Technologic Papers

Key-Logger, Video, Mouse - חלק ד': תקווה חדשה

מאת ליאור אופנהיים ויניב בלמס

הקדמה

ברוכים הבאים לחלק 0x04 (ואחרון) בסדרת מאמרי ה-KVM שלנו. בגלל שאתם בטח סקרנים ולא ישנתם בלילות מציפייה אליו, נצלול ישר לעניינים.

בסוף המאמר הקודם הצלחנו לפענח את הצופן (למרות שאנחנו כלל לא בטוחים שהדבר הזה אמור להקרא "צופן") של ה-BLOB שמכיל את עדכון ה-FIRMWARE. כעת, כל שנותר לנו לעשות על מנת לממש את התוכנית הזדונית שלנו להשתיל Key-Looger הוא להבין כיצד ה-KVM פועל.

עקרונית זה לא אמור להיות מסובך, ה-BLOB הוא סה"כ בגודל 64 קילובייט, מה שנשמע יחסית מעט קוד לעבור עליו, אבל מכיוון שרב האופקודים באסמבלי 8051 הם בית אחד, אז מדובר בכמות קוד לא קטנה (לעזאזל, ארכיטקטורה יעילה בזכרון שכמוך!). לכן, החלטנו להתמקד בשתי שאלות מרכזיות:

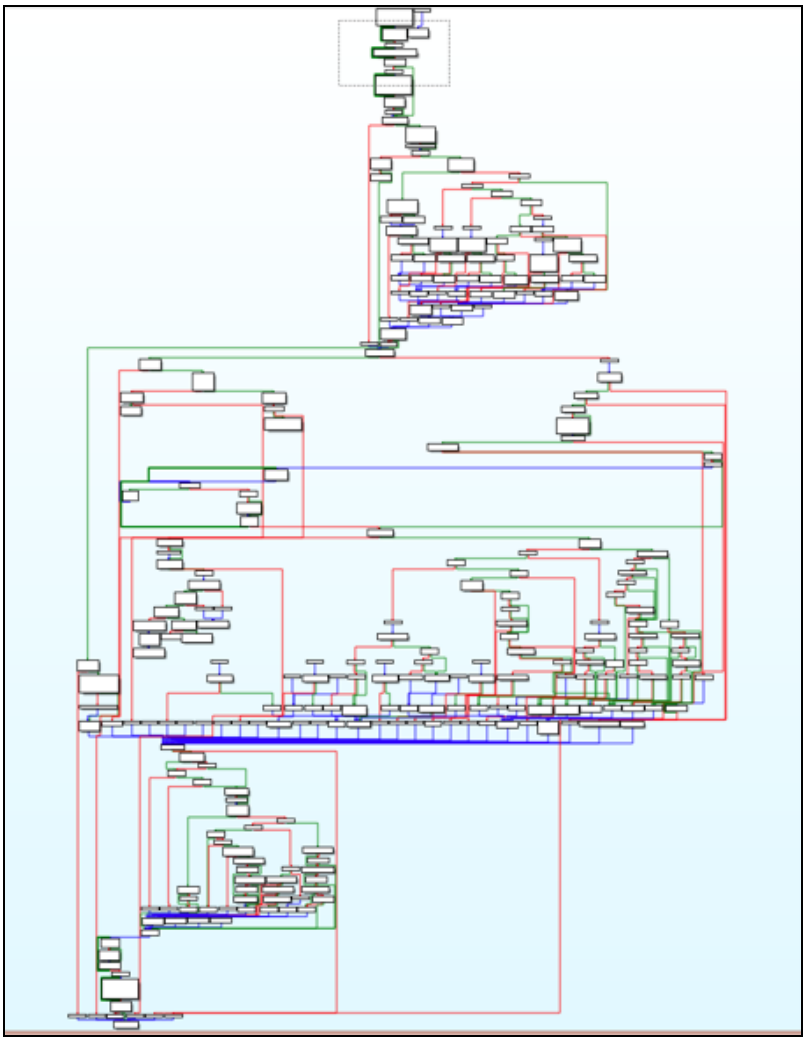
- איפה בזכרון של ה-KVM אפשר למצוא את ה-Keystrokes שהגיעו מהמקלדת?
- איך ניתן להכניס קטע קוד שלנו שיוכל לרוץ באופן קבוע, ובלי להפריע למהלך הרגיל של ה-KVM?

