EPT++

בתרגיל הקודם הגענו לנקודה בה אנחנו מסוגלים להריץ את המערכת הקיימת כ VM ולקבל נוטיפיקציות לבחירתנו.
עד עכשיו רק הכנו את המבנה של ה EPTP, ולא השתמשנו בו אף פעם :(, אז כדי לא להעליב אותו נשתמש בו...

APP

כרגיל, נבדוק ש VTX נתמך במעבד שלנו, ניצור את ה device על מנת לטרגר את תהליך האתחול שבעצם יריץ את המחשב שלנו מעל ה Hypervisor, ולבסוף נסגור את ה handle שיגרור את סגירת ה device ולסיום הוירטואליזציה שלנו.

DRIVER

כפי שניתן לראות הקוד השתנה באופן דרסטי, גם מבחינת design וגם מבחינת פונקציונאליות והוספת לוגיקה.
כרגיל, נתחיל מלהסתכל על ה driver, ב DrvCreate קוראים ל HvVMXInitialize, וב DrcClose ל HvTerminateVMX.
לכן, מעתה והלאה נתמקד בפונקציות האלה ובפונקציונליות שנגזרת מהן.

HvVMXInitialize

ראשית, קוראים ל VmxInitializer על מנת לאתחל את מבני הנתונים ההכרחיים. שנית, מקצים VMM Stack ו MSR Bitmap עבור כל מעבד. ולבסוף, קוראים ל HvDpcBroadcastInitializeGuest על מנת להתחיל את תהליך הוירטואליזציה כפי שאנו כבר מכירים (שמירת מצב, VMCLEAR, VMPTRLD, VMCS Update, VMLAUNCH).

VmxInitializer

בודקים האם יש תמיכה ב VMX, האם יש תמיכה ב EPT, בונים את המיפוי של הזיכרון הפיזי בעזרת MTRR וקוראים ל EptLogicalProcessorInitialize. לבסוף, מאפשרים שימוש ב VMX ומקצים VMXON Region + VMCS עבור כל מעבד, שימו לב שמשתמשים בפונקציות API רלוונטיות על מנת למנוע בעיות סנכרון.

EptLogicalProcessorInitialize

מאתחלים את ה page table שלנו בעזרת הפונקציה EptAllocateAndCreateIdentityPageTable, לאחר כן מאתחלים את אוגר ה EPTP, ולבסוף דואגים לשמור את שניהם במשתנה הגלובלי eptState על מנת שנשתמש בהם בהמשך.

EptAllocateAndCreateIdentityPageTable

כאמור, ב hypervisor type 3 נוקטים בשיטת ה identity mapping, כלומר נמפה כל כתובת פיזית לעצמה ב EPT.
על מנת לממש שיטה זו יש להקצות טבלת מיפוי של 1:1 עבור כל גוש של 4KB מתוך הזיכרון הפיזי הזמין (נניח 4GB).
יצירה של טבלה כזאת תצרוך המון זיכרון, ולכן ננקוט בגישה אחרת של מיפוי בת 3 רמות במקום 4, כאשר "page" ייחשב כ 2MB במקום 4KB, במידה ונצטרך 4KB נוכל ליצור עוד רמת מיפוי ספציפית ולמחוק אותה שנסיים.

על מנת לעשות זאת משתמשים במבנה VMM_EPT_PAGE_TABLE אשר מכיל את 3 רמות המיפוי, ובנוסף תחזוק דינמית של דפים שפיצלנו ל 4KB - אנחנו משתמשים רק ב PML4E יחיד, ולכן אנו ממפים 512GB ב 2MB chunks סה"כ.
מאתחלים את כל הרמות עם הרשאות של RWX ואינדקס לרמה הבאה, ועם רשימת פיצולים ריקה. עבור הרמה האחרונה PML2E, חשוב לזכור גם לשחזר את שדה ה MemoryType כפי שמוגדר ב MTRRs (EptSetupPML2Entry).

