Defend. Detect. Deter.

ARXAN AT SMAMA HANDOUT VERSION

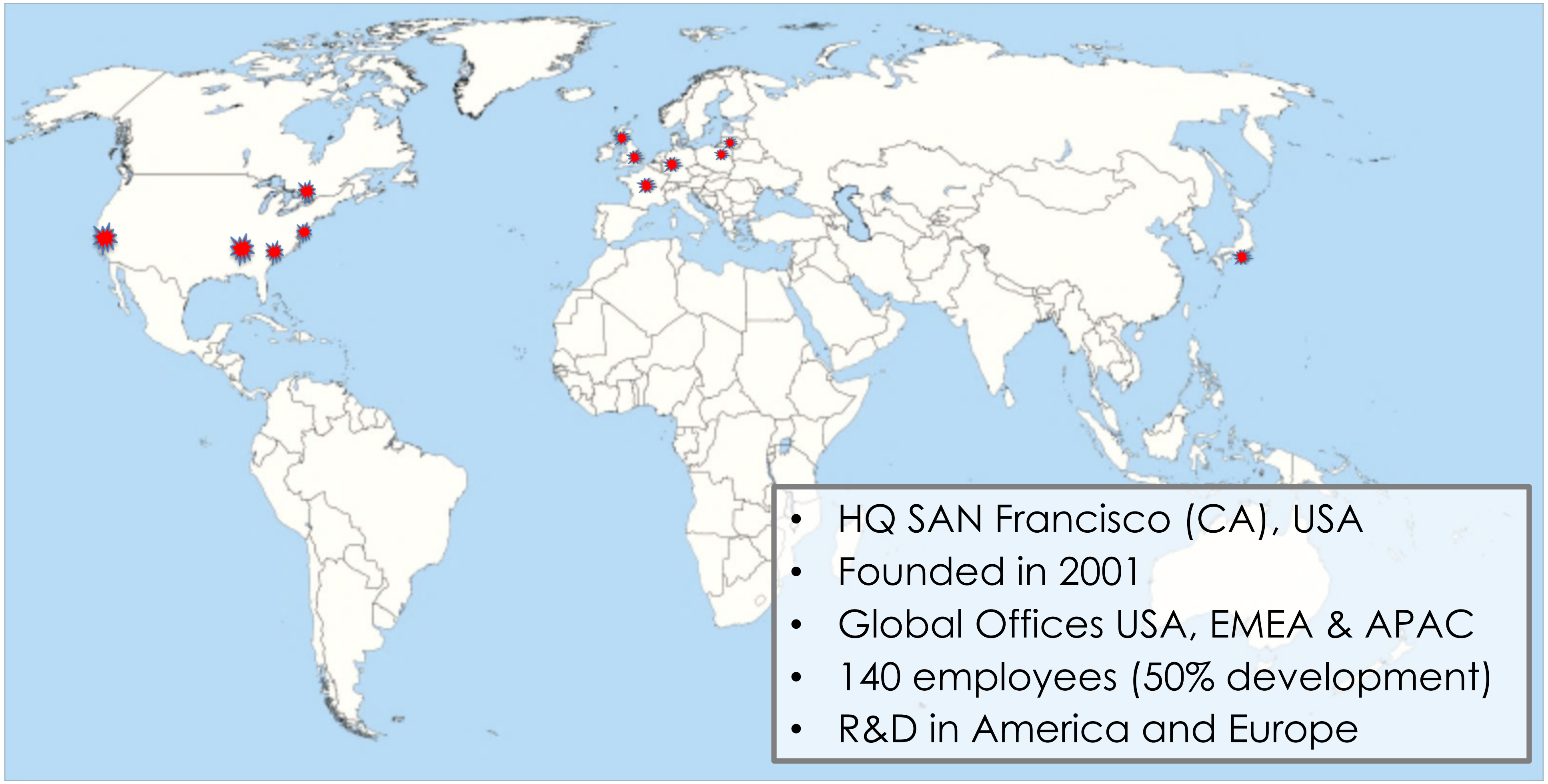
APPLICATION SECURITY - BUILD IT SECURE, KEEP IT SECURE

MIRKO BRANDNER

Agenda for 30 mins

- Intention
  - Very, very short company overview
  - Introduce all the buzzwords
  - Becoming more technical as the presentation continues
    - From Techies <u>for Techies</u>
    - As as short as possible but always enough to understand
- Solutions
  - Vulnerabilities, current Platforms & Tools
  - Focus Application Hardening Apps for Mobile Devices (iOS, Android)
    - Native and JavaScript
  - Hardening using Guards and Guard Networks

About ARXAN



- HQ SAN Francisco (CA), USA
- Founded in 2001
- Global Offices USA, EMEA & APAC
- 140 employees (50% development)
- R&D in America and Europe

Application Security

Application Hardening, WhiteBox Cryptography, Mobile Application Management

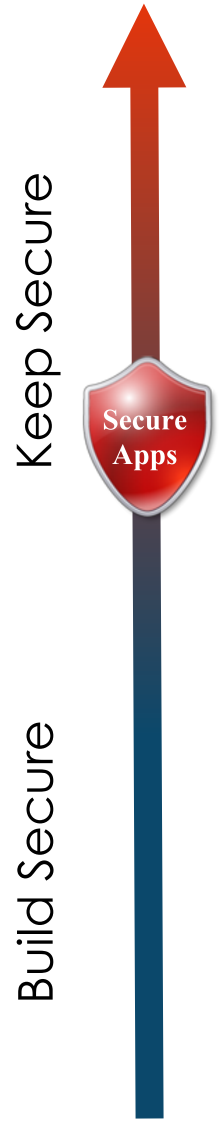
Arxan Products, Benefits and Threats

Objective	Binary Protection & JS code	Key Protection	Mobile App Management
Threats	Tampering, Reversing-Engineering, Unauthorized Access	Key Discovery	Tampering, Unauthorized Access, public access to store
Products	Binary Code Integrity: - GuardIT® and EnsureIT™ (Mobile) Source Code Integrity: - SecureJS (JavaScript)	Key Protection: TransformIT® Whitebox Cryptography	Apperian EASE
Broad Coverage	Desktop, Server Embedded & Mobile Apps Multi-Platform Coverage	Support for Major Cryptographic Algorithms: RSA, AES, ECC,DESS Multi-Platform Coverage	iOS and Android Diverse policies
Benefits	• IP Protection, Piracy –Prevention • Preserve Integrity of Code and Business Models	• Secure Premium Content or Data • Preserve Integrity of Intellectual Property and Business Models	• Onboarding, inspect, protect, sign, deploy, measure • Your independent own store • Applies parallel to MDM

Customers and Verticals

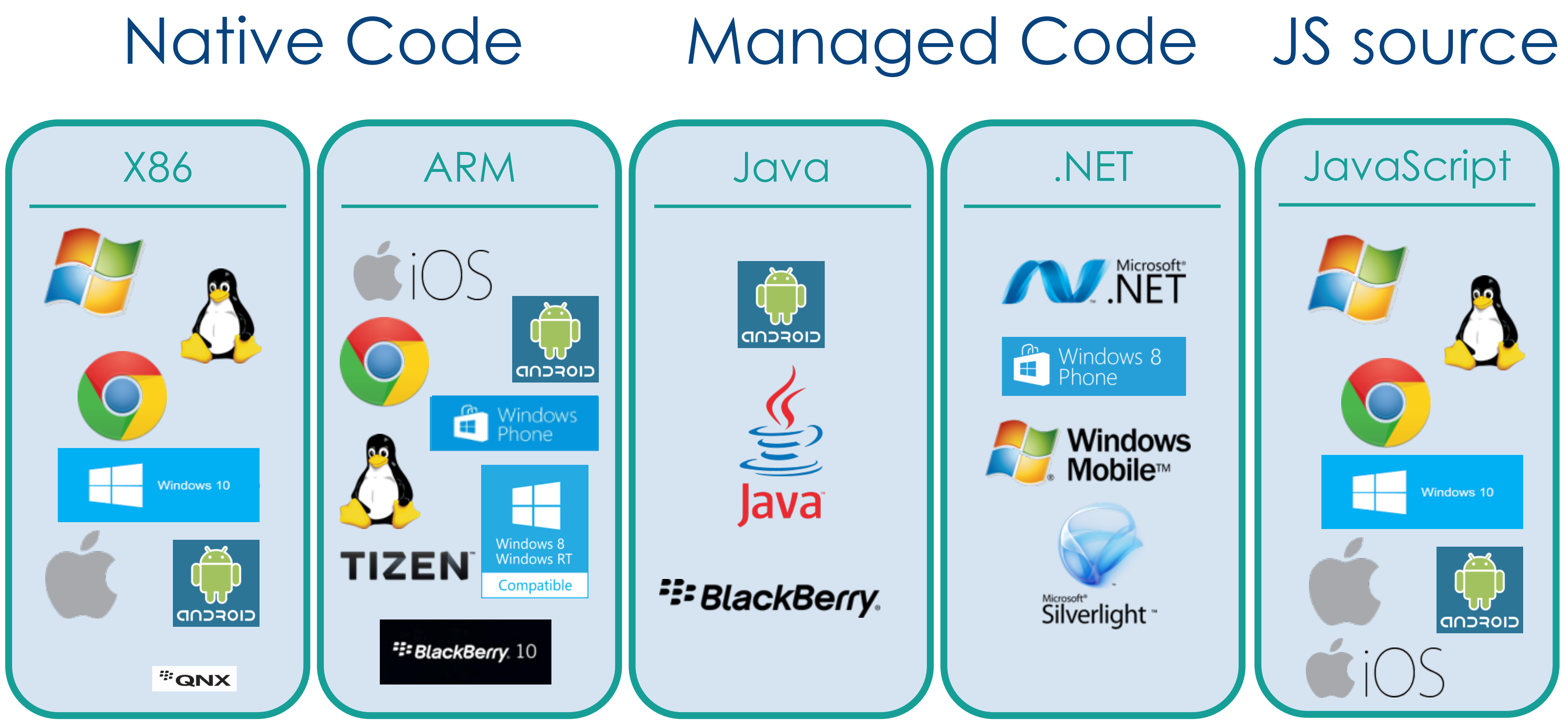
Sorry, we are not allowed to present our customers in written form to the public or provide Gartner content about Application Shielding in these follow-up slides.

