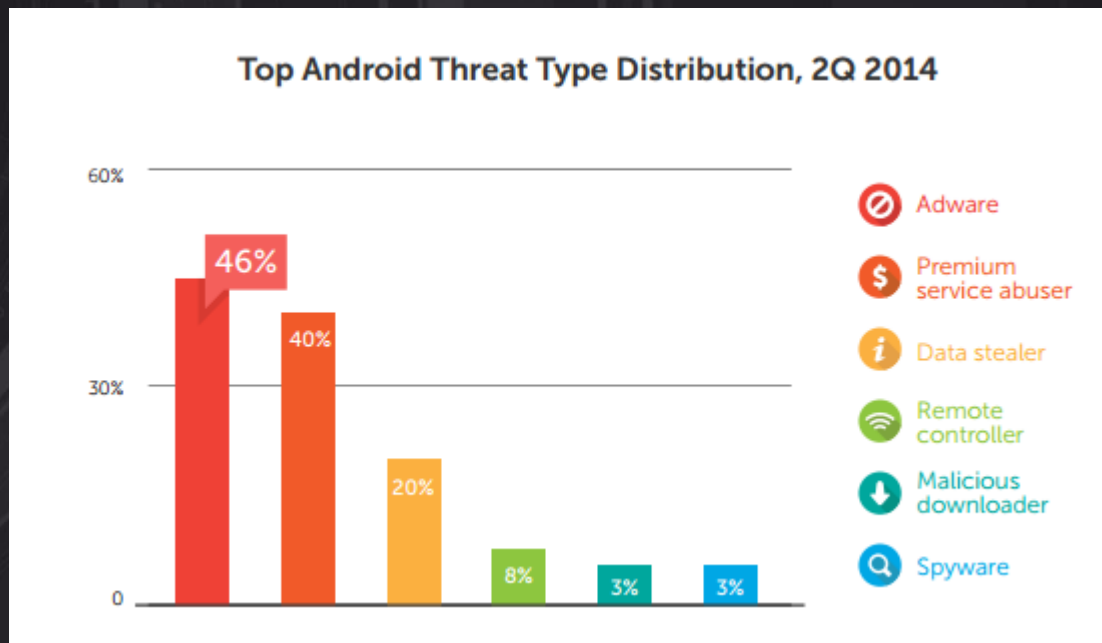
TrendLabs 2Q 2013 Security Roundup

Countries Most at Risk of Privacy Exposure Due to App Use

1	↑ Saudi Arabia	11.49%
2	★ Vietnam	8.87%
3	↑ Indonesia	8.82%
4	★ Brazil	7.98%
5	↓ India	7.87%
6	↑ Malaysia	7.57%
7	★ South Africa	6.52%
8	↓ Russia	5.64%
9	★ Algeria	5.55%
10	↑ Philippines	5.19%

