Android Developer Security Checklist
Avoiding OWASP Mobile Top10 Risks

Eby John, Yucheng Liu
{ebyjohn, yliu_leon}@gatech.edu

Analyze Tools
- AndroGuard
- SandDroid
- TaintDroid
- dex2jar
- jd-gui
- apktool

	AliExpress	DHGate Mobile	Facebook Messenger	Gag Viewer	HD Call ID	textPlus	TweetCaster	Biz Barcode	Outlook.com	FourGoats
M1	✓	✓	✓	✓		✓	✓		✓	✓
M2										
M3	✓	✓							✓	✓
M4		✓								✓
M5	✓									
M6										
M7		✓		✓	✓		✓		✓	
M8					✓					✓
M9		✓				✓				
M10		✓		✓	✓	✓	✓			✓

OWASP Mobile Top 10 Risks

M1 – Insecure Data Storage	M2 – Weak Server Side Controls	M3 - Insufficient Transport Layer Protection	M4 - Client Side Injection
M5 - Poor Authorization and Authentication	M6 - Improper Session Handling	M7 - Security Decisions Via Untrusted Inputs	M8 - Side Channel Data Leakage
	M9 - Broken Cryptography	M10 - Sensitive Information Disclosure	

Detail Analysis Results
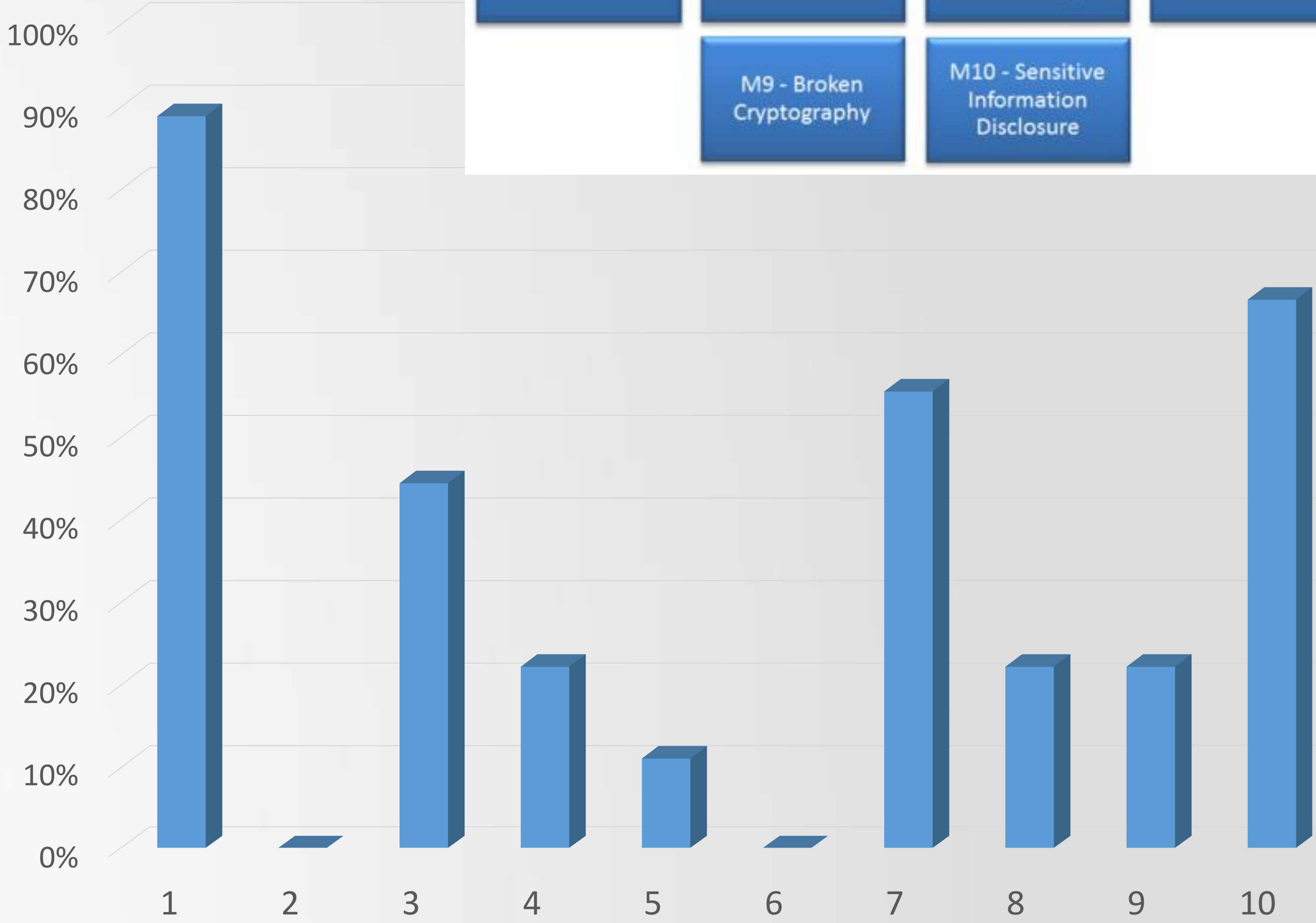
OWASP Mobile Top 10 Risks

Methodology

OWASP Mobile Top 10 Risks

M1
Is sensitive information like Usernames, Passwords, Location data, SSN, Address etc stored insecurely in files on the mobile?

Is sensitive information stored on SDCard encryted?

Is sensitive information being logged?

Is any file marked MODE_WORLD_WRITEABLE or MODE_WORLD_READABLE unnecessarily ?

M2
Is a device dependant variable being used as an authentication token?

Is data properly validated before authentication proceeds?

Does the password recovery feature allow account hijacking?

Are passwords being hashed and salted before being stored on the server?

M3
Is the SSL Certificate properly validated before establishing connection or are all certificates accepted?

Is a MiTM attack possible.

Is the Cipher used of strength sufficient? [Also M9]

Do not send sensitive data over alternate channels, such as SMS, MMS, or notifications.[Also M10]

M4
Is Input data properly validated?

Is there any Local file inclusion?

Use parameterized queries.

Intent Injection/Fuzzing: Verify actions and data are validated via an Intent Filter for all Activities.

M5
Does the application rely solely on immutable, potentially compromised values (IMEI, IMSI, UUID)?

Are there Hardcoded/default accounts in the application?

Test if direct access to backend resources is possible.

M6
Is sensitive information utilized within the application flushed from memory upon session expiration?

Ensure that sessions time out locally as well as on the server side.

When HttpClient instance is no longer needed, shut down the connection manager to ensure immediate deallocation of all system resources

M7
An intent injection attack is possible if the intent address is derived from untrusted input. e.g. URLConnection.getInputStream()

Buffer Overflows: Review all code that accepts input from users via the HTTP request and ensure that it provides appropriate size checking on all such inputs on the server-side

The best way to protect a web application from XSS attacks is ensure that your application performs validation of all headers, cookies, query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification of what should be allowed.

M8
Is sensitive information being logged?

Ensure 3rd party libraries are not leaking sensitive information.

Store sensitive information in char array instead of Java strings.

Use a key generation algorithm like PKCS#5 when using password based authentication.

Use unpredictable initialization vectors.

Ensure the key size is sufficiently large.

M9
Use standard security algorithms and their standard implementations. Avoid custom algorithms and protocols.

M10
Does the app expose sensitive information like IMEI or other uniquely identifiable details.

Check the Shared Preference xml file for sensitive information.

Does the application send sensitive information in the URL parameters without using SSL.

Are there hardcoded passwords or API keys.