השאלה השנייה נפתרה די מהר, בסוף ה-BLOB, לפני אותה שמיניית בתים מפורסמת, יש "אזור מת" שבו ניתן לשים קוד משלנו, ואז כל שנותר לעשות הוא לערוך את הקוד המקורי כך שיקפוץ אל הקוד שלנו בזמן המתאים.

לגבי השאלה הראשונה, זה קצת יותר מסובך... אין ברירה אלא להפשיל שרוולים ולצלול לקוד.

IDA, אני בוחר בך!

לאחר נבירה קצרה בקוד, הגענו לפונקציה שנראית כך:



זו היא אחת מהפונקציות המרכזיות של ה-FIRMWARE שלמעשה רצה בלולאה אינסופית וכפי הנראה מטפלת ברוב הפונקציונאליות המעניינת של ה-KVM. אז כעת, כל שנותר לעשות הוא לחפש איזה חלק בפונקציה אחראי על ההאזנה ל-Keystrokes.

לצערנו, מסתבר שלעשות RE ל-Embedded זה לא ממש טיול בפארק.

הבעיה המרכזית שנתקלנו בה היא שלא הייתה לנו קרקע יציבה לעמוד עליה. הרבה מאוד מה-FLOW של ה-FIREMWARE התבסס על קריאה של ערכים מהזכרון שנכתבו על ידי מקורות חיצוניים (אם אתם זוכרים, במעגל יש עוד 2 צ'יפים גדולים מסוג ASIC שמחוברים גם הם לזכרון). מכיוון שאין לנו מושג מה הערכים האלה אומרים, או מה הפורמט שלהם, אז אנחנו די אבודים בסבך האופקודים.

בד"כ במצב כזה אפשר להעזר ב-Dynamic RE ובעזרת ניסוי וטעייה להבין את המשמעות של הערכים, אבל במקרה שלנו, אין שום פרוטוקול דיבאגינג שיכול לעזור לנו. אולי שווה ליצור אחד?

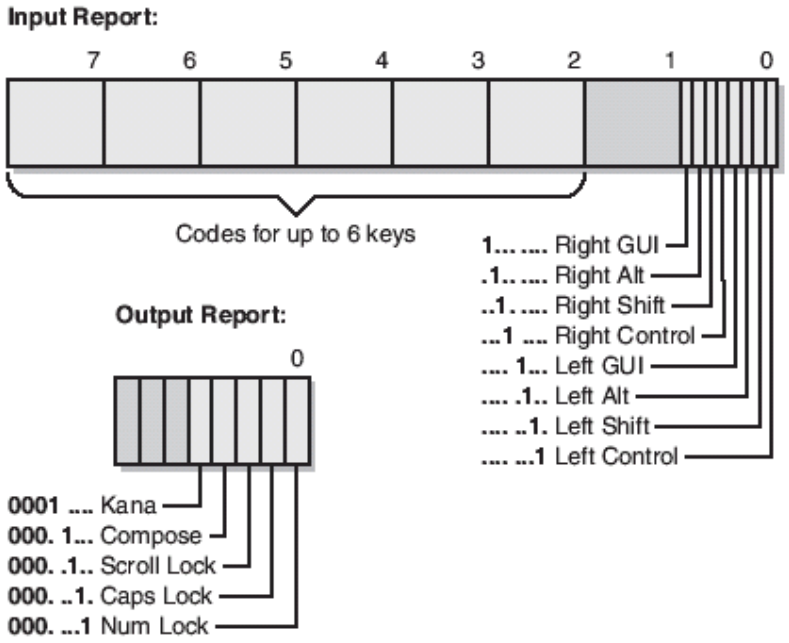
תחשבו על זה, יש לנו דרך להעלות קוד משלנו למכשיר (פשוט לשנות את הקוד ב-BLOB הלא מוצפן ואז לארוז אותו מחדש), ובנוסף, אנחנו יכולים לנצל את הפרוטוקול הסיראלי כדי ליצור ערוץ תקשורת בין המחשב לבין ה-KVM.

