דו"ח סופי

תחילה, יצרנו תוכנית הרצה בתוך enclave בסביבת SGX במצב סימולציה.
לאחר מכן, תכננו התקפת rootkit שתוכל לשבש את ריצת התוכנית הרצה בenclave, ובכך לחשוף את פגיעותה של המערכת, ואת הפירצות הקיימות ביחס להתקפות מסוג זה.
התקפת הrootkit שבה התמקדנו היתה התקפה שמטרתה לשנות את הsyscall table של מ"ה על מנת לחדור את הגנות מערכת ה-SGX בעת קריאות ל- system calls באמצעות טעינת מודול זדוני לקרנל.
התקפת ה-rootkit פועלת על גירסאות kernel 2.6.X ולא על גירסאות חדשות יותר.
בבואנו להמשיך לחלקו הבא של הפרוייקט, גילינו שאין תמיכה למערכת SGX בגרסאות קרנל אלו והתמיכה החלה מגירסאות kernel 4.2.X.
בשלב זה הבנו יחד עם מנחה הפרוייקט שלא נוכל ליצור התקפה דומה לגירסאות הקרנל החדישות יותר מטעמי ההגנה הגבוהה שבהם.

בעקבות זאת, הוחלט יחד עם המנחה שנמשיך להתעמק בהכרת מערכת ה-SGX וה-API שלה הידוע במורכבותו ובפרט בתהליך ה-attestation (עדות) הייחודי למערכת זו על מנת ליצור תכנית מורכבת יותר שתדגים את יכולות ה-SGX.
במשך מס' שבועות התעמקנו בהכרת המערכת ובהכנת מצגת מפורטת על תהליך העדות (מצ"ב), תוך שנפגשנו מדי שבוע עם המנחה שבחן הבנתנו בעקרונות הקריפטוגרפיים שעליהם מבוססת המערכת (הצפנה סימטרית, הצפנה אסימטרית, פרוטוקולי סיגמא ומחויבות).

בחלק הסופי של הפרויקט עבדנו על הכנת הדגמה להוכחה באפס ידיעה על בסיס ארכיטקטורת SGX:
התכנית שכתבנו רצה ב- enclave המחביא סוד כלשהו, ואפליקציה חיצונית הרוצה לקבל ממנו את הסוד הזה ולשכנע אותו שהיא אמינה.
ההוכחה התבססה על בסיס התסריט הבא: לאפליקציה החיצונית יש מערך עם שני מספרים (0,1) והיא רוצה להוכיח ל-enclave ששני המספרים הם שונים.
האפליקציה מעבירה ל-enclave את המערך והוא, בהסתברות של 50%, מחליט האם לשנות את סדר המספרים או לא. לאחר מכן ה-enclave שואל את האפליקציה החיצונית האם שונה סדר המספרים.
תהליך זה חוזר על עצמו 42 פעמים ואם כל תשובותיה של האפליקציה החיצונית נכונות- ה-enclave בוטח באפליקציה החיצונית.
בקוד שלנו הדגמנו מה קורה כאשר אפליקציה אמינה (בעצם אפליקציה ששני המספרים שלה באמת שונים) מנסה לשכנע את ה-enclave באמינותה, ומה קורה כאשר אפליקציה זדונית (בעצם אפליקציה ששני המספרים שלה זהים) מנסה לשכנע את ה-encave באמינותה.

כל הקוד נכתב ב-C וב-++C בעזרת visual studio 2013 על בסיס ה-Intel SGX SDK.

מדי שבוע נפגשנו עם המנחה שבחן אותנו על ידיעתנו ב-SGX, בחן את ההתקדמות שלנו בכתיבת התכנית ועזר לכוון אותנו במה להתמקד על מנת להתקדם ולהשלים את הפרוייקט.

פיתוחים עתידיים:

ישנה ביקורת רבה המופנית כנגד Intel בשל ההישענות על מנגנון העדות במערכת ה-SGX.
המבקרים טוענים שהתהליך מורכב מדי, דורש המון הן מהמפתח והן מהמשתמש, מסורבל מאוד להבנה ושימוש ועלול לגזול נתח זמן משמעותי מתהליך הפיתוח.
כמו כן, המנגנון דורש מהמשתמשים "לסמוך" על Intel, דבר שבעייתי בפני עצמו ולכן מפתחים רבים נמנעים מהשימוש בטכנולוגיית SGX רק בשל כך, אע"פ שבאפשרותה לספק יכולות הגנה יוצאות מן הכלל.
בחלקו האחרון של הפרוייקט הראינו את האפשרות להשתמש בפרוטוקול קריפטוגרפי (הוכחה באפס ידיעה) על מנת לאשר/לדחות גישה לתכנית חיצונית מתוך ה-enclave, בדומה למנגנון ה remote attestation.
הראינו באמצעות תכנית המדגימה את השימוש בפרוטוקול זה ובאמצעות גרף המראה ניתוח הסתברותי של השימוש בפרוטוקול זה כיצד ההסתברות להשיג גישה בזדון לתכנית כזאת נמוכה מאוד ואף יעילה יותר מ-remote attestation.

פירוט שעות עבודה:

פגישה עם המנחה – אנחנו נפגשים עם מנחה הפרויקט מדי שבוע כדי לשפר ולקדם את הפרויקט.
עבודה עצמאית – לאחר הפגישה, כל אחד מיישם חלק מהמשימות שקבענו לעצמנו ביחד עם המנחה. מספר שעות עבודה שבועיות: כ-10 שעות.

316295005 daniella.fertouk@gmail.com דניאלה פרטוק
203839030 atrap11@gmail.com עידו אמיתי
204326755 dan.blumen1@gmail.com דן בלומנברג