הגנה במערכות מתוכנתות - ש.ב. 4

צפריר ריהן 039811880

גיא שקד 036567055

12 בינואר 2011

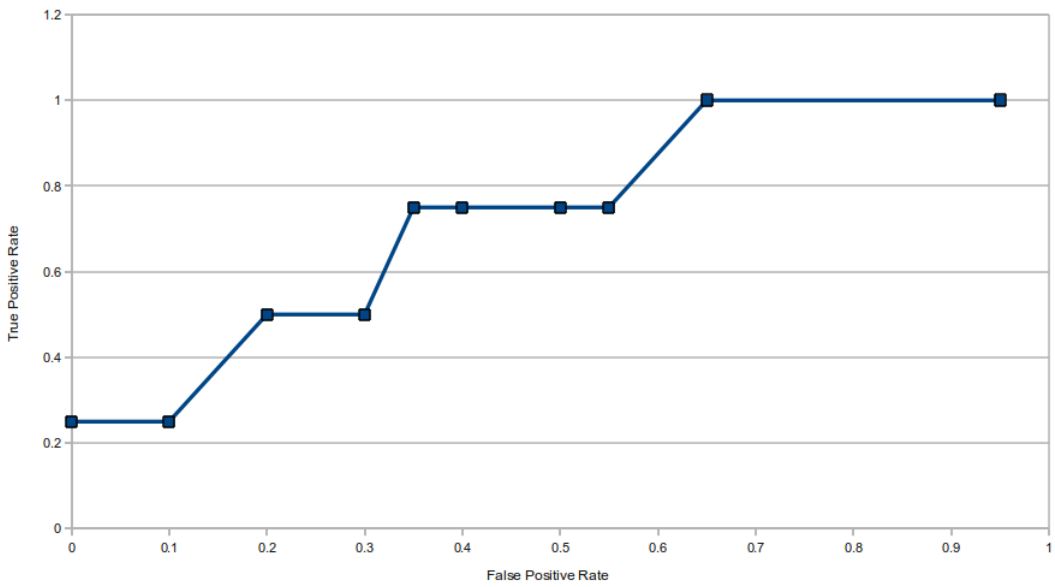
1. (א) מערכת א' עם הפרמטר 500 תניב 10 התראות. מתוכן 3 התראות אמת, כאשר יש 4 התקפות במערכת. לכן בסך הכל שיעור התראות האמת הוא 75%.

(ב) 7 מתוך ההתראות הן התראות שווא, ויש 20 מופעי התנהגות רגילה, לכן שיעור התראות השווא הוא 35%.

(ג) מערכת ב' עם הפרמטר 430 תניב 11 התראות (בהנחה ועבור שעות שבהן אין פעילות כלל לא ניתנת התראה). מתוכן 3 התראות אמת, בסך הכל שיעור התראות האמת הוא 75%.

(ד) 8 מתוך ההתראות הן התראות שווא, לכן שיעור התראות השווא הוא 40%.

(ה)



(ו) בהנחה ואין העדפה ל־false positive או ל־true-positive, נרצה להביא למקסימום את הערך -

$$TruePositive - FalsePositive$$

הערך המקסימלי עבור מדד זה (0.4) מתקבל עבור הפרמטר 500.

(ז) בהנחה ומנהל הרשת מעוניין לתפוס את כל אזעקות האמת (ולהקטין למינימום את אזעקות השווא) עליו לבחור במערכת א' בפרמטר 100 (ולקבל 65% אזעקות שווא), או במערכת ב' בפרמטר 500 (ולקבל 80% אזעקות שווא). לכן במקרה זה עדיף לבחור במערכת א'.

(ח) אם נפעיל את שתי המערכות, כאשר מערכת א' תופעל עם הפרמטר 500 ומערכת ב' עם הפרמטר 390, ונתריע רק כאשר באותה שעה שתי המערכות מתריעות על התקפה, נקבל בסך הכל 3 אזעקות (בשעות $9:00,\ 17:00,\ 22:00$) - כולן אזעקות אמת. כלומר 75% אזעקות אמת וללא אזעקות שווא כלל.

2. (א) כאשר מדיניות write down אינה נאכפת, תהליך בלתי אמין עלול לכתוב לחלקים קריטיים במערכת. ניתן לממש התקפה אפשרית על מערכת כזו באמצעות שימוש בזכויותיו של תהליך ה־browser לכתיבת דרייבר זדוני עבור המקלדת בתיקיית drivers/ - דרייבר אשר רושם את הקלדות המשתמש ושולח אותן לשרת מרוחק (keylogger).

(ב) אי אכיפת מדיניות read up עלולה לגרום למצב שבו תהליך הנחשב לאמין קורא נתונים בלתי אמינים. תוקף יכול לנצל זאת על ידי כתיבת נתונים זדוניים לתיקיית tmp/, כך שתהליך מרמה גבוהה יותר כגון application installer יקרא אותם, וינצל אותם למטרותיו של התוקף (לדוגמה, החלפת כתובתו של שרת מרוחק כדי לייצר מתקפת man in the middle).

