

## תרגיל בית 9 – תרגיל מסכם

alonrs@campus.technion.ac.il

אחראי התרגיל: אלון רשלבר

תאריך הגשה: ה-29 בינואר 2021

בתרגיל זה אנחנו נבנה תוכנת Firewall (חומת אש) אשר מסננת מידע לפי סט מוקדם של חוקים. לצורך מימוש ה-Firewall נשתמש ב-Cpp לבניית תוכנית פשוטה, וב-Bash לטובת פרסור החוקים והזנתם לתוך התוכנית שבנינו. בתרגיל זה נשים דגש על שלושה דברים:

1. נכונות – האם אתם מסננים את המידע כפי שנדרש.
2. אי זליגות זיכרון – אתם תראו שעם Cpp הרבה יותר פשוט לנהל זיכרון מאשר ב-C.
3. יעילות – אנחנו נמדוד את משך זמן הריצה שהתוכנית שלכם לוקחת.

**יש לקרוא את הוראות התרגיל עד הסוף, ורק אח"כ להתחיל לעבוד. כמו כן מומלץ (!!!!) לחלק עבודה. שימו לב:** קובץ ההוראות אמנם ארוך, אבל הפתרון די קצר. הפתרון שלנו משתמש ב-614 שורות קוד Cpp (כולל הערות ושורות ריקות) וב-17 שורות בסקריפט Bash. עם זאת, אל תחכו לרגע האחרון. הבנה לעומק של מה שנדרש מכם בתרגיל זה שווים לבערך 25% מסך ההתכוננות למבחן הסופי.

### חלק א' – קצת על רשתות תקשורת

האינטרנט שלנו עובד בעזרת שליחת וקבלת מידע בינארי ממחשבים ורכיבי חומרה שונים המחוברים זה לזה. המידע עצמו לא נשלח באופן רציף, אלא הוא מחולק לחבילות קטנות (Packets) שנשלחות באופן עצמאי (כל חבילה יכולה לשקלל בין 64 בתים לבין מספר KB). אנו לא ניכנס לנבכי עולם התקשורת והרשתות (על זה תוכלו ללמוד יותר בקורס ייעודי על רשתות), רק נאמר שבדומה למשלוח חבילות בדואר, גם במשלוח חבילות ברשת האינטרנט יש לסנן חבילות לפי שדות שונים (למשל, כתובת שולח וכתובת מקבל).

בתרגיל זה נתמקד ב-4 שדות אשר מופיעים ברוב חבילות האינטרנט אותם אנו שולחים ומקבלים:

1. כתובת מקור (Source IP Address)
2. פורט מקור (Source Port)
3. כתובת יעד (Destination IP Address)
4. פורט יעד (Destination Port)

### מושגים בסיסיים

**כתובת IP:** זהו Integer (4 בתים) אשר מציין איזה מחשב שלח את החבילה. לצורך פשטות, נניח כי כתובת זו חח"ע למחשב. כשאנחנו כותבים כתובת IP, אנו עושים זאת בפורמט הבא: כל בית ב-Integer מיוצג בצורה דצימלית (0-255), ובין כל שני בתים יש תו נקודה. למשל, כתובת ה-IP הזו: 12.133.52.4 מייצגת את ה-Integer הזה:

```
int ip = (4<<24) | (52<<16) | (133<<8) | (12);
```

**מסכה (Mask):** ניתן לייצג קבוצה של מספר כתובות IP ע"י ציון מספר הביטים (משמאל לימין) אליהם יש להתייחס, בעוד כל שאר הביטים הם Don't Care. ציון מספר הביטים נעשה ע"י הוספת סלאש (/) לאחר כתובת ה-IP, וכתובת המספר (בין 0 ל-32). לדוגמה, המסכה הבאה: 255.63.1.1/15 מציינת כי יש להתייחס ל-15 הביטים השמאליים ביותר (MSB) בכתובת ה-IP, ולשאיר ה-17 בתור Don't care.