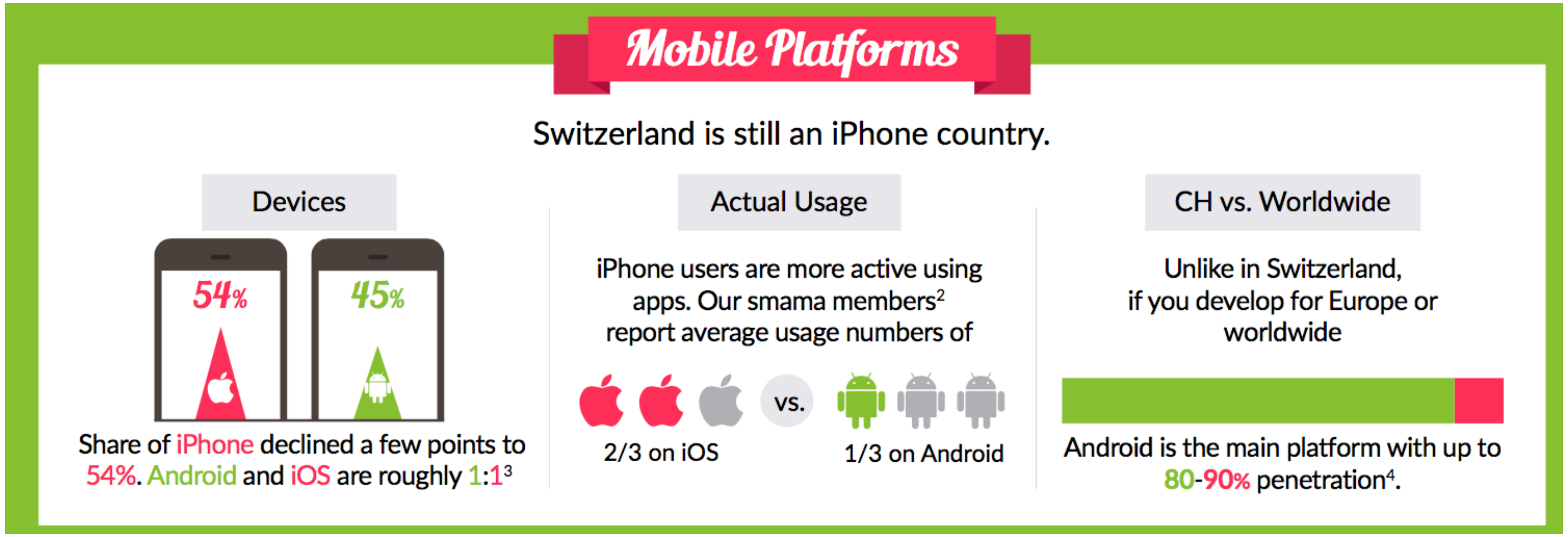
Binary Protection – no source code touched!?



Code Signing	Validates origin of code
Pen Testing	**Exposes** vulnerabilities
App Protection/App Hardening	Provide self-defense and tamper-proofing; prevents tampering, unauthorized access/analysis, code insertion
Static/Dynamic Scanning	**Identifies** vulnerabilities
Mobile Device Management/Mobile Application Management	Remote debugging and tracking for BYOD Initiatives Sets authentication policies, encrypts files saved by app
Authentication	PWs, biometrics, root detection – authenticates **user and/or environment**
Secure Development	Best practices for coding

Multi-Platform Support for Binary Protection & JS





Mobile Platforms

Switzerland is still an iPhone country.

Devices

54% 45%

Share of iPhone declined a few points to 54%. Android and iOS are roughly 1:1[3]

Actual Usage

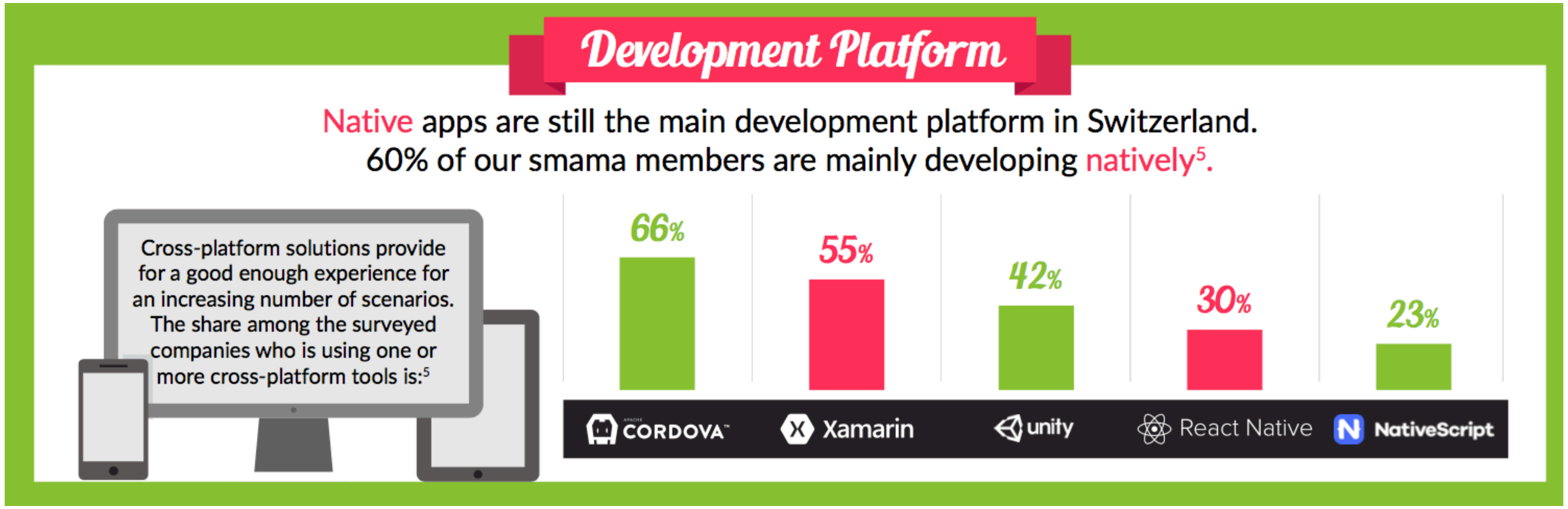
iPhone users are more active using apps. Our smama members[2] report average usage numbers of

2/3 on iOS vs. 1/3 on Android

CH vs. Worldwide

Unlike in Switzerland, if you develop for Europe or worldwide

Android is the main platform with up to 80-90% penetration[4].



Development Platform

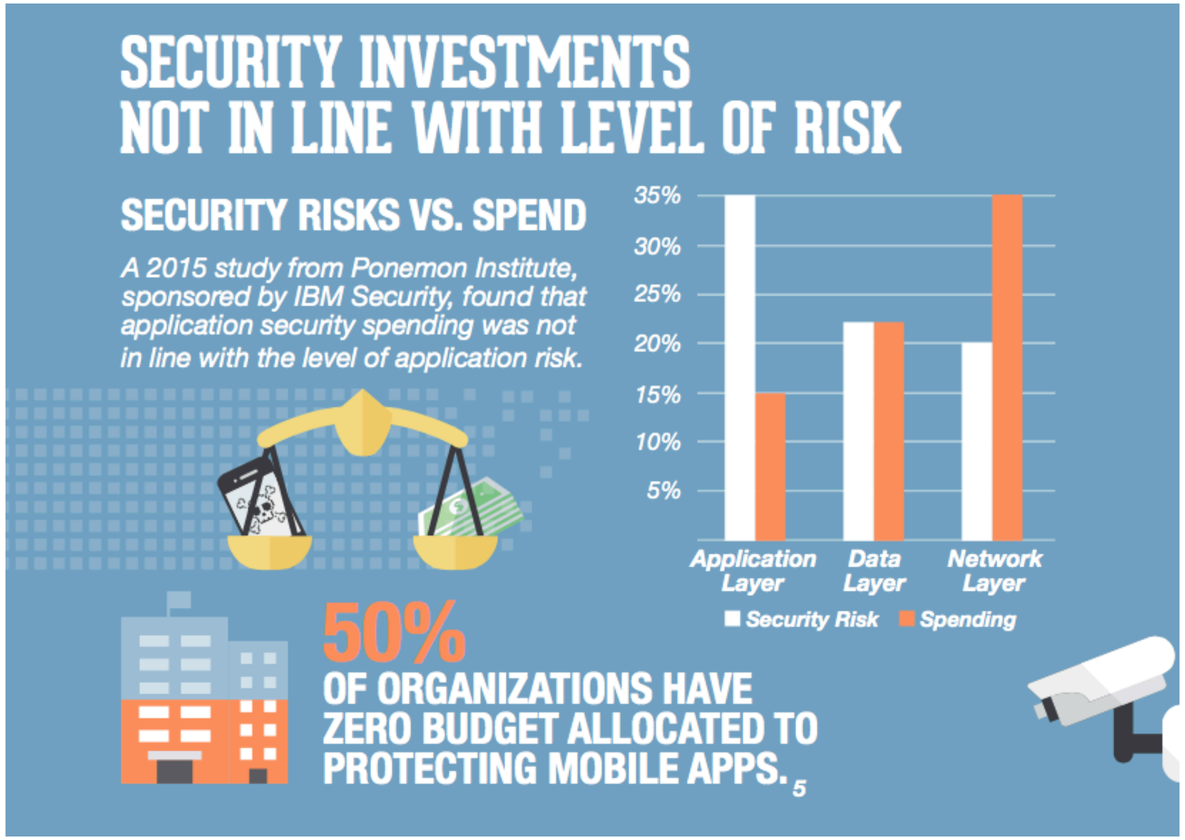
Native apps are still the main development platform in Switzerland. 60% of our smama members are mainly developing natively[5].

Cross-platform solutions provide for a good enough experience for an increasing number of scenarios. The share among the surveyed companies who is using one or more cross-platform tools is:[5]

Cordova	Xamarin	unity	React Native	NativeScript
66%	55%	42%	30%	23%

Why Application Security and Arxan at all?

Infos from IBM / Ponemon Institute

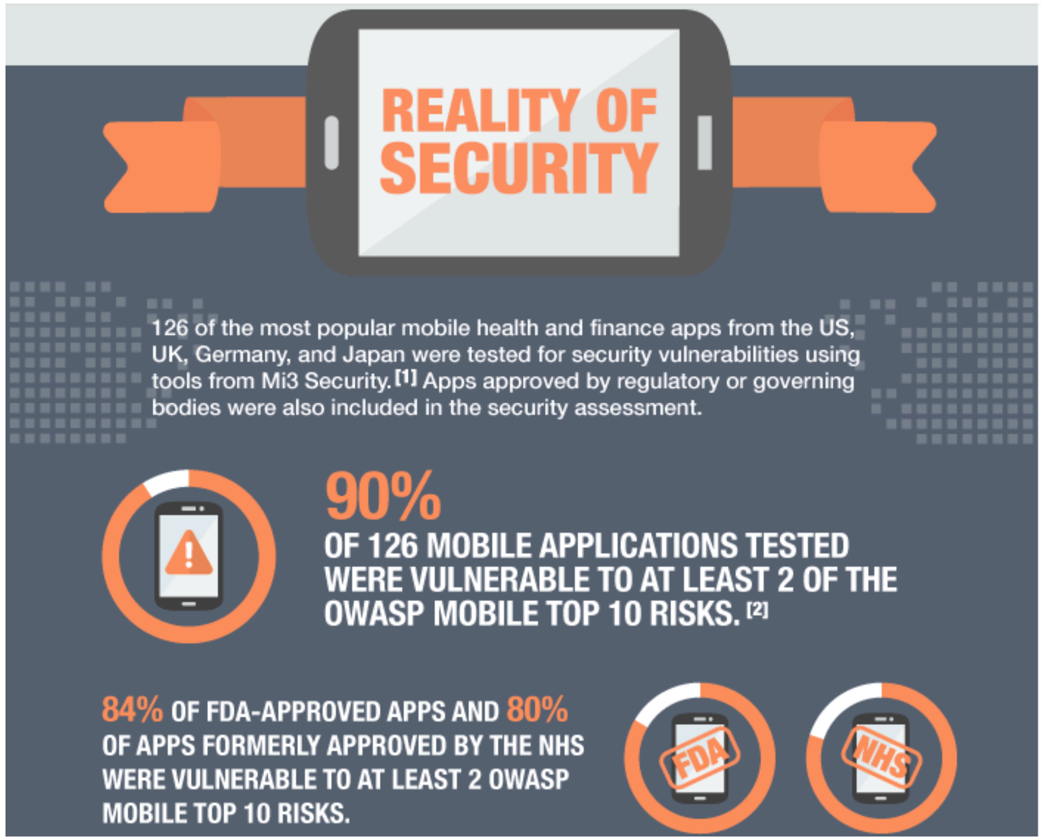


SECURITY INVESTMENTS NOT IN LINE WITH LEVEL OF RISK

SECURITY RISKS VS. SPEND

A 2015 study from Ponemon Institute, sponsored by IBM Security, found that application security spending was not in line with the level of application risk.

