מדעי המחשב

פרויקט גמר

פיתוח מערכת דירוג קלטים המתבססת על אינסטרומנטציה

מגישים:

יוסף בואקנה 301757324

וסים עודה 201162245

מנחים:

אבישי רידלמן

אופיר וייס

אוניברסיטת חיפה

מדעי המחשב

פרטים

מנחה הקורס:

ד"ר ענת אהרוני

מנחה הפרוייקט:

אבישי רידלמן

אופיר וייס

מפתחי הפרוייקט:

וסים עודה 201162245

ODEHWASSIM@GMAIL.COM

יוסף בואקנה 301757324

YOUSSEF.BAWAKNY@GMAIL.COM

תודה

רוצים להודות למנחים שלנו אבישי רידלמן ואופיר וייס אשר תרמו לנו בעשייה, גיבוש וסיום הפרויקט, לא היססו לעזור לנו להפגין מקצועיות בכל נושא שהתקשנו בו.

תודה רבה!

תוכן עניינים

1. דף הגדרת הפרויקט

שם סטודנט א	וסים עודה	תז:	201162245
שם סטודנט ב	יוסף בואקני	תז:	301757324
שם הפרויקט	פיתוח מערכת דירוג קלטים המתבססת על אינסטרומנטציה.		
מנחה:	אבישי רדלמן		
טלפון:	נייד: 0528736342	Email:	avishai.redelman@intel.com

חתימת

המנחה:_____אבישי_____

מטרת הפרויקט:	מטרת הפרוייקט היא לפתח מערכת המדרגת קלטים. המערכת מחפשת חולשות בתוכנה שתינתן על ידי המנחים ובאמצעות המערכת אפשר לדרג את הקלטים השונים שמכניסים לתוכנה על ידי קרטריונים שונים אשר קובעים את איכות הקלט וכמה נזק הוא יכול לגרום לתוכנה.
דרישות קדם	ארגון המחשב וספות סף
שלבי ביצוע תכנית עבודה	1. 5% Background material validation/fuzzing, motivation [30h]

2. 9% Pin ramp-up (+ generate simple pintool as example) [54h]
3. 9% Target ramp-up (if PFAT, need to understand functionality, …) [54h]
4. 9% Peach fuzzer ramp-up, for generating inputs against the Pintool [54h]
5. 12% Define parameters to be extracted from program execution [72h]
    a. Running time, memory heatmap, stack depth, API calls, plagiarise Valgrind
    b. Include capability in Pintool to get a copy of the message sent to the executable
5. 32% Write + Debug Pintool [192h]
6. 12% Scoring that may involve summarization/aggregation [72h]
7. 12% Report! Presentation! [72h]

שלבים עיקריים ומטרות ראשיות	1. היכרות עם הPINTOOL 2. כתיבת תוכניות בודקות תקינות קלטים שונים 3. היכרות עם התוכנה שצריכים לבדוק עליה את הקלטים 4. יצירת הקלטים השונים באמצעות ה- BATCH. 5. פיתוח מערכת ה PINTOOL-שבודקת ומדרגת את הקלטים השונים 6. סקירת הקלטים המדורגים וקביעת מי מהקלטים מקבל הציון הכי גבוה
שלבי בונוסים	
סביבה נדרשת	Pintool Batch
קו סיום משוער	ספטמבר 2015
ספרות ומאמרים	

2. תיאור הפרויקט

2.1 מטרת הפרויקט

היום תחום מדעי המחשב גדול מאי פעם, והרבה אנשים לומדים איך לתכנת ולכתוב תוכניות ומוציאים את התוכנות לקהל הרחב להשתמש בהן, אבל חלק ממתכנתים אלה למדו מהאינטרנט והתוכנות שלהם עושות בדיוק את מה שהן מיועדות להן, תוכנות אלה חשופות לנקודות תורפה ונקודות חלשות במקרי קצה, חולשה זאת יכולה לפגוע בבטיחות התוכנה להדליף מידע סודי או לקבל הרשאות שמשתמש מסויים אינו רשאי לגשת אליהם.

והאקרים היום מנצלים את העובדה שתוכנית יכולה להכיל נקודת חולשה, הוא מחפש אותה ומנסה לפרוץ למערכת דרכה.

לכן צריכים כלי שיעזור במציאת נקודות החולשה של התוכנית, כלי שיבדוק השפעת כל קלט על התוכנה.

לכן מטרת הפרויקט היא פיתוח מערכת בשפת C++ אשר תשמש לבדיקת קלטים ודירוגם לפי השפעתם על התוכנית הנבדקת.

הקוד בנוי ממספר בדיקות אשר בודקים מאפיניים מסויימים לקלט של התוכנה הנבחרת בזמן ריצה ונותן דירוג לפי תכונות קבועות מראש.

מריצים את הקוד מספר פעמים על אותו קלט כך שכל ריצה בודקת תכונה מסויימת וידועה מראש, ולפי תוצאת הקוד מדרגים את הקלט. אחרי כל הריצות על אותו קלט מתקבל דירוג כללי של הקלט שימש להשוואה למען דירוג של כלל הקלטים שנבדקה הרצתם על התוכנה.

בסוף הבדיקה מקבלים דירוג של קלטים, כך שהקלט המדורג הכי גבוהה הוא הכי משפיע על התוכנה, כלומר הוא העשוי לגרם הנזק הכי גדול לתוכנה.

פרויקט זה ייתן לנו את הכלי לאיתור נקודות חולשה בתוכנות מסויימות, ואם מאתרים נקודות חולשה אפשר לתקן אותה וכך התוכנה נהיית יותר חזקה ועמידה מפני התקפות.

2.2 דרישות הפרויקט

- ידע ושליטה מלאה בשפת C++ :קבצי הבדיקות ייכתבו בשפת C++.