The binary representation of 255.63.1.1/15 is (\* stands for don't care):  
11111111 0011111\* \*\*\*\*\*

**פורט (Port):** זהו Short (2 בתים) אשר מציין מספר אפליקציה בתוך המחשב. לא ניכנס למשמעות מעבר לכך, רק נגיד שפורט מיוצג בצורה דצימלית רגילה. ניתן לייצג טווח פורטים ע"י כתיבת שני מספרים מופרדים ע"י מקף, למשל 0-65535 מייצג את כל טווח הפורטים האפשרי.

**פקטה:** אנחנו נייצג פקטה כמחרוזת עם אוסף של 4 שדות (כפי שהוגדר לעיל), בפורמט הבא:  
src-ip=XXX.XXX.XXX.XXX, dst-ip=YYY.YYY.YYY.YYY, src-port=PRT, dst-port=PRT

שימו לב – יכולים להיות רווחים בין הפסיקים ובין תווי השווה (=), וכן השדות יכולים להיות בסדר שונה.

### חלק ב' – ה-Firewall שלנו

חומת האש (Firewall) שנבנה בתרגיל היא תוכנה אשר מקבלת כקלט רשימה של פקטות (כפי שהוגדרו לעיל), וחוק כלשהו המוגדר על שדה אחד בלבד מבין הארבעה. החוק מגדיר אילו ערכים של השדה חוקיים. התוכנית תדפיס ל-stdout את כל הפקטות שמקיימות את החוק.

#### דוגמה 1 - חוק על כתובת IP:

```
cat packets.txt | ./firewall.exe "src-ip=122.0.0.0/8"
```

שורת פקודה זו תריץ את ה-Firewall שלנו, כך שזה ידפיס למסך את כל הפקטות ב-packets.txt שכתובת המקור שלהן מכילות 122 בתור הבית הראשון.

#### דוגמה 2 – חוק על פורט:

```
cat packets.txt | ./firewall.exe "dst-port=22-22"
```

שורת פקודה זו תריץ את ה-Firewall שלנו, כך שזה ידפיס למסך את כל הפקטות ב-packets.txt שפורט היעד שלהן הוא בדיוק 22.

**שימו לב** – חוקי Firewall חייבים להיות בפורמט של טווחים, כלומר כתובות IP יכילו מסכה ופורטים יהיו כתובים כטווח (FROM-TO). כמו כן, בדומה לפקטות, גם כאן יתכנו רווחים בין תווי השווה (=), ובתחילת ובסוף המחזורית עצמה (למשל, החוק הזה תקין: " dst-port = 3-12 ")

**עצרו! אם משהו לא מובן לכם עד כאן, זה הזמן לשאול שאלות בפורום. כאן סימנו את חלק ההקדמה, ונמשיך לחלק המימוש.**

### חלק ג' – מימוש התוכנית Firewall.exe - מחרוזות

כפי שבוודאי שמתם לב, תצטרכו לפרסר מספר רב של מחרוזות לצורך עבודה תקינה עם Firewall. לשם כך, נבנה מחלקה בשם String אשר תוכל לתמוך בפעולות הבסיסיות תוך כדי ניהול עצמאי של הזיכרון אותו היא שומרת. בפרט, נרצה לתמוך בפעולות הבאות:

- קונסטרקטורים:
  - חסר פרמטרים: מאתחל מחרוזת ריקה
  - העתקה: מאתחל מחרוזת מתוך מחרוזת אחרת
  - עם פרמטר יחיד const char\*: מאתחל מחרוזת מתוך המשתנה, תוך ביצוע Clone.
- דיסטרוקטור: שחרור כל הזיכרון השמור בתוך String
- אופרטור =:
  - עם פרמטר const char\*: דורס את תוכן המחרוזת הנוכחי, תוך ביצוע Clone.
  - עם פרמטר String: דורס את תוכן המחרוזת הנוכחי, תוך ביצוע Clone לתוכן המחרוזת
- Equals:
  - עם פרמטר const char\*: מחזיר true אם מ"מ תוכן המחרוזת ב-this זהה לפרמטר.
  - עם פרמטר String: מחזיר true אם מ"מ תוכן המחרוזת ב-this זהה למחזורית בפרמטר.
- Split: מקבל רשימה של תווים לפיהם יש לפצל את המחרוזת (Delimiters), כתובת יעד לשמירת המחרוזות המפוצלות (Output), וכתובת יעד לשמירת מספר המחרוזות המפוצלות (Size).
- to\_integer: הופכת את המחרוזת למספר.
- Trim: מחזירה מחרוזת חדשה ללא רווחים בתחילת או בסוף המחרוזת.

