Security and Your Apps

Session 706

Ivan Krstic Security and Privacy Strategy
Pierre-Olivier Martel Sandbox Engineering Manager
Andrew Whalley Core OS Security Engineering
Device Security

iOS: Secure enclave, touch ID, secure boot chain, data protection
OS X: FileVault, app sandbox, system integrity protection, ...
Keychain, MDM
Network Security

HTTPS, TLS, “lock icon”

In 2015, TLS is a minimum baseline
TLS Is Not Enough

Many servers still default to TLSv1.0, from 1999

Newest version is TLSv1.2 from 2008, with a number of cryptographic improvements to the protocol
TLSv1.2 Is Not Enough

Compromise of a server key lets you decrypt all TLS traffic that was encrypted in the past.

With forward secrecy, a server key compromise only lets an attacker decrypt future traffic.

- Mitigates bulk recording of encrypted network data

TLS supports forward secrecy through specific cipher suites.
“People have entrusted us with their most personal information. We owe them nothing less than the best protections that we can possibly provide by harnessing the technology at our disposal. We must get this right. History has shown us that sacrificing our right to privacy can have dire consequences.”

–Tim Cook, February 2015
App Transport Security

By default, apps linked against iOS 9 and OS X 10.11 cannot make unprotected HTTP connections.

TLS connections require compliance with best practices:
- TLSv1.2 with forward secrecy, no known-insecure cryptographic primitives (RC4 encryption, SHA-1 certificate signatures), and key size requirements (2048 bits for RSA, 256 bits for EC).

Exceptions can be declared in your Info.plist on a case-by-case basis, or as a complete override if necessary.
System Integrity Protection

Pierre-Olivier Martel
Sandbox Engineering Manager
Defense In Depth
Defense In Depth

Security is about layers
Defense In Depth

Security is about layers
One layer failing shouldn’t defeat all security
Defense In Depth

Security is about layers
One layer failing shouldn’t defeat all security
Defense In Depth

Security is about layers
One layer failing shouldn’t defeat all security
Rely on multiple layers of protection with different properties
• Delay the advance of an attacker
• Reduce the attack surface
Defense In Depth
The origins

Sebastien de Vauban (1633–1707)
Military Expert for the King of France
Defense In Depth

The origins
Defense In Depth

The OS X model
Defense In Depth
The OS X model

- Keychain
- POSIX
- Sandbox
- Developer ID and Gatekeeper
Defense In Depth

The OS X model

- Keychain
- POSIX
- Sandbox
- Developer ID and Gatekeeper
Defense In Depth

The OS X model

- Keychain
- POSIX
- Sandbox
- Developer ID and Gatekeeper
Defense In Depth
The OS X model

- Keychain
- POSIX
- Sandbox
- Developer ID and Gatekeeper
Defense In Depth
The OS X model

- Keychain
- POSIX
- Sandbox
- Developer ID and Gatekeeper
Defense In Depth
The OS X model

- Keychain
- POSIX
- Sandbox
- Developer ID and Gatekeeper
OS X Security Model

The power of root
OS X Security Model

The power of root

Most Macs are single-user systems, where the user has administrative privileges by default.
OS X Security Model

The power of root

Most Macs are single-user systems, where the user has administrative privileges by default.

Root hidden behind a single—often weak—password.
OS X Security Model
The power of root

Most Macs are single-user systems, where the user has administrative privileges by default.
Root hidden behind a single—often weak—password.
OS X Security Model

The power of root

Most Macs are single-user systems, where the user has administrative privileges by default

Root hidden behind a single—often weak—password
OS X Security Model

The power of root

Most Macs are single-user systems, where the user has administrative privileges by default.

Root hidden behind a single—often weak—password.

Root can disable all security measures on the device.
OS X Security Model

The power of root

Most Macs are single-user systems, where the user has administrative privileges by default.

Root hidden behind a single—often weak—password.

Root can disable all security measures on the device.