- ידע ושליטה בשפת BATCH: משתמשים בשפת BATCH כדי לייצור את הקלטים של התוכנית ולהכניס אותם לתוכנית.

- ידע ומיומנות בסביבת PINTOOL-כלי המאפשר בדיקת המשאבים של תוכנית באמצע פעולה.

- מערכות הפעלה:היכרות עם מערכת ההפעלה חלונות של מיקרוסופט, הבדיקות נעשות בסביבת מערכת הפעלה חלונות של מיקרוסופט, ועל מנת לממש את הבדיקות צריך לצבור ידע באיך מערכת ההפעלה עובדת ובפרט איך הגרעין של מערכת ההפעלה עובדת: שעון, מחסנית,זיכרון וכו'...

- ידע בסיסי באבטחת מידע: פרויקט זה יכול לממש ככלי לאיתור חורי אבטחה לכן ידע בסיסי באבטחת המחשבים יעזור ביצירת בדיקות שיאפשרו זיהוי חורי אבטחה.

2.3 תכנית עבודה

i. היכרות עם שיטות בסיסיות של מציאת ויצירת חורי אבטחה בתוכניות, וגם למידה לעומק איך מערכת ההפעלה חלונות של מיקרוסופט והגרעין שלה פועלים.

ii. לחשוב על בדיקות כך שהתוצאות שלהן יכולות להעיד על זה שיש חולשה לתוכנה הנבדקת עבור קלט מסויים.לדוגמא בדיקה שמומשה בפרויקט היא בדיקת מספר הגישות לזיכרון, אם רוב הקלטים "הנורמאלים" נותנים פחות או יותר אותו מספר של גישות לזיכרון אבל יש קלט אחד או יותר שמספר הגישות לזיכרון עולה בצורה חשודה, קלט זה יקבל ציון יותר גבוהה משאר הבדיקות בעלות מספר גישות לזיכרון קטן יותר.

iii. אחרי ויש כבר רשימה של בדיקות, מתחילים להכיר את סביבת ה-Pintool שהיא הסביבה העיקרית והחיונית למימוש הפרויקט.

iv. התחלה בכתיבת הבדיקות של ה Pintool מתחילים בבדיקות הקלות והבסיסיות כמו מדידת זמן ריצה וכמות גישה לזיכרון.

v. בדיקת שהבדיקות עובדות היטב על תוכנות מובטחות כמו המחשבון או הכתבן של מערכת ההפעלה חלונות.

vi. התחלה בכתיבת הבדיקות הקשות והמסובכות כמו איתור יצירת קובץ ואיתור חיבור לרשת.

vii. בודקים את כל הבדיקות על תוכנית אחת ולוודא שהכל עובד כמתוכנן.

viii. לקבל תוכנה שצריכים לבדוק את הקלטים שלה ולדרג אותם, ללמוד מה מקבלת ואיך הקלטים שלה יירא.

ix. לכתוב ב- BATCH קובץ שיעבור ויכניס ל-Pintool את התוכנה עם כל הקלטים האפשריים, קובץ זה יוציא עבור כל בדיקה את התוצאות של כל קלט

x. דירוג הקלטים ויצירת קובץ דירוג סופי.

2.4 הגדרת משימות

הבעיה שהפרויקט מנסה לפתור היא למצוא נקודות חולשה של תוכנות באמצעות הכנסת קלטים שונים לתוכנה.

לכן המשימות לפתירת הפרויקט הן:

i. למידה על סביבת ה-Pintool ועל מערכת ההפעה.

ii. מציאת בדיקות שיעזרו באיתור נקודות חולשה בתוכניות.

iii. כתיבת הבדיקות ובדיקתן על תוכניות שהן מובטחות כמו תוכניות הבאות עם מערכת ההפעלה.

iv. כתיבת קבצי BATCH המתאימות לתוכנה שצריכים לבדוק אותה.

v. דירוג את תוצאות הבדיקות ומתן ציון סופי לכל קלט.

3. תהליכים מרכזיים

3.1 כתיבת הבדיקות

תהליך זה מתחיל בלמידה על סביבת ה- Pintool ומעבר על דוגמאות הרצה.

אחרכך חושבים על בדיקות שתוצאותן יכולות להעיד שאולי יש נקודת חולשה בתוכנה הנבדקת, רקע בסיסי באבטחת המידע יעזור במציאת סוגי בדיקות שמוצאות נקודות חולשה בתוכנות שהאקרים מנצלים אותם כדי לפרוץ לתוכנה.

חלק מהבדיקות קלות יחסית ואפשר לבדוק את נכונתן ישר, ויש גם את החלק האחר שהוא הבדיקות הקשות שלחשוב איך לממש אותם לקח הרבה זמן כמו כן מימושן ובדיקתן.

בסוף נכתבו 8 בדיקות שונות והן:

1) בדיקה שסופרת מספר הפקודות שהתבצעו.
2) בדיקת זמן ריצה.
3) בדיקת מספר הגישות אל הזיכרון.
4) בדיקת התנהגות המחסנית והעומק המקסימלי שהיא מגיעה אליו.
5) בדיקה מאתרת יצירת קבצים.
6) בדיקה הסופרת מספר הפונקציות השונות בתוך התוכנה.
7) בדיקה אם מבצעים שחרור זיכרון לאותו משאב פעמיים.
8) בדיקה המאתרת חיבור לאינטרנט וכמות הבייטים המועברים.

3.2 יצירת קלטים ובדיקתם

אחרי התהליך הראשון יש כבר קבצי DLL של כל הבדיקות, עם סיום התהליך השני מקבלים 8 קבצים של טקסט המכילים את הדירוג של כל קלט.

אחרי למידה והיכרות עם BATCH ניגשים לתוכנה שעליה נבצע את הבדיקות בודקים כמה קלטים היא מקבלת ומה סוג הקלטים שהיא מקבלת, קובעים מה טווח הקלטים שצריך לייצור, וכותבים תוכנית בשפת BATCH המכילה לולאה שמייצרת את הקלטים ומכניסה אותם לתוכנית.

