

# Introducing App Sandbox

Magical and Revolutionary Desktop Security

Session 203

**Ivan Krstić**

Core OS Security Samurai

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

# A Brief History Lesson

# Yesterday



- Took 30 years to reach 100K viruses in 2004
- In 2008, known malware count surpassed a million
- Today, tens of thousands of unique malware samples a day
- Mac users mostly spared

# Data Theft

- In 2005, practically no known data-theft malware spread by email
- By 2007, started seeing over 5000 samples in the wild
- Cumulative annual growth rate of 620%

# No One Is Keeping Attacks at Bay

- CSIS Commission on Cybersecurity for the 44th Presidency
- 2007: DoD, State, DHS, Commerce, and NASA all suffered “major intrusions” by “unknown foreign entities”
- Unclassified email of secretary of state compromised

# No One Is Keeping Attacks at Bay

- State lost terabytes of information
- DHS: Break-ins in several divisions, including TSA
- NASA: Email restrictions before launches, launcher designs compromised

# What Is Going On?

- People do not need a physics degree to drive a car safely
- Yet people seem to need a CS degree to safely use a computer

# Modern Car Safety



- Mandatory standardized crash testing performed by the government
- Redundant sensors and computers
- Damage containment
- When all else fails, there are seat belts and airbags



# Modern Computer Security

GAME OVER

- Defender must protect everything at all times; attacker must breach one protection at any time
- Emphasis on damage prevention (ASLR, NX, anti-virus), not containment
- Where is the seat belt for your computer?
- One thing goes wrong, game over

# The Unfortunate Assumption

- All programs should execute with the full privileges of the executing user, or...
- Security should exist between different users, but not different programs

# Tracing the Family Tree

The Unfortunate Assumption is 40 years old

Ritchie, Thompson: The program is the user

TCP/IP WAN

1971

1978

1983

1991

International packet-switched network

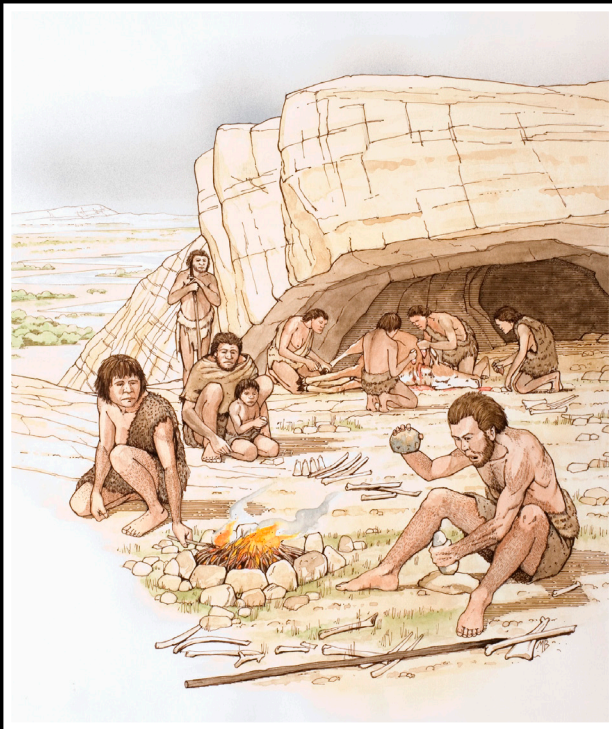
The web

# Untrusted Code?



- 1971: No conceivable way for code to appear on a machine
- Physically put all code there via tape or punched card
- Today: Untrusted code on every visited website

# Sticks and Stones

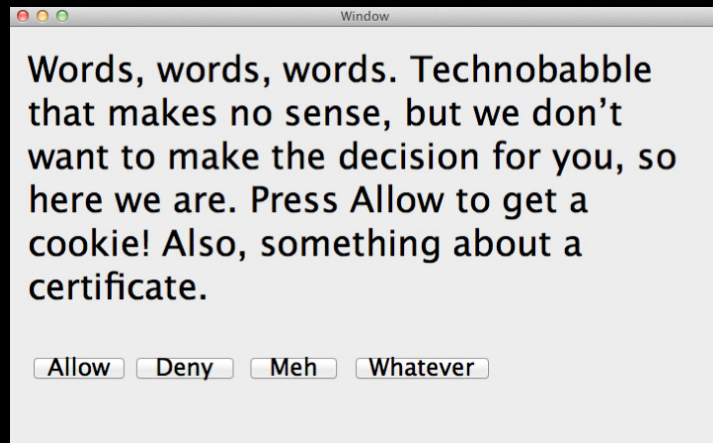


- The Unfortunate Assumption predates personal computing
- Need damage containment, regardless of cause
- Not just for malicious attacks
  - Unintentional coding errors
  - Misbehavior

