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).
הצע לכל שיטה שהצגת פתרון ל VMM על מנת למנוע את הזיהוי.

SOLUTION

כפי שציינתי, במסמך לא מפורט על דרכי זיהוי VM אך ניתן לקרוא על זה בזמנך הפנוי + זה נחמד כתרגיל מחשבתי.
(יש עוד המון דרכים + דרכי התמודדות ממה שרשום פה)
דרכי פתרון אפשריות:

- עבור VMWare לפני VT-x
  - ניתן לראות שיש מחסור בדרייברים בסיסיים (דרייבר קול).
  - קיום דרייברים וקבצים מסויימים של VMWare.
  - כתובת MAC - יש לתמוך בוירטואליזציה לתקשורת ולכן חייב לדמות PIC, ה 3 בתים הראשונים מייצגים את היצרן, כמובן שעבור VMWare יש ערכים די קבועים.
  - תהליכי מערכת וערכי רגיסטרי מאוד ספציפיים לניהול ה VM.
  - RedPill – בחינת ערך ה IDTR בעזרת הפקודה SIDT ולאבחן אותו, מסתבר ש MSB של ה IDTR הוא די קבוע (0xFF) ב VMWare\VirtualPC לעומת מערכת non-virtualized.
  - ראינו ש VMWare משתמשים ב DE ל UM וב DE ל KM, מה יקרה אם נריץ את הפקודה sidt בשניהם ? עבור UM נקבל את ה IDTR האמיתי, ואילו עבור KM נקבל את ה IDTR המאומלץ !
- וירטואליזציה לפני VT-x
  - כפי שראינו, לפני VT-x היו עושים flush ל TLB בכל פעם שיצאנו מה VM.
    ניתן לנצל את זה על מנת לגשת לאיזור זיכרון כדי שייכנס ל TLB, לאחר מכן לגשת לאותו אזור זיכרון שוב אבל הפעם למדוד כמה זמן זה לקח לנו. לאחר מכן לבצע פקודה שמידה והיינו ב VM תגרור ריצה ב VMM בוודאות (cupid למשל) ולאחר מכן לגשת שוב למקום בזיכרון ולהשוות את כמות הזמן שלקח לנו הפעם לעומת הפעם הקודמת (במידה ואנחנו ב VM, היה TLB flush).
  - וכו'
- וירטואליזציה
  - להריץ פקודות שאנחנו מצפים לתוצאה מסוימת אבל מקבלים תוצאה אחרת (למשל CPUID)
  - בדיקת זמן ריצה עבור פקודות שבוודאות גורמות ל VM Exit (cupid, invd, vmcall ולא חסר)
  - בסופו של דבר ה VM חולק אותו זיכרון פיזי כמו ה VMM, ויש לכך השפעות צד.
    מה יקרה אם נחפש בכל הזיכרון הפיזי את המילה VMWare ? כאשר ה VMM השתמש בזיכרון ושחרר אותו (free), מ"ה לא מאפסת את הזיכרון ולכן יש שאריות בזיכרון משוחרר :)