

# מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS

## עם iNalyzer

נכתב ע"י חיליק טמיר (Appsec-labs.com)

### הקדמה

ביצוע בדיקות חדירות לאפליקציות iOS הינו עסק מסובך, אין בנמצא אימולטור, אין קוד, המוצר חתום בדרך כלל ולכן הסיפור אינו פשוט. בנוסף במקרים רבים הבקשות הנשלחות מהמכשיר חתומות או מוצפנות ולכן אין אפשרות ממשית לבצע פעולות Tampering או Injecting שונות מתוך התווך או מפרוקסי. וכמובן שאין מה לחשוב בכלל על כל הכלים האוטומטים שחוסכים לנו הרבה זמן בבדיקות אבטחה רגילות (AppScan, Acunetix, WebInspect, Burp וכדומה).

במאמר זה אציג בפניכם גישה חדשנית לביצוע בדיקות אבטחת מידע לאפליקציות iOS, אדגים את התהליך מתחילתו ועד סופו, ואחשוף בפניכם, הקוראים, את השימוש ב-iNalyzer, פרי המחקר האחרון שלי בחברת AppSec-Labs.com אשר מופץ בחינם לשימושכם (החוקי).

אבל אולי לפני הכל כמה חוקים כלליים על אפליקציות iOS...



### עשרת הדיברות לניתוח אפליקציות iOS

**אנוכי ה'** - בכדי לבצע בדיקות אתה זקוק לרמת הרשאה עודפת מאשר האפליקציה, על מנת שתוכל לשכנע אפליקציה ששחור זה לבן - אתה זקוק ל-Root ולמכשיר jailbreak, ולכן כהכנה יש להשיג מכשיר ולדאוג לפתוח אותו ב-jailbreak.

**לא יהיה לך** - סורסים של האפליקציה - היית רוצה אבל זה

לא יוצא, ולכן המטרה העקרית היא לחלוב את כל האינפורמציה מתוך מה שיש בשאיפה לכמה שיותר תיעוד, החיסרון הינו שהקוד מקומפל לאסמבלי של ARM, שזה אומר שאתה צריך IDA ורקע באסמבלי בכדי להתחיל להבין מה הולך. לעומת זאת, היתרון הינו המבנה המיוחד של קוד שקומפל ל-iOS אשר מכיל בחובו המון מידע על מבנה הקוד ותפקודו.

**לא תשא** - בעול האבדן של המידע שלך. יש לבצע בדיקות על מכשיר מגובה או על מכשיר ייעודי.  
**זכור את** - המספר הסידורי של המכשיר שלך (UDID, IEMI). יש מצב טוב מאוד שהוא יופיע בכל מיני בקשות מוזרות של המערכת.

**כבד את** - חתימות הבקשות, יש מצב טוב שבקשות שנשלחות מהמכשיר חתומות, כל התערבות ישירה מהפרוקסי בטווח מהווה בזבוז זמן במקרה הטוב.

**לא תרצח** - תהליך במהלך העבודה ללא שמירת המצב, קח בחשבון שהתהליך יכול לעוף במהלך הבדיקה שלך - כדאי מאוד שתוכל להמשיך לעבוד בשניה שאתה מרים אותו בחזרה, ולכן תיעוד מדוקדק מאוד מסייע.

**לא תנאף** - ותתקין תוכניות על המכשיר ממקומות מפוקפקים. מאוד קל לחטוף Rootkit וזה מזבל מאוד את תעבורת הרשת.

**לא תגנוב** - תוכנות של מישהו אחר. תהליך הבדיקה מנטרל את מנגנון ה-DRM של אפל, אבל זה לא אומר שעכשיו אתה הולך ומחלק לכל החברים גרסאות של תוכנות בתשלום.

**לא תענה** - לשיחות טלפון במהלך הבדיקות, זה הורס את הבדיקה ומכניס אותה ל-Delay, עבור למצב טיסה או הוצא את כרטיס הסיים.

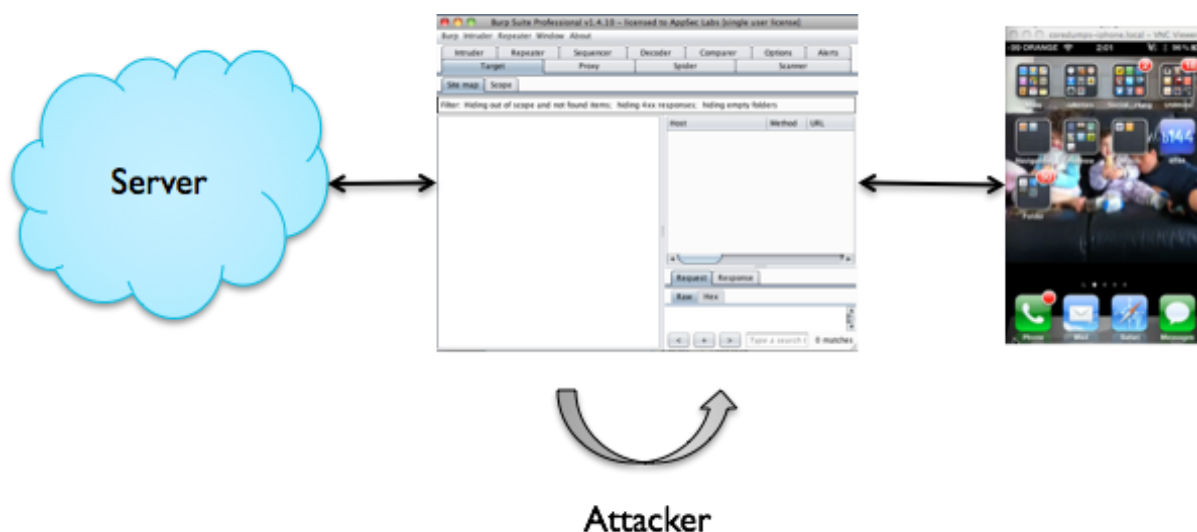
**לא תחמוד** - דרוש מכשיר iOS בכדי לבצע בדיקות אבטחת מידע על אפליקציות iOS ולו מהסיבה שאפליקציות iOS נכתבות בשפת c / ++ או Objective-c ומקומפלות לקבצים בינאריים בפורמט mach-o התואמים את המעבד (במקרה שלנו, ARM), מאחר ואין בנמצא אימלטור עבור סביבת האייפון הפרוייקט האחרון שרץ נסגר על ידי לחץ של אפל (אין לנו אפשרות לבצע בדיקות אבטחה אלא על מכשיר אמיתי). אבל זה לא אומר שהוא צריך להיות iPhone5 מפלוטוניום משובץ ביהולמים 45 קראט...



