Tackling Common Vulnerabilities

Mircea Vasiliniuc @mirceaSV

February 2018

Agenda

Security Audits

App Vulnerabilities

Leaking Data

- Screen Capture
- Keyboard Logging
- State Preservation
- *Pasteboard*

- Examples
- Demos
- Solutions

Security Audits

Static & Dynamic analysis of binary

Static analysis of source code

Automated Functional Testing

Manual security analysis techniques.

Security Audits

Static & Dynamic analysis of binary

Static analysis of source code

Automated Functional Testing

Manual security analysis techniques.

↓

Filter, sort, risk level accession

Security Audits

Filter, sort, risk level accession

- **Vulnerability Risk Level (Impact, Probability)**
- **Possible remedies**
- **Responsible Code**
- **Recommendations**

Vulnerabilities

Let's define our context...

Sensitive Data

Vulnerabilities

Let’s define our context...

Attackers

Vulnerabilities

Let’s define our context...

Implications

Vulnerabilities

Let’s define our context...

User Trust

Vulnerabilities

Let’s define our context...

Vulnerabilities

*in this presentation

Sensitive Information Disclosure

Side Channel Data Leakage

Mitigation Strategies

*in this presentation

Avoid unprotected data storage

Secure keystrokes, screen shots

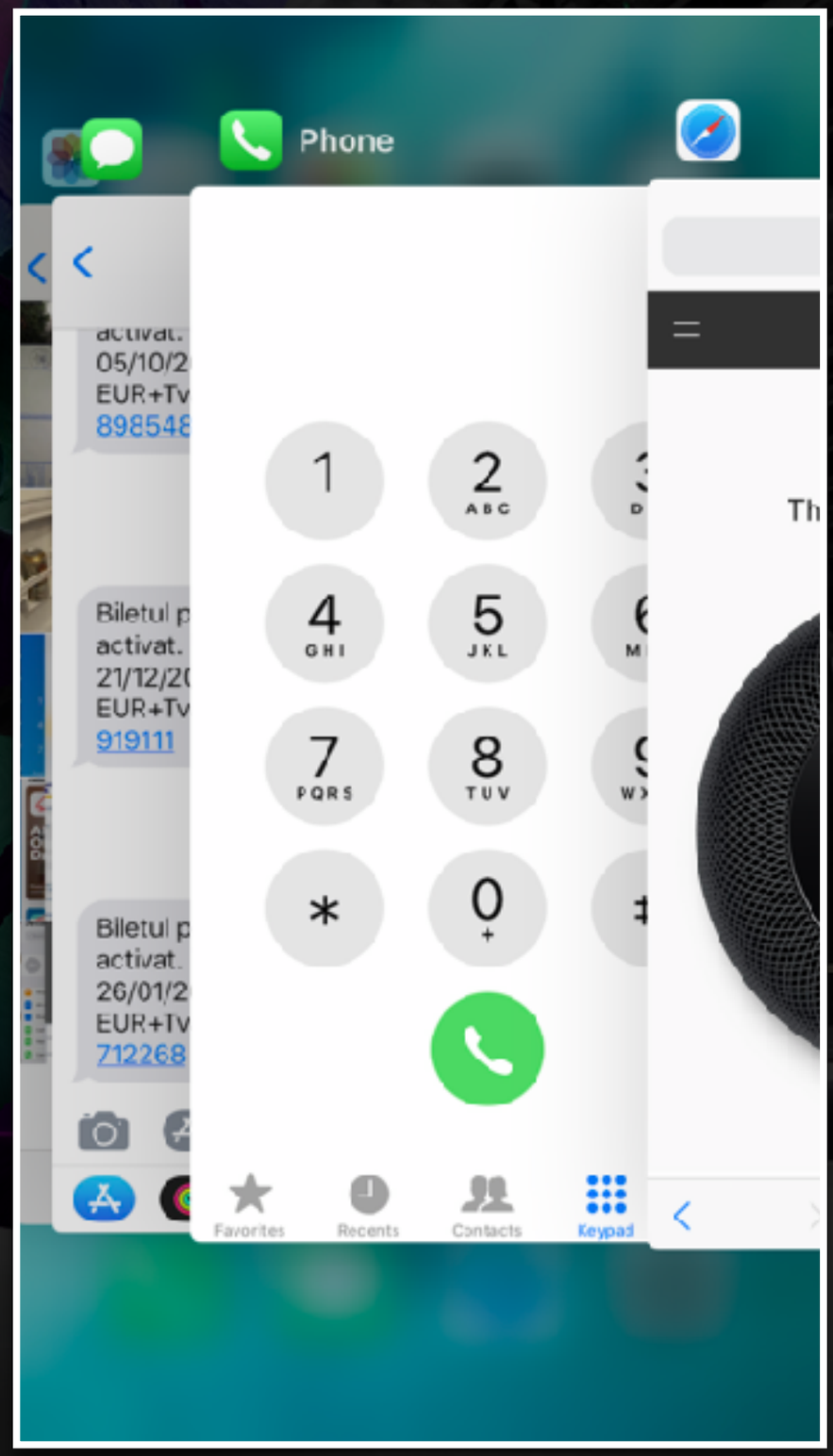
Protect agains malicious apps

Task Switcher

Screenshots used for smooth animations.

Issue:

Screenshots might capture sensitive data.

