RSAConference2015
Singapore | 22-24 July | Marina Bay Sands

CHANGE

Challenge today's security thinking

SESSION ID: MBS-F04

Mitigating Cybersecurity Threats on Mobile Platforms

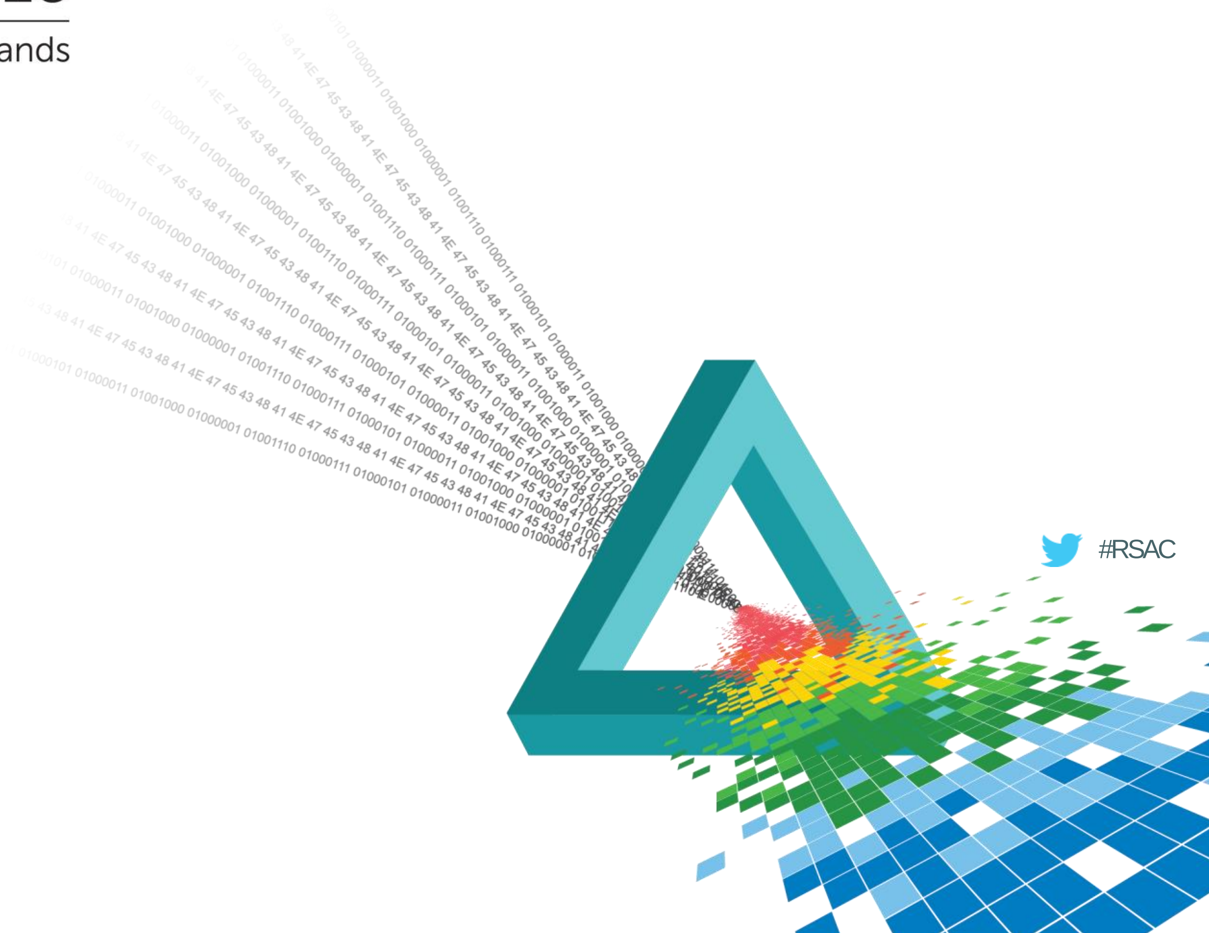
Selim Aissi, Taeho Kgil, Gyan Prakash

Agenda

- Threats
- Mitigation
- Summary

RSA®Conference2015

Singapore | 22-24 July | Marina Bay Sands



#RSAC

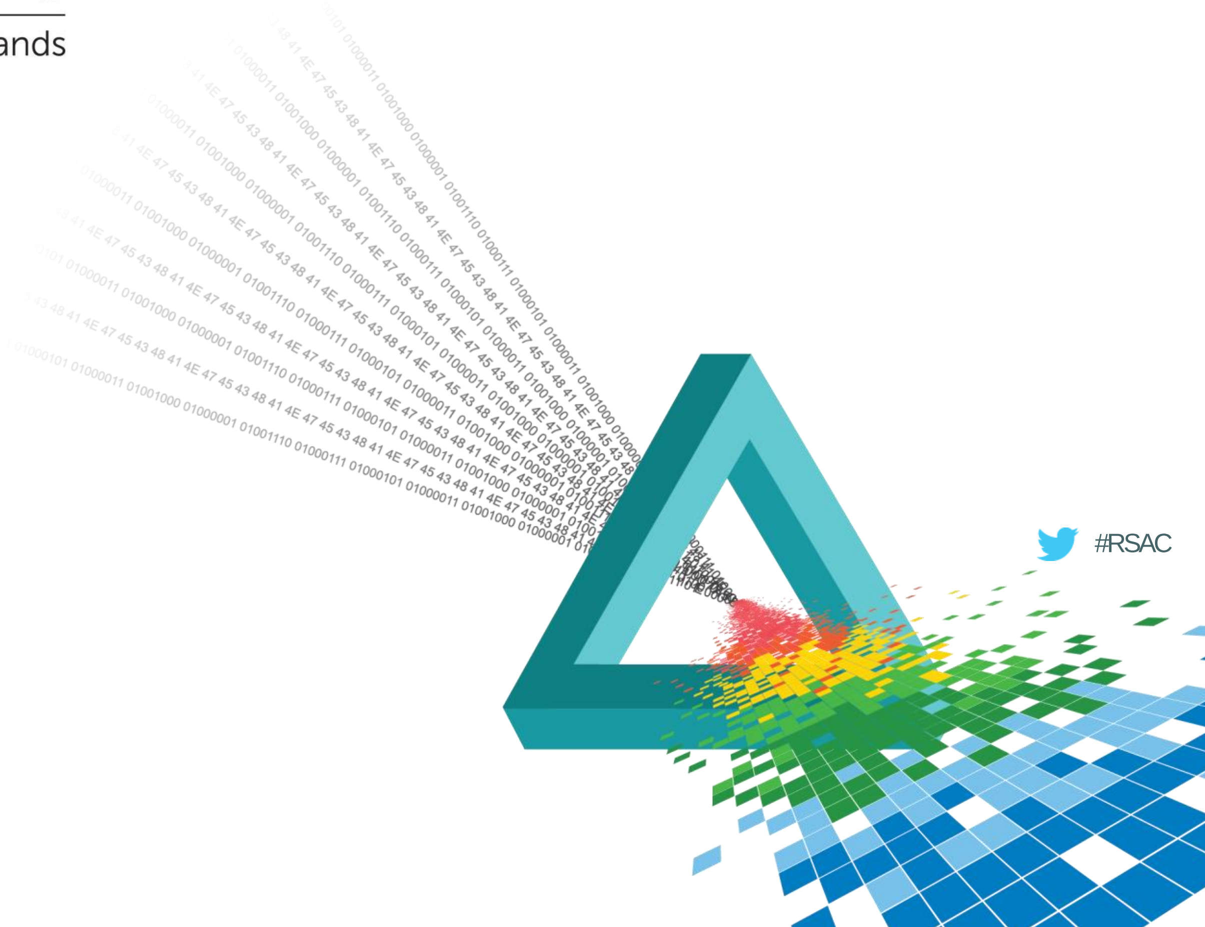
Threats

Cybersecurity Threats on Mobile Platforms

- Sensitive Information in primary stores (key, data) is leaked **(Data Leakage)**
- Sensitive Information for data-in-use is leaked **(Data Leakage)**
- Sensitive Information Not Cleared From Data-in-use **(Data Leakage)**
- Primary stores (key, data) are tampered with and set to deterministic state **(Tampering)**
- Code Module, OS platform and device is cloned **(Spoofing)**
- Code Module executing security capabilities tampered with **(Tampering)**
- Denial of Service launched on Code Module **(DOS)**