Any piece of malware is one password or vulnerability away from taking full control of the device.
OS X Security Model

The missing layer

Limit the power of root

Protect the system by default, on disk and at runtime

Provide a configuration mechanism that can’t be automatically compromised by root
System Integrity Protection

Security policy applying to every process, including privileged code running unsandboxed

Extends additional protections to system components on disk and at runtime

System binaries can only be modified by Apple Installer and Software Update, and no longer permit runtime attachment or code injection
System Integrity Protection

Developer impact

No impact on Mac AppStore applications

Potential impact for non-AppStore applications

• Modifying system binaries or framework
• Installing content in system locations
• Inspecting memory state of system processes
• Injecting libraries into system processes
Key Aspects

Filesystem protections
Runtime protections
Kernel extensions
Configuration mechanism
Platform Policy

Filesystem protections

Installer marks system locations with special flag

Kernel stops processes from

• Writing to protected files or directories
• Writing to block devices backing protected content
• Mounting over protected content

Only applies to boot and root volumes
## Platform Policy

### Filesystem protections

<table>
<thead>
<tr>
<th>System only</th>
<th>Available to developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>/System</td>
<td>(~)/Library</td>
</tr>
<tr>
<td>/bin</td>
<td>/usr/local</td>
</tr>
<tr>
<td>/usr</td>
<td></td>
</tr>
<tr>
<td>/sbin</td>
<td></td>
</tr>
<tr>
<td>/Applications</td>
<td></td>
</tr>
</tbody>
</table>

System Migration will move 3rd-party content out of system locations after upgrading.
Platform Policy

Runtime protections

Injecting code into a process is equivalent to modifying the binary on disk.

Processes are marked restricted by the kernel:

- Main executable is protected on disk.
- Main executable is signed with Apple-private entitlements.
Platform Policy

Restricted processes
Platform Policy

Restricted processes

task_for_pid() / processor_set_tasks() fail with EPERM
Platform Policy

Restricted processes

task_for_pid() / processor_set_tasks() fail with EPERM
Mach special ports are reset on exec(2)
Platform Policy

Restricted processes

task_for_pid() / processor_set_tasks() fail with EPERM
Mach special ports are reset on exec(2)
dyld environment variables are ignored
Platform Policy
Restricted processes

task_for_pid() / processor_set_tasks() fail with EPERM
Mach special ports are reset on exec(2)
dyld environment variables are ignored
dtrace probes unavailable
Platform Policy
Restricted processes

task_for_pid() / processor_set_tasks() fail with EPERM
Mach special ports are reset on exec(2)
dyld environment variables are ignored
dtrace probes unavailable

$> sudo lldb -n Finder
(lldb) process attach --name "Finder"
error: attach failed: attach failed: lost connection
Platform Policy

Restricted processes

task_for_pid() / processor_set_tasks() fail with EPERM
Mach special ports are reset on exec(2)
dyld environment variables are ignored
dtrace probes unavailable

$> sudo lldb -n Finder
(lldb) process attach --name "Finder"
error: attach failed: attach failed: lost connection
Platform Policy

Kext signing

Extensions have to be signed with a Developer ID for Kexts certificate.

Install into /Library/Extensions

`kext-dev-mode boot`-arg is now obsolete.
Platform Policy

Kext signing

Extensions have to be signed with a Developer ID for Kexts certificate
Install into /Library/Extensions

kext-dev-mode boot-arg is now obsolete

$> sudo nvram boot-args='kext-dev-mode=1'
Platform Policy

Kext signing

Extensions have to be signed with a Developer ID for Kexts certificate
Install into /Library/Extensions

kext-dev-mode boot-arg is now obsolete

$> sudo nvram boot-args='kext-dev-mode=1'
Configuration Mechanism

Disable System Integrity Protection (subject to change)

• Boot to Recovery OS (Command+R on boot)
• Launch “Security Configuration” from the “Utilities” menu
• Change configuration and apply

Configuration is stored in NVRAM

• Applies to the entire machine
• Persists across OS install
Summary
Summary

New security policy applying to every process
Summary

New security policy applying to every process
Protect the system by default, on disk and at runtime
Summary

New security policy applying to every process

Protect the system by default, on disk and at runtime

- Restrict write access to system location
Summary

New security policy applying to every process

Protect the system by default, on disk and at runtime

- Restrict write access to system location
- Prevent runtime attachment and code injection into system binaries
Summary

New security policy applying to every process

Protect the system by default, on disk and at runtime
• Restrict write access to system location
• Prevent runtime attachment and code injection into system binaries

3rd-party content must be migrated out of system locations
Summary

New security policy applying to every process

Protect the system by default, on disk and at runtime

- Restrict write access to system location
- Prevent runtime attachment and code injection into system binaries