# Making a Better Wheel

- In the last 20 years, we learned what does not work
- The Unfortunate Assumption does not work
- Neither does security UI
- “If you’re explaining, you’re losing”
- If you have to show words to the user, there is no security

# What Users See



- Security dialogs are a black box; clicking Permit or Allow maximizes the likelihood of getting work done
- Pavlovian conditioning to ignore security

# Security UI



- When is the last time your airbag asked you for permission to deploy?
- “We’d like to detonate the nitroguanidine charge. Accept or Deny?”
- “The identity of the rhombohedral sodium azide could not be verified. Retry or Abort?”



# What Does Work?



- Principle of Least Privilege
- PDP-11/70 had no segmented memory. a.out!
- Eventually: Kernel/user separation, later user/process separation within userland
- Desktop OSes caught up with Mac OS X and Windows NT

# What Does Work?

- x86 CPU rings, protected memory, user separation, process separation
- Each iteration reduced privilege
- But the reduction stopped at processes

# Processes and Privileges

- A user's programs today run with that user's full privileges
- No way to deliver fine-grained privileges to parts of a process (sub-PID)
- One part of a program needs privileges, all of the program has it
- Certainly true in C—arbitrary pointers

# A Tale of Modern Times

# Today



- The Internet brought many apps, many vendors
- Trivial to download apps
- Computers are always on a network
- Security challenge: Isolate data between programs

# Mac OS X Challenge

- Filesystem-centric user experience
- Apps have always run with full user privileges
- Developers can not express intended app behavior
- OS can't construct a last line of defense

# Status Quo



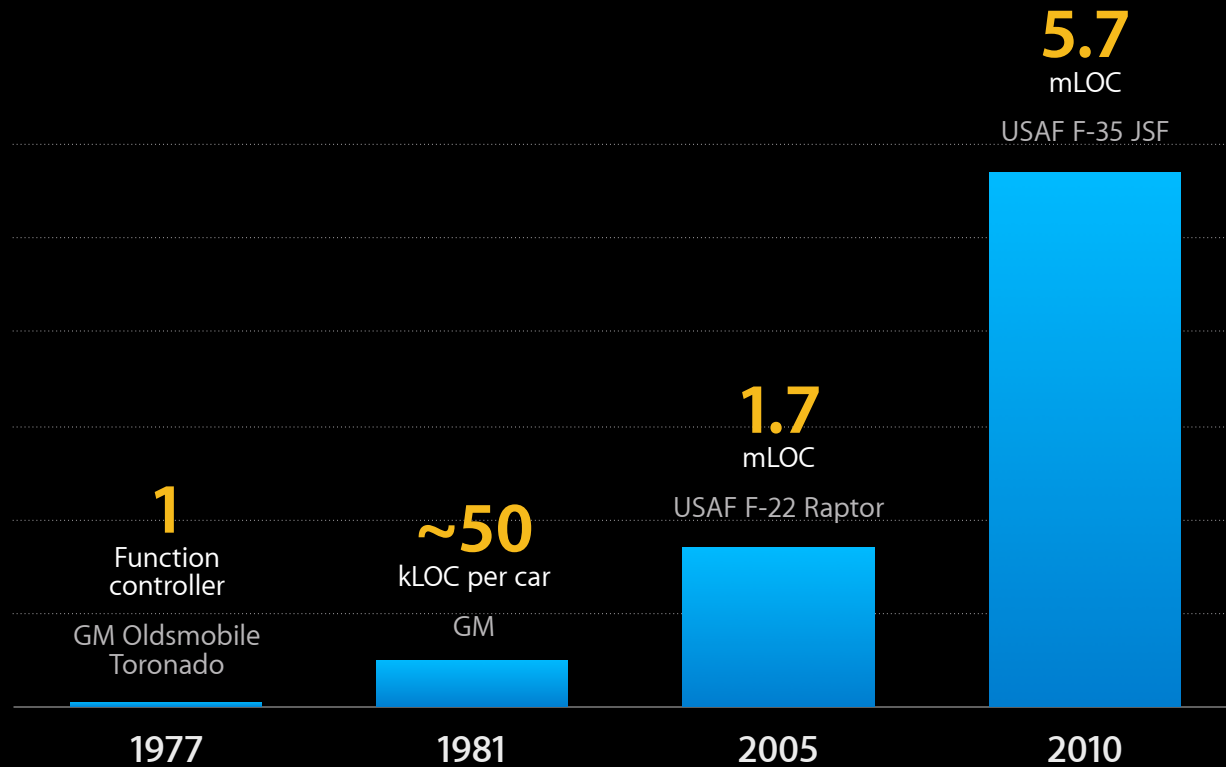
- WatchGrassGrow.app can read all your email and send it to Croatia
- Did I mention I'm Croatian?

