When Security Gets in the Way
PenTesting Mobile Apps That Use Certificate Pinning

Justine Osborne  Alban Diquet
Outline

What is Certificate Pinning?
• Definition and Background
• Consequences for Mobile Blackbox Testing

iOS
• Certificate Pinning Within an iOS App
• Intercepting the App's Traffic: MobileSubstrate Extension

Android
• Certificate Pinning Within an Android App
• Intercepting the App's Traffic: Custom JDWP Debugger

Conclusion
Outline

What is Certificate Pinning?

• Definition and Background
• Consequences for Mobile Blackbox Testing

iOS

• Certificate Pinning Within an iOS App
• Intercepting the App's Traffic: MobileSubstrate Extension

Android

• Certificate Pinning Within an iOS App
• Intercepting the App's Traffic: Custom JDWP Debugger

Conclusion
Certificate Pinning and SSL

Hard-code in the client the SSL certificate known to be used by the server

- Pin the server's certificate itself
  - Takes the CA system out of the equation
- Pin the CA certificate used to sign the server's certificate
  - Limit trust to certificates signed by one CA or a small set of CAs

Significantly reduces the threat of a rogue CA and of CA compromise

- Implemented in Chrome 13 for Google services
- In Mobile Apps: Square, Twitter, Card.io...
Mobile Blackbox Testing

Intercepting the App's HTTPS traffic using a proxy
• Usually simple: Add the proxy's CA certificate to the device trust store
• This will not work if the App does certificate pinning

Beating certificate pinning as a penetration tester
• Change the certificate(s) or SSL validation methods within the App?
  • Re-package and side-load the new binary
• Use a debugger?

Introducing new tools to make this easy:
• iOS SSL Kill Switch
• Android SSL Bypass
Outline

What is Certificate Pinning?
• Definition and Background
• Consequences for Mobile Blackbox Testing

iOS
• Certificate Pinning Within an iOS App
• Intercepting the App's Traffic: MobileSubstrate Extension

Android
• Certificate Pinning Within an Android App
• Intercepting the App's Traffic: Custom JDWP Debugger

Conclusion
Several APIs to do network communication on iOS

- NSStream, CFStream, **NSURLConnection**

**Most iOS Apps use NSURLConnection**

- High level API to perform the loading of a URL request
- Verifies the server's certificate for *https*: URLs
- Developers can override certificate validation
  - To disable certificate validation (for testing only!)
  - To implement certificate pinning
NSURLConnection

NSURLConnection has the following constructor:

- \((id)\)initWithRequest:\((NSURLRequest *)request\)
  delegate:\((id <NSURLConnectionDelegate>)delegate\)

The delegate has to implement specific methods

- Those methods get called as the connection is progressing
- They define what happens during specific events
  - Connection succeeded, connection failed, etc...
- Two documented ways to do custom certificate validation
Connection Authentication

- `connection:willSendRequestForAuthenticationChallenge:`
- `connection:canAuthenticateAgainstProtectionSpace:`
- `connection:didCancelAuthenticationChallenge:`
- `connection:didReceiveAuthenticationChallenge:`
- `connectionShouldUseCredentialStorage:`

Connection Completion

- `connection:didFailWithError:`

MethodGroup

- `connection:willCacheResponse:`  \textit{required method}
- `connection:didReceiveResponse:`  \textit{required method}
- `connection:didReceiveData:`  \textit{required method}
- `connection:didSendBodyData:totalBytesWritten:totalBytesExpectedToWrite:`  \textit{required method}
- `connection:needNewBodyStream`
- `connection:willSendRequest:redirectResponse:`  \textit{required method}
- `connection:didFinishLoading:`  \textit{required method}
Connection Authentication

- connection:willSendRequestForAuthenticationChallenge: **Strategy 1**
- connection:canAuthenticateAgainstProtectionSpace:
- connection:didCancelAuthenticationChallenge:
- connection:didReceiveAuthenticationChallenge:
- connection:shouldUseCredentialStorage:

Connection Completion

- connection:didFailWithError:

MethodGroup

- connection:willCacheResponse: **required method**
- connection:didReceiveResponse: **required method**
- connection:didReceiveData: **required method**
- connection:didSendBodyData: totalBytesWritten: totalBytesExpectedToWrite: **required method**
- connection:needNewBodyStream
- connection:willSendRequest:redirectResponse: **required method**
- connection:didFinishLoading: **required method**
Connection Authentication
- `connection:willSendRequestForAuthenticationChallenge:`
- `connection:canAuthenticateAgainstProtectionSpace:`
- `connection:didCancelAuthenticationChallenge:`
- `connection:didReceiveAuthenticationChallenge:`
- `connection:shouldUseCredentialStorage:`

Strategy 1

Strategy 2 (deprecated)

Connection Completion
- `connection:didFailWithError:`

MethodGroup
- `connection:willCacheResponse:` required method
- `connection:didReceiveResponse:` required method
- `connection:didReceiveData:` required method
- `connection:didSendBodyData:totalBytesWritten:totalBytesExpectedToWrite:` required method
- `connection:needNewBodyStream`
- `connection:willSendRequest:redirectResponse:` required method
- `connection:didFinishLoading:` required method
Jailbroken iOS Development

MobileSubstrate

• Available on jailbroken devices
• “de facto framework that allows 3rd-party developers to provide run-time patches to system functions”
• MobileSubstrate patches are called “extensions” or “tweaks”
MobileSubstrate Extension

One example: WinterBoard

• Hooks into the SpringBoard APIs
• Allows users to customize their home screen
#import "HookedNSURLConnectionDelegate.h"

%hook NSURLConnection

// Hook into NSURLConnection's constructor
-(id)initWithRequest:(NSURLRequest *)request delegate:(id <NSURLConnectionDelegate>)delegate {
    // Create a delegate "proxy"
    HookedNSURLConnectionDelegate* delegateProxy;
    delegateProxy = [[HookedNSURLConnectionDelegate alloc] initWithOriginalDelegate: delegate];

    return %orig(request, delegateProxy); // Call the "original" constructor
}

%end
iOS SSL Kill Switch

Forwarding method calls to the original delegate

@implementation HookedNSURLConnectionDelegate : NSObject

...