**3rd-party content must be migrated out of system locations**

Configuration mechanism in the Recovery OS
The Keychain and Touch ID

Andrew R. Whalley
Core OS Security Engineering
Protecting Data
Protecting Data
Protecting Data
Protecting Data
Keychain
Keychain

A very specialized database
Keychain

A very specialized database

Efficiently searched by attributes
Keychain

A very specialized database
Efficiently searched by attributes
Optimized for small payloads
The Keychain in a Nutshell

Item creation in Swift
The Keychain in a Nutshell

Item creation in Swift

let secret = "top secret"
let secretData = secret.dataUsingEncoding(NSUTF8StringEncoding)!
The Keychain in a Nutshell

Item creation in Swift

```swift
let secret = "top secret"
let secretData = secret.dataUsingEncoding(NSUTF8StringEncoding)!

let status = SecItemAdd(attributes, nil)
```

```swift
let status = SecItemAdd(attributes, nil)
```
let secret = "top secret"
let secretData = secret.dataUsingEncoding(NSUTF8StringEncoding)!

let attributes = [
    kSecClass as String       :  kSecClassGenericPassword as String,
    kSecAttrService as String : "myservice",
    kSecAttrAccount as String : "account name here",
    kSecValueData as String   : secretData!
]

let status = SecItemAdd(attributes, nil)
The Keychain in a Nutshell

Other SecItem calls
The Keychain in a Nutshell

Other SecItem calls

```swift
let status = SecItemCopyMatching(query, &data)
let status = SecItemDelete(query)
let status = SecItemUpdate(query, attributes)
```
The Keychain in a Nutshell
Some considerations
The Keychain in a Nutshell

Some considerations

Factor keychain code into a simple, testable unit
The Keychain in a Nutshell

Some considerations

Factor keychain code into a simple, testable unit

• Wrapper class
The Keychain in a Nutshell

Some considerations

Factor keychain code into a simple, testable unit

• Wrapper class

Use the highest data protection level you can
The Keychain in a Nutshell

Some considerations

Factor keychain code into a simple, testable unit
• Wrapper class

Use the highest data protection level you can
• kSecAttrAccessibleWhenUnlocked default and best
The Keychain in a Nutshell

Some considerations

Factor keychain code into a simple, testable unit

• Wrapper class

Use the highest data protection level you can

• kSecAttrAccessibleWhenUnlocked default and best
• kSecAttrAccessibleAfterFirstUnlock for background apps
The Keychain in a Nutshell

Some considerations

Factor keychain code into a simple, testable unit

• Wrapper class

Use the highest data protection level you can

• kSecAttrAccessibleWhenUnlocked default and best
• kSecAttrAccessibleAfterFirstUnlock for background apps
• kSecAttrAccessibleAlways will be deprecated in iOS 9
The Keychain in a Nutshell

Some considerations

Factor keychain code into a simple, testable unit

- Wrapper class

Use the highest data protection level you can

- `kSecAttrAccessibleWhenUnlocked` default and best
- `kSecAttrAccessibleAfterFirstUnlock` for background apps
Apple Watch
Reducing Password Prompts
Reducing Password Prompts
Reducing Password Prompts
Shared web credentials
Safari Saved Passwords
Websites and Apps
Shared Web Credentials
Save to Safari
let username = "j.appleseed@icloud.com"
let password = SecCreateSharedWebCredentialPassword().takeRetainedValue()

SecAddSharedWebCredential("www.macosforge.org", username, password){ error in
    // handle error
}
Username:
j.appleseed@icloud.com

Password:

Confirm Password:

Create Account
Automatically fill out web forms using your contact info, previous names and passwords, or credit card info.

- Use Contact Info
- My Info
- Names and Passwords
- Credit Cards
- Saved Credit Cards
Login

Email: j.appleseed@icloud.com
Password: ************

Login
SecRequestSharedWebCredential("www.macosforge.org", .None)
{
    credentials, error in
    if CFArrayGetCount(credentials) > 0 {
        let dict = unsafeBitCast(CFArrayGetValueAtIndex(credentials, 0),
                                    CFDictionaryRef.self) as Dictionary
        let username = dict[kSecAttrAccount as String]
        let password = dict[kSecSharedPassword as String]
        login(username, password)
    }
}
SecRequestSharedWebCredential("www.macosforge.org", .None) 
{
    credentials, error in
    if CFArrayGetCount(credentials) > 0 {
        let dict = unsafeBitCast(CFArrayGetValueAtIndex(credentials, 0),
            CFDictionaryRef.self) as Dictionary
        let username = dict[kSecAttrAccount as String]  
        let password = dict[kSecSharedPassword as String] 
        login(username, password)
    }
}
Associated Domains