50% OF ORGANIZATIONS HAVE ZERO BUDGET ALLOCATED TO PROTECTING MOBILE APPS.[5]



REALITY OF SECURITY

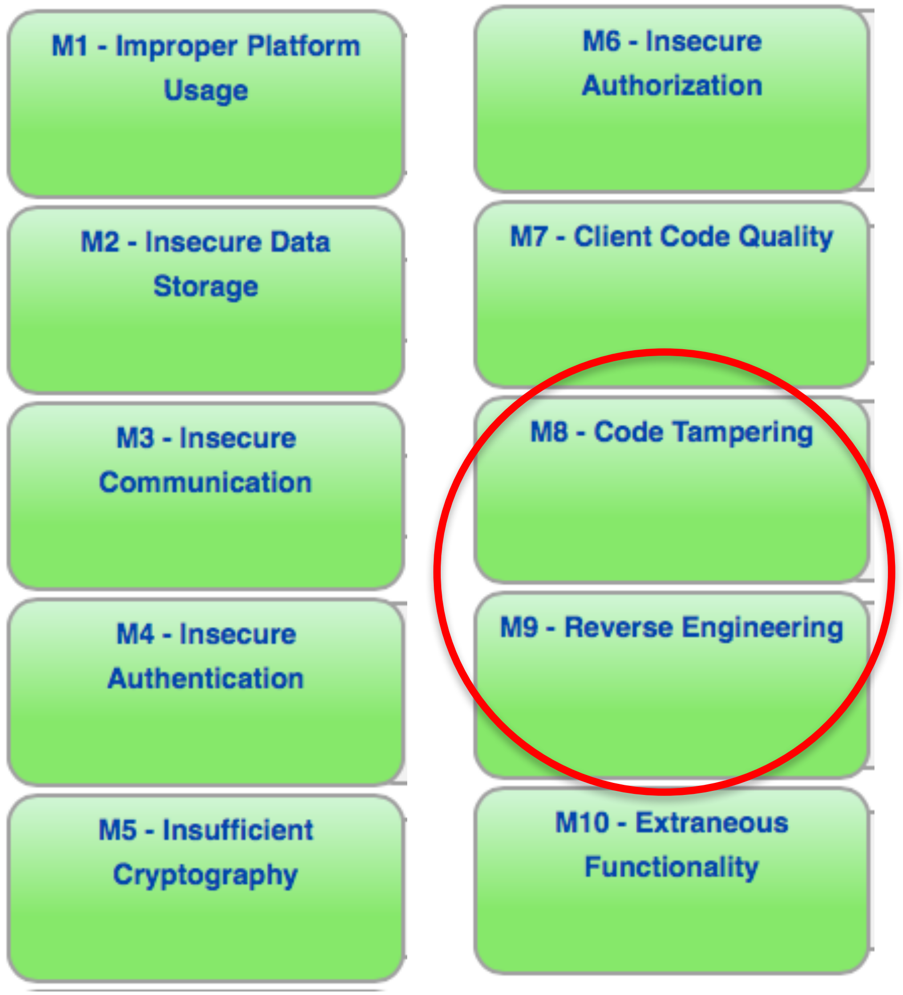
126 of the most popular mobile health and finance apps from the US, UK, Germany, and Japan were tested for security vulnerabilities using tools from Mi3 Security.[1] Apps approved by regulatory or governing bodies were also included in the security assessment.

90% OF 126 MOBILE APPLICATIONS TESTED WERE VULNERABLE TO AT LEAST 2 OF THE OWASP MOBILE TOP 10 RISKS.[2]

84% OF FDA-APPROVED APPS AND 80% OF APPS FORMERLY APPROVED BY THE NHS WERE VULNERABLE TO AT LEAST 2 OWASP MOBILE TOP 10 RISKS.

Sicher entwickeln / Sicher bleiben

https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10



M1 - Improper Platform Usage

M2 - Insecure Data Storage

M3 - Insecure Communication

M4 - Insecure Authentication

M5 - Insufficient Cryptography

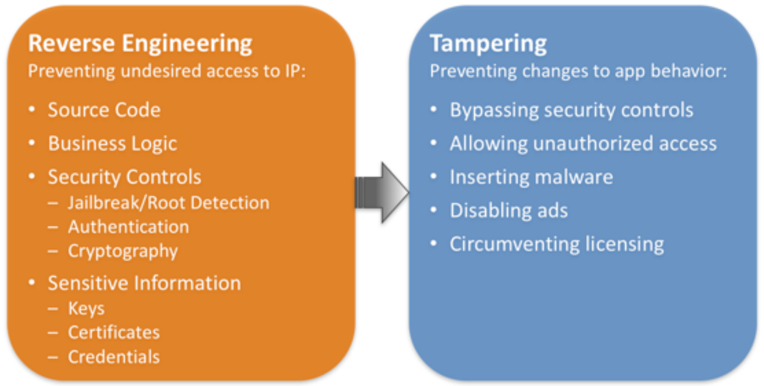
M6 - Insecure Authorization

M7 - Client Code Quality

M8 - Code Tampering

M9 - Reverse Engineering

M10 - Extraneous Functionality

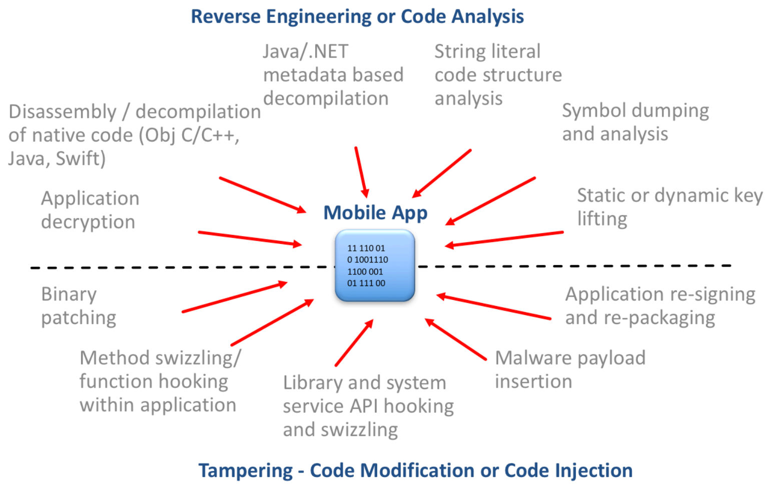


Reverse Engineering
Preventing undesired access to IP:

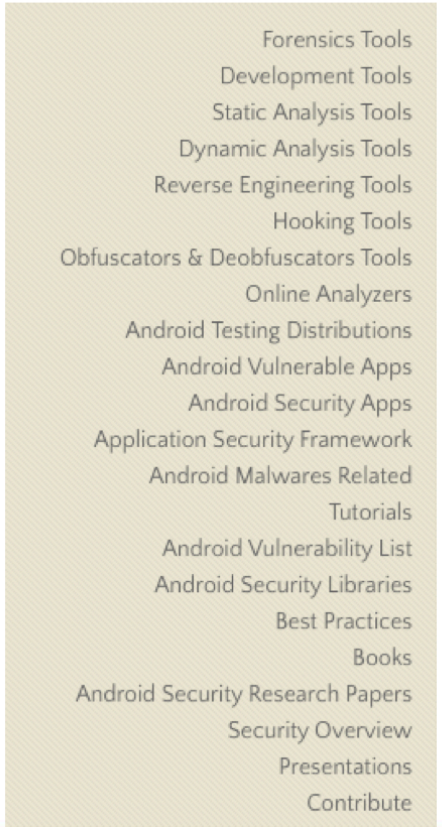
- Source Code
- Business Logic
- Security Controls
  - Jailbreak/Root Detection
  - Authentication
  - Cryptography
- Sensitive Information
  - Keys
  - Certificates
  - Credentials

Tampering
Preventing changes to app behavior:

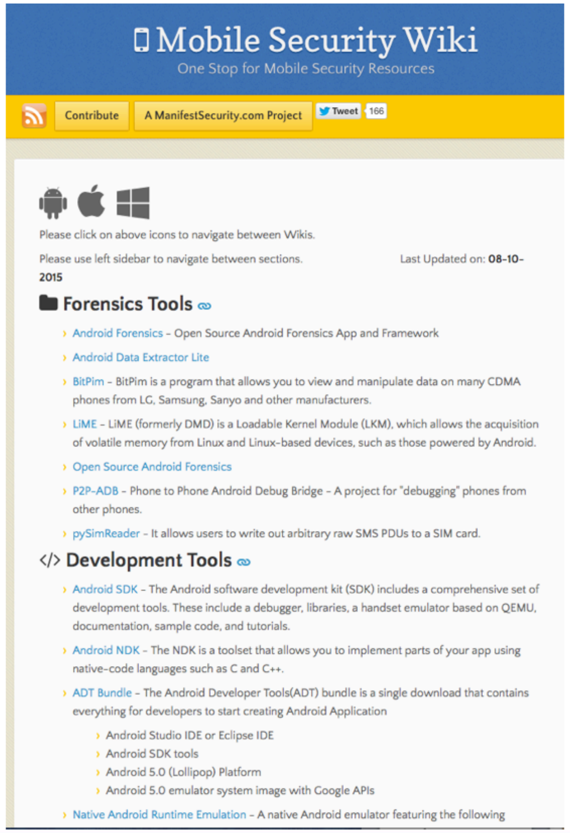
- Bypassing security controls
- Allowing unauthorized access
- Inserting malware
- Disabling ads
- Circumventing licensing


Reverse Engineering or Code Analysis
Java/.NET metadata based decompilation
String literal code structure analysis
Disassembly / decompilation of native code (Obj C/C++, Java, Swift)
Symbol dumping and analysis
Application decryption
Mobile App
Static or dynamic key lifting
11 110 01
0 1001110
1100 001
01 111 00
Binary patching
Application re-signing and re-packaging
Method swizzling/ function hooking within application
Library and system service API hooking and swizzling
Malware payload insertion
Tampering - Code Modification or Code Injection

Some Public Tools



Forensics Tools
Development Tools
Static Analysis Tools
Dynamic Analysis Tools
Reverse Engineering Tools
Hooking Tools
Obfuscators & Deobfuscators Tools
Online Analyzers
Android Testing Distributions
Android Vulnerable Apps
Android Security Apps
Application Security Framework
Android Malwares Related
Tutorials
Android Vulnerability List
Android Security Libraries
Best Practices
Books
Android Security Research Papers
Security Overview
Presentations
Contribute



Mobile Security Wiki
One Stop for Mobile Security Resources

Please click on above icons to navigate between Wikis.

