Throw Your (App)Integrity Out the Window

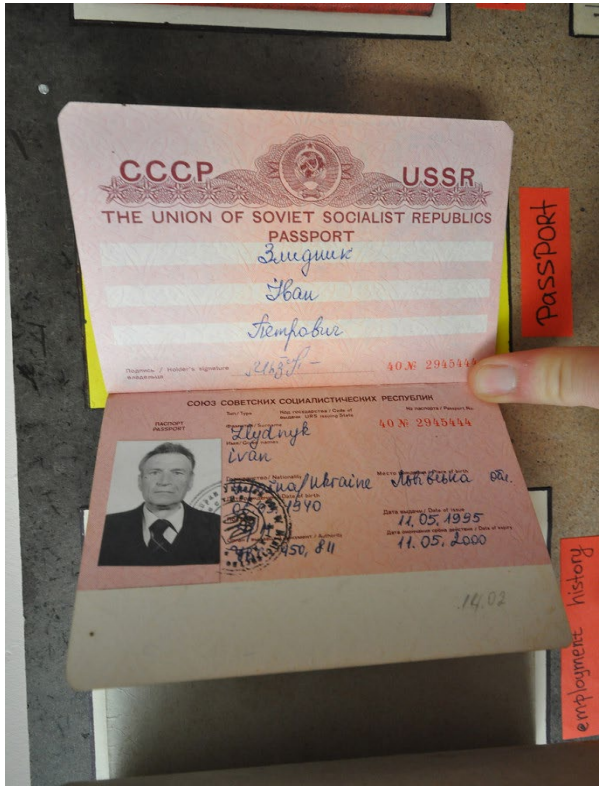
Bypassing Device Integrity Checks on iOS

May 20, 2023

Priyank Nigam

Bio

- Red Team @Microsoft
- Previous
  - Offensive Security Consulting
  - App Dev (C++)
- Research Interests
  - AppSec (Web/mobile)
  - IoT
  - Network Sec
  - MS Azure
- Blue team @Home
  - My toddler in YOLO'ing, RED team mode



Agenda

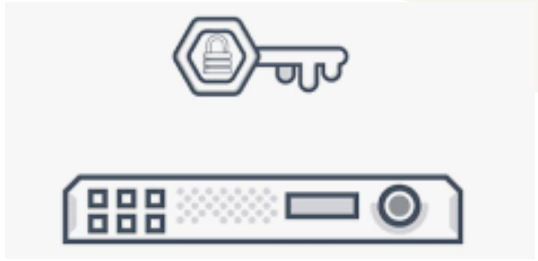
- Why Device Integrity?
- Overview of the Apple's DeviceCheck Framework
  - DCDevice
  - DCAppAttestService
- Bypass App Implementations
  - Runtime Instrumentation
  - Attacks on Server-side APIs
- Best Practices
  - For App developers
  - For Red teamers

Why Device Integrity

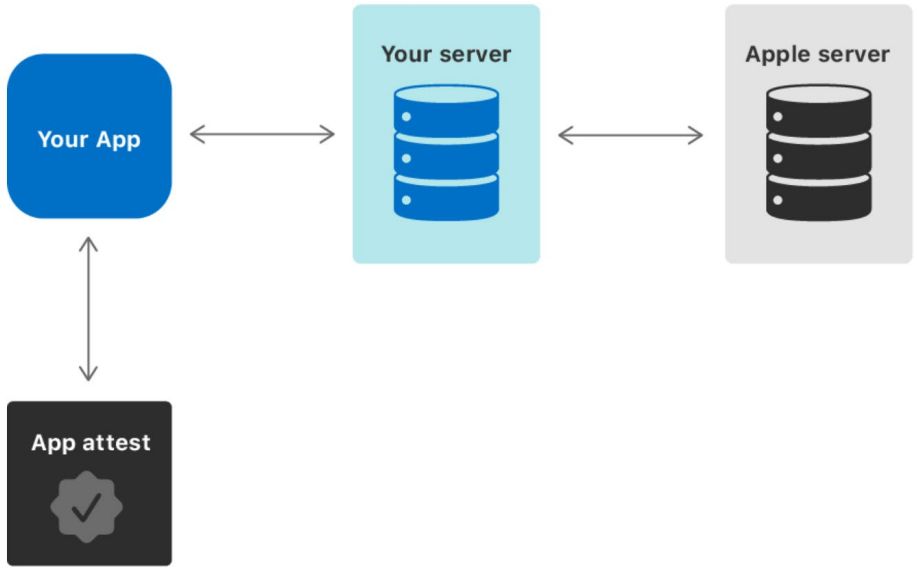
App's logic cannot be relied to perform security checks on itself

Leverage Hardware-based, cryptographic key that uses Apple servers to certify that the key belongs to a valid instance of your app.