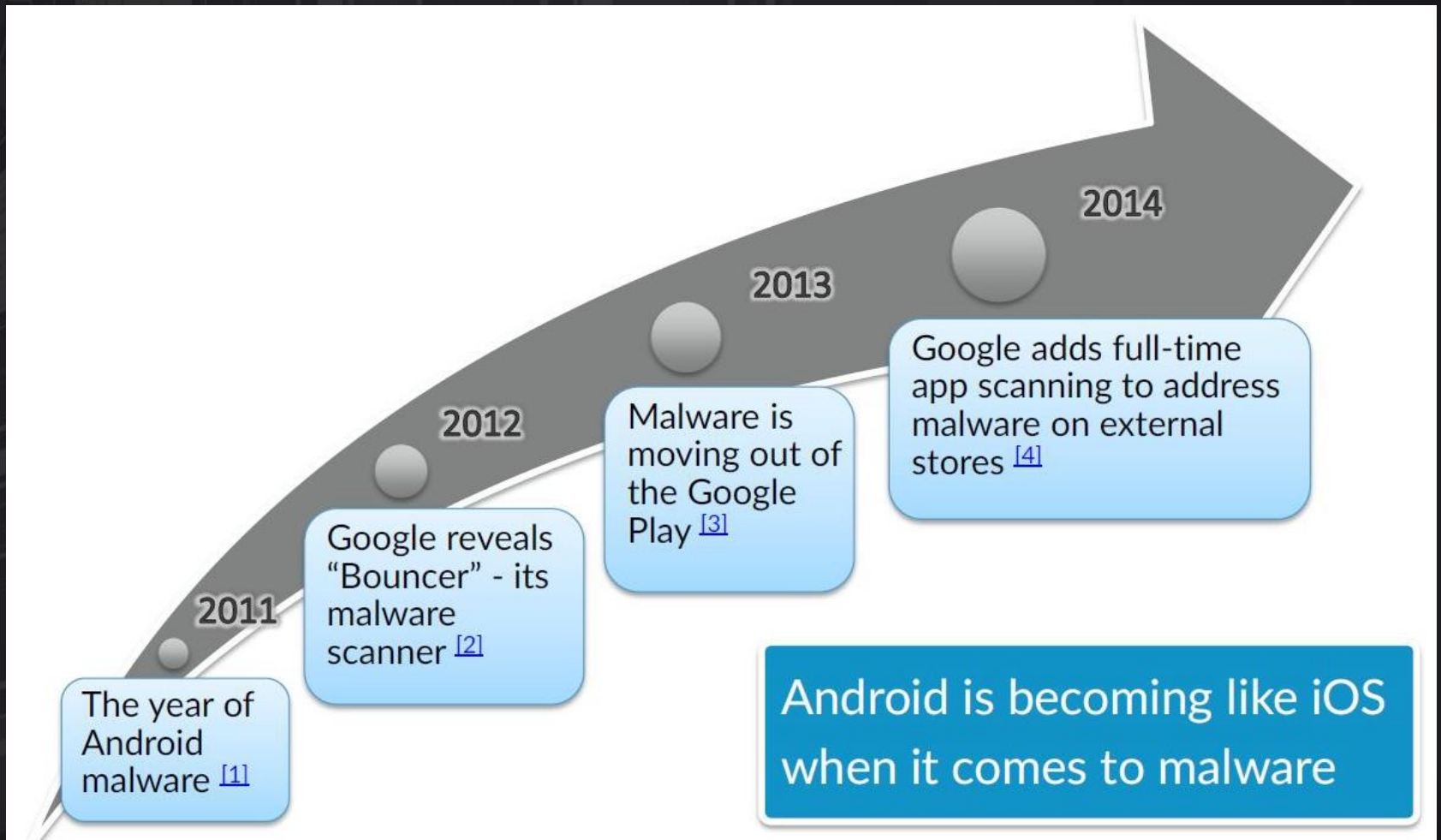


TrendLabs 2Q 2014 Security Roundup



<http://www.trendmicro.com.ru/media/report/turning-the-tables-on-cyber-attacks-2q-2014-report-en.pdf>