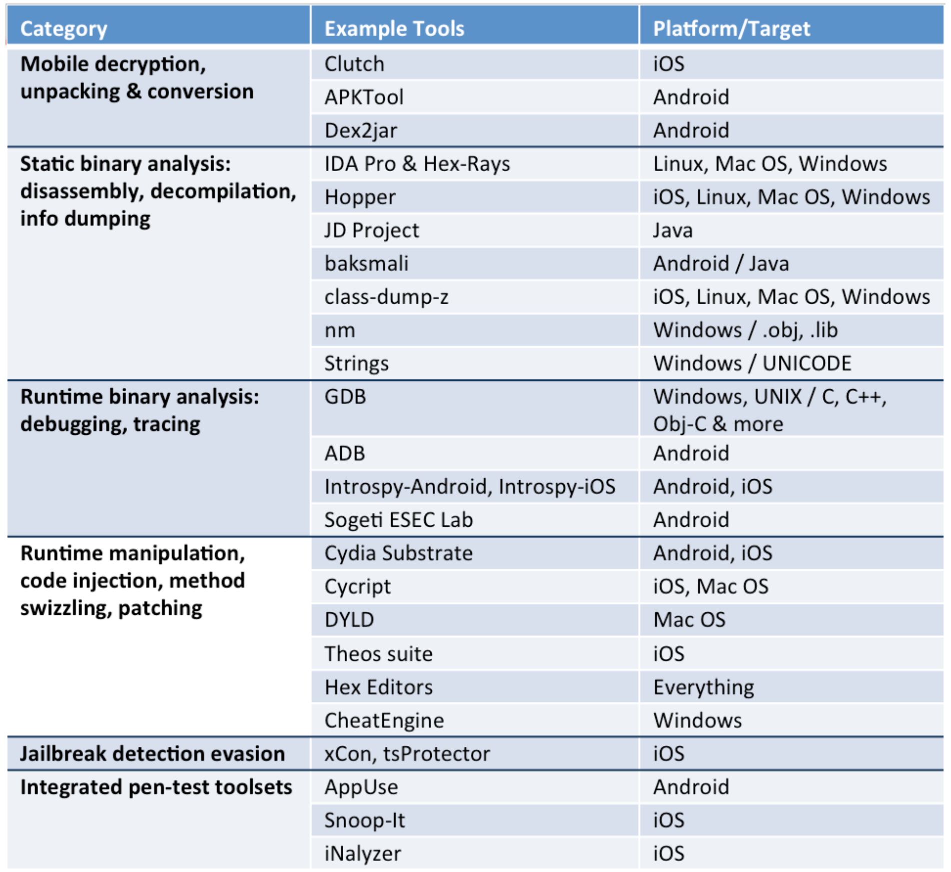
Please use left sidebar to navigate between sections. Last Updated on: **08-10-2015**

Forensics Tools

- Android Forensics - Open Source Android Forensics App and Framework
- Android Data Extractor Lite
- BitPim - BitPim is a program that allows you to view and manipulate data on many CDMA phones from LG, Samsung, Sanyo and other manufacturers.
- LiME - LiME (formerly DMD) is a Loadable Kernel Module (LKM), which allows the acquisition of volatile memory from Linux and Linux-based devices, such as those powered by Android.
- Open Source Android Forensics
- P2P-ADB - Phone to Phone Android Debug Bridge - A project for "debugging" phones from other phones.
- pySimReader - It allows users to write out arbitrary raw SMS PDUs to a SIM card.

</> Development Tools

- Android SDK - The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials.
- Android NDK - The NDK is a toolset that allows you to implement parts of your app using native-code languages such as C and C++.
- ADT Bundle - The Android Developer Tools(ADT) bundle is a single download that contains everything for developers to start creating Android Application
  - Android Studio IDE or Eclipse IDE
  - Android SDK tools
  - Android 5.0 (Lollipop) Platform
  - Android 5.0 emulator system image with Google APIs
- Native Android Runtime Emulation - A native Android emulator featuring the following



Category	Example Tools	Platform/Target
Mobile decryption, unpacking & conversion	Clutch	iOS
	APKTool	Android
	Dex2jar	Android
Static binary analysis: disassembly, decompilation, info dumping	IDA Pro & Hex-Rays	Linux, Mac OS, Windows
	Hopper	iOS, Linux, Mac OS, Windows
	JD Project	Java
	baksmali	Android / Java
	class-dump-z	iOS, Linux, Mac OS, Windows
	nm	Windows / .obj, .lib
	Strings	Windows / UNICODE
Runtime binary analysis: debugging, tracing	GDB	Windows, UNIX / C, C++, Obj-C & more
	ADB	Android
	Introspy-Android, Introspy-iOS	Android, iOS
	Sogeti ESEC Lab	Android
Runtime manipulation, code injection, method swizzling, patching	Cydia Substrate	Android, iOS
	Cycript	iOS, Mac OS
	DYLD	Mac OS
	Theos suite	iOS
	Hex Editors	Everything
	CheatEngine	Windows
Jailbreak detection evasion	xCon, tsProtector	iOS
Integrated pen-test toolsets	AppUse	Android
	Snoop-It	iOS
	iNalyzer	iOS

How We Protect Binary Applications

some Guards for various platforms

Detect
attacks at run time

- Checksum
- Anti-Debug
- Resource Verification
- Jailbreak/Root Detection
- Swizzle / Hook Detection

Defend
Against compromise

- Advanced Obfuscation
- Encryption
- Pre-Damage
- Metadata Removal

Deter
to ward off attacks

- Repair
- Custom Reactions
- Shut Down (Exit, Fail)
- Alert / Phone Home

Protected App

- Self-defending
- Tamper-resistant
- Hardened against hacking attacks & malware exploits

… more Guards and Parameters

for multiple platforms, not always the same on all platforms

- Encryption Wrapper
- Authentication
- Value Verification
- Renaming
- Resource Encryption
- Data Obfuscation
- Custom Guards
- Garbage Code
- ... more to come

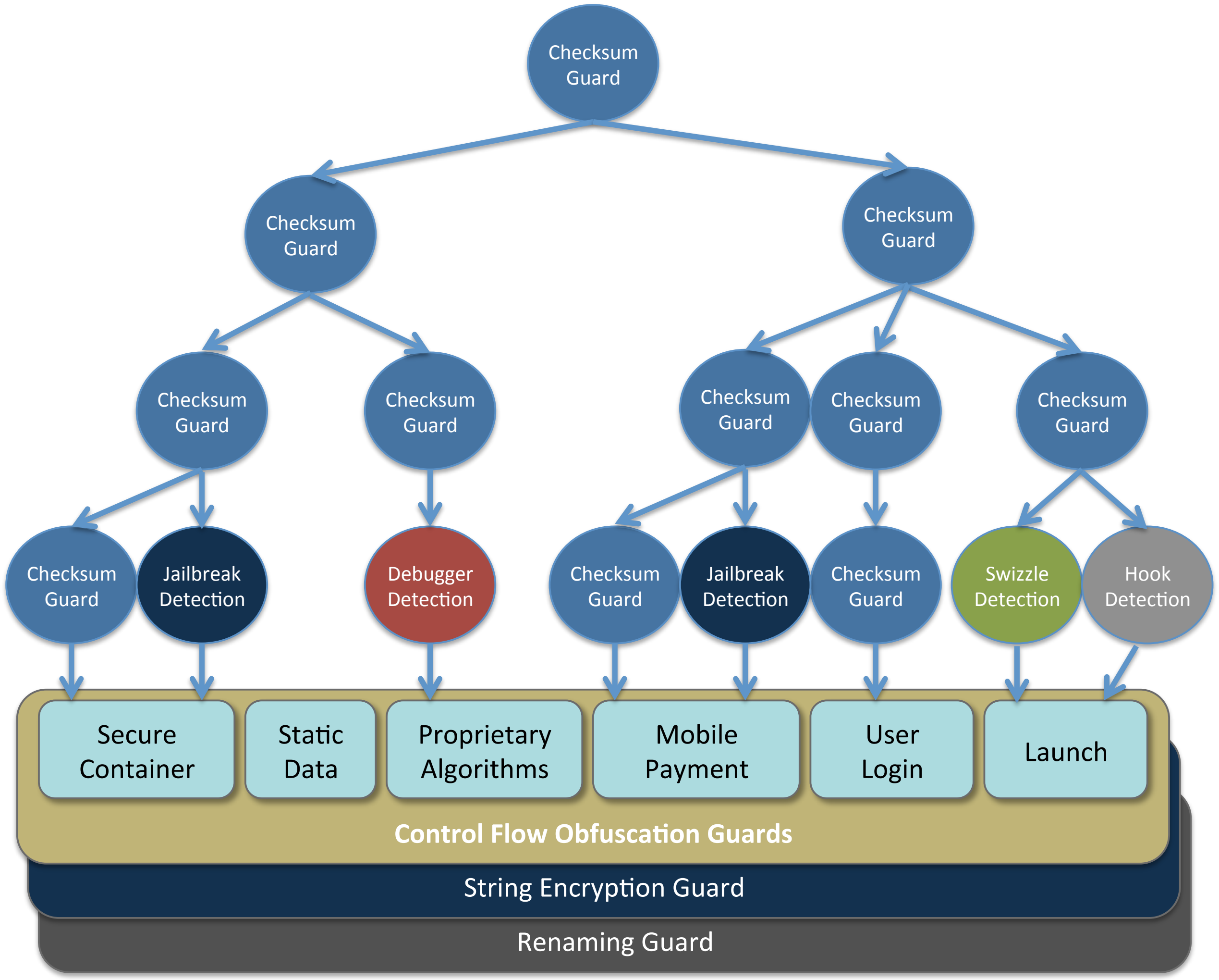
... more possibilities

Parameters

- Different algorithms
- Performance vs Security
- Range / Invocations
- Execution Probabilities
- Seeding
- Logs (e.g. for renaming)
- Debugging
- Actions
- …

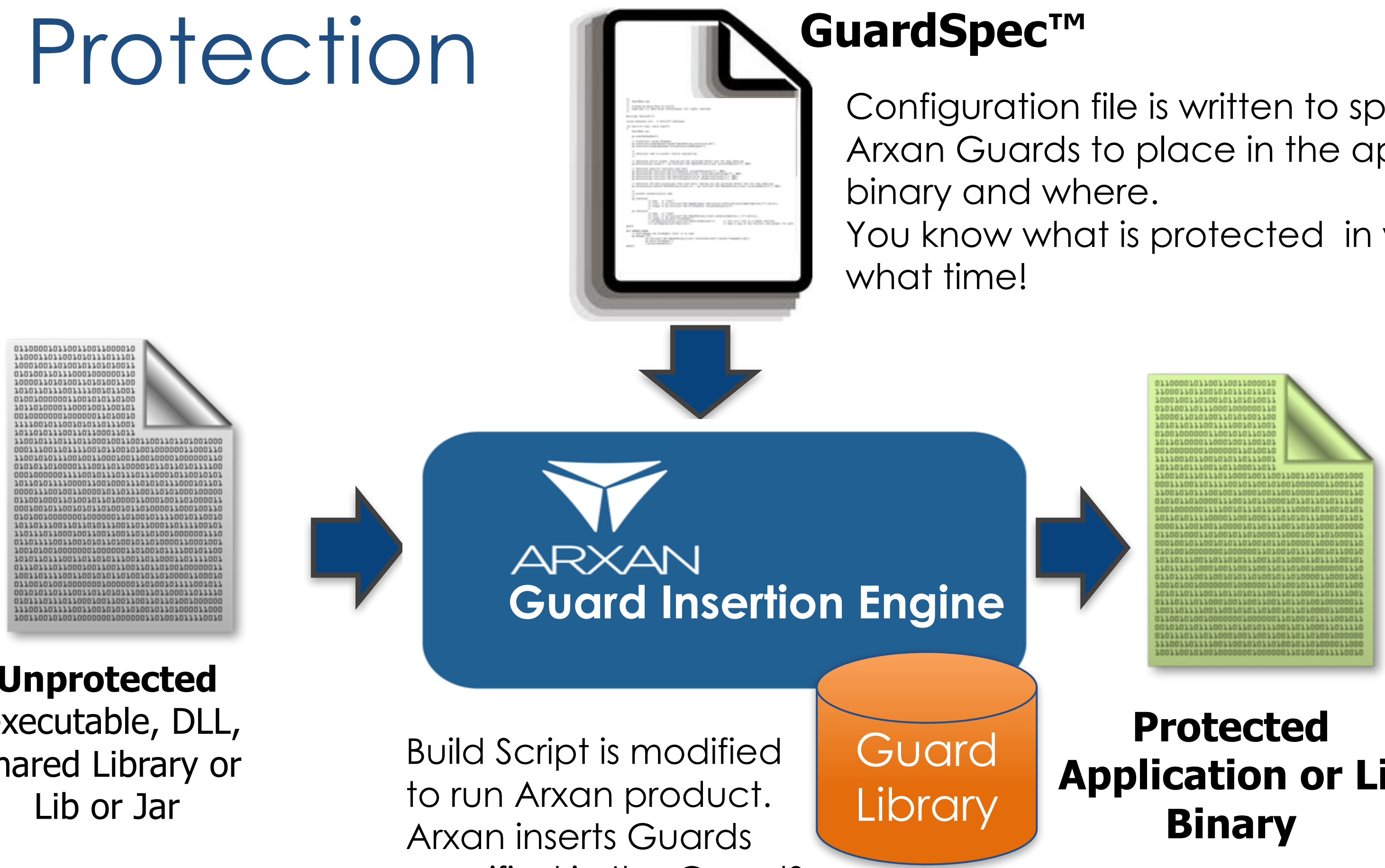
What makes Arxan successful?