שימו לב לקובץ הממשק **string.h**. את הקוד יש לכתוב בקובץ **string.cpp**.  
**עצרו! אם יש משהו שלא ברור בנוגע לאופן פעולת מחלקת String – זה הזמן לשאול שאלות בפורום.**

### חלק ד' – מימוש התוכנית Firewall.exe – שדות

בשביל לפרסר שדות שונים מסוג IP ומסוג Port, נבנה ראשית מחלקת אב המייצגת שדה. בשם Field. מחלקה זו תתמוך ב:

- קונסטרקטור המקבל מחרוזת: פרמטר זה יגדיר את המחרוזת אותה אנו מצפים לקבל בתחילת השדה (למשל src-ip או dst-port), וישמור אותה בתוך המחלקה.
- דיסטרקטור: ישחרר את כל הזיכרון של המחלקה.
- `get_type`: מתודה אשר מחזירה 0 אם השדה הוא Ip ו-1 אם השדה הוא פורט. שימו לב: ניתן לממש את מתודה זו ב-Field בלבד, לא חייבים להיעזר בדריסה!
- `set_value`: מתודה אשר תגדיר את סט הערכים החוקיים לאותו שדה (למשל, 120.0.0.0/8). מחזירה True אם לא היו בעיות בפרסור הקלט. שימו לב: המימוש של מתודה זו מתשנה עבור המחלקות היורשות מ-Field (ישנה דריסה!)
- `match_value`: מתודה המחזירה True רק אם הערך שהתקבל בפרמטר תואם את סט הערכים שהודגרו ע"י `set_value` (למשל, בהמשך לדוגמה הקודמת, עבור 120.4.5.6). שימו לב: המימוש של מתודה זו מתשנה עבור המחלקות היורשות מ-Field (ישנה דריסה!)
- `match`: מתודה אשר מקבלת מחרוזת של פקטה (ראו חלק א') ומחזירה True אם"מ הפקטה תואמת את החוק המוגדר ע"י השדה. שימו לב – במימוש מתודה זו מומלץ לקרוא ל-`match_value`. שימו לב: ניתן לממש את מתודה זו ב-Field בלבד, לא חייבים להיעזר בדריסה!

עתה, נבנה שתי מחלקות בנות – Ip ו-Port אשר ירשו מ-Field ויתמכו במימוש שונה וספציפי עבור כל אחת מהמתודות `set_value` ו-`match_value`.

**הערה:** לצערנו, לא הספקנו ללמוד בקורס מנגנונים מתקדמים ב-Cpp אשר יכולים באופן אוטומטי לקרוא למתודות של מחלקות בנות לפי הסוג שלהן. לכן, במימוש של `set_value` ו-`match_value` במחלקה **Field**, תצטרכו באופן ידני לבדוק מה הטיפוס של המחלקה (ע"י `get_type`), ולקרוא למתודה הרלוונטית של אחת מהמחלקות הבנות Ip או Port.

שימו לב לממשקים המוגדרים לכם בקבצים **ip.h**, **port.h**, **field.h**. את הקוד יש לכתוב בקבצים **field.cpp**, **ip.cpp**, **port.cpp**. כמו כן, מומלץ ואף רצוי (!!) להשתמש במחלקת String שיצרנו בחלק ג' לטובת פרסור המחרוזות השונות.