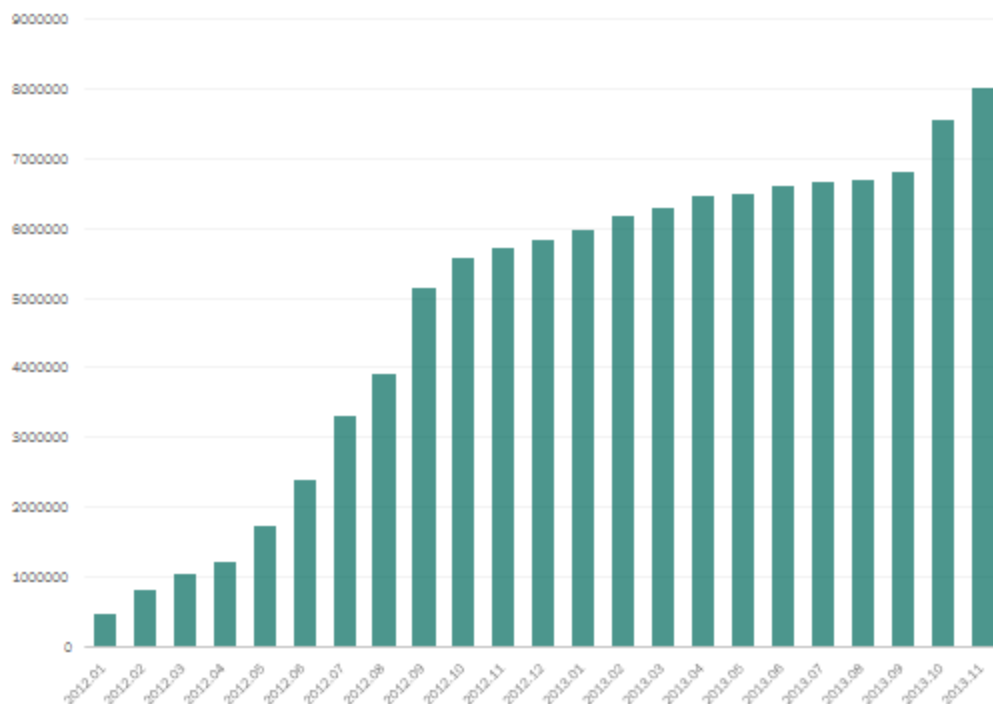
Google's Focus on Malware



Android Malware

ПОПУЛЯРНОСТЬ ЗЛОВРЕДОВ ДЛЯ ANDROID

Зловредные APK



10,000,000+

Число зловредных приложения
(APK в коллекции Лаборатории)

120,000

Среднее число новых образцов

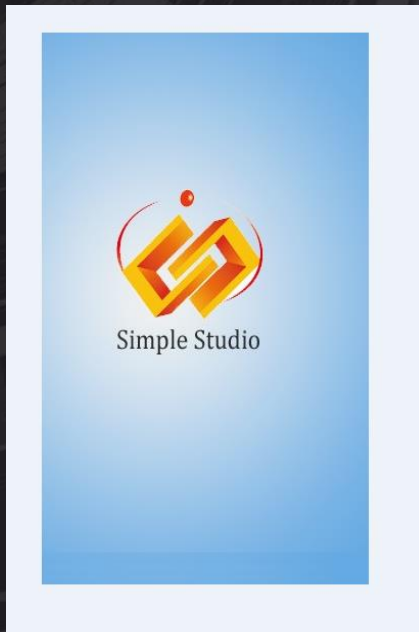
Source: Kaspersky Lab Security Network

KASPERSKY
lab

SPERASOFT

Check Images

- Hide executable code in pictures
- Hide entire APK in pictures



Malware

За просмотр
запрещенного(Педофилия,Зоофили
я и т.д.) порно ваш телефон
блокирован!



Все фото и видео материалы с вашей камеры
переданы на рассмотрение.
Для разблокировки вашего телефона и
удаление материалов
вам необходимо оплатить штраф 1000 руб. в
течении 24 часов
Для этого вам нужно пополнить Номер
+79147011354
В ближайшем терминале оплаты.
ВНИМАНИЕ: При попытке избежать штрафа
Все данные будут направлены в публичные
источники



FBI Criminal Investigation

#356440047053168

US

Prohibited content

This device is locked due to the
violation of the federal laws of the
United States of America:

- Article 161
 - Article 148
 - Article 215
 - Article 301
- of the Criminal Code of U.S.A.

Your device was used to visit
websites containing pornography.

Following violations were detected:



Amount of fine is \$200.



You can settle the fine with
MoneyPak express Packet
vouchers.

As soon as the money
arrives to the Treasure
account, your device will be
unblocked and all
information will be
decrypted in course of 24
hours.

We made a photo with your camera, it
will be added to the investigation.

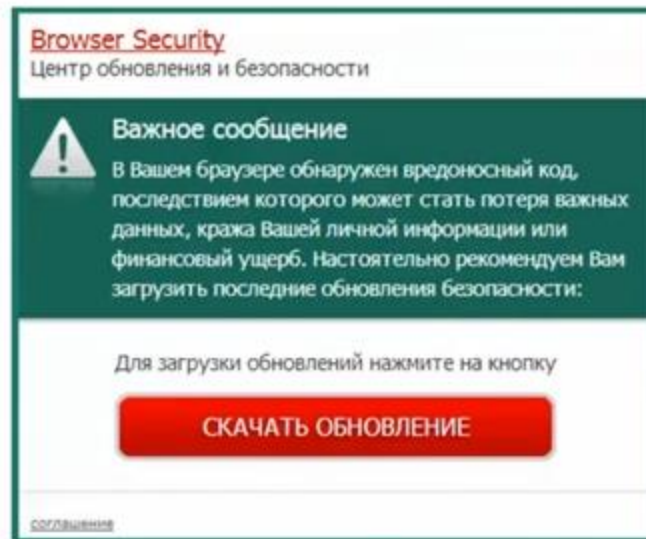
All your contacts are copied. If you do
not pay the fine, we will notify your
relatives and colleagues about the
investigation.