Protecting the Protection



- Each Guard provides security itself
- Fine grained control, fully customizable
- Multi-Layered Guard Network
- with Defense in Depth (first Guard layer protects code, additional Guard layers protect lower-level Guards)
- Risk-Based, Custom Created for Each Application
- Randomized Binary Implementation for Automated Variability (every build looks different)
- Think Templates for Guards!

Adding Protection



GuardSpec™

Configuration file is written to specify which Arxan Guards to place in the application binary and where.
You know what is protected in which way at what time!

Unprotected executable, DLL, Shared Library or Lib or Jar

Build Script is modified to run Arxan product. Arxan inserts Guards specified in the GuardSpec into the unprotected binary. Guard Library contains many different Guard Types and thousands of Guard instances.

Protected Application or Lib Binary

Guards fully dissolve into binary and cannot be isolated or identified. No accompanyin g libraries or need to connect to Arxan at run-time.

Integration with focus on iOS / Android

iOS

- Xcode/xcodebuild integration per **Xcode plug-in / xcodebuild**
- Integration into LLVM toolchain

Java

- Integration with **gradle** (therefore also Android Studio)
- APK protection

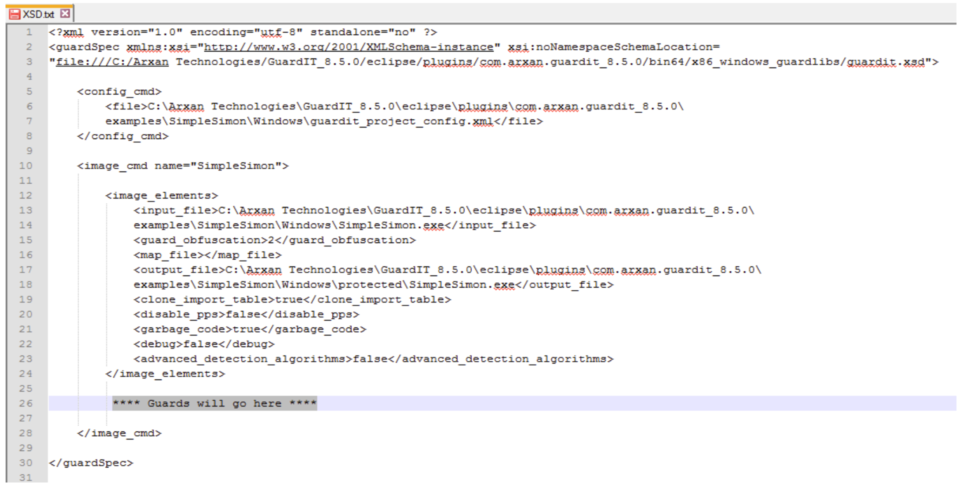
Native

- Integration with **ndk-build**
- Integration into LLVM toolchain

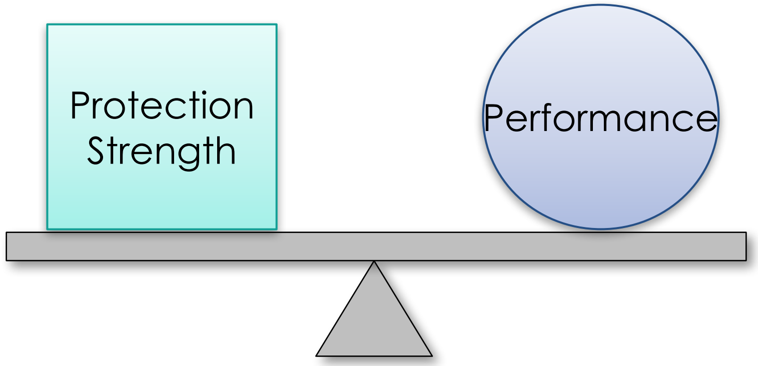
- Protection (Guard Insertion) Engines usable per IDE or build management / command line, therefore usable from Jenkins etc.

Guards and managable Impact

- Guards are protection primitives that are inserted into your application.
- Guards are inserted for each binary based on the configuration of each Guard defined in the GuardSpec.
- Guards add security but also increase code size and execution time which can be adjusted by using parameters. How much is a matter of *good* configuration.



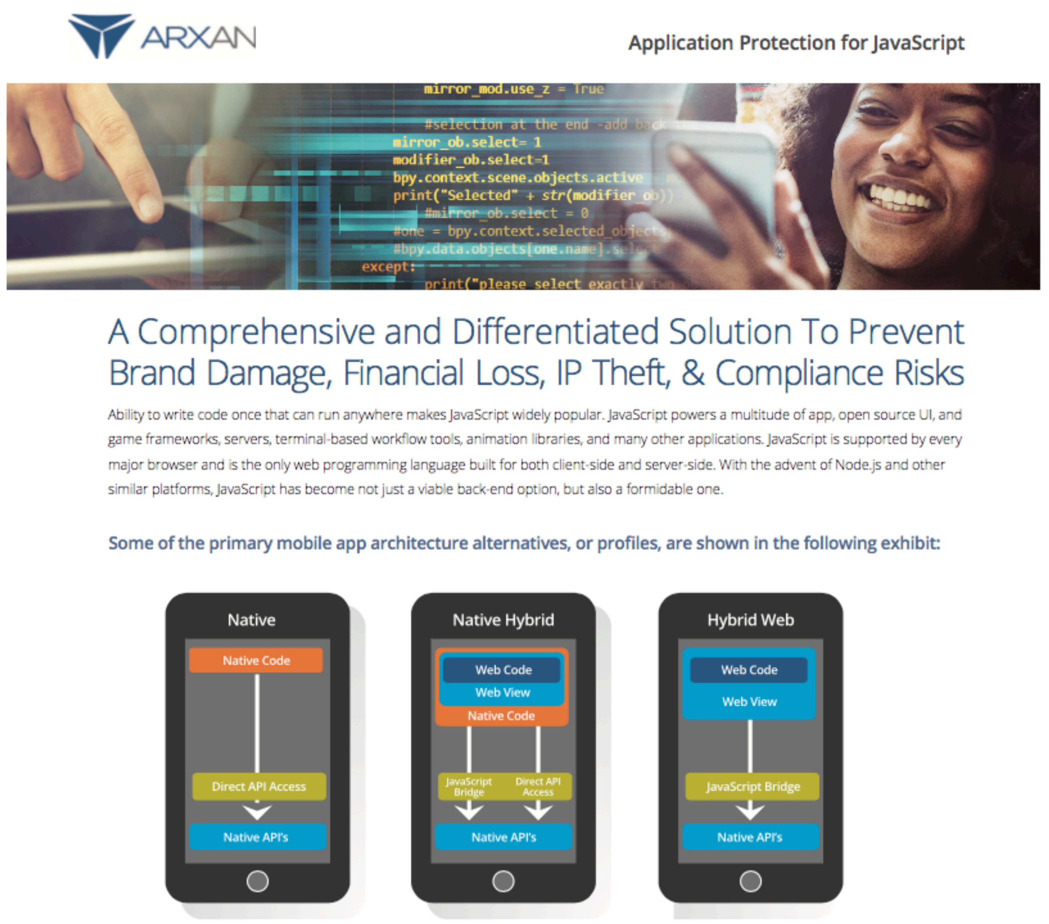
```xml
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<guardSpec xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation=
"file:///C:/Arxan Technologies/GuardIT_8.5.0/eclipse/plugins/com.arxan.guardit_8.5.0/bin64/x86_windows_guardlibs/guardit.xsd">

    <config_cmd>
        <file>C:\Arxan Technologies\GuardIT_8.5.0\eclipse\plugins\com.arxan.guardit_8.5.0\
        examples\SimpleSimon\Windows\guardit_project_config.xml</file>
    </config_cmd>

    <image_cmd name="SimpleSimon">

        <image_elements>
            <input_file>C:\Arxan Technologies\GuardIT_8.5.0\eclipse\plugins\com.arxan.guardit_8.5.0\
            examples\SimpleSimon\Windows\SimpleSimon.exe</input_file>
            <guard_obfuscation>2</guard_obfuscation>
            <map_file></map_file>
            <output_file>C:\Arxan Technologies\GuardIT_8.5.0\eclipse\plugins\com.arxan.guardit_8.5.0\
            examples\SimpleSimon\Windows\protected\SimpleSimon.exe</output_file>
            <clone_import_table>true</clone_import_table>
            <disable_pps>false</disable_pps>
            <garbage_code>true</garbage_code>
            <debug>false</debug>
            <advanced_detection_algorithms>false</advanced_detection_algorithms>
        </image_elements>

         **** Guards will go here ****

    </image_cmd>

</guardSpec>
```



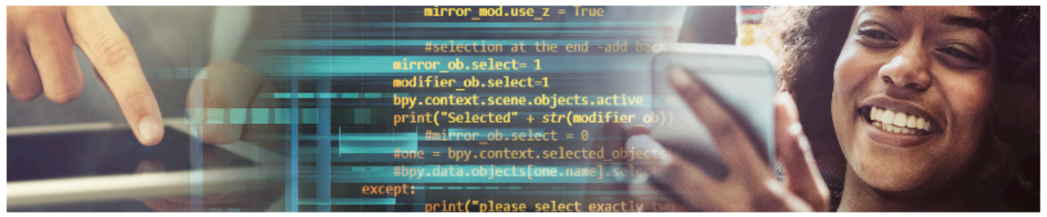
Application Protection of JavaScript

- JavaScript Protection (SecureJS)
  - source transformation protection
  - supports ECMAScript 5,6
  - not binary protection
  - in the cloud solution or on premise
  - supporting multiple architecture development platforms, hybrid apps
  - Very fast obfuscation
  - Static and dynamic protection (Guards)

- Apache Cordova Solution
  - Example package
  - Several enhanced topics in development
  - Other frameworks in work



ARXAN Application Protection for JavaScript

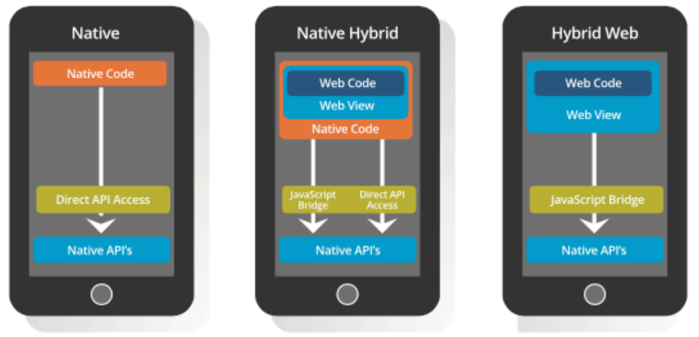


```
mirror_mod.use_z = True
#selection at the end -add back
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active
print("Selected" + str(modifier_ob))
#mirror_ob.select = 0
#one = bpy.context.selected_objects
#bpy.data.objects[one.name].sel
except:
print("please select exactly two
```