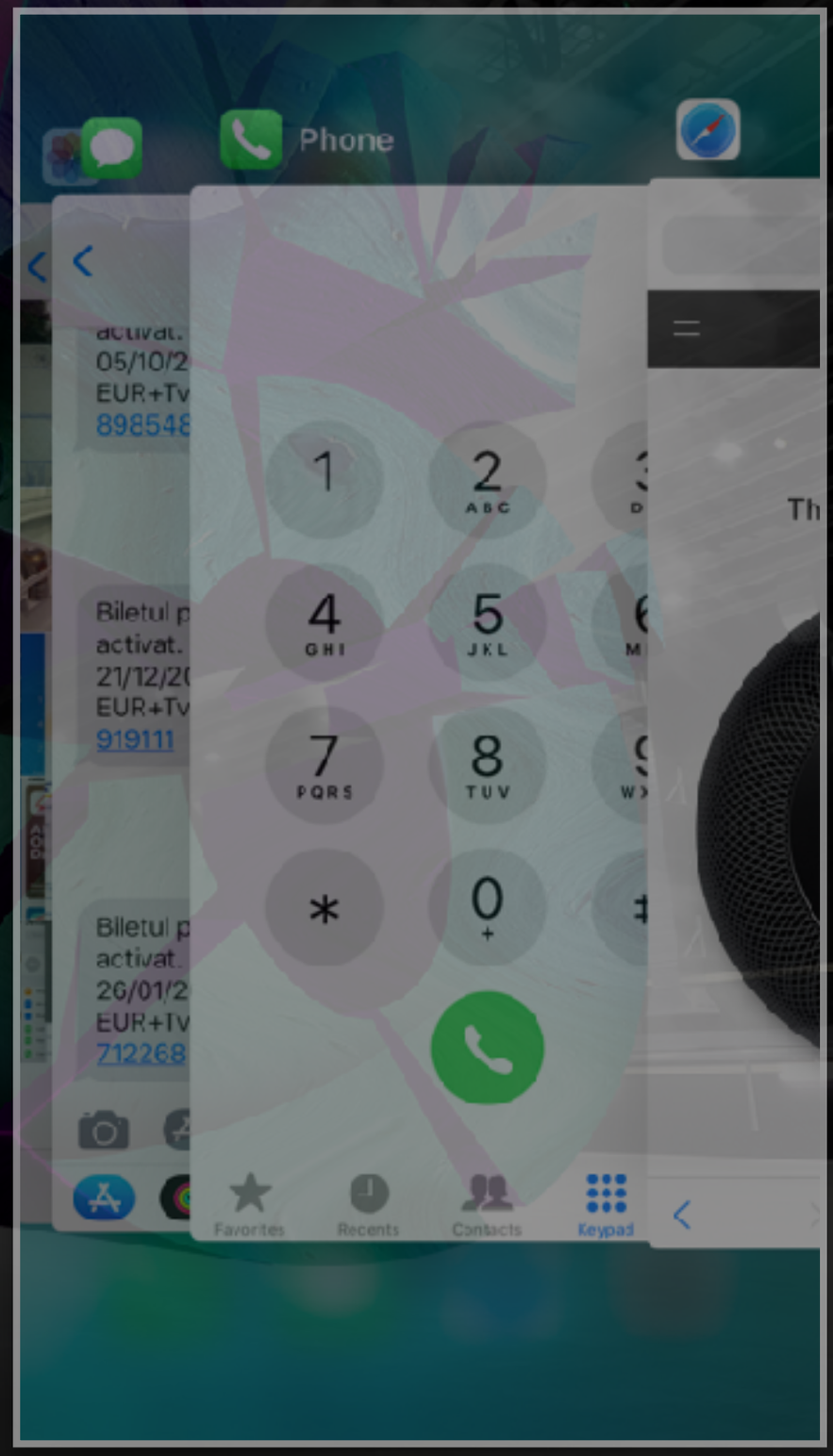


Task Switcher

Screenshots used for smooth animations.

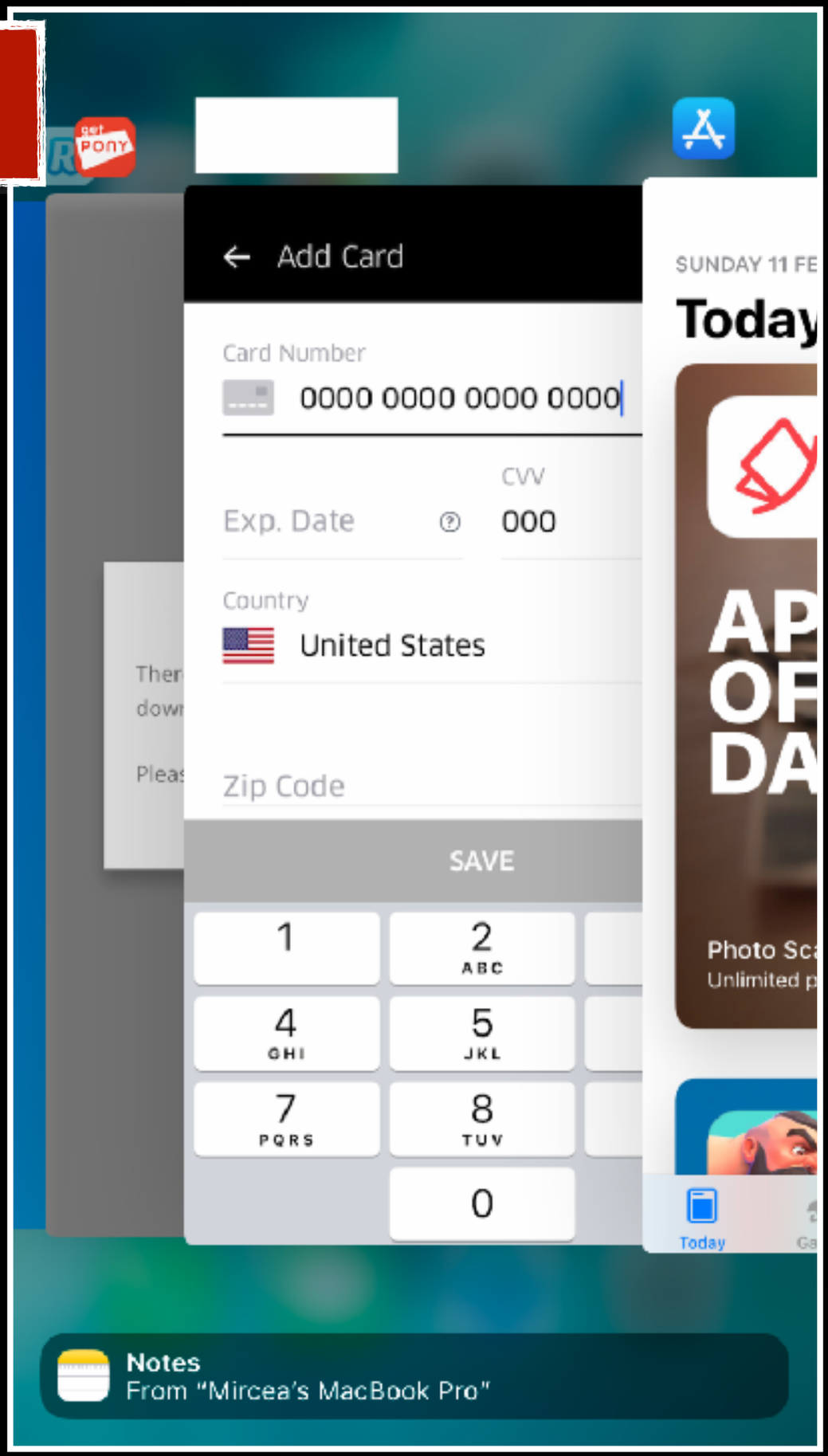
Issue:
Screenshots might capture sensitive data.

Impact: Hight
Probability: Low
Security Risk: Medium
Sanitization Effort: Reduced



