A Practical Guide to the App Sandbox

Session 710
Ivan Krstić
Core OS Honey Badger

These are confidential sessions—please refrain from streaming, blogging, or taking pictures
Modern Car Safety

• Mandatory standardized crash testing performed by the government
• Traction control, blind spot warnings, lane-drift alerts
• But, damage containment
• When all else fails, there are seat belts and airbags
Traditional Desktop Security

• Defender must protect everything at all times, attacker must breach one protection at any time
• Emphasis on damage prevention (ASLR, NX, antivirus), not containment
• One thing goes wrong, game over
• No seatbelt and airbag for the computer
The Unfortunate Assumption

• All programs should execute with the full privileges of the executing user
  ▪ Or, security is a barrier between different users, not different programs
• But most modern computer devices are single-user systems
• Not every app should have access to the most sensitive data
  ▪ Apps should only have access to the resources they need
An Unfortunate Example

- The unfortunate assumption does not work
- Compromising any app must not grant access to all user data
Security UI Does Not Work

• Security dialogs are mysterious and opaque; riddles wrapped inside enigmas
• Clicking “Permit” or “Allow” maximizes the likelihood of getting work done
• “If you’re explaining, you’re losing”
• Pavlovian conditioning to ignore security
Landscape Changes

• Many apps, many developers
• Computers are always on a network
• Easier than ever to find and run new software
• Security challenge: Isolate data between programs
Software Reality

• Complex systems will always have vulnerabilities
  ▪ Complexity is never decreasing
• Single buffer overflow can ruin your user’s day
• Frameworks and libraries you don’t control
  ▪ Every WebView instance: Millions of lines of code and a full-featured JavaScript engine
• No limit on exploit damage
App Sandbox
App Sandbox

• Introduced in OS X Lion
• More secure applications
• Drive security policy by user intent
• Contain exploit damage
• Reduce ability for a compromised or misbehaving application to steal, corrupt, or destroy user data
Key Concepts

• Developer expresses what an app is supposed to be able to do
• Each app runs in its own container
• User controls access to documents
  • Special cases (e.g., recent items, drag and drop) work automatically
Key Components
Key Components

- Entitlements
- Containers
- PowerBox
- XPC Services
Entitlements

• What apps can do is determined by the developer-specified entitlements in the code signature
• Just a property list, editable in Xcode
• Simple, easy to understand
Entitlements

• User-selected files, Downloads folder
• Personal information
  ▪ Address book, calendars, location
• Assets: Music, movies, pictures
• Network client, server
• Devices
  ▪ Camera, microphone, printing, USB, FireWire, Bluetooth, serial
• Application groups and scripting/automation targets
Key Components

- Entitlements
- Containers
- Powerbox
- XPC Services
Key Components

- Entitlements
- Containers
- Powerbox
- XPC Services
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

open("/Users/krstic/Library/foo")
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

open("/Users/krstic/Library/foo")
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

NSHomeDirectory()
HOME=~/.Library/Containers/App/
CFFIXED_USER_HOME=~/.Library/Containers/App/

NSHomeDirectory()
HOME=~/Library/Containers/App/
CFFIXED_USER_HOME=~/Library/Containers/App/

NSHomeDirectory()

"/Users/krstic/Library/Containers/App"
Key Components

- Entitlements
- Containers
- Powerbox
- XPC Services
Key Components

- Entitlements
- Containers
- Powerbox
- XPC Services
Powerbox

• Cocoa NSOpenPanel/NSSavePanel
• Trusted mediator process
• Clear declaration of user intent
  ▪ Drives security policy
  ▪ Sandboxed apps cannot synthesize user input events
NSOpenPanel

~/Documents

Powerbox  AppKit
Key Components

- Entitlements
- Containers
- Powerbox
- XPC Services
Key Components

- Entitlements
- Containers
- Powerbox
- XPC Services
XPC Services

• Very easy app and framework privilege separation
• Services have their own entitlements
• No fork/exec, process lifecycle managed by XPC
• Only available to the containing app
Putting It All Together

TextEdit
TextEdit

Process

• Prepare entitlements
• Code sign program
• Run and verify App Sandbox status
• Look for violations
Demo
TextEdit

Exploitation

• The attacker only has access to documents that the user opened during this TextEdit run
• No ability to access or modify other apps or documents
• Need multiple vulnerabilities for a successful exploit
Advanced App Sandbox
Security-Scoped Bookmarks
Security-Scoped Bookmarks

• Preserve access to user-chosen files and folders across system reboot
• Per-user app configuration—Input and output folders, commonly accessed files
• Document formats that contain references to files
Security-Scoped Bookmarks

• App scope
  com.apple.security.files.user-selected.read-{write,only}
  ▪ Locked to the app and user that created them
Security-Scoped Bookmarks

• App scope

com.apple.security.files.user-selected.read-{write,only}

• Locked to the app and user that created them
User Picks File
Security-Scoped Bookmarks

• Document scope

com.apple.security.files.bookmarks.document-scope

- Allows a document format to contain references to files (but not folders) that travel with it
- Bookmark must be stored in the document file/bundle itself
- Cannot point to system or hidden locations (~/Library)
User Creates Doc