בתהליך זה עוברים בדיקה בדיקה ועבור כל בדיקה בודקים השפעת כל קלט עליה.

בסוף מתקבל 8 קבצים כל קובץ מהווה תוצאת בדיקה מסויימת ובכל קובץ נמצאים כל הקלטים והציון שלהם.

עם סיום תהליך זה ממיינים את הקבצים לפי התוצאות שלהם כך כשמסתיים התהליך הנוכחי מקבלים 8 קבצים ממויינים כל קובץ מכיל את התוצאות של כל הקלטים ממויינים מהקטן לגדול.

3.3 דירוג תוצאות הבדיקות

אחרי התהליך השני בא התהליך השלישי והאחרון, כרגע קבצי ה-BATCH יצרו קובץ לכל בדיקה המכיל את התוצאות של כל הקלטים עבור אותה בדיקה, קבצים אלה התוצאות שבתוכם ממוינות בסדר עולה.

מה שנותר לעשות הוא להחליט לגבי כל בדיקה מהו פקטור השפעתה על הציון הסופי, כלומר כמה חשוב שהבדיקה תהיה תקינה או שלפי התוכנה שמבצעים עליה הבדיקה כמה משמעותי שינוי בתוצאות של הבדיקה.

אחרי שקובעים לכל בדיקה את אחוז השפעתה על הציון הכללי, עוברים בדיקה בדיקה ומחשבים עבור כל קלט ציון סופי, אחרי המעבר על כל הבדיקות סוכמים את כל התוצאות אל תוך ערך אחד עבור כל הבדיקות לקלט אחד.

ועם זה הפרויקט הסתיים, קובץ הפלט הסופי Rate.txt יכיל הציון הסופי של כל קלט על התוכנה,

ואפשר לבדוק איזה קלט קיבל הציון הכי גבוהה והוא הקלט שייגרום הכי נזק לתוכנה.

4. סיכונים טכניים וקשיים למימוש הדרישות והתגברות עליהן

4.1 סיכונים טכניים וקשיים למימוש הדרישות:

i. למידת כלי חדש (Pintool) בזמן קצר על מנת לעמוד בלוח זמנים של הפרויקט.

ii. קשיים בבחירת הבדיקות וקבלת החלטות לגבי הבדיקות היעילות ביותר.

iii. קושי בקביעת את הפקטור של כל בדיקה (מידת השפעתן על הציון הכללי).

iv. קושי במימוש חלק מהבדיקות.

4.2 התגברות על הבעיות:

v. לימוד חומר בצורה אינטנסיבית.

vi. ייעוץ והנחיות ממנחי הפרויקט.

vii. חיפוש פתרונות באינטרנט.

5. מדריך למתכנת

Checkers (*)

בפריוקט ישנם 8 קבצי CPP שהם מהווים את הבדיקות שנכתבו על מנת לעקוב אחר השפעת הקלטים עליהן והן:

i. instcount.cpp

תחילה, הבדיקה מאתחלת את המונה (icount) לאפס שישמש לספירת מספר הפקודות שיתבצעו בזמן ריצת התוכנה על הקלט.

בזמן ריצת התוכנה, הבדיקה משתמשת בפונקציה INS_InsertCall שמוגדרת ב-PINTOOL, כך שבכל פעם לפני שהתוכנה תתחיל בביצוע פקודה, INS_InsertCall תקרא לפונקציה docount() על מנת לקדם את המונה icount באחד.

בסוף ריצת התוכנה, הבדיקה תשמור את תוכן המונה icount בקובץ בשם inscount.csv שהוא יהיה פלט בדיקה זו, כלומר inscount.csv יכיל את מספר הפקודות שהתבצעו בזמן ריצת התוכנה על הקלט.

ii. Memtrace.cpp

תחילה, הבדיקה מאתחלת את המונה (counter) לאפס שישמש לספירת מספר הגישות לזיכרון בזמן ריצת התוכנה על הקלט. וגם תקרא לפונקציה INS_AddInstrumentFunction שמוגדרת בPINTOOL, כך שתוסיף את הפונקציה שהגדרנו בשם Instruction ככלי לשימוש בגרעין הפקודות.

בזמן ריצת התוכנה, כל פעם שהתוכנה ניגשת לביצוע פקודה, הבדיקה קוראת לפונקציה Instruction ואז Instruction קוראת לפונקציה שמוגדרת בPINTOOL בשם INS_MemoryOperandCount שמקבלת פקודה ומחזירה את מספר האופרנדים בפקודה שהם גישות לזכרון, אז מוסיפים את מה שמחזירה INS_MemoryOperandCount למונה counter.

אם נרצה לבדוק גם את סוג הגישה לזיכרון (קריאה או כתיבה לזיכרון), אז אחרי שקראנו ל INS_MemoryOperandCount אפשר להשתמש בפונקציות INS_MemoryOperandIsRead שמקבלת את הפקודה ואת מספר התא של האופרנד שרצינו לבדוק, הפונקציה תחזיר ערך אמת רק אם אופרנד זה הוא עבור גישה לקריאה מהזכרון, והפונקציה INS_MemoryOperandIsWritten שמחזירה ערך אמת רק אם אופרנד זה הוא לכתיבה לזיכרון.

בסוף ריצת התוכנה, הבדיקה תשמור את תוכן המונה counter בקובץ בשם MemTrace.csv שהוא יהיה פלט בדיקה זו, כלומר MemTrace.csv יכיל את מספר הגישות לזכרון שהתבצעו בזמן ריצת התוכנה על הקלט.

iii. stackdepth.cpp

תחילה, הבדיקה מאתחלת את maxDepth לאפס שישמש בכדי לשמור בתוכו את ערך המחסנית המקסימלי שהגיעה אליו אהתוכנה בזמן ריצתה על הקלט. וגם תקרא לפונקציה INS_AddInstrumentFunction שמוגדרת בPINTOOL, כך שתוסיף את הפונקציה שהגדרנו בשם Instruction ככלי לשימוש בגרעין הפקודות.