App entitlement
Associated Domains

Server JSON

https://example.com/apple-app-site-association

```json
{
    "webcredentials": {
        "apps": [
            "YWBN8XTPBJ.com.example.app",
            "YWBN8XTPBJ.com.example.app-dev"
        ]
    }
}
```
Associated Domains

Server JSON

https://example.com/apple-app-site-association

{
  "webcredentials": {
    "apps": ["YWBN8XTPBJ.com.example.app", "YWBN8XTPBJ.com.example.app-dev"]
  },
  "activitycontinuation": {
    "apps": ["YWBN8XTPBJ.com.example.app"]
  }
}
Associated Domains
Server JSON

https://example.com/apple-app-site-association

"activitycontinuation":
{
  "apps" : [
    "YWBN8XTPBJ.com.example.app"
  ]
},

Associated Domains

Server JSON

https://example.com/apple-app-site-association

"applinks":
{
   "apps" : [],
   "details" :
   {
      "YWBN8XTPBJ.com.example.app": [
         "/example/content/*"
      ]
   }
}
Associated Domains

Server JSON

https://example.com/apple-app-site-association
Associated Domains

Server JSON

https://example.com/apple-app-site-association

For iOS 9: No need to sign JSON
Avoiding Asking for Passwords

iCloud Keychain
Avoiding Asking for Passwords

iCloud Keychain
iCloud Keychain
iCloud Keychain

For all passwords that can be used on multiple devices

- Add `kSecAttrSynchronizable` to all SecItem calls
iCloud Keychain

For all passwords that can be used on multiple devices
• Add `kSecAttrSynchronizable` to all SecItem calls

A few caveats
• Updating or deleting items affects item on all devices
• See SecItem.h
Keychain
Keychain

Store all secrets in the keychain
Keychain

Store all secrets in the keychain

Protect them at the highest level possible
Keychain

Store all secrets in the keychain

Protect them at the highest level possible

Use SharedWebCredentials and iCloud Keychain
Device Specific Credentials
Device Specific Credentials

Examples

• Limited use tokens and cookies
• Encrypted messaging keys
• Keys with specific protection requirements
Device Specific Credentials

Examples

- Limited use tokens and cookies
- Encrypted messaging keys
- Keys with specific protection requirements

kSecAttrAccessibleWhenUnlockedThisDeviceOnly
Device Specific Credentials

Examples

• Limited use tokens and cookies
• Encrypted messaging keys
• Keys with specific protection requirements

kSecAttrAccessibleWhenUnlockedThisDeviceOnly
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
Device Specific Credentials

Examples

• Limited use tokens and cookies
• Encrypted messaging keys
• Keys with specific protection requirements

kSecAttrAccessibleWhenUnlockedThisDeviceOnly
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
kSecAttrAccessControl
Architecture
iOS security domains

User Space
- Application
- Security Framework

Kernel
- Process Separation
- KeyStore
Architecture
iOS security domains

User Space
- Application
- Security Framework

Kernel
- Process Separation
- KeyStore

Secure Enclave
- Touch ID
Architecture
iOS security domains

User Space
- Application
- Security Framework

Kernel
- Process Separation
- KeyStore

Secure Enclave
- Touch ID
Architecture
iOS security domains

User Space
- Application
- Security Framework

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Touch ID

Convenience

- Don’t need to enter your passcode all the time
Touch ID

Convenience
• Don’t need to enter your passcode all the time

Security
• Stronger passcode
• Lock immediately
Touch ID

APIs

- LocalAuthentication
- Keychain Access Control Lists
if PresentedFingerprintMatchesEnrolled() then
    DoSomething()
Local Authentication

- User Space
  - Application

- Kernel
  - Process Separation

- Secure Enclave
  - Touch ID
Local Authentication

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
Local Authentication

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID (✓)
LocalAuthentication

User Space

- Application

Kernel

- Process Separation

Secure Enclave

- Touch ID
Local Authentication

User Space
- Application
- DoSomething()

Kernel
- Process Separation

Secure Enclave
- Touch ID
LocalAuthentication

User Space
- Application
- DoSomething()