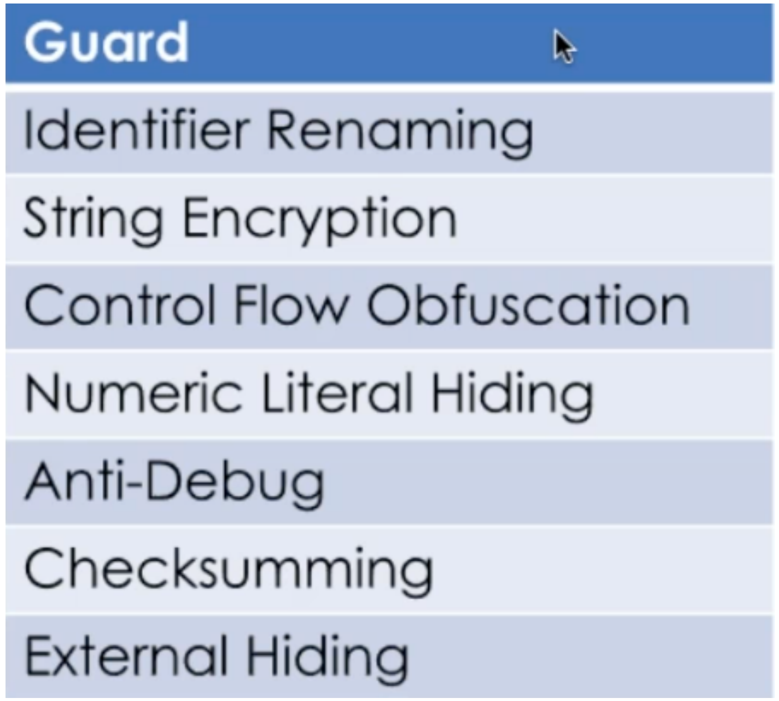
A Comprehensive and Differentiated Solution To Prevent Brand Damage, Financial Loss, IP Theft, & Compliance Risks

Ability to write code once that can run anywhere makes JavaScript widely popular. JavaScript powers a multitude of app, open source UI, and game frameworks, servers, terminal-based workflow tools, animation libraries, and many other applications. JavaScript is supported by every major browser and is the only web programming language built for both client-side and server-side. With the advent of Node.js and other similar platforms, JavaScript has become not just a viable back-end option, but also a formidable one.

Some of the primary mobile app architecture alternatives, or profiles, are shown in the following exhibit:



SecureJS Features



Guard
Identifier Renaming
String Encryption
Control Flow Obfuscation
Numeric Literal Hiding
Anti-Debug
Checksumming
External Hiding

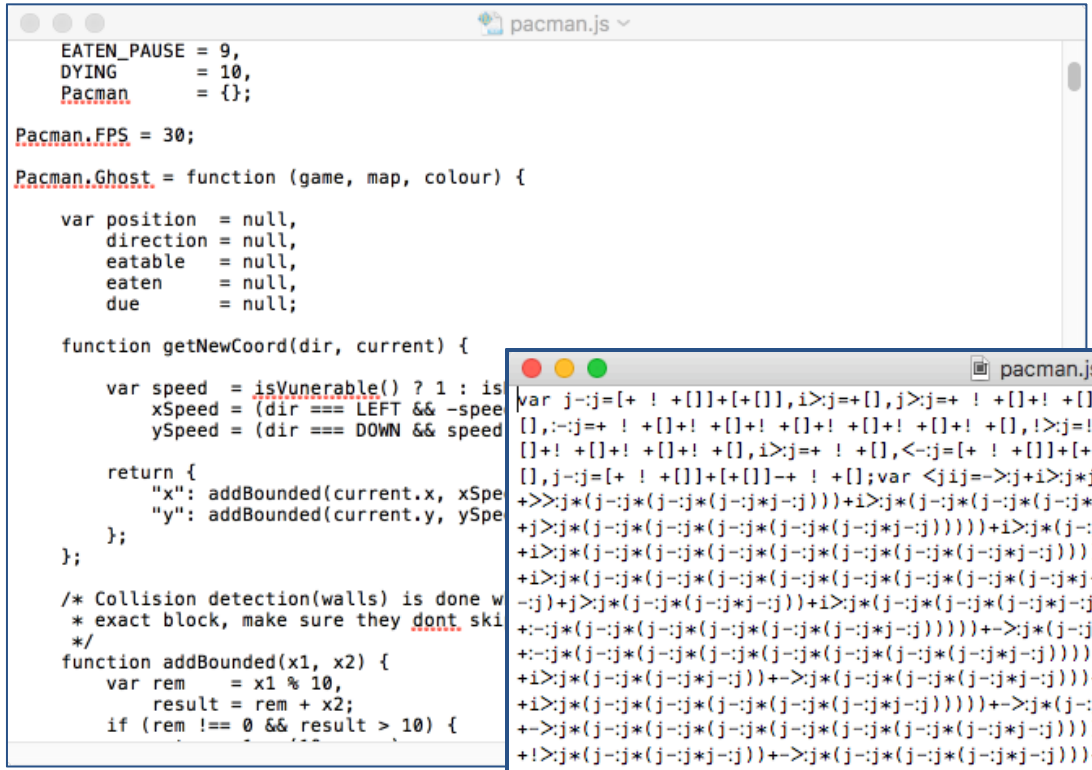
Guards include Operator Removal and Minification and dynamic Guards are interwoven

Useful performance?

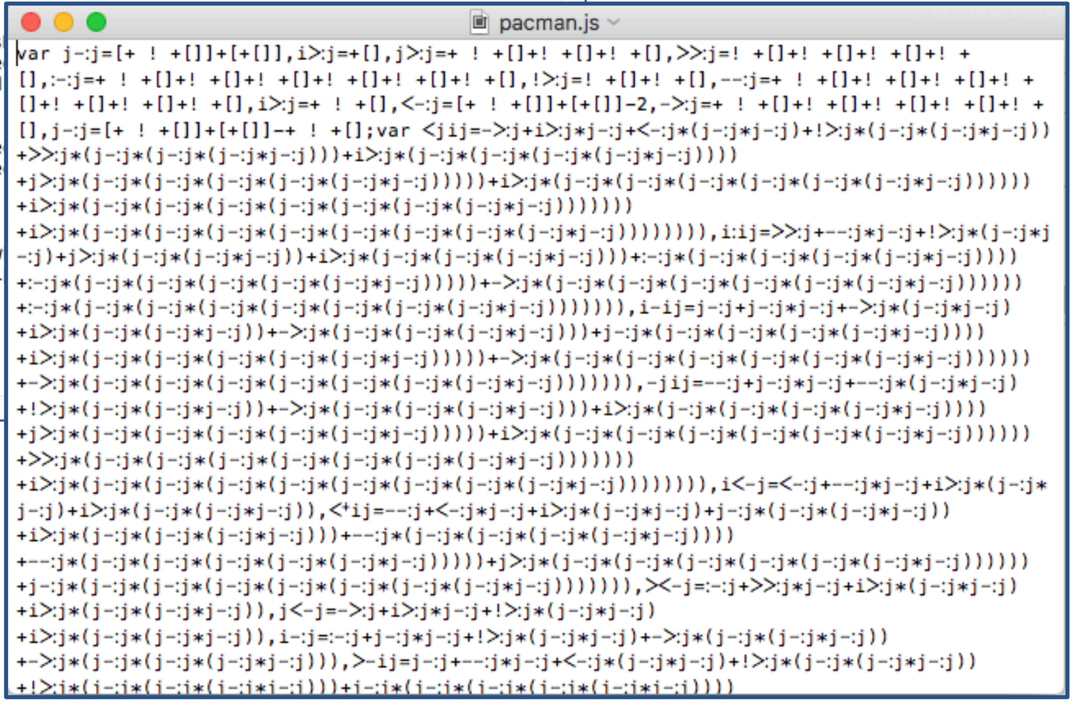
- Product has been performance and externally pen tested against competition with very good results

- Generally available for 4 months now
  - In evaluation with plenty of early bird companies for 6 months
  - Applied even in Web Gaming
  - Cordova solutions / examples
  - More enhancements, size and performance optimization upcoming
  - More Guards and features for server environments

JS - Which One Would You Rather Hack?


pacman.js

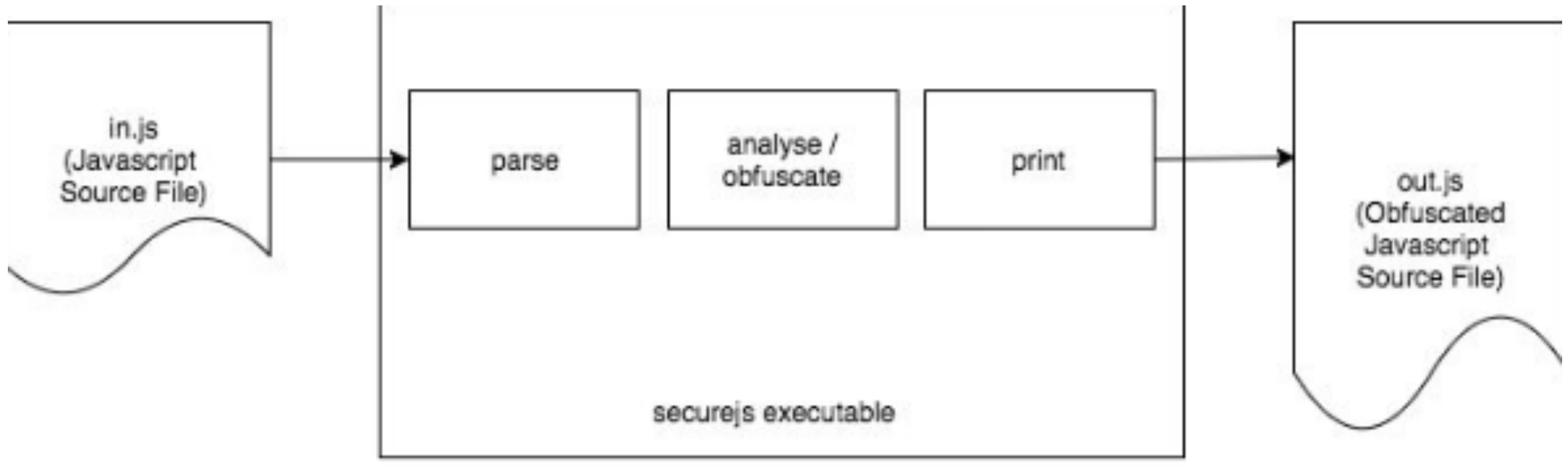
```
    EATEN_PAUSE = 9,
    DYING       = 10,
    Pacman      = {};

Pacman.FPS = 30;

Pacman.Ghost = function (game, map, colour) {

    var position  = null,
        direction = null,
        eatable   = null,
        eaten     = null,
        due       = null;

    function getNewCoord(dir, current) {

        var speed  = isVunerable() ? 1 : is
            xSpeed = (dir === LEFT && -spee
            ySpeed = (dir === DOWN && speed

        return {
            "x": addBounded(current.x, xSpe
            "y": addBounded(current.y, ySpe
        };
    };

    /* Collision detection(walls) is done w
     * exact block, make sure they dont ski
     */
    function addBounded(x1, x2) {
        var rem    = x1 % 10,
            result = rem + x2;
        if (rem !== 0 && result > 10) {
```