אחרי שהבהרנו את הנקודות האלו, בואו נסקור את שיטת העבודה המוכרת לנו עד כה ומה נדרש בכדי להתאימה לבדיקות iOS.

## זה מה יש

עד עכשיו כאשר רצינו לבצע בדיקות על מערכת רשת היינו מרימים סביבה מהתצורה הבאה:



דהיינו, שרת חיצוני, פרוקסי ונייד. זה נחמד עבור אפליקציות פשוטות: נכנסים עם הפרוקסי, מחברים איזה Scanner ונותנים לעסק לרוץ על אוטומט בזמן שאנחנו מתמקדים בבדיקות ידניות. אלא שיש כמה שאלות מכריעות שאנחנו צריכים לענות עליהן בניתוח שלנו לפני שנוכל לומר שהבדיקה הסתיימה:

### מה אני יודע על המערכת?

"יוסטון יש לנו בעיה" - לאן האפליקציה מתחברת? האם אני יודע בוודאות שכיסיתי את כל נקודות הממשק של המערכת על ידי בדיקה ידנית? מה אחוז הכיסוי שלי? מה פיספסתי...?

← אין לנו יכולת טובה לברר מהם נקודות הקצה של המערכת מול העולם החיצון!

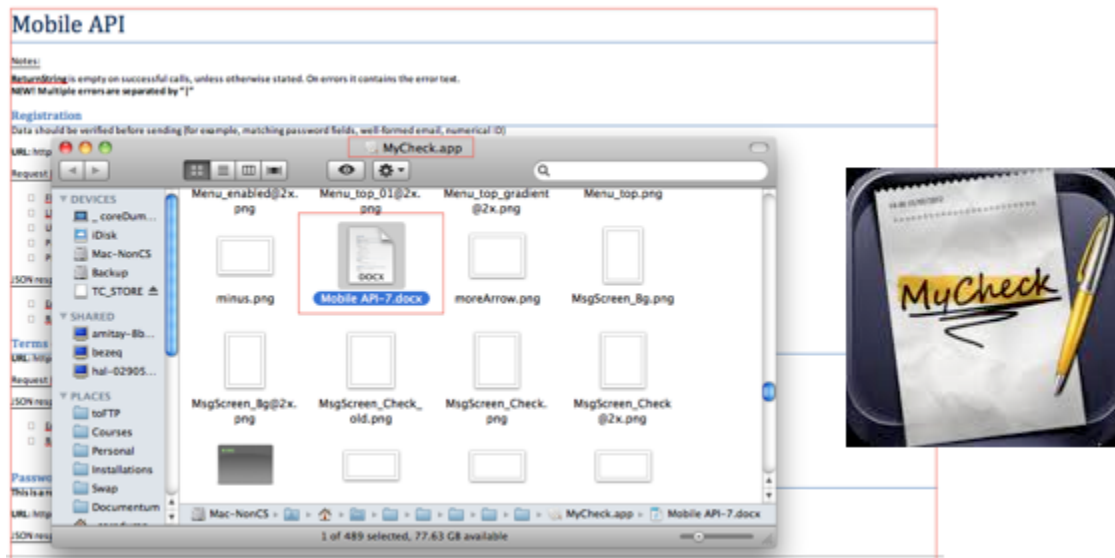
"תחתום לי פה" - מה קורה אם אנחנו רוצים לבצע בדיקות על אפליקציה, כאשר הקליינט שולח בקשות חתומות, או שהוא שולח בקשות דרך 3G, ואין לנו יכולת טובה לייטר אותן ולשחק איתן. יש לנו סיכוי גבוה לפספס כיסוי ובעיות, מאחר וכל הבקשות שלנו ייפלו על ולידציות או על התאמות של צד שרת.

← אין לנו יכולת טובה להתעסק עם חתימות, במקרה שהקליינט חותם בקשות: **Game Over!**

"עיזים" - איזו פונקציונאליות מתחבאת לי בקליינט ואני לא יודע עליה? "לא ראיתי אותה במהלך הבדיקות שלי.", "לא ידעתי על קיומה...", "איך אני יודע בוודאות גמורה שכיסיתי את כל המערכת ואין לי עיזים במוצר?"

← אין לנו יכולת טובה להכיר את כל הפונקציונאליות הנחבאת של המערכת.

"איפה המפתח של הצוללת" - האם יש מפתחות רגישים בתוך הקוד? האם ישנם נתונים רגישים אחרים שמוחבאים באפליקציה? לפעמים המפתחים עושים דברים מטופשים כמו להעלות את האפליקציה יחד עם מסמך שלם המתאר את כל ה-API (לתשומת ליבה של גב' רפאלי).



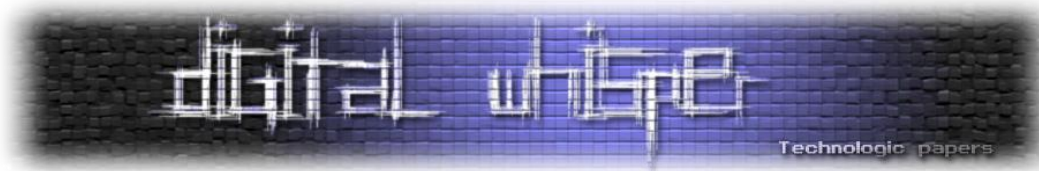
← אין לנו יכולת טובה של הכרת המערכת והקבצים שלה והתוכן שלהם!