Malware

MALWARE

1. Browser update

Landing pages

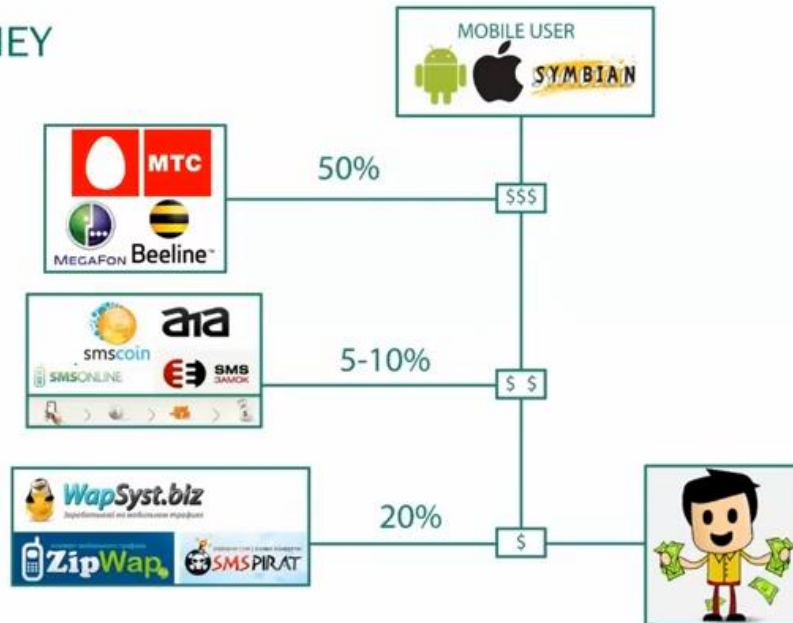


Malware

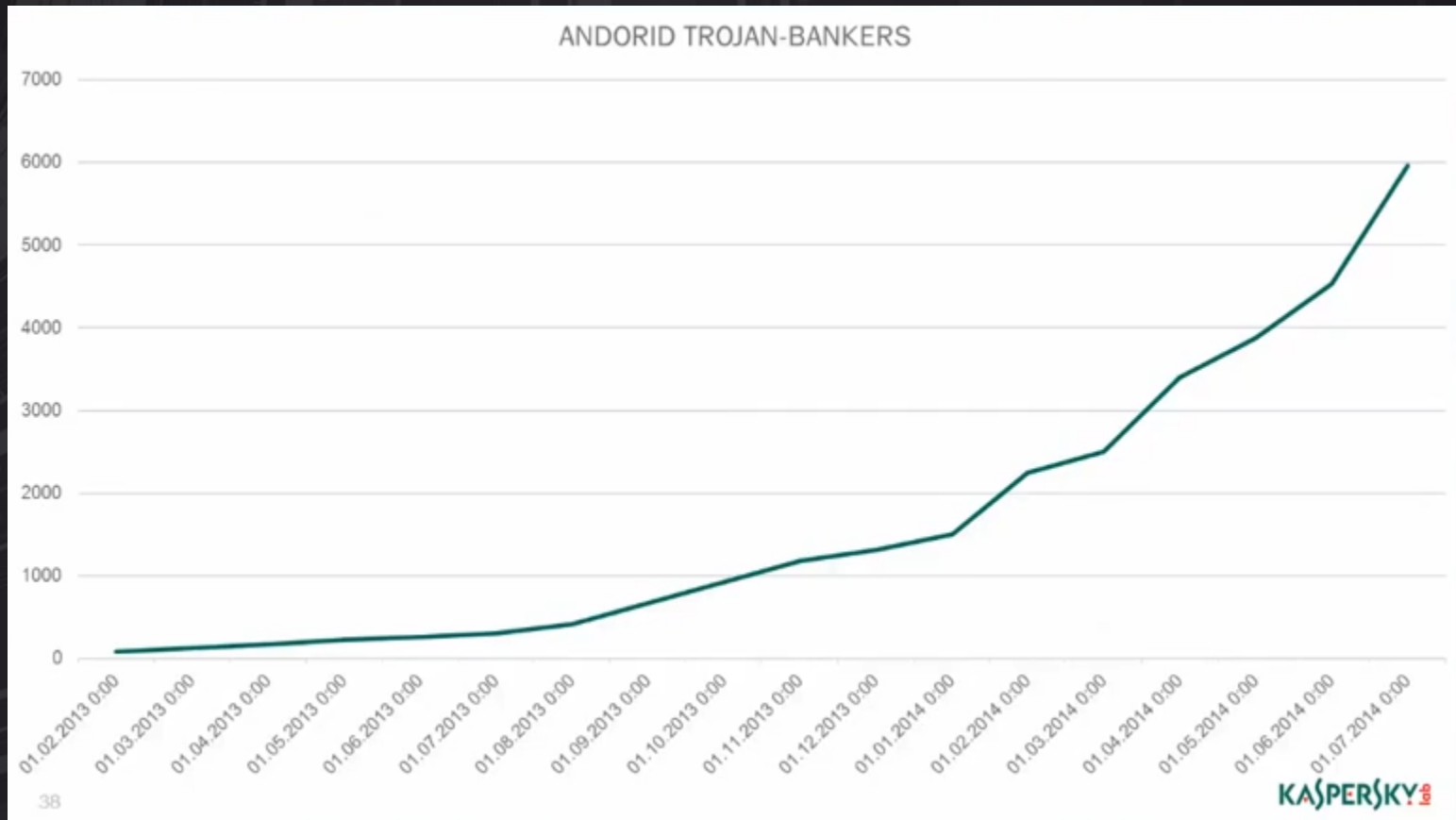
MONEY

- > \$54,000,000 украдено в прошлом году
- > \$27,000,000 мобильным операторам
- > \$2,700,000 смс билингам
- > \$4,800,000 партнеркам