pacman.js

```
var j-:j=[+ ! +[]]+[+[]],i>:j=+[],j>:j=+ ! +[]+! +[]+! +[],>>:j=! +[]+! +[]+! +[]+! +
[],:-:j=+ ! +[]+! +[]+! +[]+! +[]+! +[]+! +[],!>:j=! +[]+! +[],--:j=+ ! +[]+! +[]+! +[]+! +
[]+! +[]+! +[]+! +[],i>:j=+ ! +[],<-:j=[+ ! +[]]+[+[]]-2,->:j=+ ! +[]+! +[]+! +[]+! +[]+! +
[],j-:j=[+ ! +[]]+[+[]]-+ ! +[];var <jij=->:j+i>:j*j-:j+<-:j*(j-:j*j-:j)+!>:j*(j-:j*(j-:j*j-:j))
+>>:j*(j-:j*(j-:j*(j-:j*j-:j)))+i>:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))
+j>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))+i>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))))
+i>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))))
+i>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))))),i:ij=>>:j+--:j*j-:j+!>:j*(j-:j*j
-:j)+j>:j*(j-:j*(j-:j*j-:j))+i>:j*(j-:j*(j-:j*(j-:j*j-:j)))+:-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))
+:-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))+->:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))))
+:-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))))),i-ij=j-:j+j-:j*j-:j+->:j*(j-:j*j-:j)
+i>:j*(j-:j*(j-:j*j-:j))+->:j*(j-:j*(j-:j*(j-:j*j-:j)))+j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))
+i>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))+->:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))))
+->:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))))),-jij=--:j+j-:j*j-:j+--:j*(j-:j*j-:j)
+!>:j*(j-:j*(j-:j*j-:j))+->:j*(j-:j*(j-:j*(j-:j*j-:j)))+i>:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))
+j>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))+i>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))))
+>>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))))
+i>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))))),i<-j=<-:j+--:j*j-:j+i>:j*(j-:j*
j-:j)+i>:j*(j-:j*(j-:j*j-:j)),<+ij=--:j+<-:j*j-:j+i>:j*(j-:j*j-:j)+j-:j*(j-:j*(j-:j*j-:j))
+i>:j*(j-:j*(j-:j*(j-:j*j-:j)))+--:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))
+--:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j)))))+j>:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))))
+j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))))),><-j=:-:j+>>:j*j-:j+i>:j*(j-:j*j-:j)
+i>:j*(j-:j*(j-:j*j-:j)),j<-j=->:j+i>:j*j-:j+!>:j*(j-:j*j-:j)
+i>:j*(j-:j*(j-:j*j-:j)),i-:j=:-:j+j-:j*j-:j+!>:j*(j-:j*j-:j)+->:j*(j-:j*(j-:j*j-:j))
+->:j*(j-:j*(j-:j*(j-:j*j-:j))),>-ij=j-:j+--:j*j-:j+<-:j*(j-:j*j-:j)+!>:j*(j-:j*(j-:j*j-:j))
+!>:j*(j-:j*(j-:j*(j-:j*j-:j)))+j-:j*(j-:j*(j-:j*(j-:j*(j-:j*j-:j))))
```

- “Ugly” one liner
- Actually regular, executable Code
- Contains:
  - Interwoven Guards
  - Transformations like renaming in overall project
  - Debugger detection
  - Checksums
  - And more …

Processing from Cmd line / Jenkins / Cloud

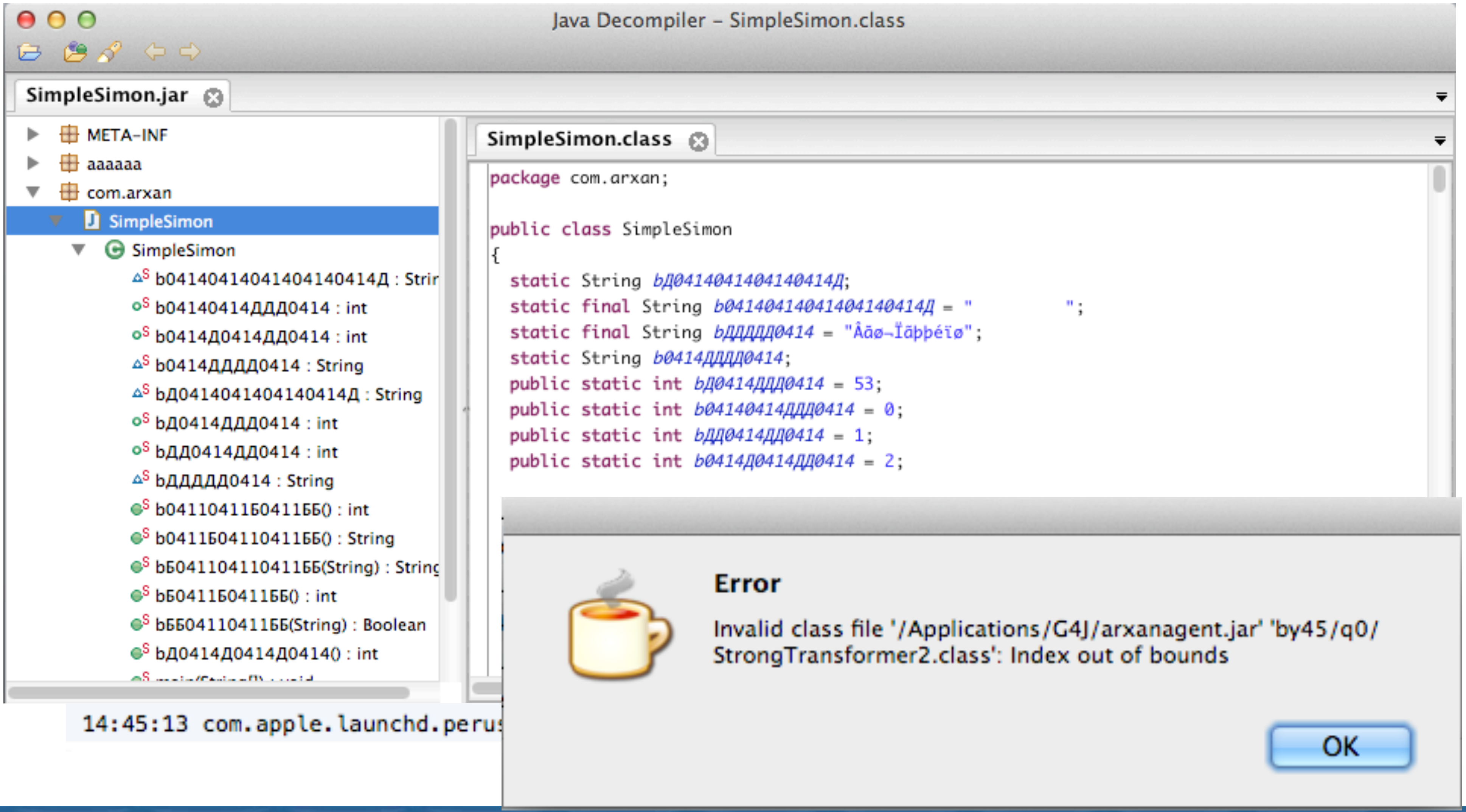


```
securejs -i in.js -o out.js
```

Inputs

- Single files
- Whole folders, ZIP files
- Various options
- Configuration file

Java Protection Example and Decompilation


Java Decompiler – SimpleSimon.class

SimpleSimon.jar

```
META-INF
aaaaaa
com.arxan
  SimpleSimon
    SimpleSimon
      b0414041404140414041 4Д : Strir
      b04140414ДДД0414 : int
      b0414Д0414ДД0414 : int
      b0414ДДДД0414 : String
      bД04140414041404 14Д : String
      bД0414ДДД0414 : int
      bДД0414ДД0414 : int
      bДДДДД0414 : String
      b04110411Б0411ББ() : int
      b0411Б04110411ББ() : String
      bБ041104110411ББ(String) : String
      bБ0411Б0411ББ() : int
      bББ04110411ББ(String) : Boolean
      bД0414Д0414Д0414() : int
```

SimpleSimon.class

```java
package com.arxan;

public class SimpleSimon
{
  static String bД0414041404140414Д;
  static final String b04140414041404140414Д = "          ";
  static final String bДДДДД0414 = "Âãø¬Ïãþþéïø";
  static String b0414ДДДД0414;
  public static int bД0414ДДД0414 = 53;
  public static int b04140414ДДД0414 = 0;
  public static int bДД0414ДД0414 = 1;
  public static int b0414Д0414ДД0414 = 2;
```

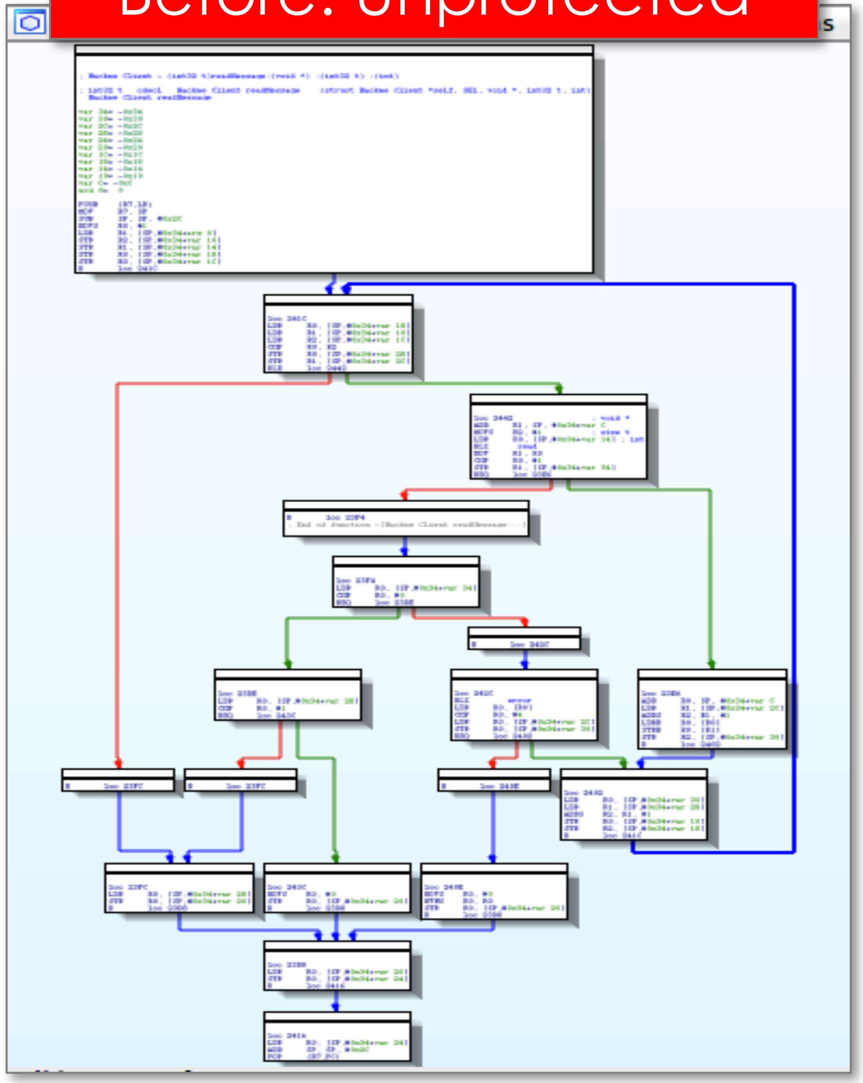
```
14:45:13 com.apple.launchd.perus
```