הבעיה בכל השאלות הללו היא שמתוך הפרוקסי אנחנו לא יכולים לענות עליהן, אנחנו לא יודעים אם כל הבקשות הפוטנציאליות עברו דרכינו או אם כל המסכים הרלוונטיים נטענו על המכשיר. במקרים רבים התווכח מוצפן כך שגם בדיקות רגילות של טימפור הם עסק בעייתי ואנחנו נקבל המון false-positive.

### Cycript - לשנות את הגישה

אנו יודעים שהאפליקציה עצמה יודעת לבצע את החתימה ואת כל השלבים עד שהבקשה יוצאת לשרת, ולכן היינו רוצים דרך נוחה להזין לאפליקציה ערכים מזוייפים או שונים. מה עוד שאם היתה לנו אפשרות לדבג את האפליקציה היינו יכולים לשנות כתובות זיכרון וערכים on-the-fly, ולתת לאפליקציה לשלוח את המסרים המטומפרים שלנו אל השרת. אבל כמו שאמרנו עבור objc, הקומפילציה היא לקובץ mach-o בשפת מכונה ללא שפת ביניים, ולכן היכולות הכלליות שלנו בדיבוג האפליקציה היא לעבוד עם gdb. למזלנו, ישנו בחור בשם [Jay Freeman](#) (ידוע גם כ-saurik) שמאוד אוהב לאתגר את החברה עם התפוח והוא מוציא כל פעם כלים נפלאים (סידיה לדוגמא) לשימוש הכלל, אחד מהכלים הללו הוא [Cycript](#).





עם Cycript אנחנו יכולים להתעסק עם ה- Runtime בנוחות של javascript ו-objC, הבעיה היחידה בשימוש Cycript הינו הצורך להכיר את האובייקטים השונים של המערכת ואת הסלקטורים והמתודות שהוא יודע לקבל.

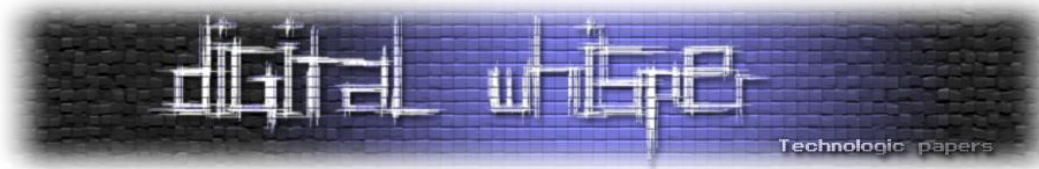
כמו שאמרנו - אנחנו מבצעים פה black-box אז מאיפה נשיג את האינפורמציה?  
התשובה היא: מהמאקו (mach-o)!

## "מאקו"-רה פה?

כן, אנחנו הולכים לנצל את המבנה הייחודי של קובץ ה"מאקו" (mach-o) לטובת חליבת אינפורמציה על המערכת שלנו.

[המבנה של קובץ שכזה](#) אורז בתוכו את כל התבניות של האובייקטים שהוא צריך בכדי לרוץ. במילים אחרות אני יכול לשאול את קובץ המערכת שלנו אילו אובייקטים ומתודות הוא צריך בכדי לרוץ. דוגמא נוספת: אנחנו משתמשים ב-otool של Apple על מנת לתשאל קובץ מאקו, ובעיקר מעניין אותנו אילו נתונים כתבו המפתחים שלו לתוך סגמנט ה-ObjC שלו:

```
coreDumps-MacBook-Pro-2:SpringBoard.app _coredump$ otool
Usage: otool [-fahLDtdorSTMRIHvVcXm] <object file> ...
-f print the fat headers
-a print the archive header
-h print the mach header
-l print the load commands
-L print shared libraries used
-D print shared library id name
-t print the text section (disassemble with -v)
-p <routine name> start disassemble from routine name
-s <segname> <sectname> print contents of section
-d print the data section
-o print the Objective-C segment
-r print the relocation entries
-S print the table of contents of a library
-T print the table of contents of a dynamic shared library
-M print the module table of a dynamic shared library
-R print the reference table of a dynamic shared library
-I print the indirect symbol table
-H print the two-level hints table
-v print verbosely (symbolically) when possible
-V print disassembled operands symbolically
-c print argument strings of a core file
-X print no leading addresses or headers
-m don't use archive(member) syntax
-B force Thumb disassembly (ARM objects only)
coreDumps-MacBook-Pro-2:SpringBoard.app _coredump$
```



אז נריץ את otool על קובץ המערכת שלנו (במקרה זה את SpringBoard של iOS 5.0.1) ונבקש את כל הרשומות המופיעות בסגמנט ה-ObjC של קובץ ה-mach-o (ביקשנו רק 50 שורות):

```

coreDumps-MacBook-Pro-2:SpringBoard.app _coredump$ otool -o SpringBoard | head -i 50
SpringBoard:
Contents of (__DATA,__objc_classlist) section
0020ed48 0x248a64
  isa 0x248a50
  superclass 0x0
  cache 0x0
  vtable 0x0
  data 0x210868 (struct class_ro_t *)
    flags 0x0
    instanceStart 132
    instanceSize 132
    ivarLayout 0x0
    name 0x20443a SBSpringBoardMetaHostingWindow
    baseMethods 0x210890 (struct method_list_t *)
    entsize 12
    count 2
    name 0x1b33bd hitTest:withEvent:
    types 0x207519 @20@0:4{CGPoint=ff}8@16
    imp 0x12389
    name 0x1b7da8 _isWindowServerHostingManaged
    types 0x2070b5 c8@0:4
    imp 0x4c31
    baseProtocols 0x0
    ivars 0x0
    weakIvarLayout 0x0
    baseProperties 0x0
Meta Class
  isa 0x0
  superclass 0x0
  cache 0x0
  vtable 0x0
  data 0x210840 (struct class_ro_t *)
    flags 0x1 R0_META
    instanceStart 20
    instanceSize 20
    ivarLayout 0x0
    name 0x20443a SBSpringBoardMetaHostingWindow
    baseMethods 0x0 (struct method_list_t *)
    baseProtocols 0x0
    ivars 0x0
  