- > \$19,000,000 партнерам

MONEY



Bankers



SVPENG

SVPENG FUNCTIONALITY

The image displays two side-by-side screenshots of the SVPENG mobile application interface, illustrating the login and registration process.

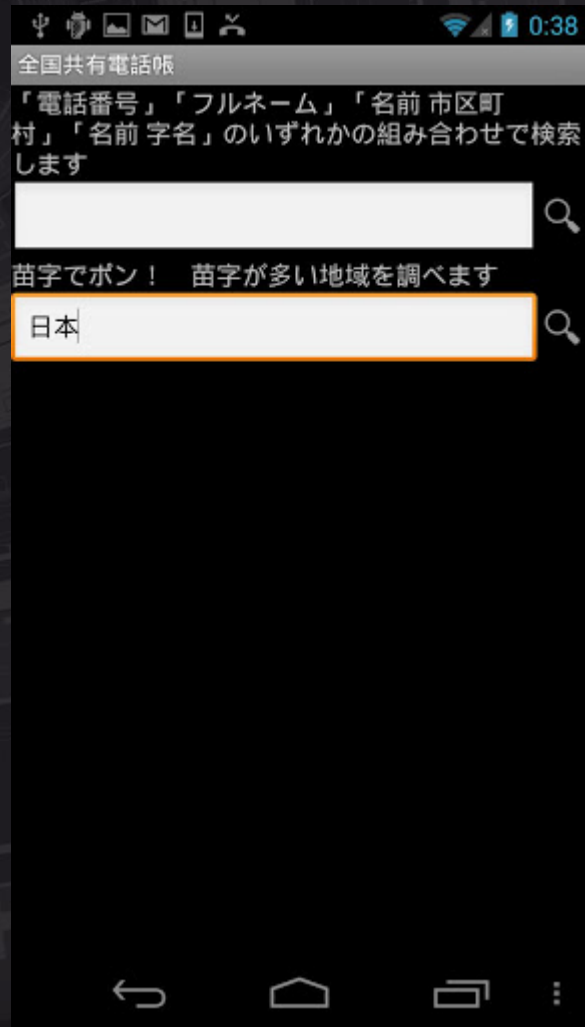
Left Screenshot (Login Phase):