# Software Reality

- Complex systems will always have vulnerabilities
  - Complexity is never decreasing
- Single buffer overflow can ruin your user's day
  - Not just in your code, but in all your frameworks and libraries
- No limit on exploit damage



# Complexity: An Aside



# Complexity

## The Punchline



- Modern cars: 30-100 ECUs, ~100 mLOC total

# The Situation Is Untenable

- String of high-profile breaches and compromises recently across a number of companies
- Personal information exposed, financial and identity fraud
- Theft of company property, danger to national security
- Users are hurting
  - Time
  - Money
  - Sense of comfort and enjoyment of technology

# A Better Model



# iOS Sandbox



- Apps cannot touch other apps
- Apps cannot touch the system
- Limits damage of exploits, mistakes
- Trivial app uninstall
- Key iOS security element

# Sandbox

## Implementation

- Almost fully low-level SPI
- Gates filesystem access, network access, signals, Mach and IOKit lookups, etc.
- Kernel access control mechanism based on MAC framework
- Strong daemon adoption in Mac OS X

# Sandbox and Desktop GUIs

- Requires static knowledge of required resources
- Enforces restrictions—does not make it easier to separate privilege
- Inappropriate for desktop apps

# App Sandbox



# App Sandbox

- Secure GUI apps for Mac OS X
- Support Mac App Store
- Limit exploit exposure
- Control filesystem, network access
- Can not steal, corrupt, or delete user data
- Sandbox for enforcement, deep changes throughout the OS

# App Sandbox

## Design Goals

- Drive security policy by user intent
- Damage containment when all else has failed
- Make it easy for developers to prevent confused deputies and better separate privilege
- No perfect security, but significantly elevates the bar

# Mac App Store Restrictions

| Prohibition                | Policy | App Sandbox |
|----------------------------|--------|-------------|
| Filesystem sprawl          | x      | x           |
| Installing other apps      | x      | x           |
| Root privileges            | x      | x           |
| Kernel extensions          | x      | x           |
| Undocumented functionality | x      | x           |

# Key Ideas

- Developer expresses what an app is supposed to be able to do
- Each app runs in its own container
  - Bound to code identity
- User controls access to documents
  - Access does not persist across application relaunch
  - Special cases (recent items, drag and drop) work automatically

# Key Components

1. Entitlements
2. Containers
3. Kernel Enforcement
4. Powerbox
5. 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
- Very different than Android permission model
  - Less than 15 total entitlements in Lion

# Entitlements

- Filesystem
  - User-selected files, Downloads folder
- Network client, server
- Devices
  - Camera, microphone, printing, USB bus
- Personal information
  - Address book, calendars, location
- Assets
  - Music, movies, pictures

# Key Components

1. Entitlements
2. Containers
3. Kernel Enforcement
4. Powerbox
5. XPC Services



```
HOME=~/.Library/Containers/App/  
CFFIXED_USER_HOME=~/.Library/Containers/App/
```