באופן זה אפשר לבנות CUSTOM KVM DEBUGGER משלנו! איך זה עובד? בכל פעם בוחרים מספר נקודות מעניינות בקוד, ומחליפים את האופקוד המקורי ב"קפיצה" לפונקציית ה-DEBUGGING שלנו (שנמצאת ב-"אזור המת" של ה-BLOB) ומעדכנים את ה-KVM עם ה- FIRMWARE הערוך.

הפונקציה משתמשת בערוץ הסיראלי על מנת לתקשר עם ה-DEBUGGER שנמצא על המחשב, שיכול לתת לה פקודות בסיסיות כמו: קריאה וכתיבה לזכרון, קריאה ושינוי רגיסטרים, המשכת הריצה וכו'.

חמושים ב- KVM DEBUGGER, חזרנו לתקוף את הקוד. כעת, כשאנחנו מסוגלים לדגום את הזכרון ולהבין איך ה-FLOW של הקוד מתנהג בכל מיני מצבים, החלה להתבהר התמונה הכללית - הקושחה היא למעשה Man-In-The-Middle ל-"USB HID Reports".

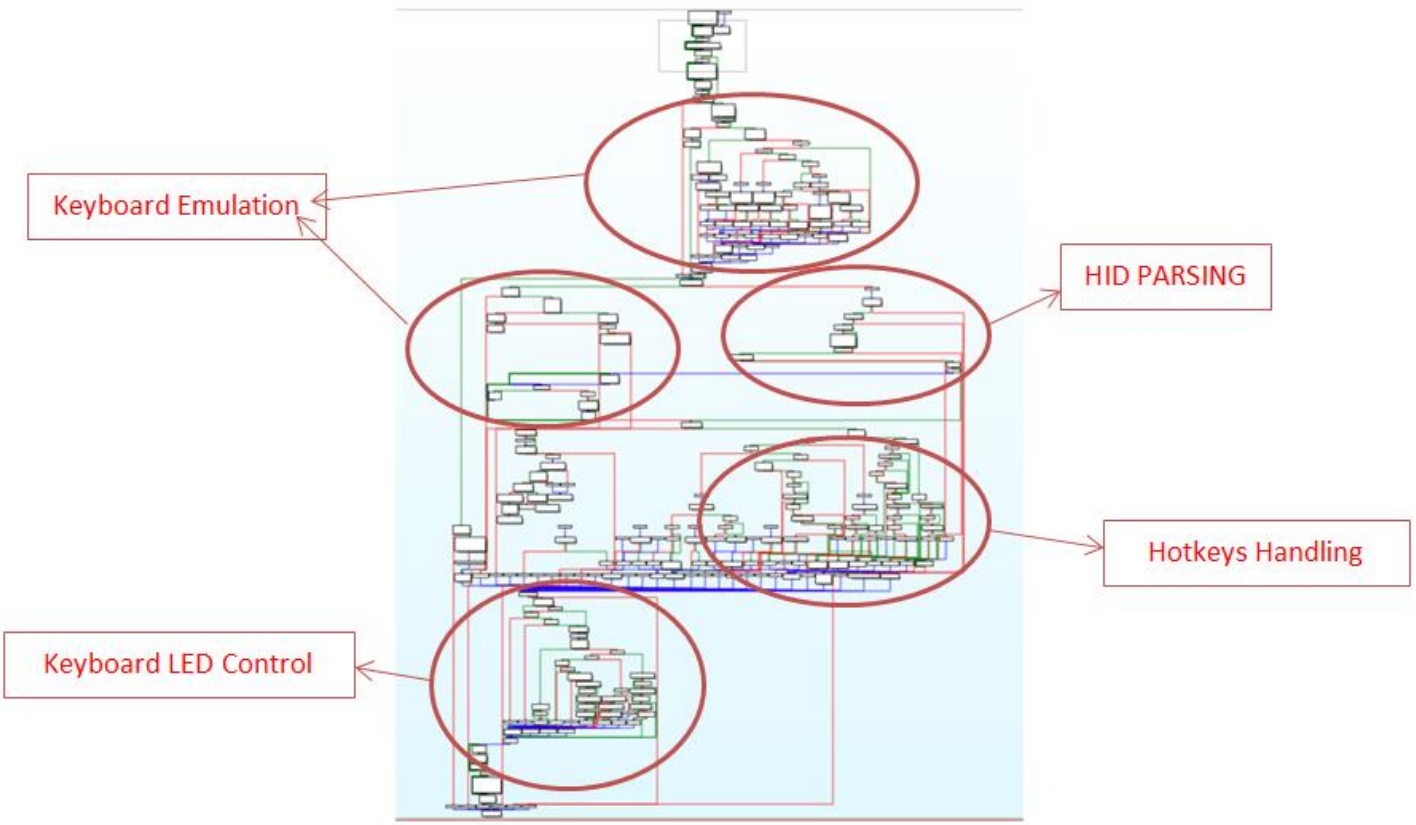
מה זה HID? שאלה טובה. HID זה ראשי תיבות של Human Interface Device, ולמעשה זו היא תת מחלקה פרוטוקול ה-USB שמאגדת את פרוטוקולי התקשורת בין המחשב לדברים כמו: עכבר, מקלדת, גו'יסטיק וכו' (אתם יודעים, מכשירים עם ממשק לבן אנוש). נתמקד בהודעות הקשורות למקלדת: InputReport ו-Output Report:



הודעות HID מהמקלדת למחשב (Input Report) מכילות בית אחת אשר מכיל את התו הלחוץ (בקידוד מיוחד של HID) וModifiers כגון: alt,shift,control. הודעות מהמחשב למקלדת (Output Report) נשלחות מהמחשב למקלדת ומכילות מידע על אילו LED-ים צריכים להיות דלוקים (במילים אחרות, המקלדת לא שולטת על ה-LED-ים שלה באופן עצמאי, אלא מקבלת הוראות לכך מהמחשב). הפרוטוקול תומך בעד חמישה LED-ים שונים, למרות שברב המקלדות יש רק שלושה CapsLock,ScrLock,NumLock (הידעתם? בחלק מהמקלדות ביפן יש LED רביעי שנקרא KANA).

נקודה מעניינת: כחלק מהנסיונות שלנו לפתוח את הקידוד של ה-BLOB ניסינו לראות אם על הפעלת כל מיני פעולות מתמטיות על המידע אנחנו נקבל שכיחות גבוהה של מחרוזות UNICODE\ASCII ב-BLOB. מסתבר שטעינו בקידוד! ה-BLOB היה מלא במחרוזות, אבל מסוג HID. טוב, אין כמו חוכמה שלאחר מעשה.

ובכן, עכשיו שאנחנו מבינים שה-KVM נגיש להודעות ה-HID שעוברות בין המחשב למקלדת, הבה נחזור לפונקציה המרכזית (שרוברסה למשעי, אני חייב להודות):



נעבור בקצרה על החלקים המרכזיים בה:

- HID Parsing: החלק הזה הופך את כל קלט מהמקלדת, שמגיע בפורמט HID, לפורמט ASCII רגיל.
- Hotkeys Handling: בודק האם ה-Keystroke שנקלט קשור ל-hotkey של ה-KVM ומבצע אותו (נגיד ScrLock+ScrLock+2 מעביר אוטומאטית את ה-KVM למחשב בפורט 2).
- Keyboard LEDs Control: במקרים מסויימים ה-KVM משנה את הקונפיגורציה של ה-LED-ים של המקלדת בעצמו, על ידי שליחה של ה-Output Report המתאים, לדוגמא כאשר עוברים מפורט אחד לאחר וה-KVM רוצה לשחזר את ה-STATE הנכון של מצב המקלדת לאותו פורט.
- Keyboard Emulation: ה-KVM יכול להתחזות למקלדת ולהקליד בעצמו תווים לבחירתו

רגע?! מה?! ה-KVM מסוגל, בלי שום קשר למקלדת, לשלוח Keystrokes משלו למחשב? הרי זה חלומו הרטוב של כל כותב רושעות ל-KVM-ים. מה שזה בעצם אומר זה שה-Keylogger שלנו יכול לא רק להאזין לקלט מהמקלדת, אלא גם "לתקתק" כל מיני פקודות למחשב. בנוסף, הפונקציונאליות הזו מאפשרת ל-KVM לשלוח מידע לרושעה שיושבת על המחשב עצמו (על ידי שימוש ב-USB SNIFFER).

