Avoiding damage, shame and regrets
data protection for mobile
client-server architectures
Real-world security is intuitive, evolution trained us for it
Meet Dodo birds!

Alice

Bob

data protection for client-server apps
They are chatting together

tweet hello

Alice

Bob
Here comes Eve..

..the eavesdropping Fennec Fox
Eve eavesdrops
tweet
ack
danger
ear radars: ON 😈
data protection for client-server apps
Birds fly away, Eve doesn’t hear them

secure place

hear nothing 😭

data protection for client-server apps
Real-world security

Risk (threat): Eve hears your secrets

Mitigation: physically move away from Eve
Cyber-world security

evolution did not prepare you for that!
Apple Secure Coding Guide

Every program is a potential target. Your customers’ property and your reputation are at stake.

What we protect?

User’s data!

- in memory
- in motion
- in storage

data protection for client-server apps
Data in motion
Problem: Layer 1

There are hackers..

and threats these hackers exploit..

to create damage
Meet Alice-the-App and Bob-the-Server

Alice-the-App

Bob-the-Server

data protection for client-server apps
Client and Server are communicating

Alice-the-App

passw: 123456

HTTP 1.1

Bob-the-Server

data protection for client-server apps
Eve-the-Hacker

data protection for client-server apps
Here Eve-the-Hacker comes!

passw: 123456
HTTP 1.1
Here Eve-the-Hacker comes!

data protection for client-server apps
Problem: Layer 2

Let’s go deeper..

To avoid threats we need secure programming
Alice decides to implement security
puts on paper hat!

data protection for client-server apps
Bob decides to implement security

builds the fence!

data protection for client-server apps

@vixentael
..and they decide to use HTTPS!

HTTPS out of the box
But it’s not really secure..

HTTPS out of the box

{"passw": "123456"}

data protection for client-server apps

@vixentael
Intercept traffic using proxy (1)

data protection for client-server apps

@vixentael
Intercept traffic using proxy (2)

any.do

login

e-mail

password

plain text

data protection for client-server apps
Intercept traffic using proxy (3)

- SSL experimenting with Android Top100 apps
- Intercepting the App Store’s Traffic on iOS
  - http://bit.ly/1H3xMrs

In the image:
- Bank app login
- Phone number
- IMEI

Data protection for client-server apps

@vixentael
What helps Eve to eavesdrop?

- plain HTTP
  - self-signed certificates
  - HTTPS with old cipher-suites
- using vulnerable libraries and bad examples from StackOverflow
  - SSL without SSL certificate pinning
  - not encrypting user data
Problem: Layer 3

As the result,

Programming is rarely secure
Software is buggy

AFNetworking SSL verification bug (v2.5.1-2.5.2)

Apple “goto fail” vulnerability
http://www.dwheeler.com/essays/apple-goto-fail.html

Out-of-the-box SSL is frequent subject to attacks
https://eprint.iacr.org/2013/049.pdf
http://noxxi.de/howto/ssl-debugging.html
Software is buggy. Why?

- Copying bad code from StackOverflow
- Debugging by tearing security suites apart
- Avoiding “complicated” security documentation
Cyber-world security

- is easy to f*ck up
- is inconvenient to implement
What shall we do?

- use good practice and brain
- use good tools
- minimize re-inventing the wheel
Realize threat vectors

Bad cryptography
No access control
Authentication bypass
Credential reuse
Session hijacking
Denial of Service
Data leakage

...
Implementing security tools yourself is a threat

Anyone can invent a security system that he himself cannot break

— Schneier's Law


data protection for client-server apps
Do not re-implement existing things

data protection for client-server apps
Use great tools

- scientific background
- trust big guys
- good track record

libsodium/NaCL
OTRKit
RNCryptor
MIHCrypto
Themis

https://github.com/mochtu//libsodium-ios
https://github.com/ChatSecure/OTRKit
https://github.com/RNCryptor/RNCryptor
https://github.com/hohl/MIHCrypto
https://github.com/cossacklabs/themis

data protection for client-server apps
@vixentael
Apple open sourced crypto

Apple published its corecrypto library, which implements MD2, MD4, MD5, DES, 3DES, RC2, RC4, etc. Welcome to 1998. developer.apple.com/cryptography/

A crypto library from a company that has repeatedly proven that they produce bad, insecure code. What could go wrong
Swift CommonCrypto wrapper

Supported Algorithms

The `Digest` class supports the following algorithms:

- MD2
- MD4
- MD5
- SHA1
- SHA224
- SHA256
- SHA384
- SHA512

old & boring

okay for now

https://github.com/iosdevzone/IDZSwiftCommonCrypto/
https://news.ycombinator.com/item?id=10733215

data protection for client-server apps
Armoring your SSL
Do your SSL/TLS right

SSL has a lot of problems  https://www.cossacklabs.com/avoid-ssl-for-your-next-app.html