Use the service to cryptographically sign server requests using the certified key



DeviceCheck Overview



- Reduce fraudulent use of your services
- `DCDevice` – Verify device authenticity while maintaining user privacy
  - Use case – Identify devices which have already taken advantage of a promo
- `DCAppAttestService` – Verify App validity via one-time challenge-response implementation, leveraging a unique, hardware-backed key.
  - Use cases – Game cheats, ad removal, verify if app is not running a jailbroken device

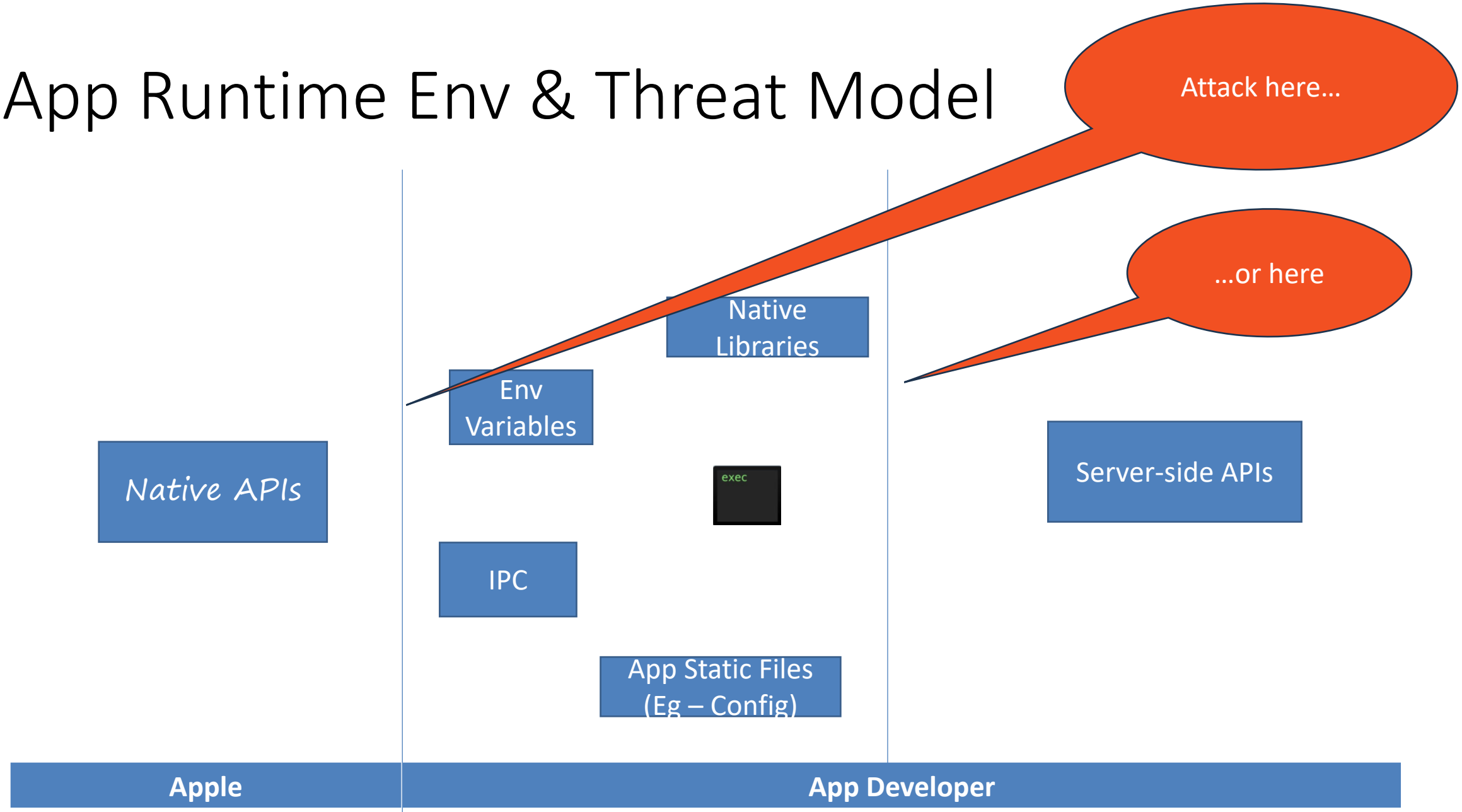
DeviceCheck Overview

- No single policy can eliminate all fraud.
  - Cannot definitively pinpoint Jailbroken device
- However, this info can be integrated into the overall risk assessment of the device

Attacks on the app implementations

- Runtime Instrumentation
- Attacks on Server-Side APIs

App Runtime Env & Threat Model



Dynamic Instrumentation Crash course FRIDA

Frida[1] lets you inject snippets of JavaScript or your own library into native apps

Full access to the target processes

Process memory	Hook functions	Call native functions

[1]https://frida.re/

Bypass Device Integrity Checks

Easy Mode- Swizzle these to false!

- `DCDevice isSupported()`
- `DCAppAttestService isSupported()`

- Returns a boolean value that indicates whether the device supports the corresponding DeviceCheck APIs.

Check if the DeviceCheck API is loaded

```
for (var className in ObjC.classes)
{
        if (ObjC.classes.hasOwnProperty(className))
        {
                console.log(className);
        }
}
```

Runtime Instrumentation

Bypass DCDevice API

```
if DCDevice.current.isSupported { // Always test for availability.

    DCDevice.current.generateToken { token, error in

        guard error == nil else { /* Handle the error. */ }

        // Send the token to your server.

    }

}
```

```
***entered -[DCDevice isSupported]
[-] New Return Value:- 0x0
```

```
if (ObjC.available) {

try {