```

כפי שאתם רואים אנחנו מקבלים את מבנה האובייקט, השם שלו, אילו פרמטרים הוא מקבל ומחזיר. אז בואו נראה אילו אובייקטים ישנם בקובץ המערכת עם זיקה ללוח הנעילה (LockView):

```

coreDumps-MacBook-Pro-2:SpringBoard.app _coredump$ otool -o SpringBoard | grep name | sort -u | grep -i lockview
name 0x1c60dd _lockView
name 0x1c7741 _deviceLockView
name 0x1fe4cf deviceLockView
name 0x204737 SBDeviceLockViewOwner
name 0x20681b SBDeviceLockViewDelegate
name 0x1c0653 deviceLockView
name 0x1c2102 attemptDeviceUnlockWithPassword:lockViewOwner:
name 0x1c2281 unlockFromSource:playSound:lockViewOwner:
name 0x1c22ab unlockWithSound:lockViewOwner:
name 0x1c71df _zoomInDeviceLockViewWithDelay:
name 0x1c7362 _shouldZoomDeviceLockView
name 0x1c73f8 _zoomOutDeviceLockViewWithDelay:
name 0x1c75c5 deviceLockViewEmergencyCallButtonPressed:
name 0x1c75ef deviceLockViewCancelButtonPressed:
name 0x1c7612 deviceLockViewPasscodeEntered:
name 0x1c7631 deviceLockViewPasscodeDidChange:
name 0x1c7652 deviceLockViewWillAnimateMaximization:
name 0x1c7679 deviceLockViewWillAnimateMinimization:
name 0x1d3c44 initWithFrame:deviceLockView:
name 0x205460 SBDeviceLockViewWithKeyboard
name 0x20547d SBDeviceLockViewWithKeyboardPhone
name 0x20549f SBDeviceLockViewWithKeyboardWillCant
name 0x2054c3 SBDeviceLockViewWithKeypad
name 0x2054de SBDeviceLockViewWithKeypadWillCant
name 0x205500 SBDeviceLockViewWithKeypadPhone
name 0x205578 SBDeviceLockView
coreDumps-MacBook-Pro-2:SpringBoard.app _coredump$
  
```

מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS עם iNalyzer

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



Cycript. אנחנו יכולים לראות שהסלקטור שבחרנו ממומש בארבעה אובייקטים ובהם ב-SBDeviceLockViewDelegate SBSlidingAlertDisplay וכו'.

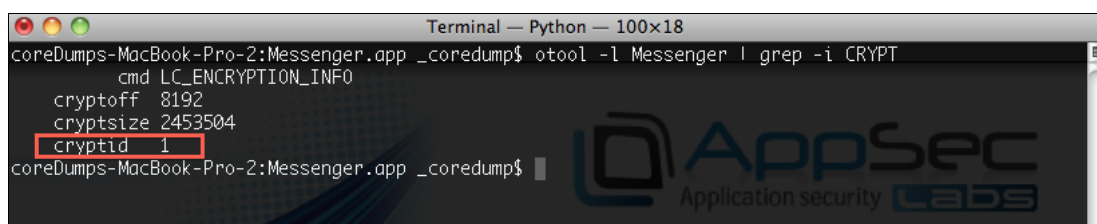
אם כך אנחנו יכולים להשתמש ב-Cycript בכדי לשחק עם האפליקציה מתוך הכרות יותר מעמיקה של המתודות והאובייקטים שלה. זאת אומרת שאנחנו נתעסק עם האפליקציה במקום עם התקשורת - ולסמוך עליה שהתעבורה תצא באופן שאנחנו רוצים.

כל זה מדבר על מקרה שבו האפליקציה אינה מוצפנת, אך הקבצים שמגיעים מה-AppStore מוצפנים - ולכן חשוב שנדבר קצת על התהליך כאשר מדובר באפליקציה מוצפנת על ידי iOS.

## עצור! סימא!

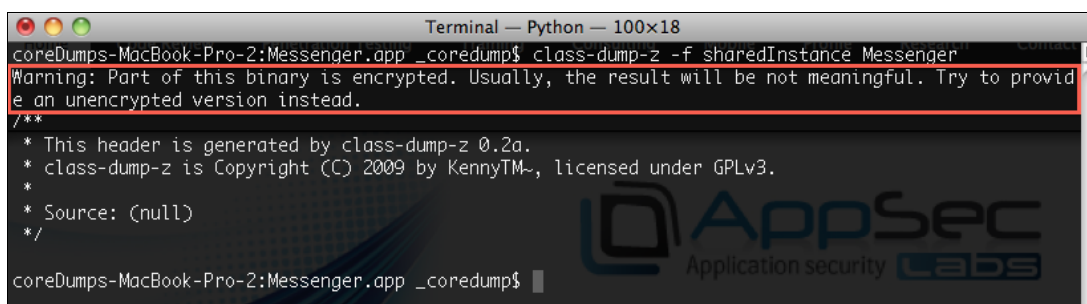
סקירה קצרה של תהליך ההצפנה: תהליך ההורדה של אפליקציות מה-AppStore כולל בתוכו הצפנה של קובץ המאקו של האפליקציה על ידי השרת של אפל. ההצפנה מתבצעת עם מפתחות פרטיים של המכשיר, דבר המונע באופן תיאורתי ממשתמש להריץ אפליקציות ממכשיר אחר. האפליקציות לא יעבדו מאחר והמפתחות לא תואמים את מפתחות ההצפנה ששיכות למכשיר שהוריד את האפליקציה.

ניתן בקלות לראות האם הקובץ מאקו מוצפן על ידי otool:



```
Terminal — Python — 100x18
coreDumps-MacBook-Pro-2:Messenger.app _coredump$ otool -l Messenger | grep -i CRYPT
cmd LC_ENCRYPTION_INFO
cryptoff 8192
cryptsize 2453504
cryptid 1
coreDumps-MacBook-Pro-2:Messenger.app _coredump$
```

בדוגמא לעיל אנו רואים כי הדגל cryptid דולק ולכן אנו יודעים כי קובץ מאקו זה מוצפן, אם נבקש מה-class-dump-z להציג את המחלקות השונות נקבל אזהרה כמו זו המלווה בפלט חסר:



```
Terminal — Python — 100x18
coreDumps-MacBook-Pro-2:Messenger.app _coredump$ class-dump-z -f sharedInstance Messenger
Warning: Part of this binary is encrypted. Usually, the result will be not meaningful. Try to provide
an unencrypted version instead.
/**
 * This header is generated by class-dump-z 0.2a.
 * class-dump-z is Copyright (C) 2009 by KennyTM-, licensed under GPLv3.
 *
 * Source: (null)
 */
coreDumps-MacBook-Pro-2:Messenger.app _coredump$
```

class-dump-z מציין בפנינו כי הקובץ מוצפן ולכן אין באפשרותו לשלוף ממנו מידע, הוא מציע להשתמש בקובץ שאינו מוצפן.