בזמן ריצת התוכנה, כל פעם שהתוכנה ניגשת לביצוע פקודה, הבדיקה קוראת לפונקציה Instruction , ואז Instruction קוראת לפונקציה שמוגדרת בPINTOOL בשם INS_InsertCall על מנת להוסיף קריאה לפונקצית עזר שיצרנו בשם MaxDepth(REG esp1) לפני ביצוע הפקודה, את ערך esp1 שהפונקציה MaxDepth תקבל גם יהיה קלט בקריאה ל INS_InsertCall. אחרי הקריאה ל-MaxDepth הפונקציה תחילה שומרת את תוכן esp1 במשתנה static REG base, רואים שמשתנה זה סטטי לכן ערכו לא נדרס אחרי סיום ריצת הפונקיה הזו. עכשיו הפונקציה בודקת אם ערך המשתנה שאתחלנו בתחילת הבדיקה (maxDepth) קטן מערך base-esp1, אם זה מתקיים אז התוכנה הגיעה לעומק מחסנית גדול יותר מהעומק המקסימלי שהמחנית הגיעה אליו בזמן הריצה שעבר לכן הפונקציה מעדקנת את תוכן המשתנה maxDepth להיות (base-esp1) ומסימת, אם maxDepth היה גדול או שווה ל base-esp1 אז התוכנית מסתימת בלי לשנות את ערך המשתנה maxDepth.

בסוף ריצת התוכנה, הבדיקה תשמור את תוכן maxDepth בקובץ בשם stackdepth.csv שהוא יהיה פלט בדיקה זו, כלומר stackdepth.csv יכיל את עומק המחסנית המקסימלי שהגיעה אליו התוכנה בזמן ריצתה על הקלט.

iv. Filetrace.cpp

תחילה, הבדיקה מאתחלת את המונה (fileCount) לאפס שישמש לספירת מספר הקבצים שהתוכנית יצרה או פתחה(לקריאה או כתיבה) בזמן הריצה שלה. וגם תקרא לפונקציה PIN_InitSymbols בכדי לאתחל טבלת סמלים מכיוון שPINTOOL לא יודעת לקורא סמלים אם לא קוראים ל PIN_InitSymbolsלפני תחילת ריצת ה PINTOOL.

בסוף ריצת התוכנה, הבדיקה תשמור את תוכן fileCount בקובץ בשם fileTrace.csv שהוא יהיה פלט בדיקה זו, כלומר fileTrace.csv יכיל את מספר הקבצים שהפונקציה יצרה או פתחה בזמן ריצתה.

במערכת ההפעלה של מיקרוסופט-חלונות- ישנן שתי פונקציות שהן מבצעות ייצירת קובץ והן:createfileA & createfileW, אם מאתרים קריאות לפונקציות הללו נגישים לפונקציה שמקבלת את שם הקובץ הנפתח ועוד כמה פרמטרים שהפונקציות מקבלות בפרויקט זה מעניין רק לדעת אם הייתה יצירת קובץ ולא את השם של הקובץ, אבל בהחלט אם רוצים לשנות את תוצאת הבדיקה להכיל את שם הקובץ הנוצר, פשוט להוסיף לקובץ הפלט את המשתנה funcName .

v. FunctionsCount.cpp

תחילה, הבדיקה מאתחלת ווקטור בשם functions שמקבל משתנים מסוג STRING וגם תאתחל iterator בשם it שיעבוד על הווקטור functions על מנת לעזור לנו בחיפוס בתוך הוקטור, כך שבזמן ריצת התוכנה, הבדיקה תוסיף את הפונקציות שהתוכנה משתמשת בהם לווקטור בכדי שבסוף נספור כל פונקציה שהתוכנה קראה לה רק פעם אחת, כלומר אם התוכנה קראה לפונקציה מסוימת יותר מפעם אחתף הבדיקה תוסיף את הפונקציה רק פעם אחת לווקטור functions כך שבסוף ריצת התוכנה אורך הווקטור functions יהיה מספר הפונקציות השונות בתוך התוכנה. הבדיקה גם קוראת לפונקציה PIN_InitSymbols בכדי לאתחל את טבלת הסמלים לPINTOOL.

בזמן ריצת התוכנה, הבדיקה קוראת לפונקציה שהגדרנו בשם Image בכל פעם שהתוכנה הולכת להריץ תת-תוכנה או משהו דומה (למשל בכל הרצת קבצי dll. או exe.), הפונקציה Image מקבלת משתנה מסוג IMG (בשם img), אז בתחילת הפונקציה Image בודקים על ידי IMG_IsMainExecutable שמוגדרת ב-PINTOOL אם הקלט img הוא ההפעלה הראשית, אם IMG_IsMainExecutable לא מחזירה ערך אמת אז לא עושים כלום, אחרת אם החזירה ערך אמת, עוברים על כל טבלת הסמלים של img ועבור כל פונקציה בודקים אם שמה קיים בתוך הווקטור functions, אם הוא לא בווקטור אז נוסיף את שם הפונקציה לווקטור.