Error

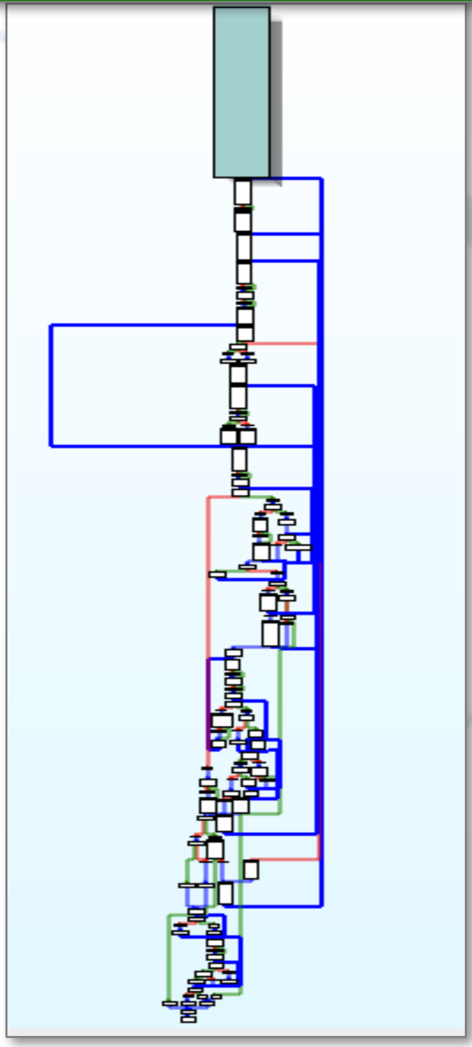
Invalid class file '/Applications/G4J/arxanagent.jar' 'by45/q0/StrongTransformer2.class': Index out of bounds

Passive Guard Example: Control Flow Obfuscation

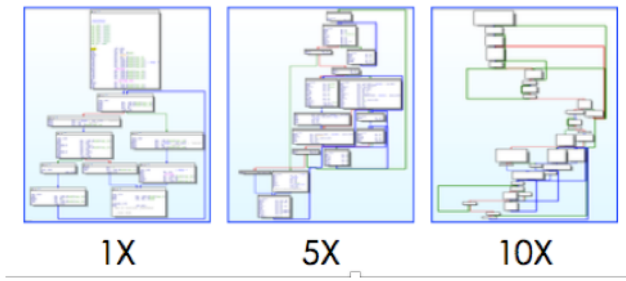
Before: Unprotected



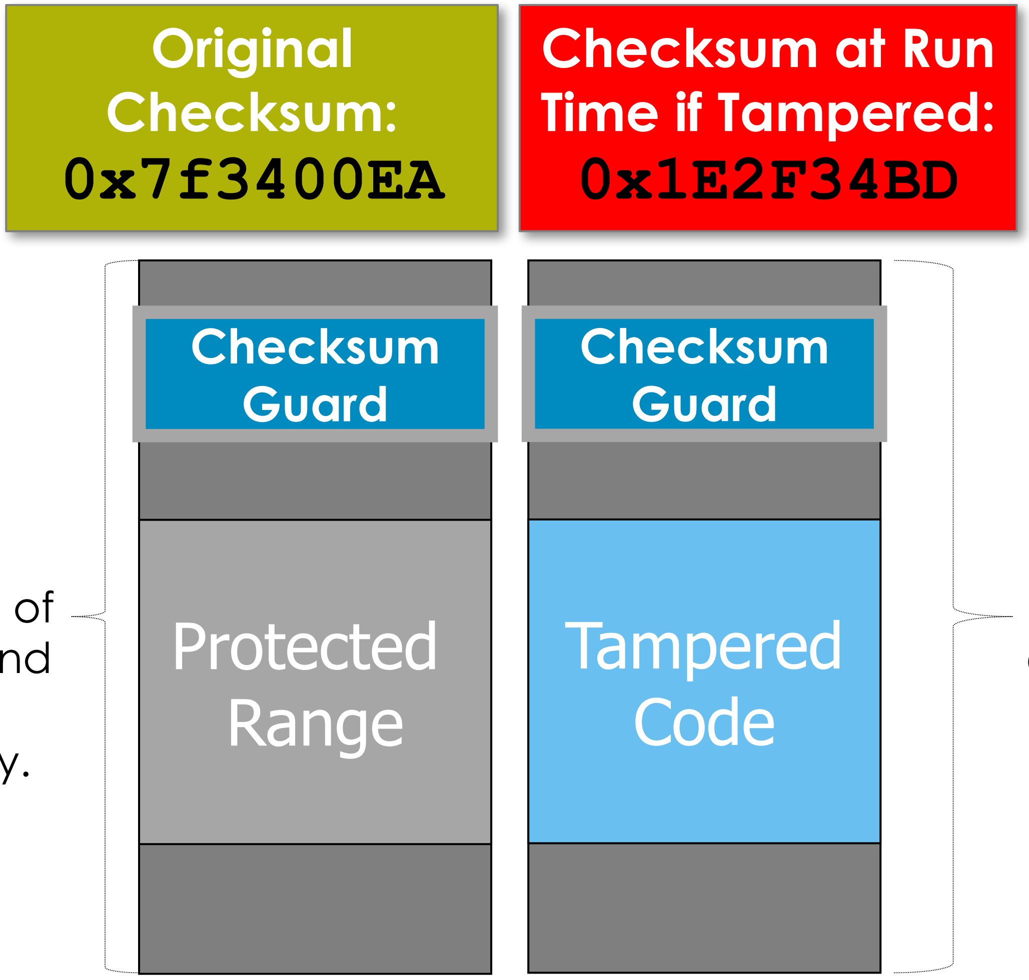
After: Protected



Makes code very difficult to read and breaks static analysis. Techniques used include inserting dummy code, instruction substitution, path merging, block and symbol shuffling, inlining, opaque predicates, jump instructions, and more



Checksum Guard detects Tampering

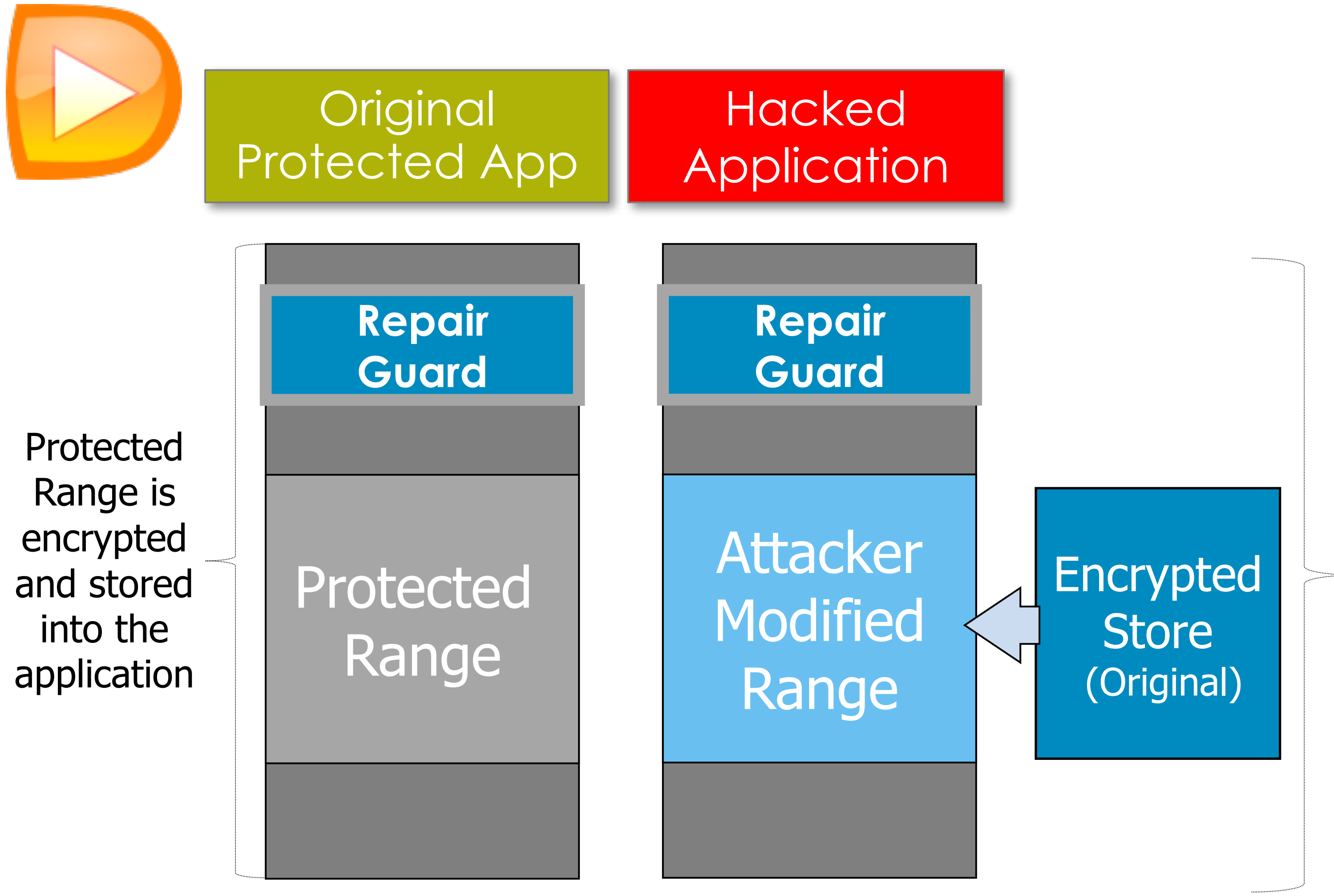


Checksum of Protected Range of binary created and hidden in the protected binary.

At runtime a Checksum of the Protected Range is created and compared against Original Checksum.

Guard triggers if no match

React: Self-Repairing a Tampered App



Protected Range is encrypted and stored into the application

If app is found to be tampered with then the Protected Range can self-heal using the master version to overwrite the tampered version so program execution can continue.

Thank you!
Mirko Brandner – mbrandner@arxan.com

Additional Information Available at Arxan.com

Mobile Application Protection Handbook

Financial Services

Enterprise Apps

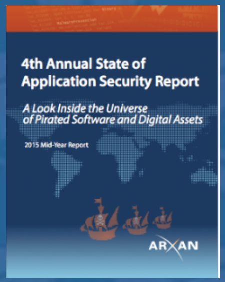
Healthcare

Gaming

Digital Media

ARXAN

Defend Against Application Risks & Attacks
by Protecting the Integrity of Your Binary Code


4th Annual State of
Application Security Report

*A Look Inside the Universe
of Pirated Software and Digital Assets*

2015 Mid-Year Report

ARXAN

State of Mobile App Security

Apps Under Attack

Research Report
Special Focus on Financial, Retail/Merchant and Healthcare/
Medical Apps

Volume 3 – November 2014
(Previously titled: State of Security in the App Economy)

ARXAN

Financial Services Edition

5th Annual State of Application Security Report
Perception vs. Reality

January 2016

ARXAN

White Paper

ARXAN

Securing Cryptographic Keys

A Sophisticated Implementation of White-Box Cryptography Using Arxan's
TransformIT® Solution

Version 1.0

2016

STATE OF
APPLICATION
SECURITY

ARXAN

AN IN-DEPTH LOOK INTO
MOBILE FINANCE & PAYMENT APPS

PERCEPTION

PERCEPTION OF
SECURITY

APPLICATION
EXECUTIVES

APPLICATION
USERS