כדי להתגבר על המכשול הזה אנו נזקקים להבנה קצרה של תהליך הפענוח: בזמן הפעלת האפליקציה,

מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS עם iNalyzer

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

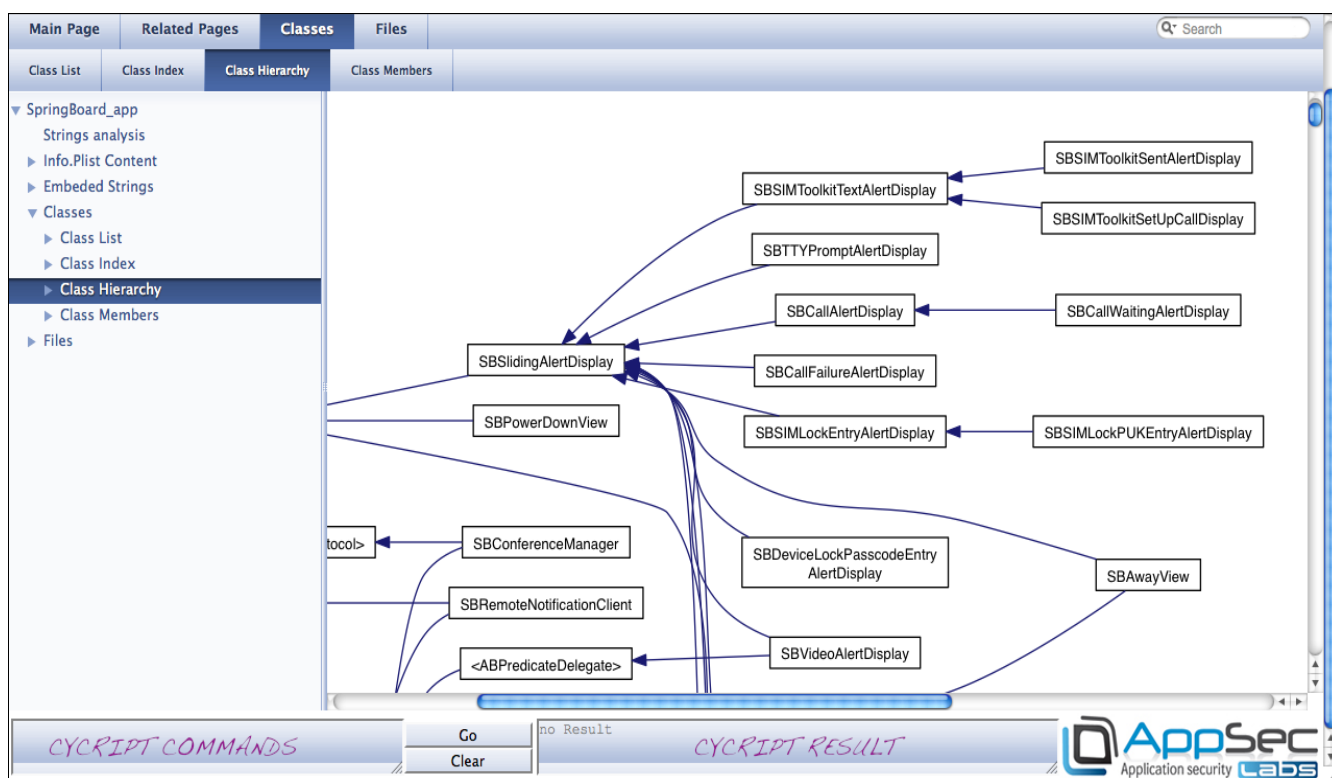
המערכת טוענת את המפתחות ומבצעת פענוח של הסגמנט המוצפן בתוך קובץ המאקו (mach-o) שהורד מה-AppStore. לאחר שהפענוח הושלם האפליקציה מתחילה לרוץ.

זאת אומרת שהאפליקציה נמצאת במצב מפוענח שניה לפני שהיא מתחילה לרוץ, אם נוכל להתחבר אל האפליקציה עם gdb נוכל לקבוע bp על כתובת התחלתית וממנה לבצע dump לזיכרון של אותו מקטע. ואז אנחנו יכולים לערוך את קובץ המאקו כך שהוא יכיל את הגרסא הלא מוצפנת שנזרקה מהזכרון ואז נוכל לקרוא ל-class-dump-z שיעשה את הקסמים שלו.

## ה-iNalyzer!

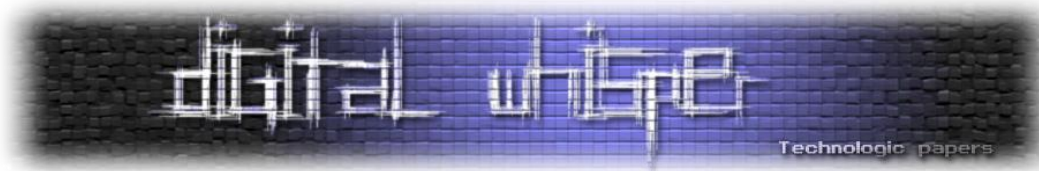
ה-iNalyzer מהווה סביבה מיוחדת לבדיקות של מערכות מבוססות iOS. הוא אוסף את כל הנתונים מתוך קובץ המערכת ומתוך class-dump-z ואז מחולל ממשק Command&Control (מבוסס דוקסיג'ן) עבור Ccrypt, כך שקיבלנו ממשק של סביבת בדיקות מלאה.