VmxVmcallHandler

כאמור, VMCALL מאפשר לנו לקרוא לפונקציונליות של ה VMM כאשר אנחנו ב non-root operation, אך בפועל כל מה שזה עושה זה לגרום ל VM-Exit עם EXIT_REASON_CALL. לכן כל שעלינו לעשות זה לקבוע צורת קריאה לבחירתנו, למשל פרמטר ראשון יהיה control code ושלושת הפרמטרים הבאים יהיו Inputs.
כעת, כמובן שניתן לממש control codes רבים כמו למשל VMCALL_VMXOFF שיסיים את הוירטואליזציה בצורה נורמטיבית במקום לעשות hacks של קבלת נוטיפיקצייה על CPUID עם magics וכו' וכו'.

EptPageHook

אנו מעוניינים לממש פונקציה שתקבל כתובת וירטואלית בקרנל ותודיע לנו כאשר כתובת כלשהי מאותו page הורצה. נתחיל מלרשום את הפונקציה EptVmxRootModePageHook אשר שמה hook על הפונקציה הרצויה בהנחה שאנחנו ב VMX root operation, ולאחר מכן את הפונקציה EptHandleEptViolation אשר מטפלת בקפיצה של ה hook.

בהנחה שיש לנו את 2 הפונקציות האלה, כאשר אנחנו לפני VMLAUNCH נוכל פשוט לקרוא לה על מנת לשים את ה hook, וכאשר אנחנו אחרי נממש VMCALL control code ששם את ה hook הרלוונטי (שכן אסור לשחק עם ה EPT ב non-root), ולאחר מכן חשוב לזכור לעשות TLB flush כי כבר התחילו להשתמש ב EPT הזה (אחרי VMLAUNCH).

שימו לב שעל מנת לשים את ה hook ב root mode אנחנו צריכים לפצל את המיפוי של 2MB לעוד רמה של 4KB, ועל מנת לעשות זאת צריך להקצות זיכרון מה שלא תמיד ניתן לעשות ב root-mode (IRQL issues).
על מנת להתמודד עם בעיה זו, נתחזק buffer שיוקצה ב non-root mode ונשמור אותו ב guestState.

EptVmxRootModePageHook

מחלצים את הכתובת הפיזית של הכתובת הוירטואלית שקיבלנו ומחלקים את הבלוק 2MB המתאים ל 4KB קטנים בעזרת הפונקציה EptSplitLargePage. לאחר מכן מחלצים את ה PTE המתאים בעזרת הפונקציה EptGetPml1Entry ושמים הרשאות NX על מנת שנקבל EPT Violation כאשר ינסו להריץ את הקוד שרצינו. לבסוף, חשוב לא לשכוח לעשות flush ל TLB במידה ואנחנו אחרי VMLAUNCH.

EptHandleEptViolation, EptHandlePageHookExit

טיפול ב VM-Exit מסוג EPT Violation, מחלצים את ה PTE של ה guest PA כאשר ה EPT Violation קפץ, לאחר מכן בודקים האם זה נגרם כתוצאה מהרצה של NX page. במידה וכן, עלינו להחליף את ההרשאות של הדף להיות execute, להזכיר למעבד להריץ את הפקודה מחדש, ולעשות flush TLB.

HvTerminateVMX

משחררים את כל המשאבים הגלובלים שהקצנו, ולאחר מכן עבור כל מעבד משחררים מבני נתונים ספציפיים שלו וקוראים ל VMCALL עם control code של VMXOFF על מנת לסיים את הוירטואליזציה.

כעת, הריצו ובדקו שלא מקבלים BSOD, ושמקבלים את ההודעות דיבוג המתאימות...
יש לנו קוד קצת יפה יותר, פתרנו כמה בעיות סנכרון ועשינו PageHook די מעפן - יום מוצלח סה"כ.