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
{
    // Forward the call to the original delegate
    return [origiDelegate connection:connection didReceiveResponse:response];
}
iOS SSL Kill Switch

Intercepting calls to certificate validation methods

@implementation HookedNSURLConnectionDelegate : NSObject

...

- (void)connection:(NSURLConnection *)connection
  willSendRequestForAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge
{
  // Do not forward... Accept all certificates instead
  if([challenge.protectionSpace.authenticationMethod isEqualToString:NSURLAuthenticationMethodServerTrust])
  {
    NSURLConnectionCredential* serverCred;
    serverCred = [NSURLCredential credentialForTrust:challenge.protectionSpace.serverTrust];
    [challenge.sender useCredential:serverCred forAuthenticationChallenge:challenge];
  }
}

Outline

What is Certificate Pinning?
- Definition and Background
- Consequences for Mobile Blackbox Testing

iOS
- Certificate Pinning Within an iOS App
- Intercepting the App's Traffic: MobileSubstrate Extension

Android
- Certificate Pinning Within an Android App
- Intercepting the App's Traffic: Custom JDWP Debugger

Conclusion
Certificate Validation and Pinning on Android

- Device trust store cannot be modified by user until Android 4.0 (ICS)
- Certificate pinning can be implemented using an App specific trust store
- Common methods of certificate pinning outlined on Moxie’s blog:
  - [http://blog.thoughtcrime.org/authenticity-is-broken-in-ssl-but-your-app-ha](http://blog.thoughtcrime.org/authenticity-is-broken-in-ssl-but-your-app-ha)
Many possible ways to implement a bypass

- Decompile/Patch/Recompile/Resign/Sideload
- Custom VM/ROM with hooks built in
- Native code hooking (Mulliner) or native code debugger (gdb, vtrace)
- JDWP debugger
What is the Java Debug Wire Protocol (JDWP) ?

- Standard Java debugging protocol
- Programmatic debugging through Java APIs
  - Java Debug Interface (JDI)
- Python bindings available through AndBug
What can we do with a JDWP debugger?

- Normal debugging tasks: set breakpoints, step, etc...
- Once suspended we can:
  - Get the current thread, frame, frame object, local variables and arguments references
  - Load arbitrary classes, instantiate Objects, invoke methods, get and set local variables and arguments values
  - And more...
Certificate Pinning on Android

Two common ways to do SSL on Android
• javax.net.ssl.HttpsURLConnection
• org.apache.http.*

Certificate pinning
• Create SSLSocketFactory with custom TrustManager
javax.net.ssl.HttpsURLConnection
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

```java
HttpsURLConnection urlConn = (HttpsURLConnection)url.openConnection();
urlConn.setSSLSocketFactory(sslContext.getSocketFactory());
```
Certificate Pinning on Android

org.apache.http.*

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

```java
SSLContext sf = new SSLContext(pinningSSLContext);
Scheme httpsScheme = new Scheme("https", 443, sf);
SchemeRegistry schemeRegistry = new SchemeRegistry();
schemeRegistry.register(httpsScheme);
```
Bypass certificate pinning with JDWP debugger

- Break on certificate pinning implementation classes/methods
- On breakpoint use JDI APIs to perform SSL bypass
  - Directly manipulate objects, local variables, call methods, etc.
  - Force use of “trust all” TrustManager
Simple implementation for first version

- Plugin architecture, user plugins implement
  - `setupEvents()` – set breakpoints, method entry events, etc...
  - `handleEvents()` – handle events that were set
- `SSLBypassJDIPlugin` included with tool
- Future versions will explore more comprehensive solutions
Android SSL Bypass

DEMO
Outline

What is Certificate Pinning?
- Definition and Background
- Consequences for Mobile Blackbox Testing

iOS
- Certificate Pinning Within an iOS App
- Intercepting the App's Traffic: MobileSubstrate Extension

Android
- Certificate Pinning Within an Android App
- Intercepting the App's Traffic: Custom JDWP Debugger

Conclusion
Our Tools

iOS SSL Kill Switch
• Tested on iOS 4.3 and iOS 5.1
• https://github.com/iSECPartners/ios-ssl-kill-switch

Android SSL Bypass Tool
• Tested on Android 2.3.3 and 4.0.3
• https://github.com/iSECPartners/android-ssl-bypass

Comments / Ideas?
• justine@isecpartners.com
• alban@isecpartners.com
The End

QUESTIONS?
Certificate pinning on iOS


MobileSubstrate

- [http://iphonedevwiki.net/index.php/MobileSubstrate](http://iphonedevwiki.net/index.php/MobileSubstrate)

Certificate pinning on Android

- [http://blog.thoughtcrime.org/authenticity-is-broken-in-ssl-but-your-app-ha](http://blog.thoughtcrime.org/authenticity-is-broken-in-ssl-but-your-app-ha)

iSEC Partners on GitHub

- [https://github.com/iSECPartners](https://github.com/iSECPartners)