כמה מהיתרונות של [iNalyzer](#): הפעלה אוטומטית של class-dump-z, איסוף של כל המידע שאנחנו צריכים לטובת הבדיקות, כלי אחד שיעשה את כל העבודה השחורה ויתן לנו להתעסק רק עם המערכת ו-Ccrypt בלי חתימות, בלי משחקים; דרך אוטומטית לבצע פענוח dump:



מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS עם iNalyzer

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



הנה מספר דוגמאות למידע שה-iNalyzer מספק:

הצגת קישורים חיצוניים, לטובת מיפוי נקודות התממשקות מול שרתים חיצוניים:

Main Page	Related Pages	Classes	Files
▼ Messenger_app		20 23233	http://login.facebook.com
Strings analysis		21 23234	http://m.facebook.com/profile.php?id=%@
▶ Info.Plist Content		22 23235	http://maps.google.com/maps?daddr=%@
▶ Embedded Strings		23 23236	http://maps.google.com/maps?ll=%@
▶ Classes		24 23237	http://maps.google.com/maps?q=%@
▶ Files		25 23238	http://maps.google.com/maps?saddr=%@&daddr=%@
		26 23239	http://www.apple.com/
		27 23240	http://www.youtube.com
		28 23248	https://
		29 23249	https://api.facebook.com/method/
		30 23250	https://graph.facebook.com/
		31 23251	https://m.facebook.com/a/faceweb_exception_log.php
		32 23252	https://m.facebook.com/dialog/
		33 23253	https://m.facebook.com/mobile/messenger/help?locale=%@
		34 23254	https://m.facebook.com/r.php?locale=%@&cid=%@
		35 23255	https://s-external.ak.fbcdn.net/safe_image.php
		36 23256	https://www.apple.com/appleca/0

Go no Result  
Clear

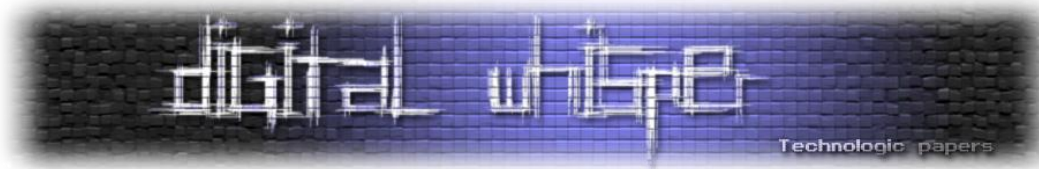
AppSec Application security

הצגת ממשקי URI שבשימוש, לטובת מיפוי הפעלות חיצוניות של אפליקציות אחרות והזרקות:

Main Page	Related Pages	Classes	Files
▼ SpringBoard_app			0 20019 text field:initWithSelectionFromCharacterRangeToCharacterRange.
Strings analysis			<b>URI strings</b>
▶ Info.Plist Content		1 19835	doubletap://com.apple.camera
▶ Embedded Strings		2 19836	doubletap://com.apple.mobilephone?view=FAVORITES
▶ Classes		3 19837	doubletap://com.apple.mobileslideshow-Camera
▶ Files		4 19838	doubletap://com.apple.springboard-Search
		5 20063	facetime-lock://
		6 20065	facetime-show://
		7 21684	http://itunes.apple.com/us/app/ibooks/id364709193?mt=8
		8 22460	itms://?action=music
		9 23114	music://playImmediately
		10 23615	photos-event://?uicmd=show-import
		11 23935	radr://5614542
		12 25995	telemergency://
		13 25997	tellock://
		14 25999	telshow://
		15 26788	x-web-search:///?%@
		16 26789	x-web-search://wikipedia/?%@

מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS עם iNalyzer

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



הצגת משפטי sql שבשימוש, לטובת ניתוח פגיעויות של הזרקות מקומיות ומרוחקות:

```

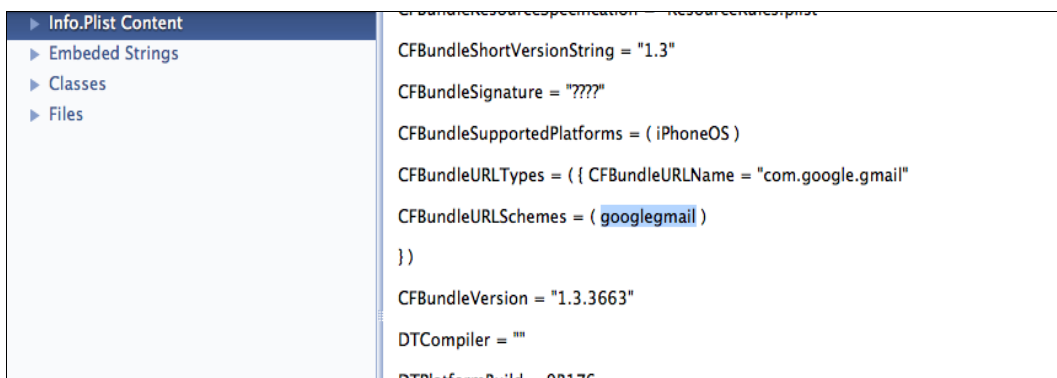
Strings analysis

Analysis of Strings found in the executable

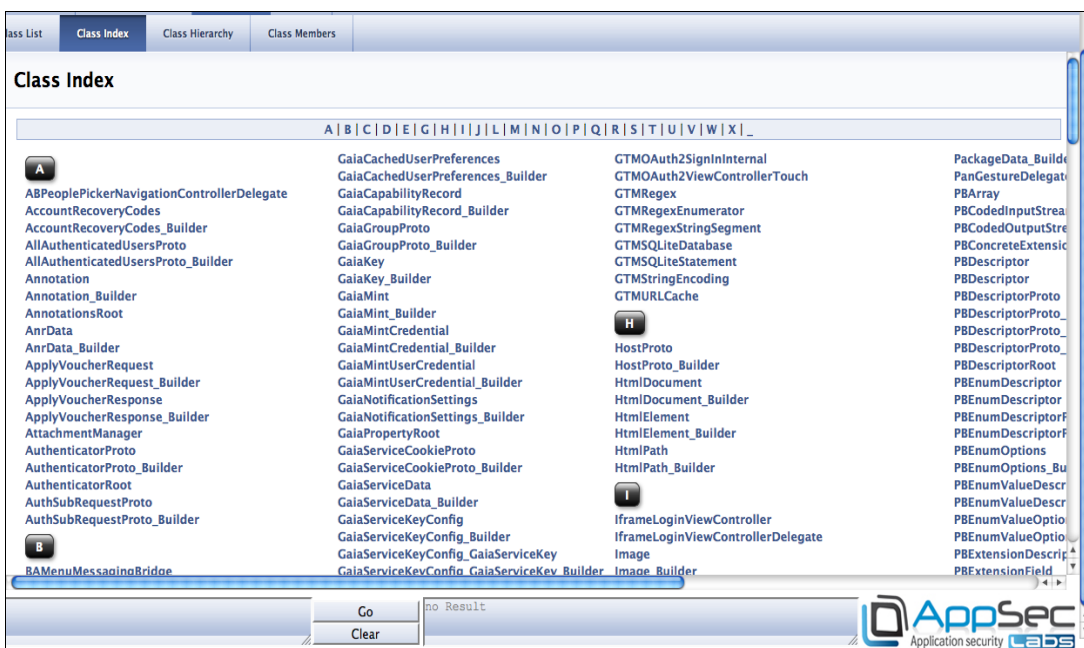
SQL Strings

1 11387 SELECT EXISTS ( SELECT 1 FROM 's' WHERE text=?);
2 11388 SELECT data FROM tiles_table WHERE id=?;
3 11389 SELECT data, storage_type, path FROM 's' WHERE text=?;
4 11390 SELECT storage_type, path, text_type FROM 's' WHERE text=?;
5 6810 DELETE FROM 's' WHERE text=?;
6 8849 INSERT OR REPLACE INTO 's' values (?,?,?,?,?);
7 8850 INSERT OR REPLACE INTO tiles_table values (d,?);
    
```