var className = "DCDevice";
var funcName = "- isSupported";
var hook = eval('ObjC.classes.' + className + '["' +
funcName + '"]');

Interceptor.attach(hook.implementation, {
    onLeave: function(retval) { console.log("[*]
Class Name: " + className);
    console.log("\t[-] Type of return value: " +
typeof retval);
    console.log("\t[-] Original Return Value: " +
retval);
    newretval = ptr("0x0")
    retval.replace(newretval)
    console.log("\t[-] New Return Value: " +
newretval) } }); }
catch(err) { console.log("[!] Exception2: " +
err.message); } }

else {
 console.log("Objective-C Runtime is not
available!");
}
```

Level-up

- What if -[DCDevice isSupported] is not called?

```
*** entered +[DCDevice currentDevice]

Caller: 0x1047c0350 /private/var/containers/Bundle/Application/2405AEA1-60E4-4105-8855-28FA0CBDC23A/[redacted].app/[redacted]
[redacted]!-[RNDeviceCheck getDeviceToken]

retval: <DCDevice: 0x280d2b3c0>

*** exiting +[DCDevice currentDevice]

*** entered -[DCDevice generateTokenWithCompletionHandler:]

Caller: 0x1047c0394 /private/var/containers/Bundle/Application/2405AEA1-60E4-4105-8855-28FA0CBDC23A/[redacted]app/[redacted]
[redacted]!-[RNDeviceCheck getDeviceToken]

generateTokenWithCompletionHandler: <__NSStackBlock__: 0x16baa2538>

retval: <__NSStackBlock__: 0x16baa2538>

*** exiting -[DCDevice generateTokenWithCompletionHandler:]
```

Runtime Instrumentation

Patch the calling method…

```
-(void) getDeviceToken {
  if (@available(iOS 11.0, *)) {
    [DCDevice.currentDevice generateTokenWithCompletionHandler:^(NSData *  token, NSError * _Nullable error){
      NSLog(@"deviceToken: %@", token);
      NSString *tokenString = [token base64EncodedStringWithOptions: 0];
      _successCallback(@[tokenString]);
    }];
  } else {
    NSError *error = [NSError errorWithDomain:@"com.microsoft.[REDACTED]" code: 1 userInfo:@{NSLocalizedDescriptionKey:@"Please update IOS version to at least 11.0+"}];
    _errorCallback(error);
  }
}
```

