# VIRTUALIZATION

## HANDS ON

השתדלו לחפש ולדעת להתמצא כמה שיותר ב **intel manual** - יש שם הכל !
לאחר שאתם מסיימים כל תרגיל - תעברו על הפתרונות כדי לראות אם יש דברים שלא חשבתם עליהם.

שימו לב שהמסמך מכסה את ה core של וירטואליזציה, ויש עוד מה הרבה ללמוד, מוזמנים לקרוא בזמן הפנוי:

- Nested Virtualization - מה קורה כאשר VM רוצה להריץ VM.
- Virtualization Detection - שיטות לזיהוי ריצה בתוך VM, למשל Red Pill \ No Pill \ ScoopyNG.
- VM escape - סה"כ מנגנון מסובך, CVE-2008-0923, CVE-2019-0964 וכו' וכו'.
- Bluepill, Vitriol, SubVirt - VM-based Rootkits (VMBR) וכו'.

### VMX FOR BOTS

כתוב תכנית C ב UM שבודקת האם המעבד תומך ב VMX, בין היתר הצג גם את יצרן המעבד.
האם יש דרך יותר קלה לבדוק במה המעבד שלי תומך ?

### BPKNOCK

ברצוננו לממש מנגנון תקשורת client-server בין החלק ה UM של הוירוס שלנו ב VM לחלק ה Hypervisor-י, התקשורת תתבצע כך שה UM יוכל לשלוח MAGIC בגודל DWORD ל hypervisor אשר יענה לו ב DWORD חזרה - חשוב על דרך לא טרוויואלית למימוש התקשורת שתמזער את הסיכויים לתפוס אותנו.
לעת עתה, ממש את הרעיון שלך רק מהצד ה UM-י, כלומר כתוב תכנית C שמקבלת בשורת הפקודה DWORD ומדפיסה את התשובה של ה hypervisor (במידה והיה ממומש).
מה הדרך הלגיטימית לממש את התקשורת הזאת ?

### THE FLOOR IS LAVMM

אנחנו כותבים malware שלא היינו רוצים שייחקר, לכן אנו רוצים שיתנהג שונה במידה והוא רץ בתוך VM.
הצע לפחות 3 שיטות אפשריות לבדיקה האם אתה רץ תחת VMM (בכל ארכיטקטורה שראינו עד כה - גם לפני VT-x).
נסה להציג פתרונות אפשריים ל VMM על מנת למנוע את הזיהוי.

### VMX++

כתוב דרייבר שמבצע את הפעולות הבאות כאשר יוצרים אותו:

1. גורם ל VMX להיות enabled.
2. מדפיס את כל ערכי ה MSRs שקשורים ל VMX (אחד מהם, למשל, הוא IA32_VMX_PINBASED_CTLS)
3. מדפיס את CR0, CR4 - הבן את השדות החשובים.

## HYPERVISOR FROM SCRATCH

מצורף מימוש של VMX hypervisor ב 6 שלבים, כאשר בכל שלב מוסיפים קוד רלוונטי.

1. עבור חלקים 1 עד 4, הבן כל חלק לעומק (מסמך מפורט לכל חלק), טען אותו למכונה וודא שהכל עובד.
2. לאחר שאתה מגיע ל P4, ומסיים להבין אותו - שנה את הקוד בכך שתממש את החלק של השרת עבור BPKNOCK, וודא שאתה מקבל 0x13371337 עבור 0x73317331 כאשר אתה מריץ את תכנית ה UM שלך מתרגיל BPKNOCK.
3. לאחר שאתה מגיע ל P6, שים לב שיש חלקים חסרים בקוד שמהווים חלק ממימוש Hidden Hooks, השלם את החלקים החסרים על מנת לגרום להם לעבוד (מצורף מסמך שמסביר מה החלקים הגדולים בקוד עושים).

## SOLUTIONS & TEMPLATES & KNOWN PROBLEMS & CHILL

בעיות מוכרות שתתקע עליהם המון המון זמן

- לא ניתן להריץ דרייברים 32-bit בסביבת 64-bit, לכן וודא שהמכונה והדרייבר שלך באותו קו (x64)
- ב x64 אין ASM inline, לכן כתבו את החלקים ה UM שלכם לסביבת x86. (ב x64 תשתמשו ב ASM file)
- שימו לב שיש לכם ב VM מעבד וירטואלי יחיד עם ליבה אחת, שסימנתם הכל ב Virtualization engine, בנוסף שנו את ה firmware ל BIOS במקום UEFI.
- אין צורך לטעון דרייבר עם sc, השתמש ב OSRLoader.
- כדי להדפיס דברים מדרייבר ניתן להשתמש ב DbgPrint, ניתן לצפות בעזרת sysinternals\DbgView.
  - וודא ש Capture Kernel, Enable Kernel Verbose Output, Pass-Through, Capture Events מסומנים.
  - פתח בעזרת regedit את
    HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager
    הוסף אליו מפתח Debug Print Filter, ואליו ערך DWORD-י בשם IHVDRIVER עם הערך 0xFFFF.
- מומלץ לעשות shared folder בין התיקייה עם הנתיב קימפול שלכם למכונה (אח"כ תגשו עם \\vmware-host\).
- קמפלו את הבינאריים שלכם עם VS, וודאו שאתם מקמפלים סטטית את הספריות ההכרחיות פנימה.
- קחו snapshot עם מכונה שמקונפגת עם כל הכלים לעיל.
- במידה ואתם רוצים לדבג, השתמשו בדיבוג קרנלי מרוחק וכתבו DbgBreakPoint בקוד שלכם.

**בהצלחה !**