(ג) לא ניתן לעשות זאת, על פי הגדרת write down. נבחין כי ה־browser נמצא ברמה הנמוכה ביותר, ומהגדרת write down הוא יכול לכתוב רק לרמה בה הוא נמצא או לרמות נמוכות יותר. היות ותיקיית drivers/ נמצאת ברמת crucial, הגבוהה ביותר, לא ייתכן כי כתוצאה מפעולה של ה־browser תיכתב אפליקציה לתיקיית applications/.

(ד) i. עדיף לוותר על מדיניות read up ־ כאשר מדיניות write down מבוטלת, תוקף יכול לנצל פרצה בדפדפן על מנת להתקין אפליקציה או דרייבר. במימוש שבו רק מדיניות read up מבוטלת, התקנת אפליקציה או דרייבר מבוצעים על ידי תהליכים אמינים הניגשים למידע שהורד על ידי הדפדפן, ולא על ידי הדפדפן עצמו. כך **התוכנה שעליה סומכים בוחרת** לאיזה מידע לא בטוח לגשת, בניגול למצב שבו **התוכנה שעליה פחות סומכים** בוחרת לכתוב מידע שיסווג כ"אמין".

ii. לאחר השינוי, המערכת חשופה להתקנת אפליקציות ודרייברים ממקורות לא ידועים. כדי להתמודד עם בעייה זו, נוסיף דרישה כי דרייבר או אפליקציה המורדים מהאינטרנט צריכים להיות מאושרים מראש על ידי יצרן המערכת, וחתומים בחתימה דיגיטלית. נשנה את המימוש של תהליכי התקנת אפליקציות ודרייברים כך שיבדקו שהאפליקציה או הדרייבר חתומים כנדרש על ידי המפתח הפומבי של יצרן המערכת.

(ה) משמעות הויתור על מדיניות write up היא שתהליך אמין שיש בידיו מידע סודי מסוגל לכתוב מידע זה לכל מקום במערכת, בפרט לרמות הרשאה נמוכות יותר, בהן רצים תהליכים פחות אמינים. מכיוון שמדיניות read down מאפשרת לתהליכים כאלה לקרוא את המידע שברמתם, ויתור זה מאפשר זליגת מידע סודי לתהליכים פחות אמינים, שייתכן כי נפלו קורבן למתקפה זדונית.

3. (א) כל המחרוזות $S_0, S_1 \dots S_{100}$ נגישות לשרת. לכן הוא יכול בשלב ראשון לחשב את p_0 ולשמור גם אותו, ועת בקשת הזדהות לשלוח למשתמש את המספר i, המשתמש יחשב את p_i וישלח לשרת, והשרת ישווה $p_i \stackrel{?}{=} p_{i-1} \oplus S_{101-i}$ כדי לבדוק האם אכן מדובר במשתמש לגיטימי.

נשים לב שפונקציית ה־xor היא לא חד כיוונית, לכן השרת למעשה יכול לחשב בעצמו את הערך שהוא מצפה לקבל מהמשתמש, ולחילופין לחשב את כל אחד מה־p_i באופן מלא ומפורש.

(ב) i. תוקף offline יוכל להתחזות לגלית בקלות. כפי שציינו בסעיף א' אצל השרת נשמר כל המידע הנחוץ לחישוב כל אחד מהמפתחות p_i ולכן אם תוקף השיג גישה למידע הנשמר אצל השרת הוא למעשה יודע את כל המידע שאליס יודעת (בנוגע להזדהות) ויכול להתחזות לה באופן מושלם.

לעומת זאת, כאשר משתמשים בפונקציה **חד כיוונית** כפי שנלמד בכתה, תוקף שמשיג את המידע הנשמר אצל השרת (p_0) לא יכול לחשב את f^{-1} בקלות, ולכן לא יוכל להתחזות לגל בקלות.

ii. תוקף online לא יוכל להתחזות לגלית בקלות. בכל תהליך הזדהות גלית מחשבת ערך שמבוסס על S_i אקראי שאין לתוקף מידע קודם לגביו, ולכן מבחינתו תהליך ההזדהות שקול להצפנה באמצעות פנקס חד פעמי.

לעומת זאת, הוא יוכל להתחזות לגל אם הוא לא מוגבל חישובית, מרגע שהתוקף "ראה" שני מופעים p_i, p_j הוא יכול לעבור על כל הסדרות האפשריות שיכלו לשמש את גל ולמצא את זו שגל משתמש בה (לחילופין, אם הוא ראה את השלב בו המשתמש הכריז על f ו־p_0 הוא יכול לחשב מתוכן את הסדרה).

(ג) תוקף יכול להתחזות לגלית מול השרת B לאחר שהיא ביצעה 100 תהליכי הזדהות מול A. בתהליך ההזדהות ה־i מול A גלית למעשה חושפת לתוקף את S_{101-i} (הוא ראה את p_{i-1} ואת p_i ולכן הוא יודע את $p_{i-1} \oplus p_i = S_{101-i}$). לכן לאחר שאליס השתמשה ב־$\{S_i\}$ עבור שרת A הם אינם סודיים עוד, והתוקף יודע אותם ולכן יכול להתחזות לגלית באופן מושלם.