Runtime Instrumentation

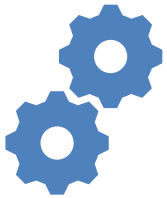
..to return null

```
const method = ObjC.classes.[REDACTED]DeviceCheck['- getDeviceToken'];
Interceptor.replace(method.implementation, new NativeCallback(function(sbai)
{
//do nothing
return 0x0;
}, 'void', ['pointer'])
);
----------------------------------------------------------------------
$frida -U -l .\RNDeviceCheck_getDeviceToken.js -f com.microsoft.REDACTED --no-
pause
```

App Crashes! Since the method's callback expects a token (Success) or error message (for errors)

Simple Solution#1 - Attack the API

Inspect the Network Traffic by installing a root CA on the device



If SSLPinning is implemented, those can be bypassed using runtime instrumentation [1]

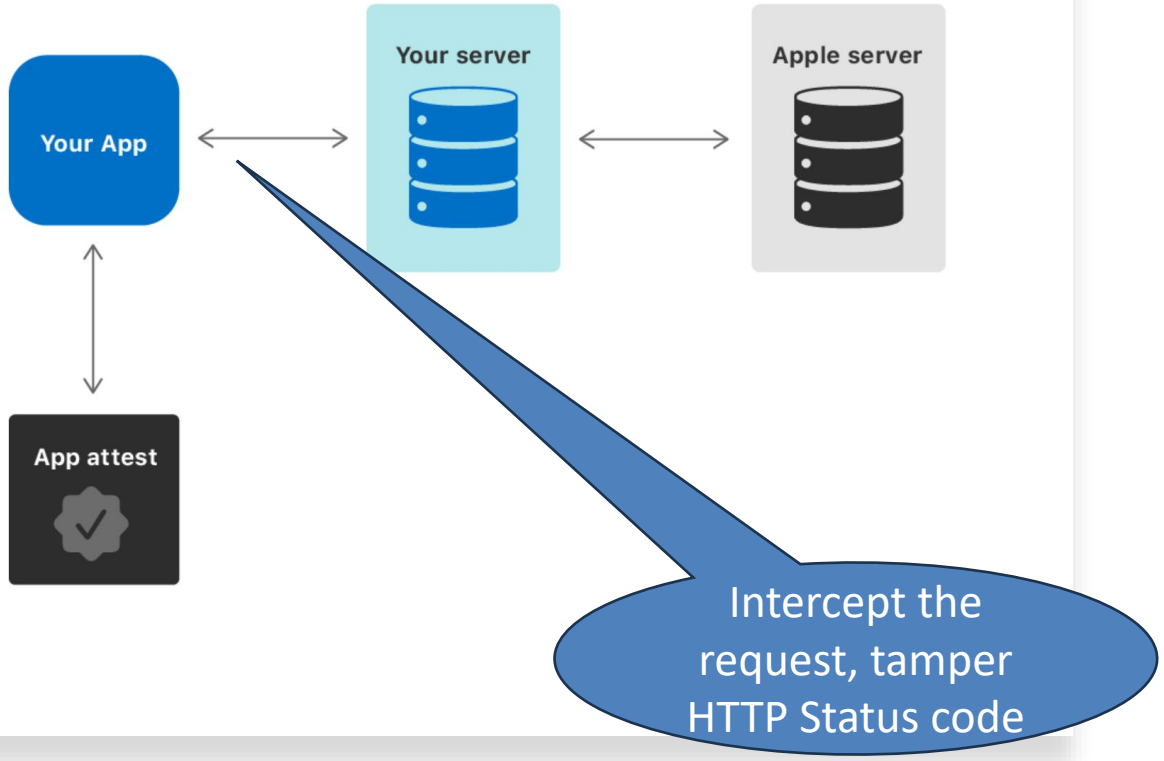
Zoom out!

For this target, the API endpoint did not distinguish between an Android and iOS client

[1]https://codeshare.frida.re/@federicodotta/ios13-pinning-bypass/

Simple(er) Solution #2 – Fool the Client

- Return a 200 OK by tampering with the HTTP Response.
- The iOS client isn't wiser.