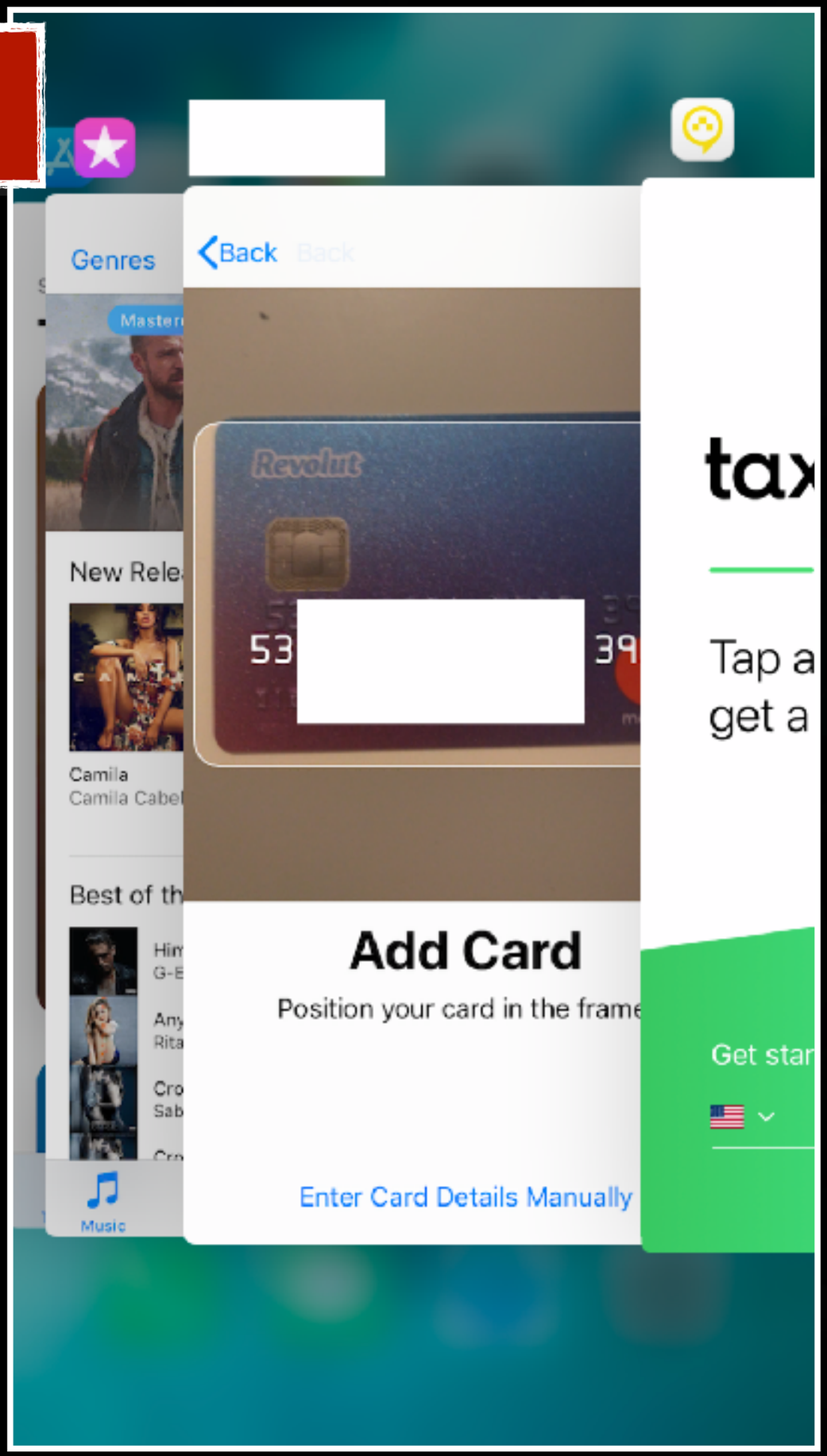
Vulnerable

Examples



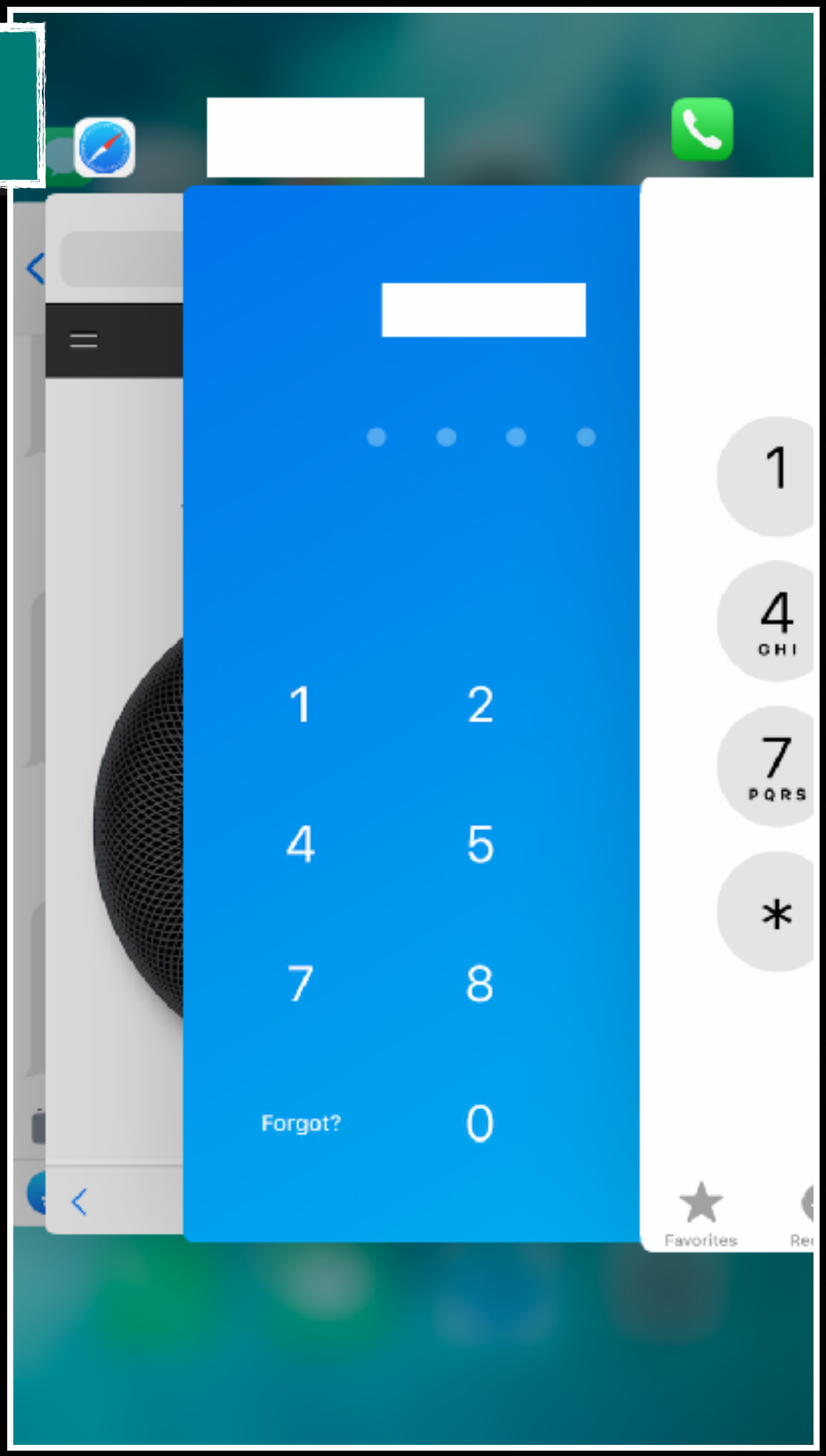
Vulnerable

Examples



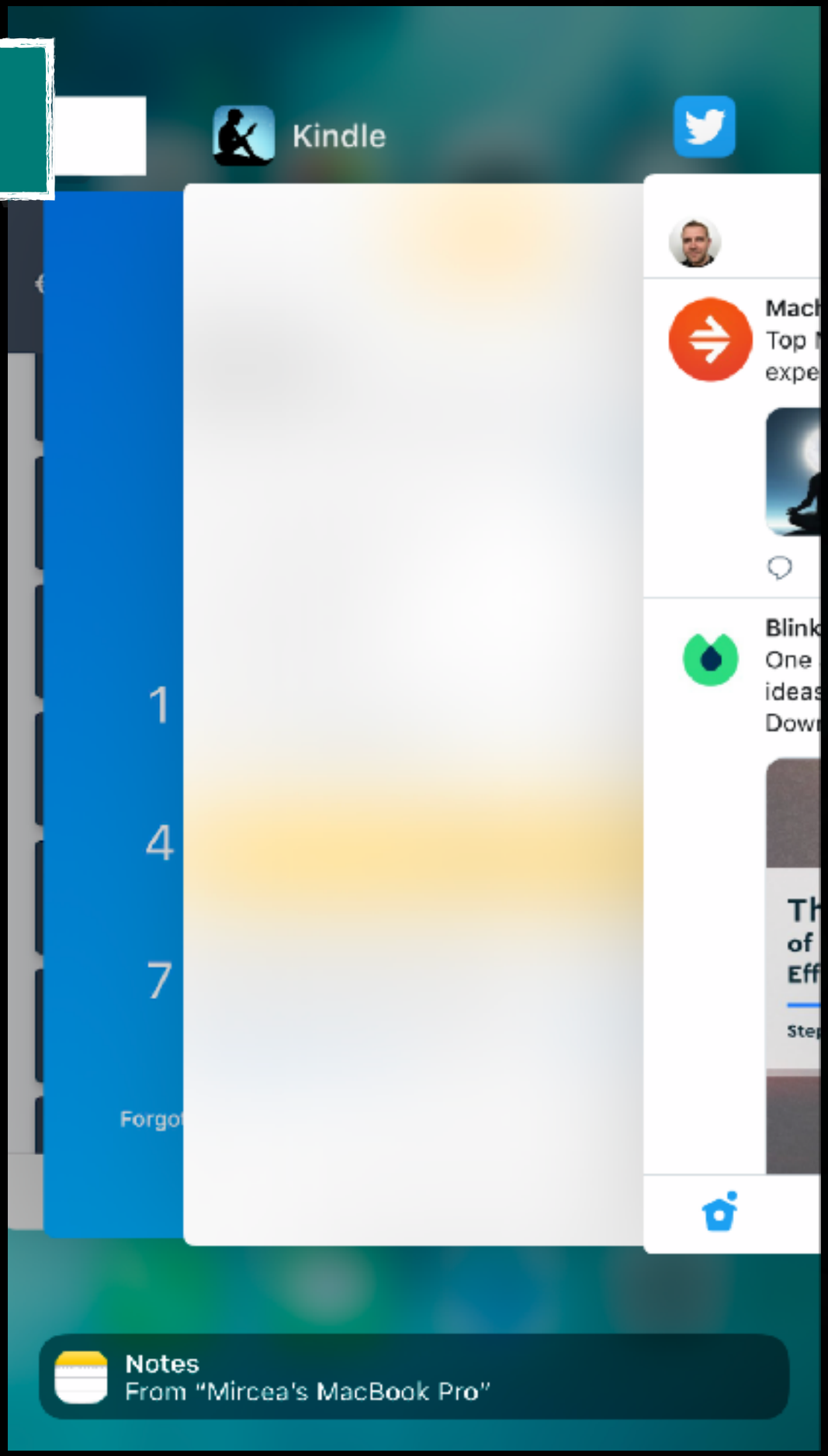
OK

Examples



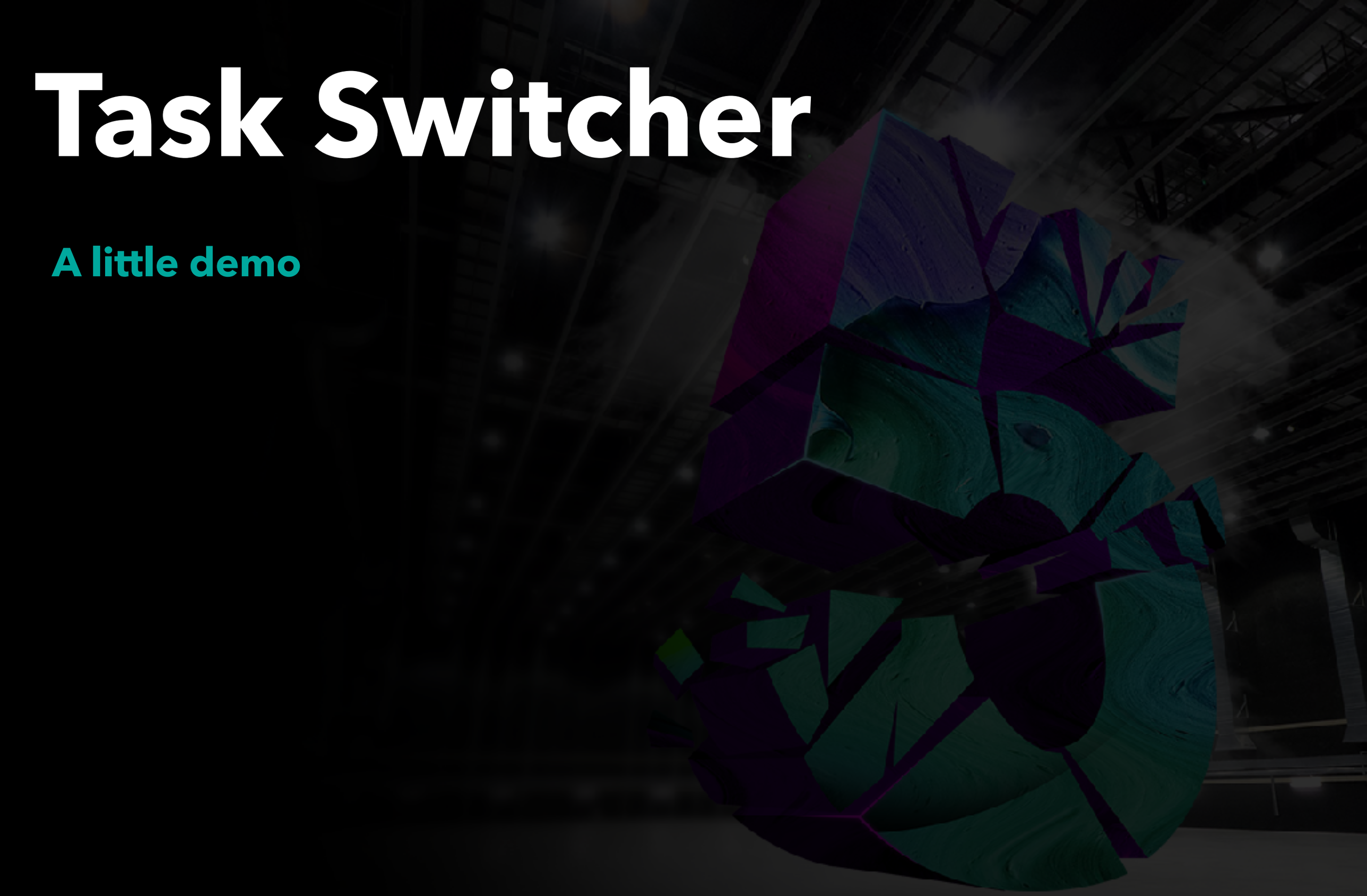
OK

Examples



Task Switcher

A little demo

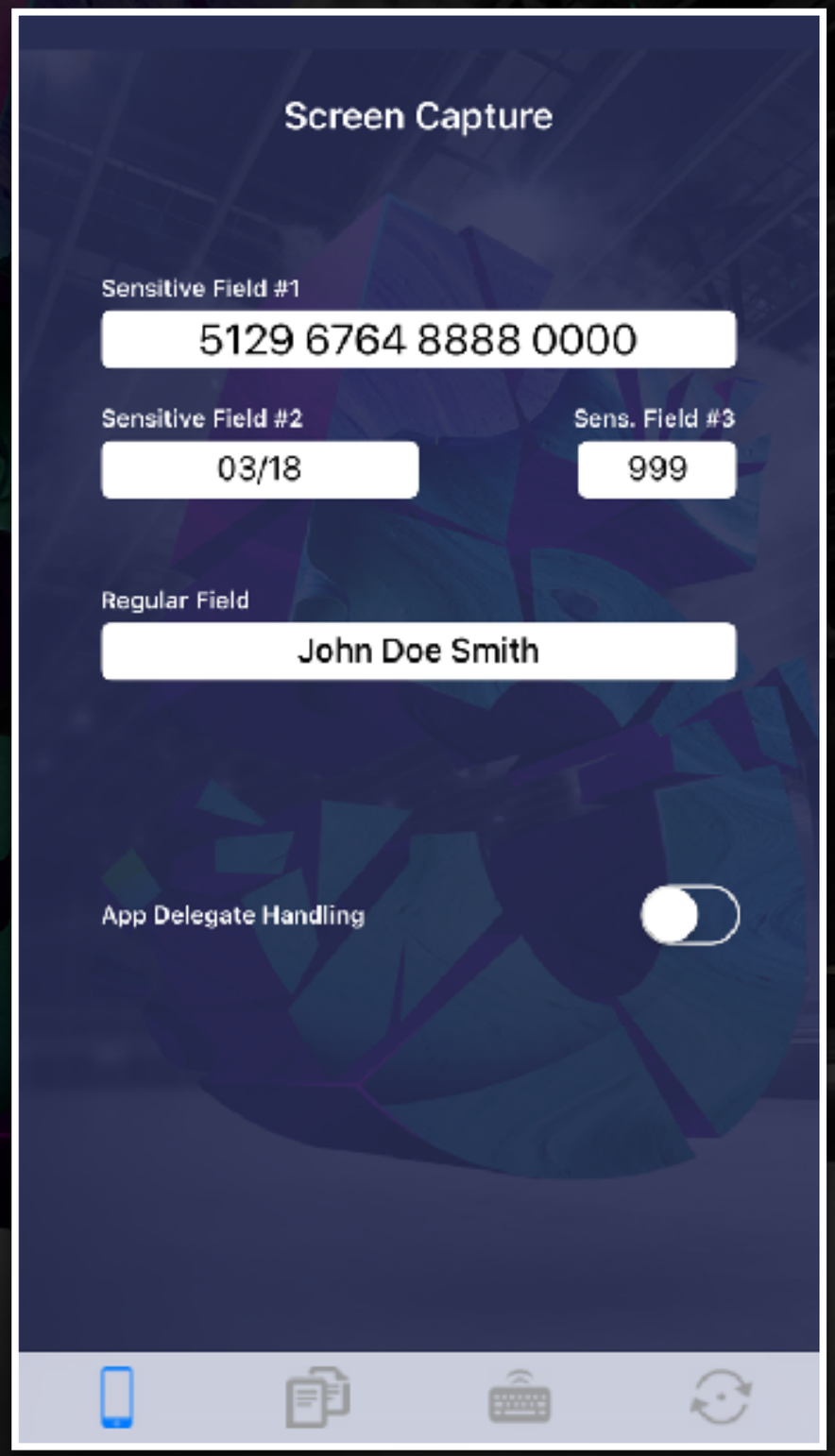


Task Switcher

Problematic Context:

- **Stolen / Unprotected Device**
- **File Access**
  - **Tools like iFunBox/iFile/iExplorer**
  - **Jailbreak**

Location:
Cache/Snapshots folder (.ktx files)



Task Switcher

A little demo