# Key Components

1. Entitlements
2. Containers
3. Kernel Enforcement
4. Powerbox
5. XPC Services

# Kernel Enforcement

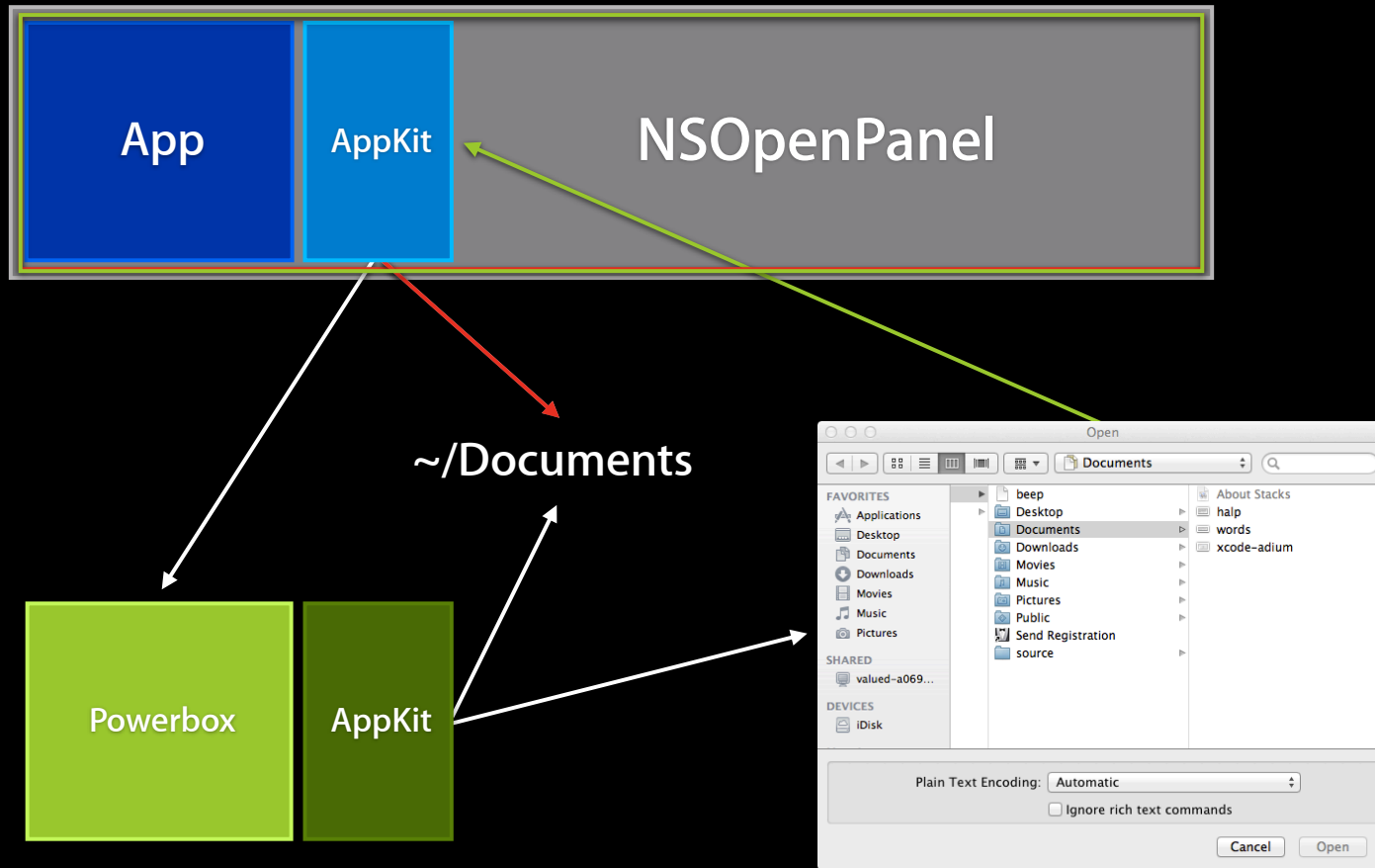
- Same mechanism as iOS
- Only container and certain system locations accessibly by default
- No direct access to the user home directory

# Key Components

1. Entitlements
2. Containers
3. Kernel Enforcement
4. Powerbox
5. XPC Services

# Powerbox

- Cocoa NSOpenPanel/NSSavePanel
- Clear declaration of user intent
- Should drive security policy
- Trusted mediator called Powerbox