Key Takeaways

Best Practices for developers

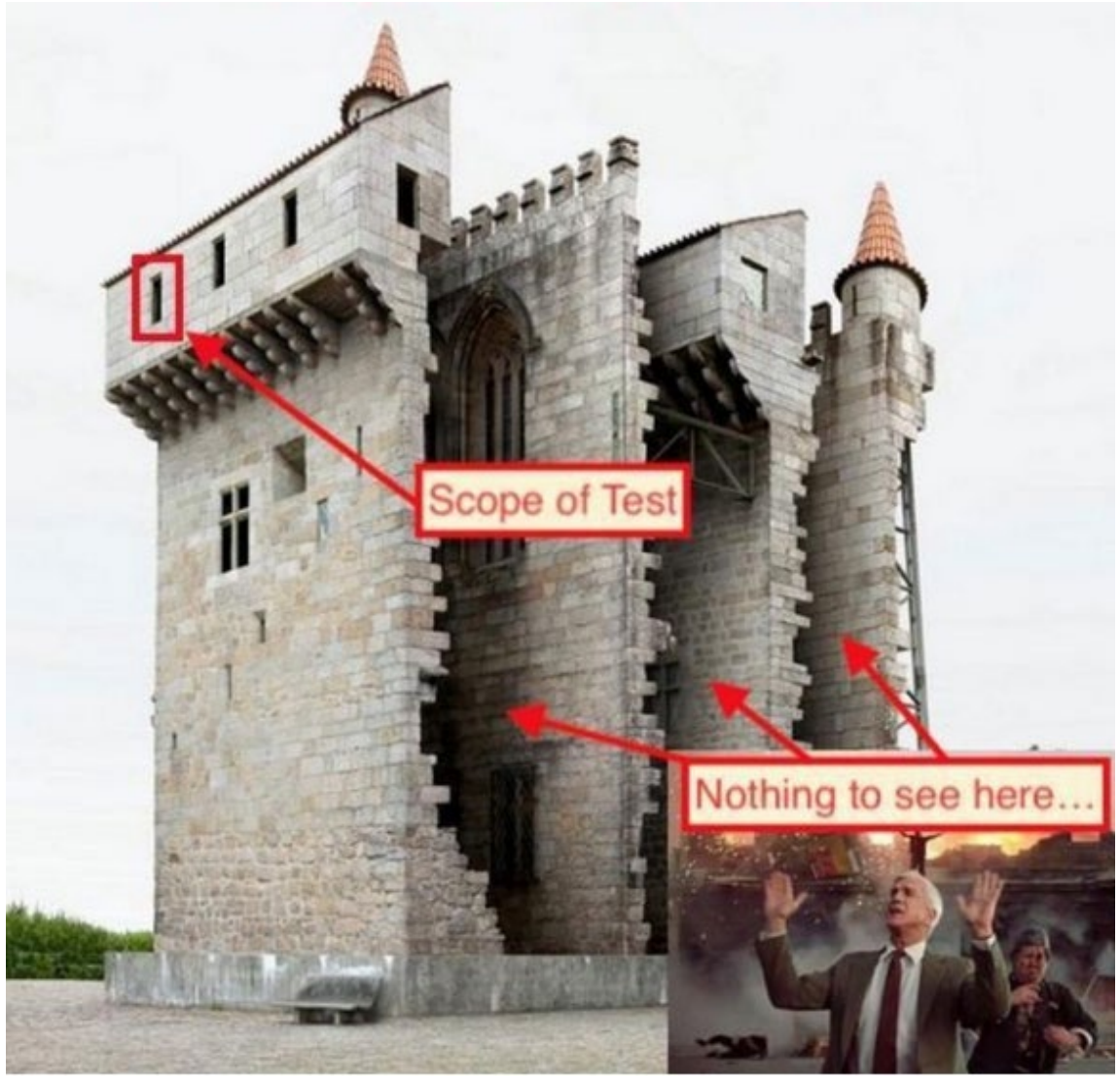
- Design solutions with Defense in depth
  - DeviceCheck Framework
  - Root detection in native code
  - SSL Pinning
  - Additional challenge-Response between the app and the app server
  - Prevent replay attacks

- Design secure fail-safe solutions
  - Your app server is down?
  - HTTP 503 – AppAttest service unavailable?

Key Takeaways

Best Practices for Red Teamers

- Think holistically about the overall solution.
- Attack different trust boundaries
- Cryptography is not attacked, it is bypassed.



Priyank Nigam - Throw your (App)Integrity out the window

Microsoft Security

Questions?

@Rev_Octo