Prevention:

1. App Delegate Level - Change view hierarchy
2. View Level - Hide sensitive information
3. Block application suspension (UIApplicationExitsOnSuspend)

Test solutions in Simulator (/Library/Developer/CoreSimulator/.../Caches/Snapshots/)

Task Switcher

A little demo

Consider embedded web views.

Prevention:

1. App Delegate Level - Change view hierarchy
2. View Level - Hide sensitive information
3. Block application suspension (UIApplicationExitsOnSuspend)

Test solutions in Simulator (/Library/Developer/CoreSimulator/.../Caches/Snapshots/)

State Preservation

Preserving Your App's UI Across Launches

Issue:

User input / progress is stored on local storage.
Data is encoded but not encrypted (!).

State Preservation

Preserving Your App's UI Across Launches

Issue:
User input / progress is stored on local storage.
Data is encoded but not encrypted (!).

Impact: High
Probability: Low
Security Risk: Medium
Sanitization Effort: Medium

State Preservation

Problematic Context:

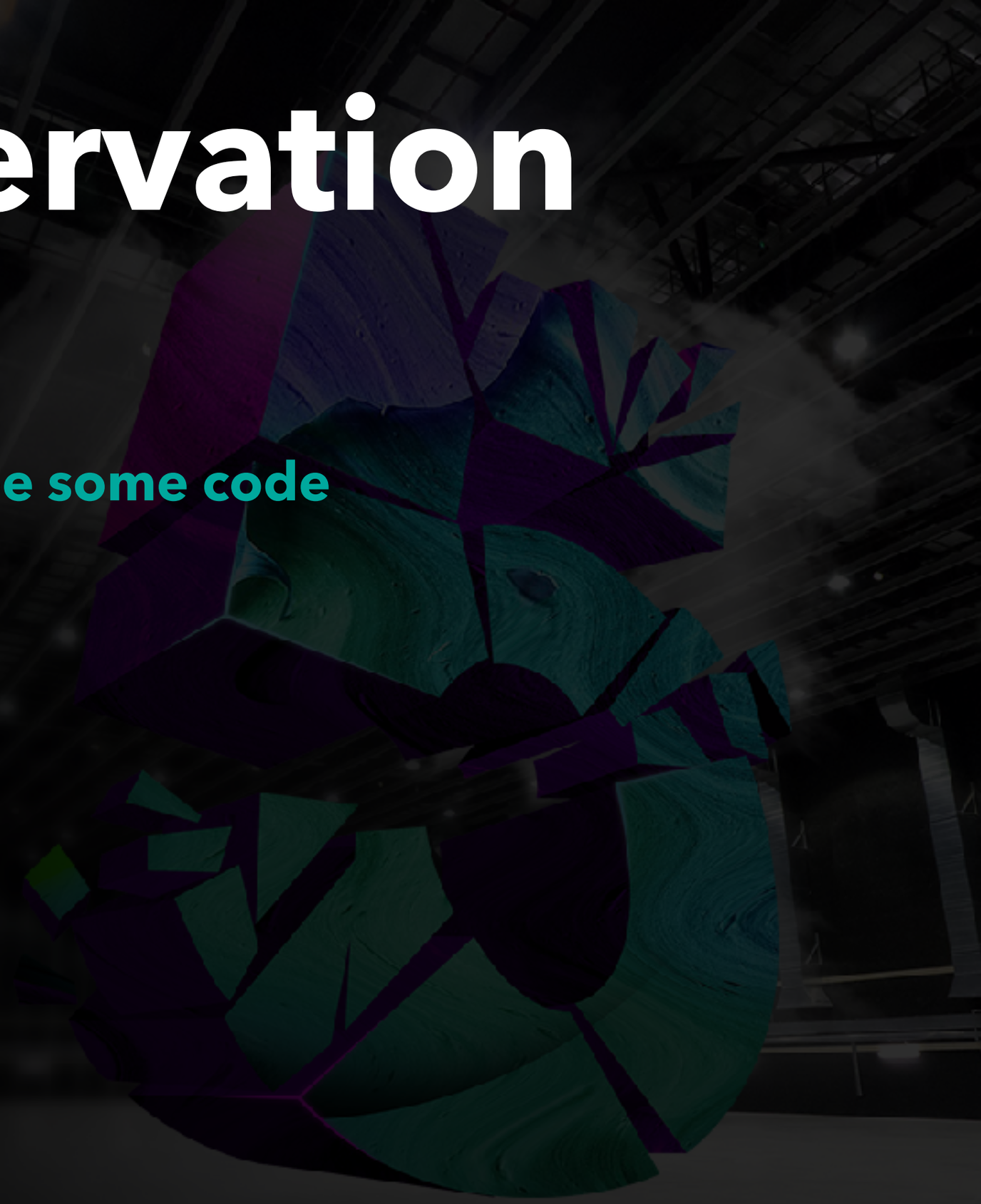
- **Stolen / Unprotected Device**
- **File Access**