My App
User Creates Doc

User Inserts Movie

My App
My App

NSData

bookmarkDataWithOptions
User Opens Doc

NSData

My App
Security-Scoped Bookmarks

• No new API, just a flag on existing NSURL methods
  + NSURLByResolvingBookmarkData:options:relativeToURL:bookmarkDataIsStale:error:
  – bookmarkDataWithOptions:includingResourceValuesForKeys:relativeToURL:error:

• Big difference—Resolution returns a security-scoped NSURL
  • Must call {start, stop}AccessingSecurityScopedResource to gain and discontinue access to resource
Application Groups
Application Groups

com.apple.security.application-groups

- Each group name must begin with Apple-assigned Team ID
- Useful for suites of different apps, or a single app and its helper(s)
- Direct IPC permitted: XPC, POSIX
- Each group is assigned a shared file system location
SMLoginItemSetEnabled()}
Related Items
Related Items

- Access to files/folders with same name, but different file extension
  - Movie player opening a subtitle file for a movie
  - TextEdit upgrading a .rtf document to a .rtfd for attachments
- 
- Requires a declaration of allowed patterns in the app’s Info.plist
Automation
Automation

• Rich history of automation on OS X
• App Sandbox does not impose restrictions on how your apps can be scripted
• But your apps were very limited in how they can script other apps
  • Scripting Terminal, Finder or Safari can be complete sandbox escapes
Apple Event Access Groups

• Access groups define groups of scriptable operations
  ▪ Commands, classes, properties
  ▪ Part of the application’s scripting interface (sdef)
  ▪ man 5 sdef

• Already in OS X applications
  ▪ Mail: com.apple.mail.compose
  ▪ iTunes: com.apple.iTunes.playback, com.apple.iTunes.library.read,
    com.apple.iTunes.library.read-write
Using an Access Group

- `com.apple.security.scripting-targets`
- Value is a dictionary
  - Keys are application code signing identifiers
  - Values are access group identifiers
Using an Access Group

Compose Mail message

<key>com.apple.security.scripting-targets</key>
<dict>
  <key>com.apple.mail</key>
  <array>
    <string>com.apple.mail.compose</string>
  </array>
</dict>
Using an Access Group
Compose Mail message

<key>com.apple.security.scripting-targets</key>
<dict>
  <key>com.apple.mail</key>
  <array>
    <string>com.apple.mail.compose</string>
  </array>
</dict>
Using an Access Group

Compose Mail message

<key>com.apple.security.scripting-targets</key>
<dict>
  <key>com.apple.mail</key>
  <array>
    <string>com.apple.mail.compose</string>
  </array>
</dict>
Using an Access Group
Compose Mail message

<key>com.apple.security.scripting-targets</key>
<dict>
  <key>com.apple.mail</key>
  <array>
    <string>com.apple.mail.compose</string>
  </array>
</dict>
Application-Run User Scripts

• Application Script Menu
• Event Handlers
  • Mail Rule
  • Aperture Import Action
  • Messages Events
• Scripts executed by the application
• Inherit application’s permissions
NSUserScriptTask

~/Library

Application Scripts

com.devID.appName
com.devID.appName
com.devID.appName
com.devID.appName
NSUserScriptTask
Running attached user scripts

• Part of Foundation.framework
• NSUserScriptTask for generic scripts
  • Supports AppleScript, Automator, and UNIX scripts
• Subclasses for specific control
  • NSUserAppleScriptTask, NSUserAutomatorTask, NSUserUnixTask
• Script runs outside the sandbox
• No entitlement required
iTunes Library Access
iTunes Library Framework

New in iTunes 11

- Access to iTunes Library media and artwork regardless of disk location
- Objective-C API instead of the XML database
- Requires com.apple.security.music.read-\{write,only\} entitlement
- Returns security-scoped NSURLs
  - {start,stop}AccessingSecurityScopedResource
App Sandbox and the Mac App Store
Mac App Store

• Technical Q&A QA1773
• All binaries must be sandboxed, including XPC services and other helper tools
• Entitlements must match app functionality
  ▪ If you don’t need it, don’t request it
  ▪ Don’t request entitlements that silence sandbox violations which have no functional impact
Mac App Store

• Understand the entitlements you’re requesting
  ▪ USB access **not required** for the user to choose files on USB media
  ▪ Incoming connections (Server) **not needed** for most network applications

• Temporary exception requests must not effectively disable the sandbox
  ▪ Scripting Finder or Terminal
  ▪ Filesystem access to /
Summary
App Sandbox

- Strong barrier against exploitation and coding errors
- Drives policy by user intent
- Complementary to Gatekeeper
- See the App Sandbox Design Guide
- Sample code available
Summary

• iOS—50 billion sandboxed apps downloaded with confidence

• Delight users with carefree apps on OS X
Related Sessions

| Efficient Design with XPC | Russian Hill  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tuesday 2:00PM</td>
</tr>
<tr>
<td>Lab</td>
<td>Location</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>OS X Sandbox Lab</td>
<td>Core OS Lab Wednesday 3:15PM</td>
</tr>
<tr>
<td>Security Lab</td>
<td>Core OS Lab Thursday 2:00PM</td>
</tr>
<tr>
<td>App Store Lab</td>
<td>Third Floor Daily 9:00AM</td>
</tr>
</tbody>
</table>
apple WWDC2013