# Key Components

1. Entitlements
2. Containers
3. Kernel Enforcement
4. Powerbox
5. XPC Services

# XPC Services

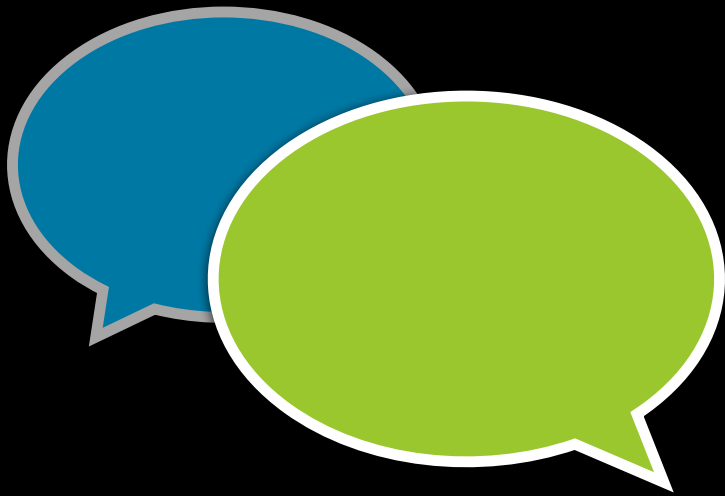
- Extremely easy app and framework privilege separation
- Services have their own entitlements
- No fork/exec—process lifecycle managed by XPC
- Only accessible to their main app



# Putting It All Together

Adium

# Adium



- Popular open source IM client
- Full-featured
- 250 source files, 75,000 lines of code

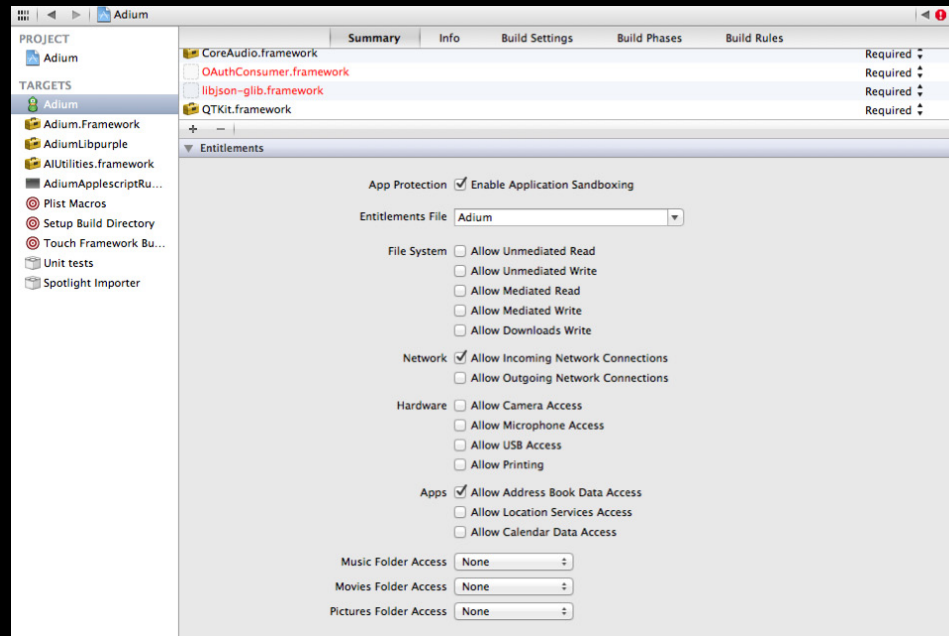
# Adium: Process

- Prepare entitlements
- Code sign program
- Run and verify App Sandbox status
- Look for violations

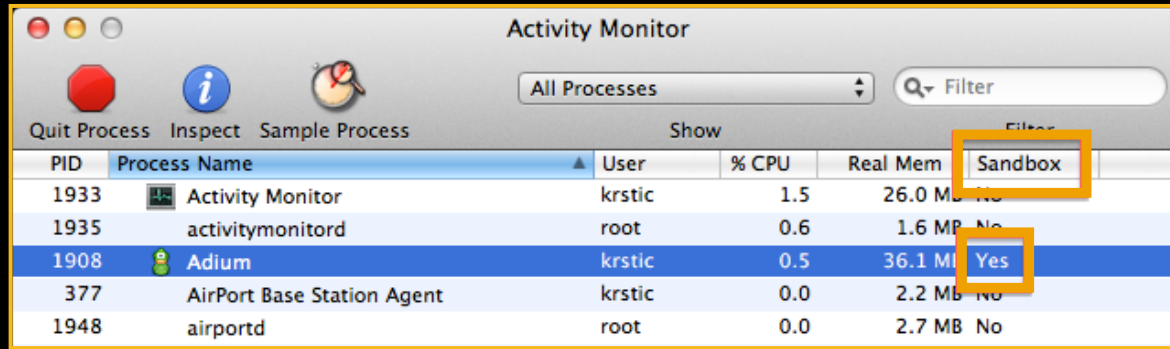
# Adium: Entitlements

```
com.apple.security.app-sandbox  
com.apple.security.personal-information.addressbook  
com.apple.security.files.user-selected.read-write  
com.apple.security.network.server
```