- Top section: A dark green rectangular area.
- Input field: A light gray box with the placeholder text "Идентификатор (Имя пользователя)".
- Buttons: A green button labeled "Продолжить" (Continue) and a white button labeled "Continue".
- Progress indicator: A horizontal bar with three steps: 1. "Ввод Логина" (Enter Login), 2. "Ввод пароля" (Enter Password), and 3. "Авторизация" (Authorization).
- Bottom section: Three icons representing different login methods: "Идентификатор" (Identifier), "Социальные сети" (Social networks), and "Демо-режим" (Demo mode).

Right Screenshot (Password Phase):

- Top section: A dark green rectangular area.
- Input field: A light gray box with the placeholder text "Идентификатор (Имя пользователя)".
- Buttons: A green button labeled "Продолжить" (Continue) and a white button labeled "Continue".
- Progress indicator: A horizontal bar with three steps: 1. "Ввод Логина" (Enter Login), 2. "Ввод пароля" (Enter Password), and 3. "Авторизация" (Authorization).
- Bottom section: A green button labeled "Отменить регистрацию" (Cancel registration).

ANDROIDOS_JIGENSHA.A



Japan porn viewer which sends ad email to all user contacts if user will not pay a monthly payment

- Impact Scope:
760,000 users' data leaked online in Japan
- Malicious Behavior:
The malware collect User's contact list includes phone number and names, then sends them to a remote server.

Antivirus Software for Android

	Avast Mobile Security 4.0				
	AVG AntiVirus Free 4.4				
	Avira Free Android Security 4.1				
	Baidu Mobile Security 5.7				
	Bitdefender Mobile Security 2.40				
	BullGuard Mobile Security 14.0				
	Cheetah Mobile Clean Master 5.10				
	Cheetah Mobile CM Security 2.6				
	ESET Mobile Security & Antivirus 3.0				
	G Data Internet Security 25.8				
	Ikarus mobile.security 1.7				
	Kaspersky Lab Internet Security 11.8				
	McAfee Mobile Security 4.4				
	Norton Norton Mobile Security 3.11				

<https://www.av-test.org/en/antivirus/mobile-devices/android/>

<http://www.av-comparatives.org/>

Kaspersky Internet Security



Free Trial Renew Upgrade

★★★★★ 4.7

Read reviews (232)

Kaspersky Internet Security for Android

Kaspersky Internet Security for Android delivers advanced anti-theft protection for smartphones and tablets, and keeps you completely protected from Internet threats when you're online shopping, banking and social networking.

- Anti-malware Protection – including Kaspersky's latest Android antivirus technologies
- Web Protection – against Internet-based attacks and phishing websites
- Anti-theft Protection – with remote access to special security features on your missing device
- Privacy Protection – to control what others can see or access when they pick up your smartphone
- Call & Text Filter – so your smartphone receives only those calls and texts you want to receive



[Help Me Choose Tool](#)
Easy as 1. 2. 3.

**Kaspersky keeps
you safe online.**

- ☒ 1 Device - 1 Year Protection
- ☐ 1 Device - 2 Years Protection
- ☐ 2 Devices - 1 Year Protection
- ☐ 2 Devices - 2 Years Protection

\$14.99

Buy Now

<http://usa.kaspersky.com/products-services/home-computer-security/security-for-android/>

S P E R  S O F T

Books

- **Android Hacker's Handbook**, Joshua J. Drake, Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A. Ridley, Georg Wicherski, 2014, John Wiley & Sons, Inc.
- **Android Security Internals: An In-Depth Guide to Android's Security Architecture**, Nikolay Elenkov, 2014, William Pollock
- **Application Security for the Android Platform: Processes, Permissions, and Other Safeguards**, Jeff Six, 2012, O'Reilly Media
- **Android Application Security Essentials**, Pragati Ogal Rai, 2013, Packt Publishing
- **Embedded Android: Porting, Extending, and Customizing**, Karim Yaghmour, 2013, O'Reilly Media

Thanks

Visit Sperasoft online:

www.sperasoft.com

www.facebook.com/Sperasoft

www.twitter.com/sperasoft