בסוף ריצת התוכנה, הבדיקה תשמור את גודל הווקטור functions בקובץ בשם FunctionsCount.csv שהוא יהיה פלט בדיקה זו, כלומר FunctionsCount.csv יכיל את מספר הפונקציות השונות בתוך התוכנה.

vi. DoubleFree.cpp

תחילה, הבדיקה מאתחלת ווקטור בשם allocWatch שמקבל משתנים מסוג ADDRINT וגם תאתחל iterator בשם it שיעבוד על הווקטור allocWatch על מנת לעזור לנו בחיפוס בתוך הוקטור, כך שבזמן ריצת התוכנה, בכל פעם שהתוכנה מקצה זיכרון מוסיפים את הכתובת שלו לווקטור allocWatch, וכאשר התוכנה משחררת זכרון, מחפשים את הכתובת שלו בווקטור allocWatch ומוצאים אותה מהווקטור, לכן אם בחיפוס הכתובת לא נמצאת בווקטור אז כתובת זו כבר נעשה לה שחרור וזאת פעם שנייה שרוצים לשחרר אותה(או שבכלל התוכנה לא הקצה את הזכרון הזה). הבדיקה גם בהתחלה קוראת לפונקציה PIN_InitSymbols בכדי לאתחל את טבלת הסמלים לPINTOOL.

בזמן ריצת התוכנה, כאשר הבדיקה קוראה לפונקצית העזר שהגדרנו בשם Image, היא מקבלת משתנה בשם img מסוג IMG, הפונקציה מחפסת ב- img אם MALLOC מופיעה כלומר יש הקצאת זיכרון, אם היא מופיעה אז על ידי RTN_InsertCall שמוגדרת ב-PINTOOL מוסיפה קריאה לפונקצית עזר בעשינו בשם MallocAfter, כך שפונקציה זו תקבל את כתובת הזכרון שיוקצא ותוסיף אותו לווקטור allocWatch. הפונקציה Image גם מחפסת הופעות של FREE(שחרור זכרון) ב- img, ואז מוסיפה על ידי RTN_InsertCall קריאה לפונקציה שמימשנו בשם Arg1Before, כך שפונקציה זו תקבל כתובת זיכרון שהתוכנה הולכת לשחרר, ותחפס את הכתובת בווטור allocWatch, אם הכתובת נמצאת בווקטור אז היא מוציאת את הכתובת ממנו, אחרת אם הכתובת לא נמצאת בווקטור אז יש שחרור כפול, לכן תוסיף את כתובת הזכרון לקובץ הפלט של הבדיקה malloctrace.csv.

בסוף ריצת התוכנה, פלט הבדיקה יהיה שמור בקובץ בשם malloctrace.csv שבתוכו יכיל את כתובות תאי הזכרון הנעשה להם שחרור כפול בזמן ריצת התוכנה.

vii. socket.cpp

בדיקה זו בודקת אם התוכנה הנבדקת מתחברת לאינטרנט והיא מחזירה את כמות הבתים שהועברו בחיבור, "0" הינה כמות הבתים כשאין חיבור.

מחפשים בתוך התוכנית הנבדקת אם הפונקציה WSASend נקראת.

WSASend : הינה פונקציה של מערכת ההפעלה חלונות שנקראת אחרי פתיחת Socket חדש והיא משמשת להעברת הנתונים מהמחשב ליעד, מתוך הפרמטרים שהיא מקבלת יש את המספר של ה- Socket והפלט של הפונקציה הינו כמות הבתים שמועברים, בפרויקט זה מתחשבים רק אם יש חיבור או לא, ואם יש חיבור מה היא כמות הבתים שהועברו.

viii. timeCheck.cpp

בדיקה זו מחשבת את זמן ריצת התוכנה על הקלט בשניות. באמצעות הפונקציה `GetTickCount()`. שהיא פונקציה של מערכת ההפעלה שמחזירה את מספר הפעימות של שעון ה-BIOS של המחשב,קוראים לפונקציה פעמיים אחת לפני עם תחילת הבדיקה והשנייה עם סיום הבדיקה, התוצאה תהיה במילי-שניות ממירים אותה לשניות ומוציאים את התוצאה לקובץ.

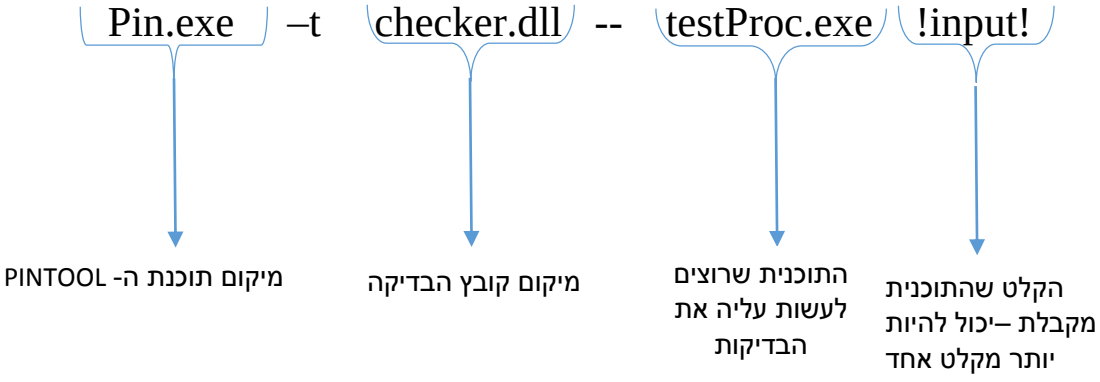
(*) קבצי ה-BATCH

לאחר שבדיקות עברו קומפיליציה מקבלים את הבדיקות בקבצים בפורמט DLL,

קבצי ה- BATCH תלויים בתוכנית שצריכים לבדוק אותה, בד"כ כותבים לולאה שעוברת על כל הקלטים האפשריים, ברור שכל תוכנית יש לה קבוצת קלטים שונה לכן צריך להתאים לכל תוכנית את Batch כך שיוציא את הקלטים המתאימים.

בתחילת כל קובץ מדפיסים הודעה שמגדירה את סוג הבדיקה לתוך קובץ חדש- קובץ זה יכיל את תוצאות הבדיקה.

אחרכך בתוך לולאה שרצה על כל הקלטים האפשריים מוסיפים את השורה הבאה:



בקובץ ה- BATCH ולפי השורה הנ"ל, !input! הינו משתנה שמקבל בכל איטירציה קלט חדש ומכניס אותו לשורה.