Kernel
- Process Separation

Secure Enclave
- Touch ID
LocalAuthentication

Use cases
LocalAuthentication

Use cases

Replace existing security barrier
LocalAuthentication

Use cases

Replace existing security barrier

Adding one when it would have been too inconvenient before
LocalAuthentication

Use cases

Replace existing security barrier
Adding one when it would have been too inconvenient before

Examples
LocalAuthentication

Use cases

Replace existing security barrier
Adding one when it would have been too inconvenient before

Examples
• Viewing especially sensitive data
LocalAuthentication

Use cases

Replace existing security barrier

Adding one when it would have been too inconvenient before

Examples

• Viewing especially sensitive data
• Confirming an operation
LocalAuthentication
Prompt at app startup
LocalAuthentication
Prompt at app startup
let context = LAContext()
context.touchIDAuthenticationAllowableReuseDuration = 30
let reasonString = "Authentication is needed for access."

context.evaluatePolicy(.DeviceOwnerAuthenticationWithBiometrics,
    localizedReason: reasonString) { success, authenticationError in
    if success {
        showMainUI()
    }
}
Touch ID Enrollment Change
Touch ID Enrollment Change
let context = LAContext()
do {
    try context.canEvaluatePolicy(.DeviceOwnerAuthenticationWithBiometrics)
    if let domainState = context.evaluatedPolicyDomainState
        where domainState == lastState {
        // Enrollment state the same
    }
    else {
        // Enrollment state changed
    }
}
catch { // Handle error }
let context = LAContext()
do {
  try context.canEvaluatePolicy(.DeviceOwnerAuthenticationWithBiometrics)
  if let domainState = context.evaluat edPolicyDomainState
    where domainState == lastState {
      // Enrollment state the same
    }
  else {
    // Enrollment state changed
  }
} catch { // Handle error}
LocalAuthentication

Recap of what’s new in iOS 9
LocalAuthentication
Recap of what’s new in iOS 9

touchIDA
• Accept a previous match
evaluatedPolicy
• Get a representation of the current set of enrolled fingers
invalidate()
• Cancel a user prompt from code
evaluateAccessControl()
• Use LocalAuthentication with Access Control Lists
Keychain
Access Control Lists
Keychain
Access Control Lists

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keychain

Access Control Lists

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keychain
Access Control Lists

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keychain
Access Control Lists

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keychain
Access Control Lists

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keychain
Access Control Lists

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Secret
Keychain
Access Control Lists

User Space
- Application
- Secret

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keychain

Access Control Lists
Keychain
Access Control Lists

Add additional protection to a saved credential
Keychain
Access Control Lists

Add additional protection to a saved credential
Take advantage of the Secure Enclave
Keychain
Access Control Lists

Add additional protection to a saved credential
Take advantage of the Secure Enclave
Examples
Keychain
Access Control Lists

Add additional protection to a saved credential
Take advantage of the Secure Enclave

Examples
• Don’t require a username and password every launch
Keychain
Access Control Lists

Add additional protection to a saved credential
Take advantage of the Secure Enclave

Examples
• Don’t require a username and password every launch
• Protect local encryption keys
let acl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
    kSecAttrAccessibleWhenUnlocked, .UserPresence, &error).takeRetainedValue()
let acl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
    kSecAttrAccessibleWhenUnlocked, .UserPresence, &error).takeRetainedValue()
Keychain Item Access Control Lists

let secret = "top secret"
let secretData = secret.dataUsingEncoding(NSUTF8StringEncoding)
var error: Unmanaged<CFErrorRef>?
let acl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
    kSecAttrAccessibleWhenUnlocked, .UserPresence, &error).takeRetainedValue()
ACL Authentication Types

.UserPresence
ACL Authentication Types

.UserPresence
.DevicePasscode
ACL Authentication Types

- .UserPresence
- .DevicePasscode
- .TouchIDAny
ACL Authentication Types

- UserPresence
- DevicePasscode
- TouchIDAny
- TouchIDCurrentSet
Touch ID and Multi Factor Authentication
Touch ID and Multi Factor Authentication

Something you know
Touch ID and Multi Factor Authentication

Something you know

• Password
Touch ID and Multi Factor Authentication

Something you know

• Password

Something you have
Touch ID and Multi Factor Authentication

Something you know
- Password

Something you have
- Physical token, smartcard
Touch ID and Multi Factor Authentication