# Adium: Choosing Entitlements



# Adium: Run and Verify



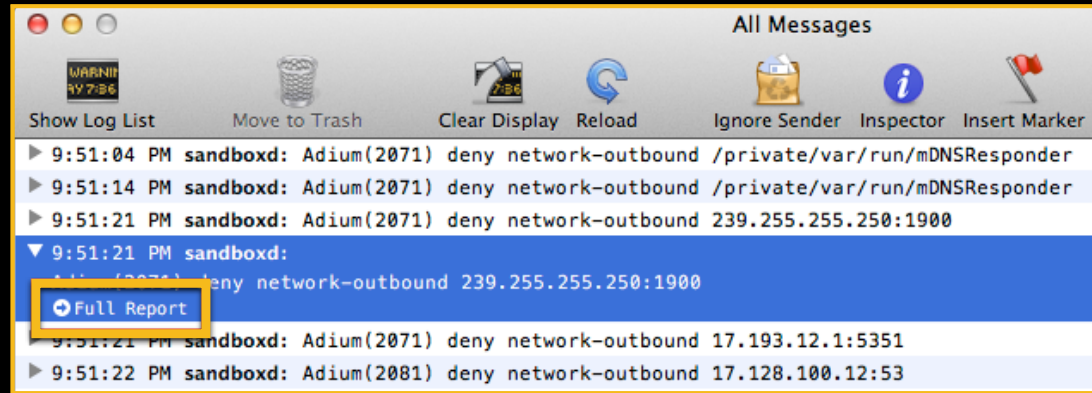
The screenshot shows the Activity Monitor application window with the following data:

| PID  | Process Name               | User   | % CPU | Real Mem | Sandbox |
|------|----------------------------|--------|-------|----------|---------|
| 1933 | Activity Monitor           | krstic | 1.5   | 26.0 MB  | No      |
| 1935 | activitymonitord           | root   | 0.6   | 1.6 MB   | No      |
| 1908 | Adium                      | krstic | 0.5   | 36.1 MB  | Yes     |
| 377  | AirPort Base Station Agent | krstic | 0.0   | 2.2 MB   | No      |
| 1948 | airportd                   | root   | 0.0   | 2.7 MB   | No      |



Activity Monitor

# Adium: Violations



Console

# Adium: Violation Report

```
Adium(2071) deny network-outbound 239.255.255.250:1900
```

```
Backtrace:
```

```
0 libsystem_kernel.dylib  
  0x00007fff8d8f __sendto + 10
```

```
1 libpurple  
  0x000000010e1d purple_upnp_discover_send_broadcast + 171
```

```
[...]
```



# Adium: Fix, Iterate

- We forgot to add the `network.client` entitlement
- Check the box in Xcode, rebuild, rerun

# Adium: Exploitation

- The attacker only has access to documents that the user exchanged with buddies during this Adium run
- No ability to access or modify other apps or documents
- Need another vulnerability for a successful exploit

# Summary

# App Sandbox

- New damage containment mechanism in Lion
- Last line of defense against exploitation and coding errors
- Not an anti-virus system; does not target intentionally malicious software
- Drives policy by user intent
- See “Code Signing and Application Sandboxing Guide”
- Sample code available

# App Sandbox: Availability

App Sandbox and the Mac App Store

Nob Hill  
Tuesday 3:15PM

Sandbox Lab

Core OS Lab B  
Wednesday 9:00AM

- Adoption very strongly encouraged for all Mac OS X applications
- Required for Mac App Store apps

# Summary



- iOS Sandbox— 14 billion app downloads with confidence
- Delight users with carefree apps on Mac OS X
- Restore sense of childlike wonder