קובץ BATCH אחד יכניס את כל הקלטים האפשריים של תוכנית היעד ויבדוק את התוצאות עבור בדיקה אחת, לכן אפשר בכל אחרי כל בדיקה רק לשנות את שם קובץ הבדיקה וקובץ הפלט, או לכתוב BATCH לכל בדיקה ולכתוב עוד קובץ BATCH שייקרא בשם runAll.bat שיריץ את כל הקבצים אחד אחר השני.

אחרי שקובץ הבדיקה ירוץ עבור קלט מסויים הוא יוציא את תוצאת הבדיקה אל תוך קובץ, את התוצאה לוקחים ומכניסים אותה ביחד עם מספר הקלט או הקלט עצמו – לצורך זיהוי הקלט בעתיד.

אחרי שהלולאה מסתיימת הקובץ שהוגדר בתחילת הקובץ יכיל את התוצאות של כל הקלטים האפשריים עבור בדיקה אחת ,

פקודת SORT תעשה מיון לתוכן הקובץ לפי תוצאת הבדיקה אל תוך קובץ חדש שיש לו את אותו שם של הקובץ המקורי אבל מתווסף לסוף השם שלו SORTED.

דוגמא להמחשה: בדיקת זמן ריצה של תוכנית שמקבלת קלט של אות **A** מאורך 1 עד 120.

קובץ ה- **BATCH** יכיל לולאה שיוסיף לקלט בכל איטירציה את האות **A** 120 פעמים.

עם סיום הבדיקה ולפני פקודת ה- **SORT** מקבלים קובץ **time.txt** שמכיל את הקלטים בסדר כרונולוגי, כל שורה תהייה מהצורה הבאה: Input 20: 0.312

במצב הזה "20" מהווה את אורך הקלט ו "0.312" את תוצאת הבדיקה.

אחרי ה-SORT נקבל קובץ בשם timeSorted.txt אשר יכיל את התוצאות ממוינות מהקטן לגדול, קבצים אלה כבר מוכנים למעבר לשלב הבא שהוא דירוג ומתן ציון כללי של כל הבדיקות.

(*) קובץ ה-Rate.exe

כשמריצים Rate.exe יוצר קובץ פלט שמכיל את הדירוג הכללי והסופי עבור כל קלט.

אחרי הרצת קבצי ה- BATCH מקבלים את התוצאות הממוינות- לכל סוג בדיקה קובץ נפרד.

התוכנית Rate.exe עוברת על כל הקבצים ומדרגת אותם.

לתוכנית יש שתי שיטות דירוג הראשונה והיא הפשוטה מבין השתיים לוקחת את התוצאה של הבדיקה מכפילה אותו במספר לינארי גדול מאפס ומוסיפה אותו לתוך מערך שיכיל את מספר הקלט ואת הציון הסופי שלו- שיהיה מאותחל לאפס בהתחלה, המספר הלינארי נקבע על ידי המתכנת והוא תלוי בפקטור השפעת והמשקל של הבדיקה משאר הבדיקות, שיטה זו משתמשים אם הציון של הבדיקה נמוך אבל בעל משמעות גדולה כגון: מספר פונקציות בתוך התוכנית, מספר הפעמים התוכנית יוצרת קובץ וכו'.

השיטה השנייה טובה לבדיקות שהתוצאות שלהן בעלות ערכים גדולים כגון: מספר הפקודות שהתבצעו, מספר גישות לזיכרון וכו'. שיטה זו מתבססת על ההפרש בין התוצאות ואם ההפרש גדול מאחוז מסויים- שהמתכנת קובע לכל בדיקה בנפרד- יתווסף לציון של הקלט הנוכחי כמספר הפעמים שהתוצאה גדולה מההפרש לדוגמא: עבור קלט א' שההפרש בין התוצאה שלו והממוצע של התוצאות הנמוכות הינו גדול פי 4 מההפרש הנקבע על ידי המתכנת יקבל ציון פי 2 גדול מקלט ב' שההפרש שלו גדול פי 2 מההפרש הנקבע על ידי המתכנת.

תוכנית זו הינה השלב האחרון בפרויקט אחרי הרצתה מקבלים קובץ Rate.txt שמכיל את הציונים של כל קלט וכך אפשר לקבוע איזה קלט הכי משפיע על התוכנית הנבדקת.

6. מדריך למשתמש

הפריוקט הזה מאפשר לבדוק את השפעת קלטים מסויימים על כל תוכנה שהיא executable .

לכן שלב ראשון הוא למצוא תוכנה המעוניינים לבדוק את השפעת הקלטים עליה.

משום שלכל תוכנה יש את הקלטים המתאימים לה, קבצי ה- BATCH בפרויקט זה הם גינירים בסיסיים

הם יוצרות קלט אקראי מתוך הא"ב באורך עד שתיים – יש קבצי מוציאים קלט באורך אחד וקלטים באורך שתיים- ומכניסים אותו לתוכנה, אפשר להתאים את קבצי ה-BATCH שייצרו קלטים לפי רצון המשתמש.

אחרי שבוחרים תוכנת יעד ומסדרים את יצירת הקלטים מפעילים קובץ runAll.bat שהוא קורא לכל קבצי ה-BATCH ומיד אחריהן קורא לתוכנית Rate.exe שמייצרת קובץ Rate.txt המכיל את הציונים של כל הבדיקות עבור כל אחד מהקלטים, חשוב לציין שקובץ ה-Rate.txt מכיל את המספר של כל קלט-קלט 1 הוא הקלט הראשון שהוכנס לתוכנה- וליד כל מספר קלט את הציון הסופי שלו.