Something you know
• Password

Something you have
• Physical token, smartcard
• iOS Device with Secure Enclave and Touch ID
Touch ID and Multi Factor Authentication

Something you know
• Password

Something you have
• Physical token, smartcard
• iOS Device with Secure Enclave and Touch ID

SecAccessControlCreateFlags.TouchIDCurrentSet
Access Control List Authentication Types

Beyond Touch ID

.UserPresence
.DevicePasscode
.TouchIDAny
.TouchIDCurrentSet
.ApplicationPassword
.PrivateKeyUsage
ApplicationPassword

kSecAttrAccessibleWhenUnlocked

5458bdf1cfd4cb6e662fe02d87620b69c01802ed8c7fa0b0843b6245db5ba0fa64cc1fd26085b78620239b75e27163e4a6a88bd8a0463525a343dad1d59e784462fb9bf70a4bdf8b8d517e8a3369e29dfc881a00415c3b7213927f013b60d092c4ce434a2a7af95f78fd106095ea7e43580799872de834b1162de3813da2bc031b07fa993f0338d539981fc502cb
ApplicationPassword

kSecAttrAccessibleWhenUnlocked

5458bdf1cfd4cb6e662fe02d87
620b69c01802ed8c7fa0b0843
b6245dbf5ba0fa64cc1fd26085
b78620239b75e27163e4a6a88
bd8a0463525a343dad1d59e78
4462fbf9bf70a4bdf8b8d517e8
a3369e29df881a00415c3b7213
927f013b60d092c4ce434a2a7af
95f78fd106095ea7e435807998
72de834b1162de3813da2bc031
b07fa993f0338d539981fc502cb
ApplicationPassword

Passcode

kSecAttrAccessibleWhenUnlocked

5458bdf1cfd4cb6e662fe02d87620b69c01802edeb8c7fa0b0843b6245dbf5ba0fa64cc1fd26085b78620239b75e27163e4a6a88bd8a0463525a343dad1d59e784462fbf9bf70a4bdf8b8d517e8a3369e29dfc881a00415c3b7213927f013b60d092c4ce434a2a7af95f78fd106095ea7e43580799872de834b1162de3813da2bc031b07fa993f0338d539981fc502cb
Application Password

Passcode → AES Key

```plaintext
kSecAttrAccessibleWhenUnlocked

5458bdf1cfd4cb6e662fe02d87
620b69c01802ed8c7fa0b0843
b6245dbf5ba0fa64cc1fd26085
b78620239b75e27163e4a6a88
bd8a0463525a343dad1d59e78
4462fbf9bf70a4bdf8b8d57e8
a3369e29dfc881a00415c3b7213
927f013b60d092c4ce434a2a7af
95f78fd106095ea7e435807998
72de834b1162de3813da2bc031
b07fa993f0338d539981fc502cb
```
The secret meeting location is row 13 of Mission right after session 706.
kSecAttrAccessibleWhenUnlocked

ApplicationPassword

b1a0d6c9b3b2726a886f0f103a
b89154ee3fb9e85ea27c78bcd
246c6262fb29ba85ab6988b7b
7758d8ae0cd89306ce2421eb33
0f900aff526a9a06fcdf040cc7c6
e5668744d792a69f9640d05a5
1d7e3e7185ae741c099257305
b882d52e7a218c8b31a51a0634
58e5b80023a7eb3e35da77bee
232d82fbb734f04ba93951de2b
8f848cd1a5c96b793f739b0d29
ApplicationPassword

Passcode → AES Key

kSecAttrAccessibleWhenUnlocked

.ApplicationPassword

b1a0d6c9b3b2726a886f0f103ab89154ee3fbd9e85ea27c78bcd246c6262fb29ba85ab6988b7b7758d8aecd89306ce2421eb330f900aff526a9a06fcdff040cc7c6ec5668744d792a69f9640d05a51d7e3e7185aee741c099257305b882d52e7a218c8b31a51a063458e5b80023a7ebee35da77bee232d82fbb734f04ba93951de2b8f848cd1a5c96b793f739b0d29
ApplicationPassword

Passcode → AES Key → kSecAttrAccessibleWhenUnlocked

ApplicationPassword

a5d3c7df546db329ed9418b7f3c5120ff5572aa4e5dc691dc06cf29b156a61e1cf1ad89c4c5e2fa58bb149b83677fe627c688d6125c0256ab7a22d130af74c6062b9155c865ffa5f58708bb498b2bd4e930ecd4c2e0a213218a987456739a3bc7f5044b7967da4618d04556d769cffe249d0cec26645bee92d14c7d614a217ead1d38509673350e13c1293a8864eefa
ApplicationPassword