נשים לב שגם אילו גלית הייתה מחלקת את אותם S_iים לשני השרתים בסדר הפוך, ומתחילה להזדהות במקביל מול שניהם תוקף היה יכול להתחזות לה מול כל אחד מהשרתים לאחר שהיא ביצעה בסך הכל 100 תהליכי הזדהות (מול שרת A או B), שכן כל תהליך כזה חושף S_i וסדר החשיפה הפוך.

4. (א) במקרה והמחשב הנייד נגנב כאשר הוא כבוי ־ סודיות הנתונים מובטחת. אנו מניחים כי אלגוריתמי ההצפנה בטוחים, לתוקף אין את המפתח הדרוש לפענוח ה־VMK או ה־$FVEK$ ולכן אינו יכול לקרא את הנתונים המוצפנים על הדיסק.

אם המחשב הנייד נגנב כאשר הוא פועל ־ ה־$FVEK$ כבר נמצא בזכרון, ולכן לתוקף יש גישה מלאה לכל הקבצים על הדיסק ־ הוא יכול להעתיק את כל תכולת הדיסק (לא מוצפנת) לדיסק אחר לפני שהוא מכבה את המחשב ־ ובכך יקבל גישה מלאה מתי שירצה לכל הקבצים.

(ב) במקרה זה החוליה החלשה במערכת היא הסמאות שבהן משתמשים המשתמשים. אם הסמאות נבחרות על ידי המשתמשים סביר להניח שהתקפת מילון תאפשר לתוקף למצא את הסמא. אבל גם אם המשתמשים בוחרים סמאות חזקות יחסית ניתן לצפות שהן יהיו קצרות יחסית, ובפרט קצרות מהסמאות האקראיות שניתן להחזיק על usb, ולכן ניתן להציע התקפות מילון גם עבורן.

פרט לבעיית אורך ואופן בחירת הסמא הפתרון המוצע שקול לפתרון בו משתמשים ב־usb (ניתן לתקוף את ה־usb עצמו כפי שניתן לתקוף את המקלדת, אך אלו לא ההתקפות בהן אנו דנים כאן). יש לציין שבדומה למשתמשים הנוהגים להחזיק את ה־usb בצמוד למחשב כך סמאות ארוכות ואקראיות גורמות למשתמשים לכתוב את הסמא בסמוך למחשב, וכך הבעיה חוזרת על עצמה.

(ג) כאשר אין במערכת רכיב TPM אין כלי שיוכל לזהות התנהגות חריגה של תוקף ולמנוע ממנו את ההתקפה. התנהגות חריגה כזו יכולה להיות העברת הדיסק הקשיח למחשב אחר, ביצוע מספר רב של נסיונות boot בתוך זמן קצר, החלפת קוד וכו'... התקפה כזו יכולה לכלול למשל - העברת הדיסק המוגן למחשב ייעודי, ש"יפציץ" את הדיסק בבקשות לפענוח ה־VMK מתוך מילון או בביצוע חיפוש ממצה. המחשב הייעודי יכול להיות מהיר למדי, וכתלות באורך המפתח ומהירות המחשב - למצא את המפתח בתוך זמן סביר.

לעומת זאת - במערכת המשתמשת ברכיב TPM ניתן לקבוע מדיניות שבה זמן מינימלי בין בקשות פענוח של ה־VMK (או מספר מוגבל של נסיונות - לפני שהמערכת ננעלת). בנוסף - כדי למנוע העברת הדיסק למחשב ייעודי שיבצע את החיפוש הממצה (או החיפוש במילון) במהירות ה־TPM יוכל לבצע בדיקת שלמות של המערכת, ולא לאפשר נסיון פענוח של ה־VMK אם הושתל בו קוד אחר או היה שינוי כלשהו בחומרה.

(ד) לא קיים מימוש פתוח של bitlocker המאפשר לקרוא קבצים מוצפנים ממערכת הפעלה שאינה חלונות. איתמר יכול להשתמש במערכת הצפנה חופשית כגון PGP, שקיימים מימושים שלה לכל מערכות ההפעלה הנפוצות, וכך גם לשמור על סודיות הנתונים שלו, וגם לאפשר גישה מלינוקס לנתונים שנשמרו בחלונות ולהפך.

(ה) i. אם המפתחות אינם שונים, קיים סיכון סטטיסטי שמשתמש מן המניין יקבל מפתח אתחול שהוא גם מפתח שחזור, ואז יוכל לקבל גישה למערכת של כל משתמש אחר בארגון תוך עקיפת ההגנות שמספק שבב ה־TPM.

ii. אין צורך במפתח שחזור במוד עבודה זה - מטרתו של מפתח השחזור היא לאפשר עליית מערכת במצב שבו שבב ה־TPM לא מאמת בהצלחה את זהות המערכת בהתאם למדדים שהוא מצפה שיהיו קבועים. מצב זה אינו קיים כששבב TPM אינו קיים, ולכן אין צורך במפתח כזה.