7. סיכומי הפגישות החודשיות

סטטוס הפרוייקט לחודש דצמבר

תאריך	13.1.2015
סטאטוס (מה עשיתי החודש...)	PINTOOL וכתיבת תוכניות בסיסיות בPINTOOLהיכרות עם פאזרים ועם ה
בעיות מיוחדות	PINTOOLהתרגלות על סביבת ה
שינויי לו"ז	התוכנית המותקפת הייתה מנוע עיבוד סקריפטים קנייני של אינטל. אבל אינטל לא אישרה שגורם חיצוני ישתמש בה. בינתיים המנחים מחפשים תוכנית חדשה.
תאריך מפגש עתידי עם המנחה	בין 18-21 בינואר
תוכנית עבודה לחודש הקרוב	-ים העושות בדיקות בסיסיות שמבצעים על כל תוכנית.PINTOOLלחשוב ולכתוב
הערות	סליחה על האיחור חשבנו שחודש ינואר הוא חודש ראשון משום שלא התקדמנו בפרויקט מספיק.

סיכום פגישות עם המנחה שנעשו במהלך החודש:

11.11- פגישה ראשונית עם המנחים והבנה על מה מדובר בפרויקט.

17.11- פגישה וירטואלית דרך המייל בה הוסברו השלבים של הפרויקט ומה צריך לעשות ברגע הנתון + חומר למידה ולינקים הקשורים לשלב הנתון.

9.12- שחזור על השלבים שלמדנו ועזרו לנו להגדיר את המחשבים שלנו כדי לעבוד באופן אופטימלי עם סביבת PINTOOLה. ונתנו לנו משימה לעשות עד לפגישה הבאה.

4.1- הגשת המשימה וקבלת הערות כלליות עליה והתקדמות לשלב הבא שהיא כפי שמילאנו לעי"ל ב- "תוכנית עבודה לחודש הקרוב".

סטטוס הפרוייקט לחודש ינואר

תאריך	5.2.2015
סטאטוס)מה עשיתי החודש...(	כתבנו בדיקות בסיסיות שמבצעים על כל תכנית בפינתול.
בעיות מיוחדות	למידה על פונקציות הגרעין של מערכת ההפעלה.
שינויי לו"ז	
תאריך מפגש עתידי עם המנחה	בסוף פיברואר
תוכנית עבודה לחודש הקרוב	לחשוב על תבנית פלט שתאפשר לנו לקרוא ולנתח את התוצאות של הבדיקות.
הערות	בשל תקופת המבחנים לא יצא לנו לעבוד על הפרויקט הרבה. אבל המנחים אמרו לנו שההתקדמות שלנו בפרויקט טובה מאוד.

סיכום פגישות עם המנחה שנעשו במהלך החודש:

25.1 – בדקנו שהבדיקות הבסיסיות עובדות, ודנו באיזה תבנית צריך להוציא את הפלט.

סטטוס הפרוייקט לחודש פברואר

תאריך	2.3.2015
סטאטוס (מה עשיתי החודש...)	התחלנו לחשוב על בדיקות מרוכבות יותר.
בעיות מיוחדות	
שינויי לו"ז	
תאריך מפגש עתידי עם המנחה	5.3.2015
תוכנית עבודה לחודש הקרוב	לכתוב ולסיים את הבדיקות המרוכבות.
הערות	

סיכום פגישות עם המנחה שנעשו במהלך החודש:

15.2 – שוחחנו עם המנחים במייל בנוגע לבדיקות שכתבנו, קיבלנו את המשוב שלהם ותיקנו מה שצריך לתקן ובתאריך 2.3 שלחנו להם את הבדיקות המתוקנות.

סטטוס הפרוייקט לחודש ____ מרץ

תאריך	7.4.2015
סטאטוס)מה עשיתי החודש...(	התחלנו לתכנת ולכתוב בדיקות מרוכבות יותר.
בעיות מיוחדות	
שינויי לו"ז	
תאריך מפגש עתידי עם המנחה	9.4.2015
תוכנית עבודה לחודש הקרוב	לסיים את הבדיקות המרוכבות.
הערות	

סיכום פגישות עם המנחה שנעשו במהלך החודש:

9.3 – נפגשנו עם המנחים ובדקו לנו את הבדיקות שמימשנו, ונתנו לנו עוד בדיקות מרוכבות שעלינו לממש.

לפני שבוע הייתה לנו בעייה במימוש הבדיקה אז שלחנו להם מייל ובדקו מה שעשינו ועזרו לנו בתיקון השגיאות שהיו במימוש.

סטטוס הפרוייקט לחודש אפריל

תאריך	7.5.2015
סטאטוס (מה עשיתי החודש...)	התחלנו לתכנת ולכתוב קוד שעובר על פלט הבדיקות שעשינו ונותן להם ציון.
בעיות מיוחדות	
שינויי לו"ז	
תאריך מפגש עתידי עם המנחה	9.4.2015
תוכנית עבודה לחודש הקרוב	תיקון כמה בעיות בבדיקות המרוכבות שעשינו והמשך בכתיבת קוד שמקבל את התוצאות ונותן להם ציונים.
הערות	

סיכום פגישות עם המנחה שנעשו במהלך החודש:

9.4 – בפגישה זו בדקנו את הבדיקות המורכבות, ונתנו לנו משימה חדשה שהיא כתיבת קטע קוד שיקלוט את תוצאות הבדיקות ולהתחיל לדרג אותם.

28.4 - איתרנו כמה בעיות באחת מהבדיקות המורכבות, נסינו לפתור אותה והמנחים נתנו לנו כמה שיטות שאולי יעזרו לנו בפתירת הבעיות,.והראנו להם איפה הגענו עם כתיבת הקוד שמקבל את התוצאות ונותן להם ציונים.