Passcode → AES Key

Password

kSecAttrAccessibleWhenUnlocked

ApplicationPassword

a5d3c7df546db329ed9418b7fc5120ff5572aa4e5d691dc06cf29b156a61e1cf1ad89c4c5e2fa58bb149b83677fe6277c688d6125c0256ab7a22d130af74c6062b9155c865ffa5f58708bb498b2bd4e930ecd4c2e0a213218a987456739a3bc7f5044b7967da4618d04556d769ccf9ce249d0cc626645bee9214c7d614a217eac1d38509673350e13c1293a8864eefa
ApplicationPassword

Passcode ➔ AES Key

Password ➔ AES Key

kSecAttrAccessibleWhenUnlocked

.ApplicationPassword

- a5d3c7df546db329ed9418b7f3c5120ff5572aa4e5dc691dc06cf29b156a61e1cf1ad89c4c5e2fa58bb149b83677fe627c688d6125c0256ab7a22d130af74c6062b9155c865ffa5f58708bb498b2bd4e930ecd4c2e0a213218a987456739a3bc7f5044b7967da4618d04556d769cffce249d0cec26645bee92d14c7d614a217eac1d38509673350e13c1293a8864eefa
The secret meeting location is row 13 of Mission right after session 706.
ApplicationPassword

Use cases
ApplicationPassword

Use cases

Server side control of local data protection
ApplicationPassword

Use cases

Server side control of local data protection
Key storage on accessories
let secret = "top secret"
let secretData = secret.dataUsingEncoding(NSUTF8StringEncoding)!

let acl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
    kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly,
    .ApplicationPassword, &error).takeRetainedValue()

let context = LAContext()
context.setCredential(password.dataUsingEncoding(NSUTF8StringEncoding),
    type: .ApplicationPassword)
ApplicationPassword

Example

```swift
let secret = "top secret"

let secretData = secret.dataUsingEncoding(NSUTF8StringEncoding)!

var error: Unmanaged<CFErrorRef>?

let acl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
    kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly,
    .ApplicationPassword, &error).takeRetainedValue()
```

Example
let secret = "top secret"
let secretData = secret.dataUsingEncoding(NSUTF8StringEncoding)!

var error: Unmanaged<CFErrorRef>?
let acl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
    kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly,
    .ApplicationPassword, &error).takeRetainedValue()

let context = LAContext()
let password = "e693b64e405e9ddc578959b97665e750"
context.setCredential(password.dataUsingEncoding(NSUTF8StringEncoding),
    type: .ApplicationPassword)
let attributes = [
    kSecClass as String: kSecClassGenericPassword as String,
    kSecAttrService as String : "myservice",
    kSecAttrAccount as String : "account name here",
    kSecValueData as String : secretData,
    kSecAttrAccessControl as String : acl,
    kSecUseAuthenticationContext as String : context
]

let status = SecItemAdd(attributes, nil)
Keeping Things Inside the Secure Enclave
Keeping Things Inside the Secure Enclave

- User Space
  - Application
  - Secret

- Kernel
  - Process Separation

- Secure Enclave
  - Touch ID
  - KeyStore
Keeping Things Inside the Secure Enclave

Asymmetric cryptography
Keeping Things Inside the Secure Enclave

Asymmetric cryptography

Asymmetric Key Pair
Keeping Things Inside the Secure Enclave

Asymmetric cryptography

Asymmetric Key Pair

Public Key

Private Key
Keeping More Inside the Secure Enclave

User Space
- Application
  - `SecKeyGeneratePair()`

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keeping More Inside the Secure Enclave
Keeping More Inside the Secure Enclave

User Space

- Application
- Public Key

Kernel

- Process Separation

Secure Enclave

- Touch ID
- Private Key
Keeping More Inside the Secure Enclave

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keeping More Inside the Secure Enclave

User Space
- Application
  - SecItemCopyMatching()

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keeping More Inside the Secure Enclave

User Space
- Application
  - `SecItemCopyMatching()`

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Private Key
Keeping More Inside the Secure Enclave

User Space
- Application
  - `SecItemCopyMatching()`

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Private Key
Keeping More Inside the Secure Enclave