Location: ***Cache/Saved Application State folder (data.data files)***

State Preservation

A little demo?

Difficult using device...Let's see some code



State Preservation

```
// MARK: UIStateRestoring Extension
extension StatePreservationViewController {
    override func encodeRestorableState(with coder: NSCoder) {
        if let inputText = textHolder?.text {
            coder.encode(inputText, forKey: "textInput")
        }
        if let image = imageHolder?.image {
            coder.encode(UIImagePNGRepresentation(image), forKey: "imageInput")
        }

        super.encodeRestorableState(with: coder)
    }

    override func decodeRestorableState(with coder: NSCoder) {
        if let inputText = coder.decodeObject(forKey:"textInput") as? String {
            textHolder?.text = inputText
        }

        if let imageData = coder.decodeObject(forKey:"imageInput") as? Data {
            let image = UIImage(data: imageData)
            imageHolder?.image = image
        }

        super.decodeRestorableState(with: coder)
    }
}
```

State Preservation

Prevention: Encryption

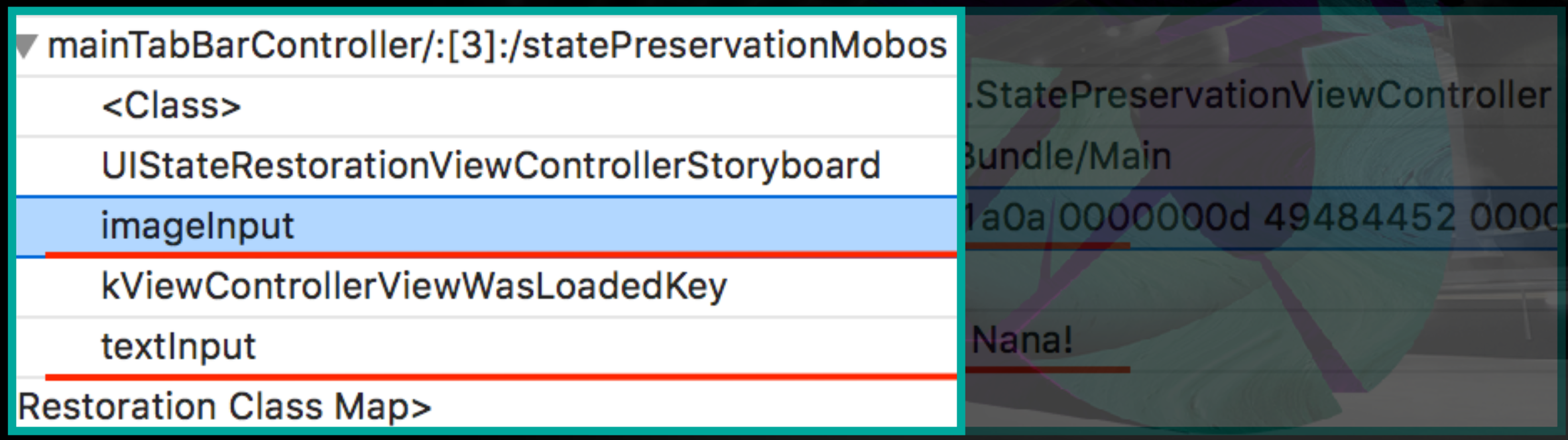
Open Source:

- **RNCryptor-objc**
- **SwiftyRSA**
- **etc.**

State Preservation

Testing

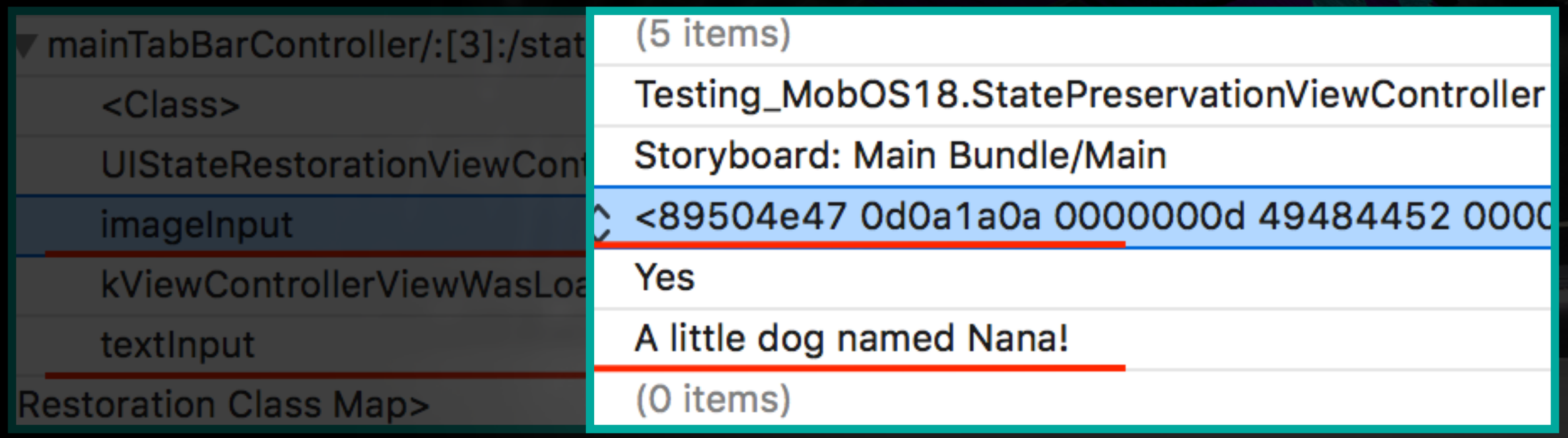
Use *"restorationArchiveTool for iOS"* from Developer Tools to decode.



State Preservation

Testing

Use *"restorationArchiveTool for iOS"* from Developer Tools to decode.