סטטוס הפרוייקט לחודש מאי

תאריך	7.6.2015
סטאטוס (מה עשיתי החודש...)	תיקנו את הבעיות שבבדיקות המרוקבות וכתבנו קוד שעובר על פלט הבדיקות שעשינו ונותן להם ציון.
בעיות מיוחדות	
שינויי לו"ז	
תאריך מפגש עתידי עם המנחה	בין 10 ל 20 ביוני
תוכנית עבודה לחודש הקרוב	נשארה לנו בעייה אחת (בקוד שעובר על פלט הבדיקות ונותן ציון) שעלינו לתקן, ואחר כך נפגש עם המנחים בכדי שיתנו לנו משימות חדשות.
הערות	

סיכום פגישות עם המנחה שנעשו במהלך החודש:

26.5 – הייתה פגישה וירטואלית דרך המייל בה הראנו למנחים את הקוד שתיקנו, וגם הראנו להם את הבעייה שעוד לו תיקנו והמנחים נתנו לנו טיפים באיך לנסות לזהות את הבעייה ולתקנה.

סטטוס הפרוייקט לחודש יוני

תאריך	7.7.2015
סטאטוס (מה עשיתי החודש...)	התחלנו לקרוא חומר שנתנו לנו המנחים שיעזור לנו לגרום לבדיקה המרוכבת שלא יכלנו לתקן לעבוד.
בעיות מיוחדות	
שינויי לו"ז	
תאריך מפגש עתידי עם המנחה	בין 10 ל 20 ביולי
תוכנית עבודה לחודש הקרוב	לגרום לבדיקה המרוכבת שלא עובדת לעבוד. ולהריץ את כל הבדיקות ולדרג את הקלטים עבור תוכנית הדוגמא שנתנו לנו.
הערות	

סיכום פגישות עם המנחה שנעשו במהלך החודש:

22.6 – הסבירו לנו איך רוצים שיראה דירוג הקלטים ולפי איזה קריטריונים לדרג.

סטטוס הפרוייקט לחודש יולי

תאריך	7.8.2015
סטאטוס (מה עשיתי החודש...)	תיקנו את כל הבעיות שהיו בבדיקות שעשינו, הרצנו את הבדיקות על תוכנית הדוגמא והתחלנו לכתוב את הספר.
בעיות מיוחדות	
שינויי לו"ז	
תאריך מפגש עתידי עם המנחה	בין 10 ל 20 באוגוסט
תוכנית עבודה לחודש הקרוב	לסיים את הספר והמצגות.
הערות	

סיכום פגישות עם המנחה שנעשו במהלך החודש:

22.7 – הייתה פגישה וירטואלית בה הראנו למנחים את תוצאות הרצת הבדיקות על תוכנית הדוגמא שנתנו לנו.

8. סיכום

מטרת פרויקט זה היא פיתוח מערכת לזיהוי נקודות חולשה של תוכנות על ידי דירוג הקלטים והשפעתם על התוכנה.

המערכת מכילה שמונה בדיקות אשר שימשו לבדיקת השפעת הקלטים על התוכנה ודירוגם, בדיקות אלה בעלי מאפיניים ידועים ובודקים תכונות ספציפיות של הקלט.

אחרי בנית המערכת הרצנו את שמונה הבדיקות על קלטים שדירוג השפעתם ידוע לנו למטרת איוליואציה וקיבלנו תוצאות לפי הציפיות שלנו.

לכל קלט קיבלנו: מספר הפקודות שהתבצעו, זמן הריצה, מספר הגישות אל הזכרון, התנהגות המחסנית והעומק המקסימלי, מספר קבצים שנוצרו, מספר הפונקציות השונות , שחרור כפול לזיכרון וחיבור לאינטרנט.

דירוג של כל קלט נקבע לפי תוצאות הבדיקות שלפיהן ידענו את הקלט הכי משפיע על התוכנה.

תהליך זה עזר לאמות פעולת המערכת ולוודא שעבור כל קלט קיבלנו פלט לפי הציפיות.

מערכת זו עזרה למציאת נקודות חולשה של תוכנות אשר פוגעות באבטחה ויכולות שהיו מיועדות להן.

9. ספרות ומקורות נוספים

1. Kit 71313 User Manual : מדריך המשתמש של Pintool מאינטל.

2. https://msdn.microsoft.com : אתר מיקרוסופט, מכיל כל הקשור לפונקציות מערכת ההפעלה חלונות.

3. Berkowits, Sion. "Pin-a dynamic binary instrumentation tool." *2012-06-13].*

https://software.intel.com/en-us/articles/pintool (2012)

4. Tool George Mason University ,"Introduction to the Pin Instrumentation"

http://cs.gmu.edu/~astavrou/courses/ISA_673_S13/PIN_lecture.pdf

5. PinTutorial- University of Virginia

http://www.cs.virginia.edu/kim/publicity/pin/tutorials/pldi07/PinTutorial.ppt

6. Introduction to Pin- Aamer Jaleel

www.jaleels.org/ajaleel/**Pin**/slides/1_Intro.**ppt**

7. "Real time Fuzzing",Charlie Miller, Independent Security Evaluators

8. Batch File Programming By Ankit Fadia

https://healholistic.files.wordpress.com/2013/08/batch-file-programming-ankit-fadia.pdf

9. INTRODUCTION TO BATCH FILES.

www.fbeedle.com/comline/79-1ch10.**pdf**

10. Windows.ppt

http://staff.csie.ncu.edu.tw/hsufh/COURSES/SPRING2012/**Windows**_2.**ppt**

11. Evolution of the Windows Kernel Archtiecture, University of Palermo

www.palermo.edu/ingenieria/Oct2009.**ppt**

12. Attacking the Windows Kernel- Black Hat

https://www.blackhat.com/.../bh.../bh-usa-07-lindsay-WP.**pdf**

13. Reddi, Vijay Janapa, et al. "PIN: a binary instrumentation tool for computer architecture research and education." *Proceedings of the 2004 workshop on Computer architecture education: held in conjunction with the 31st International Symposium on Computer Architecture*. ACM, 2004.