ממשקי CFURL אשר נרשמים לטובת המערכת ומהווים נקודות הפעלה נוספות, ופתח נוח למתקפות:

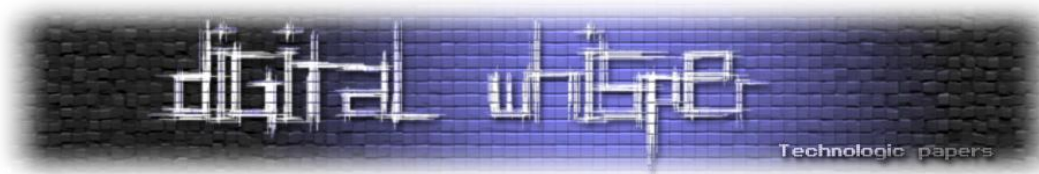


הצגת כל האובייקטים של המערכת, לטובת חשיפת פונקציונאליות בעייתית או עודפת:



מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS עם iNalyzer

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



### הצגת כל המתודות של המערכת, לטובת הפעלה ישירה שלהם על ידי Cypcript:

The screenshot displays the 'Class Members' view for the 'GmailHybrid\_app' class. The 'Functions' sub-tab is active, showing a list of methods including: accountChanged, accountsCount, accountsHost, accountWithEmail, actionSheet:clickedButtonAtIndex, actionSheet:didDismissWithButtonIndex, actionSheet:willDismissWithButtonIndex, actionSheetCancel, addAccount, addAllBoolValues, addAllDataValues, addAllDoubleValues, addAllEnumValues, addAllFixed32Values, addAllFixed64Values, addAllFloatValues, addAllGroupValues, addAllInt32Values, addAllInt64Values, addAllMessageValues, addAllSFixed32Values, addAllSFixed64Values, and addAllSInt32Values. A search bar at the bottom indicates 'no Result'.

### הצגת כל המשתנים של המערכת, לטובת התקפות טימפור (Tampering):

The screenshot displays the 'Class Members' view for the 'GmailHybrid\_app' class. The 'Variables' sub-tab is active, showing a list of variables including: \_capacity, \_count, \_data, \_field1, \_field2, \_field3, \_field4, \_field5, \_field6, \_field7, \_field8, \_field9, \_field10, \_field11, \_field12, \_field13, \_field14, \_field15, \_field16, \_field17, \_field18, \_field19, and \_field20. A search bar at the bottom indicates 'no Result'.

מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS עם iNalyzer

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



הצגת כל המאפיינים של המערכת, לטובת התקפות / Injecting / Tampering

Class Members

- accessToken : GTMOAuth2Authentication
- account : GmailNotificationSettingsChange
- addAllValuesSel : PBFieldDescriptor
- additionalAuthorizationParameters : GTMOAuth2SignIn
- additionalTokenRequestParameters : GTMOAuth2Authentication
- addValueSel : PBFieldDescriptor
- allowInsecureAuthorization : TOPAuthManager
- allowRTLLayout : GmailNativePlusNavigationItem
- appDisplayName : GIPFeedback
- applicationName : GIPNativeURL
- appName : GIPCrashReportController , GIPCrashReportData , GIPFeedbackCollectedData
- appVersion : TOPUpgradeNotifications , GIPCrashReportController , GIPCrashReportData
- arrowState : MoreMenuButton
- assertion : GTMOAuth2Authentication
- attachLogs : GIPCrashReportController
- authentication : TOPAuthManager , GTMOAuth2ViewControllerTouch , GTMOAuth2SignIn
- authManager : GmailAuthenticator
- authorizationEmail : GTMOAuth2SignInInternal
- authorizationQueue : GTMOAuth2Authentication
- authorizationTemplate : GTMOAuth2SignInInternal
- authorizationURL : GTMOAuth2SignIn
- authorizer : GTMHTTPFetcher , GTMHTTPFetcherService