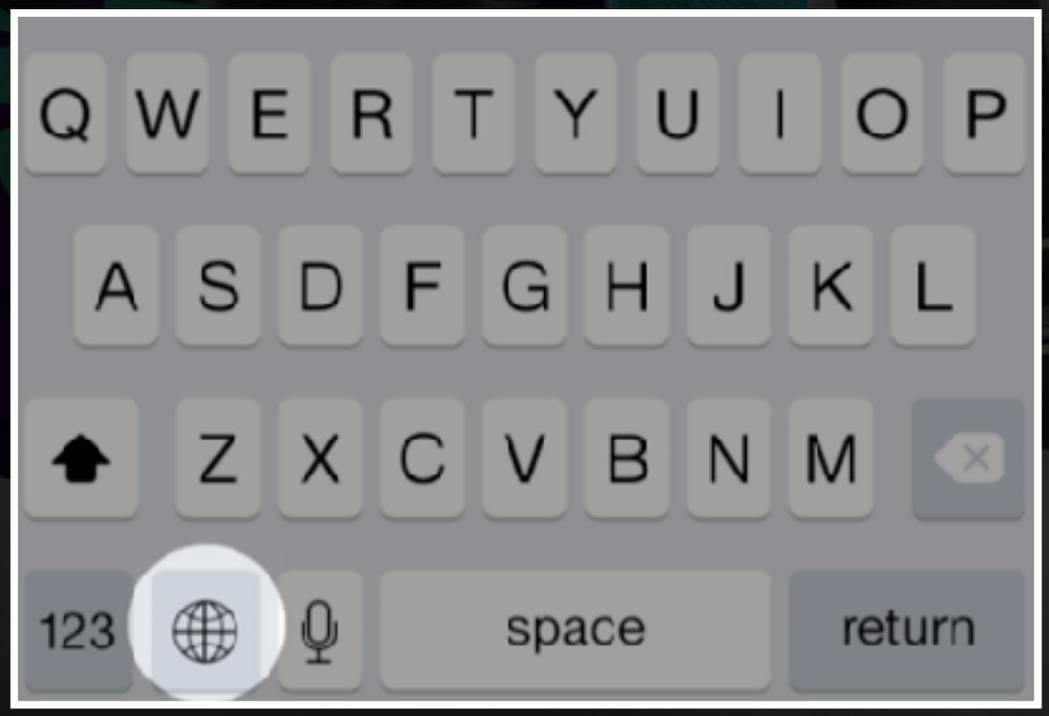


Keyboard Logging

A keyboard extension replaces the standard keyboard with a custom keyboard.

Issue:

Can lead to information disclosure through key-logging by the 3rd party developer.



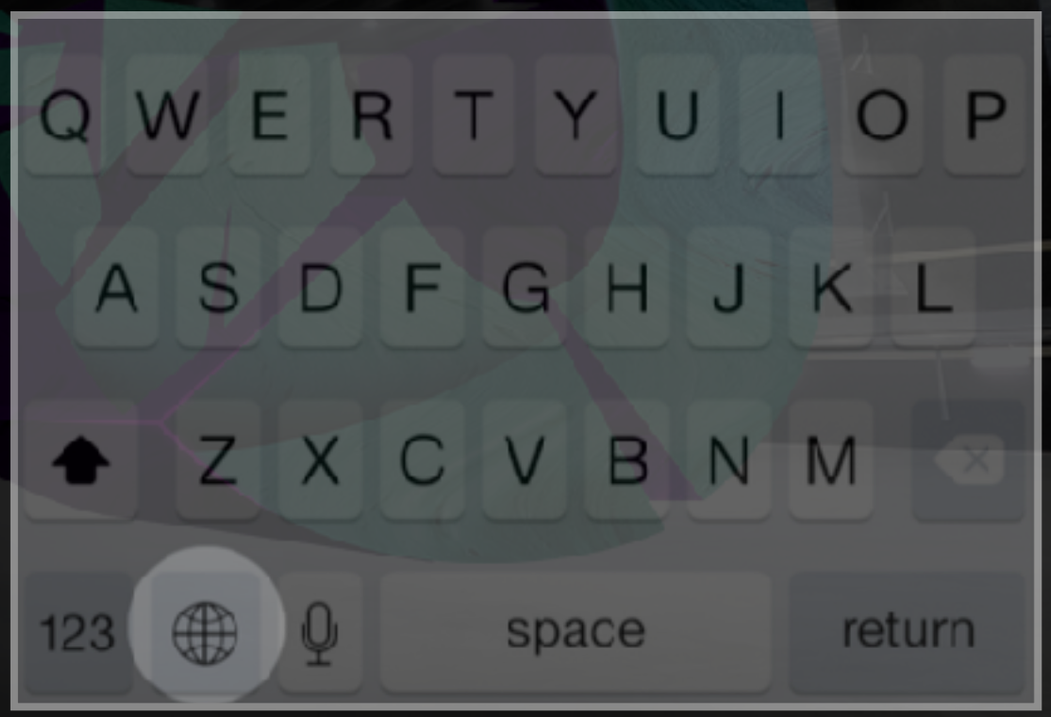
Keyboard Logging

A keyboard extension replaces the standard keyboard with a custom keyboard.

Issue:

Can lead to information disclosure through key-logging by the 3rd party developer.

Impact: Critical
Probability: Medium
Security Risk: High
Sanitization Effort: Low



Keyboard Logging

Problematic Context:

- 3rd Party Keyboards are installed
- Full Access is accepted by user

Keyboard Logging

Once “Allow Full Access” is ON,
be prepared to share data with 3rd parties

Major cases of data leaks:

- **SwiftKey (July 2016)**
- **Ai.Type (December 2017)**

Keyboard Logging

A little demo

Prevention:

Block Keyboard Extensions (!)