Best Practices for Addressing those Threats

- Setting up clear and comprehensive Security Architecture Principles
- Setting up clear, comprehensive, and achievable Security Objectives
- Defining comprehensive Mitigating Security Controls
- Translating those Controls into Platform Security Requirements



#RSAC

Mitigation

Security Architecture Principles

- Apply defense in depth (complete mediation)
- Use a positive security model (fail-safe defaults, minimize attack surface)
- Fail securely
- Run with least privilege
- Avoid security by obscurity (open design)
- Keep security simple (verifiable, economy of mechanism)
- Detect intrusions (compromise recording)
- Establish secure defaults (psychological acceptability)
- Don’t trust external systems

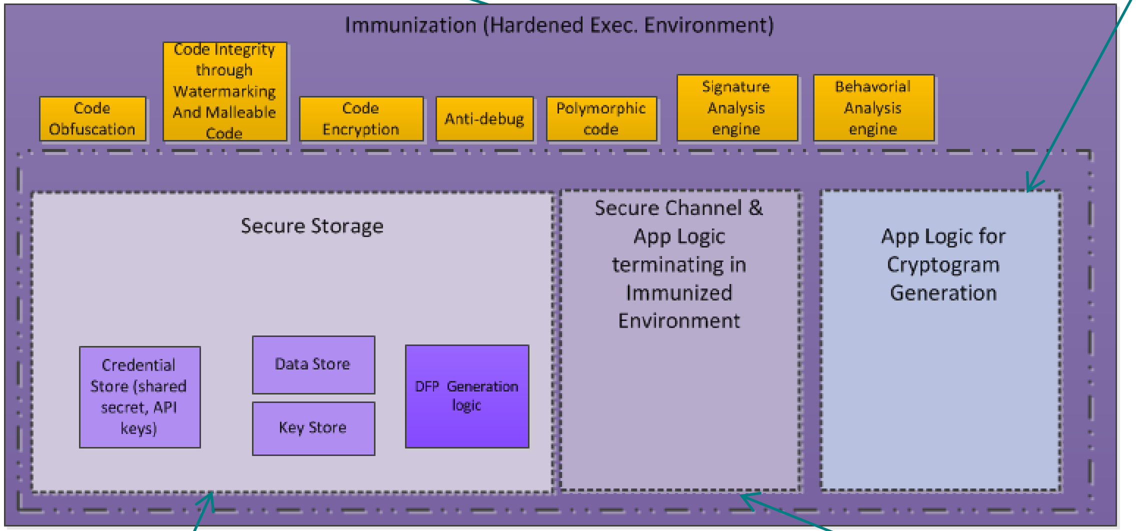
Setting the Security Objectives

- Confidentiality and integrity of cryptographic keys
- Confidentiality and integrity of cryptographic process
- Confidentiality and integrity of data-at-rest
- Confidentiality and integrity of data-in-transit
- Confidentiality and integrity of application memory & storage
- Confidentiality and integrity of virtual machine codes
- Confidentiality and integrity of user input
- Integrity of application codes

Mitigating Security Controls

Binary Immunization: resist, detect, minimize and repair from tampering

App Logic to Generate Cryptogram based on data elements in **Secure Storage**


Immunization (Hardened Exec. Environment)
Code Obfuscation
Code Integrity through Watermarking And Malleable Code
Code Encryption
Anti-debug
Polymorphic code
Signature Analysis engine
Behavorial Analysis engine
Secure Storage
Credential Store (shared secret, API keys)
Data Store
Key Store
DFP Generation logic
Secure Channel & App Logic terminating in Immunized Environment
App Logic for Cryptogram Generation

Secure Storage: Host hardened Data, Key, Credential and DFP store

App Logic to establish mutually authenticated **Secure Channel**

Platform Security Requirements

- Hardware & Firmware Security
  - Secure Boot
  - SIM Card Security
    - E.g., Mobile Identity Assurance
  - Trusted Execution Environment
    - E.g., Trusted UI, Key Management, SIM lock, HD content protection
  - Mobile Device Vendor Specific Native APIs
    - E.g., SMART APIs
- OS Platform Security
  - Application Signature Verification
  - Access Control & Application Isolation
  - Integrity Framework

Summary

- Setting up clear and comprehensive Security Architecture Principles
- Setting up clear, comprehensive, and achievable Security Objectives
- Defining comprehensive Mitigating Security Controls
- Translating those Controls into Platform Security Requirements