הצגת כל המחזורות המופיעות בקובץ המאקו, לטובת ניתוח מפגעי זליגת מידע רגיש

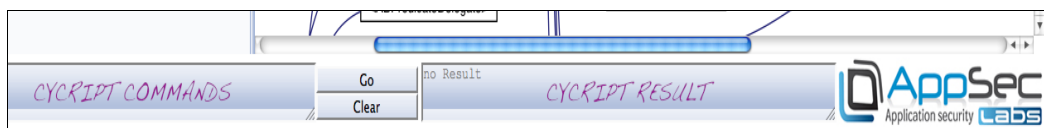
Embedded Strings

- 10721 Lydian
- 10722 L{}
- 10723 M758 (?:28[4-7]|384|4(?:6[01]|8[4-9])|5(?:1[89]|20[84])|7(?:1[2-9]|2[0-4])\d{4})
- 10724 MFMailComposeViewControllerDelegate
- 10725 MKAnnotation
- 10726 MKCircle
- 10727 MKMapViewDelegate
- 10728 MKReverseGeocoder
- 10729 MMLMDMMhMDMoo<<<<<
- 10730 MMDMd
- 10731 MMDMdyyyy
- 10732 MMDd
- 10733 MMDdjmm
- 10734 MMD Z
- 10735 MMDdyyyy
- 10736 MMDdyyyyEEEjmm
- 10737 MMDdyyyyjmm
- 10738 MQIsdp
- 10739 MQTT Connected: %@
- 10740 MQTTClientManagerConnectedChanged
- 10741 MQTTListener<@> %@ once only %d message %@ timeout %f timer %@ timeout block %@
- 10742 MQTTManager
- 10743 MQTTMessageSender
- 10744 MQTTPublisher<@> %@ success %@ failure %@ timeout %f timeoutBlock %@
- 10745 MQTT\_RECV
- 10746 MQTT\_SEND
- 10747 MXP4Z
- 10748 MYP<Z
- 10749 M'xD
- 10750 Main Panel
- 10751 Malayalam
- 10752 Malformed repeat
- 10753 Managed Context save failure! This is CATASTROPHIC! Do not ignore this, look at it or get someone to look at it!
- 10754 Managed Object Thread Violation! YOU CANNOT IGNORE THIS!!!
- 10755 Mandaic
- 10756 MapViewController
- 10757 Mark as Unread
- 10758 MarkSeenMethod
- 10759 MarkThreadMethod
- 10760 Market
- 10761 Match

מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS עם nalyzer

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)

בנוסף הממשק מכיל גם סביבת הפעלה של Cycrypt ישירות למכשיר, כך שאין צורך לפתוח SSH ולעבוד מטרמינל:



ה-iNalyzer מאפשר לי להפסיק ולהתייחס לבדיקות אבטחת מידע על מערכות iOS כבדיקות Black-Box, קבלת כלל האינפורמציה שניתן לקבל, בצורה נוחה בתוך ממשק בדיקות ידידותי. כעת במקום להשתמש בפרוקסי לביצוע התקפות כמו מערכות רשת רגילות אני הופך את האפליקציה עצמה לחוד החנית בתהליך הבידוק, ואז מבנה סביבת הבידוק שלנו נראית כמו בתרשים הבא:

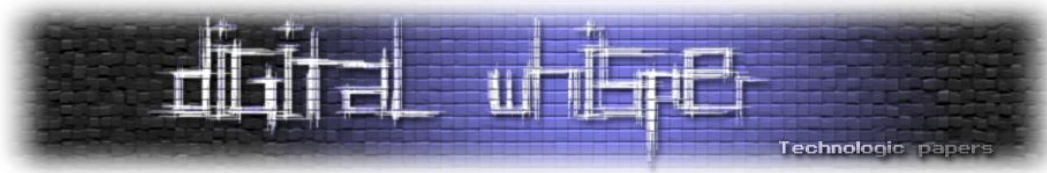


## לסיכום

במאמר זה ניסינו לדרוס כמה שיותר נושאים הקשורים לסביבת הבידוק של אפליקציות iOS, לא כיסינו את רוב הנושאים בצורה מעמיקה, אבל אני מאמין גדול ביכולת שלכם ללמוד ולהתקדם בקצב שלכם. הרעיון המרכזי היה להכיר לכם את שחקני המפתח בתהליך הזה, כמו גם להציג חלק מהמתודולוגיה שהוא כולל. בנוסף הצגתי בפניכם את ה-iNalyzer כמערכת מתקדמת לבדיקות שכאלו והדגמתי מספר מיתרונותיו.

מבוא לניתוח אבטחת מידע מתקדם באפליקציות iOS עם iNalyzer

[www.DigitalWhisper.co.il](http://www.DigitalWhisper.co.il)



אני אשמח מאוד אם תמצאו את ה-iNalyzer מסייע בתהליך הניתוח, ואשמח עוד יותר אם תחליטו לקסטם (customize) אותו בהתאם לצרכים שלכם. זאת הסיבה שהוא מופץ ככלי חינוכי ותחת קוד פתוח. [תוכלו להוריד אותו מהאתר שלו](#) ולמצוא בו עוד עדכונים וסרטונים. כל המאמר ניתן בחינם לטובת הקהילה וכולי תקווה שתמצאו אותו מועיל ומעניין, אם תרצו תוכלו לפנות אליי בשאלות ובפידפקים ([chilik@appsec-labs.com](mailto:chilik@appsec-labs.com)).

## על המחבר

המחבר הינו מומחה אבטחת מידע בעל ניסיון של יותר משתי עשורים במחקר, פיתוח, בדיקות, ליווי והדרכות בתחומי אבטחת המידע האפליקטיבית ללקוחות פיננסיים, גורמי ביטחון, משרדי ממשלה ותאגידים. בין פרסומיו הקודמים ניתן לציין את [AppUse](#) - סביבת בדיקות לאפליקציות אנדרואיד שפותח יחד עם ארז מטולה, בלץ' ([belch](#)) - כלי לאוטומציה של ניתוח ובדיקות פרוטוקולים בינאריים כגון flex ו-Java-Serializtion, וכן את הרצאותיו בכנסים בארץ כגון [OWASP IL 2011](#) ו-[OWASP IL 2012](#), כיום משמש כמדען ראשי בחברת AppSec-Labs.com ומרכז בה את תחום המחקר והחדשנות.

חיליק טמיר

מדען ראשי, [Appsec-labs.com](http://Appsec-labs.com)