via shouldAllowExtensionPointIdentifier

Keyboard Logging

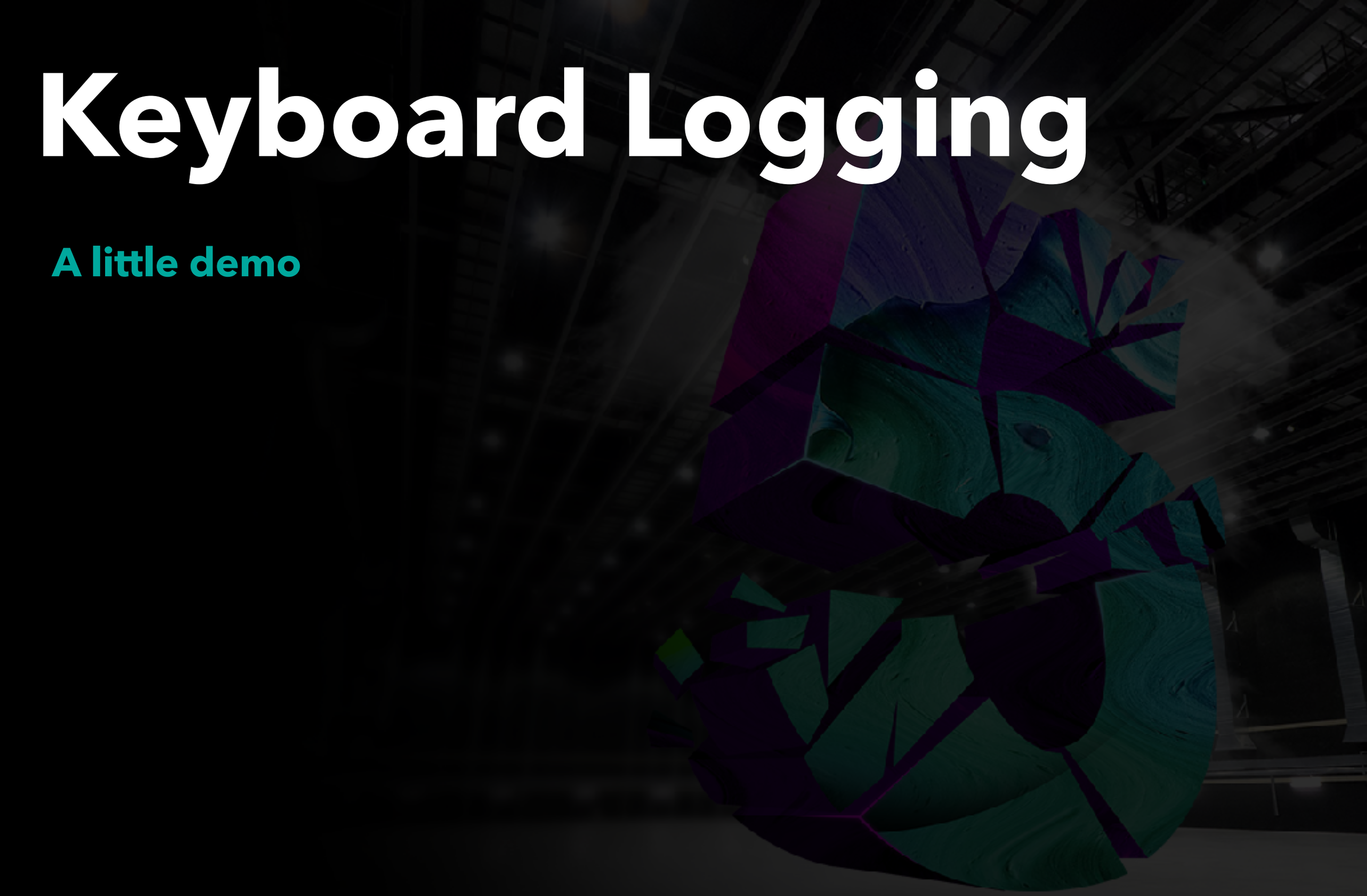
Special Case: Autocorrection

Prevention:

Disable autocorrection on sensitive fields

Keyboard Logging

A little demo



Pasteboards

Issue:

Can lead to information disclosure through copy/cut operations

Pasteboards

Issue:

Can lead to information disclosure through copy/cut operations

Impact: High
Probability: Medium
Security Risk: High
Sanitization Effort: Low

Pasteboards

A little demo

Major updates in iOS 9 and iOS 10.

iOS 9, pasteboards are EMPTY on background queues.

NOT recommended:
Clear the Pasteboard When You Switch Apps

Pasteboards

A little demo

Pasteboards

Prevention:

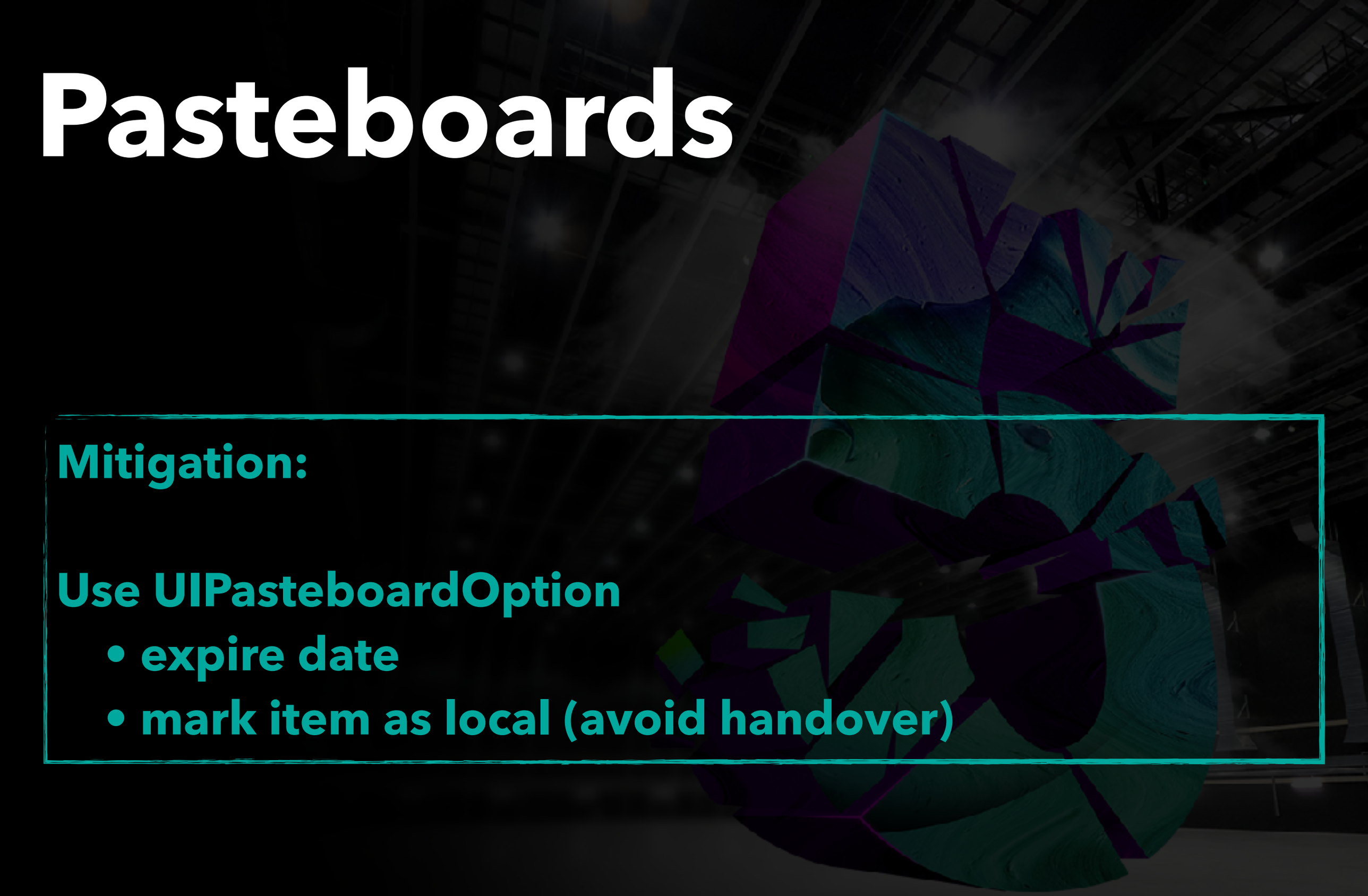
Block operations from UIResponderStandardEditActions (cut, copy)

Pasteboards

Prevention:

```
11 /// Safer Text Field
12 class RestrictedTextField: UITextField {
13     override func canPerformAction(_ action: Selector,
14                                    withSender sender: Any?) -> Bool {
15         // Avoid cut and copy operations
16         if action == #selector(cut(_:)) ||
17             action == #selector(copy(_:)) {
18             return false
19         }
20         return true;
21     }
22 }
```

Pasteboards



Mitigation:

Use UIPasteboardOption

- expire date
- mark item as local (avoid handover)

Takeaways

Vulnerabilities of medium/high risk can be resolved with reduced efforts (Majority)

Maintain User Trust is important.
Don’t even give the impression that your app is vulnerable

Assume that the user’s device will be lost or stolen

Questions

now...

...or later

mircea.dev@icloud.com
@mirceaSV

Sharing is caring

github.com/mvasiliniuc/Mobos2018

- slides
- sources
- references

Sharing is caring

Thank you!