User Space
- Application

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Private Key

SecItemCopyMatching()
Keeping More Inside the Secure Enclave

- User Space
  - Application
- Kernel
  - Process Separation
- Secure Enclave
  - Touch ID
  - KeyStore
Keeping More Inside the Secure Enclave

User Space
- Application
  - SecKeyRawSign()

Kernel
- Process Separation

Secure Enclave
- Touch ID
- KeyStore
Keeping More Inside the Secure Enclave

User Space

- Application
  - SecKeyRawSign()

Kernel

- Process Separation

Secure Enclave

- Touch ID
- Data To Sign
Keeping More Inside the Secure Enclave

User Space
- Application
- SecKeyRawSign()

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Data To Sign
Keeping More Inside the Secure Enclave

User Space
- Application
  - SecKeyRawSign()

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Data To Sign
Keeping More Inside the Secure Enclave

User Space
- Application
- SecKeyRawSign()

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Data To Sign

Data To Sign
Keeping More Inside the Secure Enclave

User Space
- Application
  - SecKeyRawSign()

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Private Key
Keeping More Inside the Secure Enclave

User Space
- Application
- Signature

Kernel
- Process Separation

Secure Enclave
- Touch ID
- Private Key
Strengthening Touch ID as a Second Factor

Example flow—enrollment
Strengthening Touch ID as a Second Factor

Example flow—enrollment

Generate keypair
Strengthening Touch ID as a Second Factor

Example flow—enrollment

Generate keypair

Send public key to server
Strengthening Touch ID as a Second Factor

Example flow—enrollment

Generate keypair
Send public key to server
Server records public key
Strengthening Touch ID as a Second Factor

Example flow—verification
Strengthening Touch ID as a Second Factor

Example flow—verification

Server sends a challenge
Strengthening Touch ID as a Second Factor

Example flow—verification

Server sends a challenge

App calls `SecKeyRawSign()`
Strengthening Touch ID as a Second Factor

Example flow—verification

Server sends a challenge
App calls `SecKeyRawSign()`
User presents finger
Strengthening Touch ID as a Second Factor

Example flow—verification

Server sends a challenge
App calls `SecKeyRawSign()`
User presents finger
App sends signed data back to server
Strengthening Touch ID as a Second Factor

Example flow—verification

Server sends a challenge
App calls SecKeyRawSign()
User presents finger
App sends signed data back to server
Server verifies signature against stored public key
Asymmetric Keys in the Secure Enclave

Generated private keys are

- EC P256
- Not extractable

Operations

- SecKeyRawSign()
- SecKeyRawVerify()
Summary
Summary

Overview of the keychain
Summary

Overview of the keychain
Avoiding password prompts
Summary

Overview of the keychain
Avoiding password prompts
Touch ID APIs
Summary

Overview of the keychain
Avoiding password prompts
Touch ID APIs
  • LocalAuthentication
Summary

Overview of the keychain
Avoiding password prompts
Touch ID APIs
  • LocalAuthentication
  • Keychain ACLs
Summary

Overview of the keychain
Avoiding password prompts
Touch ID APIs
  • LocalAuthentication
  • Keychain ACLs
Advanced features
Summary

Overview of the keychain

Avoiding password prompts

Touch ID APIs
  • LocalAuthentication
  • Keychain ACLs

Advanced features
  • App passwords
Summary

Overview of the keychain
Avoiding password prompts
Touch ID APIs
  • LocalAuthentication
  • Keychain ACLs
Advanced features
  • App passwords
  • Secure Enclave protected private keys
More Information

Technical Support
Apple Developer Forums
http://developer.apple.com forums

Keychain Services Documentation

Shared Web Credentials Reference
More Information

Documentation
iOS Security White Paper

iOS Security White Paper
http://developer.apple.com/support/technical

General Inquiries
Paul Danbold, Core OS Evangelist
danbold@apple.com
## Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Privacy and your App</td>
<td>Pacific Heights</td>
<td>Tuesday 2:30PM</td>
</tr>
<tr>
<td>Networking with NSURLSession</td>
<td>Pacific Heights</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Related Labs</td>
<td>Frameworks Lab C</td>
<td>Wednesday 9:00AM</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Security and Privacy Lab</td>
<td>Frameworks Lab B</td>
<td>Thursday 9:00AM</td>
</tr>
</tbody>
</table>