To survive you need to:

- disable backward compatibility
- use long keys
- use strong ciphers (EC vs RSA)
- pin SSL certificate
- use cheat sheet  https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

data protection for client-server apps  @vixentael
Do you pin SSL certificate?

hey, mobile devs! do you implement SSL pinning for your apps? #iosdev

29% sure

71% not yet

14 votes • Final results
SSL/TLS in short

client asks certificate
server sends cert
client verifies cert
- domain,
- expiration date,
- asks CA if cert is valid and not revoked
key negotiation
encrypted data

data protection for client-server apps

@vixentael
Where can it break?

client asks certificate
server sends cert
key negotiation
encrypted data

client verifies cert
- domain,
- expiration date,
- asks CA if cert is valid and not revoked
SSL pinning

client verifies cert
- compares cert against pinned cert

client asks certificate
server sends cert

key negotiation

encrypted data

data protection for client-server apps
SSL pinning on iOS

```swift
- (void)connection:(NSURLConnection *)connection
    willSendRequestForAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge {

    SecTrustRef serverTrust = challenge.protectionSpace.serverTrust;
    id<NSURLAuthenticationChallengeSender> sender = challenge.sender;
    SecCertificateRef certificate = SecTrustGetCertificateAtIndex(serverTrust, 0);
    NSData * remoteCertificateData =
        CFBrandingRelease(SecCertificateCopyData(certificate));

    NSString * cerPath = [[NSBundle mainBundle] pathForResource:@"MyLocalCertificate"
ofType:@"cer"];    
    NSData * localCertData = [NSData dataWithContentsOfFile:cerPath];
    if ([remoteCertificateData isEqualToData:localCertData]) {
        NSURLCredential * credential = [NSURLCredential credentialForTrust:serverTrust];
        [sender useCredential:credential forAuthenticationChallenge:challenge];
    } else {
        [sender cancelAuthenticationChallenge:challenge];
    }
}
```

https://possiblemobile.com/2013/03/ssl-pinning-for-increased-app-security/
https://www.paypal-engineering.com/2015/10/14/key-pinning-in-mobile-applications/

data protection for client-server apps

@vixentael
SSL pinning more easy :)  

Swift lib for HTTPS and SSL pinning

```swift
let certData = NSData(contentsOfFile: 
NSBundle.mainBundle().pathForResource("lvwenhancom", ofType: "cer")!!
...

.addSSLPinning(LocalCertData: certData) { () -> Void in 
    print("Under Man-in-the-middle attack!")
}
```

https://github.com/johnlui/Pitaya
https://github.com/iSECPartners/ssl-conservatory

data protection for client-server apps

@vixentael
So, we’re done?

Nah.

SSL is not enough :(
Government MitM

Казахстан внедряет свой СА для прослушивания всего TLS-трафика

По словам Управляющего директора по инновациям АО «Казахтелеком» Нурланы Мейрманова, пользователям сети Интернет необходимо установить национальный сертификат безопасности, который будет доступен через Интернет-ресурсы АО «Казахтелеком». «Пользователю необходимо зайти на сайт www.telecom.kz и установить на своих устройствах выхода в сеть Интернет данный сертификат, следуя пошаговой инструкции по установке»

http://habrahabr.ru/post/272207/

data protection for client-server apps

@vixentael
Implementing Forward Secrecy
Forward Secrecy: Threat

Eve records encrypted traffic

New crypto vulnerability allows to extract keys

Eve physically extracts keys from one of the birds

Eve decrypts all encrypted traffic

data protection for client-server apps
Forward Secrecy: Mitigation

ephemeral keys + key rotation scheme

= Forward Secrecy

SSL/TLS has forward secrecy but it’s weak:
Using ephemeral key

- open session
- key negotiation (RSA or EC)
- create symmetric temp key
- use temp key to encrypt messages during session
- close session

Data protection for client-server apps
Implementing ephemeral keys

1. establish session
2. encrypt message with SecureSession before sending
3. decrypt message after receive
4. encrypt history with SecureCell