ומה לגבי הערוץ חזור? האם הרושעה על המחשב יכולה להעביר מידע לרושעה שעל ה-KVM? אז מסתבר שכן, משום שה-KVM נגיש להודעות ה-Output Report שנשלחות מהמחשב למקלדת (זוכרים? אלו עם הLEDים). על גבי האיתותים האלה אפשר להעביר מידע, ואם אתם דואגים שהמשתמש יראה שהLED-ים על המקלדת שלו מהבהים בצורה חשודה, אז אפשר להשתמש בל-LED-ים הרביעי והחמישי בלבד להעברת המידע (רק צריך להזהר עם מחשבים ביפן ☺).

ובכן, בקווים כלליים, זו תוכנית הפעולה שרקמנו:

- באורח פלא, ה- KVM מודבק עם גרסת ה-Firmware המרושעת שלנו .
- אם המחשב מחובר לאינטרנט, הרושעה "מקלידה" לתוך המחשב פקודה אשר מוריד ומריצה את הוירוס שלנו.
- כעת, הוירוס שעל המחשב יכול להתפשט באמצעות ה- KVM גם למחשבים שאינם מחוברים לאינטרנט, על ידי הקלדה של תוכן הוירוס למחשב החדש (בניסוי שלנו, הוירוס "הקליד" את התוכן של עצמו כ-BASE64, ולאחר מכן הפך את ה-BASE64 לבינארי באמצעות certutil).
- לאחר שהודבקו כל המחשבים שמחוברים ל-KVM, הוירוסים יכולים לתקשר אחד עם השני באמצעות ה-KVM, ולהדליף מידע ממחשב פנימי למחשב אינטרנטי ומשם לשרת שלנו.

אני מקווה שהנקודה ברורה. עצם חיבור ה-KVM לשני מחשבים מאפשר לנו "לדלג" מהאחד לשני, גם אם המחשבים נמצאים ברשתות מבודדות לחלוטין.

וכיצד, אתם שואלים, ניתן להדביק שלא באורח פלא את ה-KVM? קיימות מספר אפשרויות:

- תן לי 30 שניות בפרטיות עם ה-KVM שלך, ואני מעדכן לו את הקושחה ("Evil Maid")
- תרחיש מוגזם במעט, אבל עקרונית אפשר לתקוף גם את מערך האספקה של המכשיר, ולהדביק אותו באחד מהשלבים ("Interdiction").
- ל-KVM-ים מדגמים מתקדמים יותר מזה שחקרנו יש אפשרות לעדכן את הקושחה מהרשת . נצחון קל!
- אמנם לא מצאנו אחת כזו בדגם שלנו, אבל לפי האינטרנט KVMים אינם חסינים לחולשות

מה ניתן לעשות על מנת למנוע תקיפה מהסוג הזה? אז מסתבר שבשוק ה-KVM-ים יש דגמים "מאובטחים" שפחות או יותר מתקנים את כל הנקודות החלשות שתקפנו (אין עדכוני גרסא, הפרדה פיזית בין מעגלים של פורטים שונים ועוד). מה שכן, החבר'ה האלה עולים פי חמש מהדגמים הלא מאובטחים.

בנוסף, חשבנו על פתרון תוכנתי שיכול להקשות במידת מה על תקיפות מן הסוג הזה. הקונספט הוא לכתוב service שיסרוק את ההקשות מקלדת ויתריע כאשר הוא מזהה דפוסי הקלדה שמקורם ברובוט ולא באדם אנושי (נגיד, הפרשי זמן קבועים בין תו לתו). מספיק שהסורק יקפיץ MessageBox בכדי לשבש את פעולת ה-Malware שמוטמע ב-KVM. תודה לדרור רפפורט שעזר לנו בחלק זה של המחקר והשמיש את הרעיון.

סיכום

זהו, עד כאן להפעם. מקווים שנהנתם מהמסע, ושלמדתם דבר או שניים על עולמם המופלא של ה-KVM-ים. את המאמר ניתן גם למצוא בפורמט של "הרצאה ב-DEFCON" פה.

ואגב, כל המחקר הנ"ל נעשה במסגרת קבוצת Malware & Vulnerability של צ'קפוינט. אם גם אתם אוהבים לעשות מחקרי אבטחה מגניבים (ולקבל על זה כסף!), אתם יותר ממוזמנים לשלוח קורות חיים ל-yanivb@checkpoint.com