Themis has built-in forward secrecy inside SecureSession object

https://github.com/cossacklabs/themis

data protection for client-server apps
Implementing ephemeral keys

```swift
self.transport = [Transport new];
self.session = [[TSSession alloc] initWithUserId:[name dataUsingEncoding:NSUTF8StringEncoding] privateKey:privateKey
callbacks:self.transport];

NSData * wrappedData = [self.session wrapData:dataToSend error:&error];
NSData * unwrappedMessage = [self.session unwrapData:receivedData error:&error];

self.secureStorageEncryptor = [[TSCellSeal alloc] initWithKey:encryptionKey dataUsingEncoding:NSUTF8StringEncoding];
NSData * encryptedEvent = [self.secureStorageEncryptor wrapData:message dataUsingEncoding:NSUTF8StringEncoding]
context:nil
error:&error];
```

https://github.com/cossacklabs/mobile-websocket-example

data protection for client-server apps
Data in storage
iOS data protection

<table>
<thead>
<tr>
<th>Availability</th>
<th>File Data Protection</th>
<th>Keychain Data Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>When unlocked</td>
<td>NSFileProtectionComplete</td>
<td>kSecAttrAccessibleWhenUnlocked</td>
</tr>
<tr>
<td>While locked</td>
<td>NSFileProtectionCompleteUnlessOpen</td>
<td>N/A</td>
</tr>
<tr>
<td>After first unlock</td>
<td>NSFileProtectionCompleteUntilFirstUserAuthentication</td>
<td>kSecAttrAccessibleAfterFirstUnlock</td>
</tr>
<tr>
<td>Always</td>
<td>NSFileProtectionNone</td>
<td>kSecAttrAccessibleAlways</td>
</tr>
<tr>
<td>Passcode enabled</td>
<td>N/A</td>
<td>kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly</td>
</tr>
</tbody>
</table>
Storing in plain text is a bad idea.

Data protection for client-server apps.

Plain text chat history in Skype.
What we need to do

1. Choose good storage library with efficient crypto
2. Embed it on read/write
3. Store keys safely
Storage libraries

RNCryptor example

```swift
let data: NSData = ...
let password = "Secret password"
let ciphertext = RNCryptor.encryptData(data, password: password)
```

Themis SecureCell example

```swift
NSData * message = ...;
NSData * masterKey = ...;
TSCellSeal * cellSeal = [[TSCellSeal alloc] initWithKey:masterKey];
NSData * encryptedMessage = [cellSeal wrapData:message
    context:nil
    error:&error];
```
Storing the keys

**Valet example**

```
VALValet * myValet = [[VALValet alloc] initWithIdentifier:@"Druidia"
                        accessibility:VALAccessibilityWhenUnlocked];
[myValet setString:@"mysecretpassword" forKey:@"istorepasswords" ];
```

https://github.com/square/Valet

**SSKeychain example**

```
[SSKeychain setPassword:@"mysecretkey" forService:@"myappsecrets" account:@"alice" ];
```

https://github.com/soffes/sskeychain

Data protection for client-server apps

@vixentael
Storing the keys: Computable obfuscation

key = KDF(sqrt(42) * len(user_id) / parity(user_id))

compute key and use KDF to derive

```c
#import <CommonCrypto/CommonKeyDerivation.h>

NSData * myPassData = ...;
NSData * salt = ...;

int rounds = CCCalibratePBKDF(kCCPBKDF2, myPassData.length,
                             salt.length, kCCPRFHmacAlgSHA256, 32, 100);

unsigned char key[32];
CCKeyDerivationPBKDF(kCCPBKDF2, myPassData.bytes, myPassData.length,
                      salt.bytes, salt.length, kCCPRFHmacAlgSHA256, rounds, key, 32);
```


data protection for client-server apps
Ending notes
Practical app security step by step

1. Use HTTPS with good TLS settings
2. Enable SSL pinning
3. Encrypt user data in motion with ephemeral keys
4. Encrypt stored data and protect the key
Alice is more secure now

SSL pinning

ephemeral keys

encrypted storage data

data protection for client-server apps

@vixentael
Bob is more secure now

data protection for client-server apps
Chatting is more secure

HTTPS
SSL pinning
ephemeral keys

5720b3c2 fe674f54
73e10ad4 ...

@vixentael
Security is full of adventures and discoveries.
And fun.

and shiny metal birds!
The last slide

@vixentael

iOS developer at stanfy.com
[creating awesome mobile and IoT apps]

take care!

data protection for client-server apps

@vixentael
More to read

★ The Mobile Application Hacker's Handbook
https://books.google.com.ua/books?id=UgVhBgAAQBAJ

★ Designing Secure User Interfaces

★ CryptoCat iOS app security audit
https://nabla-c0d3.github.io/documents/iSEC_Cryptocat_iOS.pdf

★ Storing secret keys
More to watch

★ All talks of Moxie Marlinspike

https://www.youtube.com/watch?v=ibF36Yyeehw
https://www.youtube.com/watch?v=8N4sb-SEpcg
https://www.youtube.com/watch?v=tOMiAeRwpPA