**עצרו! אם יש משהו שלא ברור בנוגע לאופן פעולת המחלקות Field, Ip, Port – זה הזמן לשאול שאלות בפורום.**

### חלק ה' – מימוש התוכנית Firewall.exe – הקובץ הראשי

בחלק זה תצטרכו לכתוב את פונקציית ה-main של התוכנית. הפונקציה הזו קוראת את הקלט הסטנדרטי ואת הארגומנט הראשון של התכנית, מפרסרת אותם, ומדפיסה לפלט הסטנדרטי את כל הפקטות שעומדות בחוק שהוכנס כארגומנט (ראו דוגמה לאופן הקריאה לתוכנית בחלק ב').

**שימו לב: אתם לא צריכים לכתוב הכל מאפס.** במימוש שלכם, יש להשתמש בפונקציות מהספרייה `libinput.so` שאנו כתבנו. בפרט, אלו הן המתודות:

- `check_args`: מקבלת כפרמטרים את הארגומנטים של המתודה `main`, ובודקת שהם תקינים. אם לא, מודפסות הודעות שגיאה למסך, וחוזר ערך שונה מ-0. יש לצאת מהתוכנית עם סטאטוס שגיאה במידה והפונקציה נכשלת.
- `parse_input`: מקבלת **רפרנס** לאובייקט מסוג Field, ומדפיסה ל-`stdout` את כל הפקטות שנקלטו ב-`stdin` שעומדות בחוקים שהודגרו ב-Field (ע"י הרצת המתודה `match` המוגדרת בממשק של Field).

**שימו לב:** את הקוד יש לכתוב בקובץ בשם **main.cpp**.

**עצרו! אם יש משהו שלא ברור בנוגע לאופן פעולת המתודה main, או הפונקציות ב-libinput.so – זה הזמן לשאול שאלות בפורום.**

### חלק ו' – בניית MakeFile

- עליכם לייצר Makefile אשר מקמפל את כל קבצי ה-cpp, ומבצע לינקוג' לפי ההוראות הבאות:
- על המחלקות String, Field, Ip, Port להיות מלונקג'ות לכדי ספרייה דינאמית בשם libfirewall.so. חלק זה קריטי, שכן הספרייה libinput.so עושה שימוש ב-Field וב-String, ולכן היא משתמשת בספרייה libfirewall.so.
- על הקובץ main.cpp להלונקג' לכדי תכנית בשם firewall.exe. שימו לב להצהיר כי אתם משתמשים בספריות הדינאמיות libinput.so ו-libfirewall.so (משום שאתם עושים בהן שימוש...)

**שימו לב:** כשעובדים עם Cpp, הקונבנציה היא להשתמש במשתנה CXX במקום CC, וכן ב-CXXFLAGS במקום ב-CFLAGS.

**שימו לב:** מנגנון יצירת ספרייה דינאמית ב-Cpp זהה לחלוטין לזה שב-C. בפרט, Cpp יודעת להשתמש במחלקות המוגדרות בספריית דינאמיות אשר קומפלו בעזרת Cpp (נסו לחשוב באיזה אופן זה קורה).

**עצרו! אם יש משהו שלא ברור בנוגע ל-Makefile – זה הזמן לשאול בפורום.**

### חלק ז' – דיבוג ומדידת יעילות

1. וודאו כי התוכנית רצה כמו שצריך, שאין דליפות זיכרון, ושאין בעיות כלשהן. במידה ויש, תוכלו למצוא אותן כפי שלמדנו בעזרת GDB.
2. לשימושכם, נתונים לכם שני סקריפטים לביצוע בדיקות (test0.sh, test1.sh) וכן קבצי פקטות כקלט (test0-pkts.in, test1-pkts.in). וודאו שהתוצר שלכם זהה לתוצר שאנחנו סיפקנו (test0.out, test1.out).
3. שימו לב כי בסיום הריצה התכנית פולטת ל-stderr את סך זמן ריצת התוכנית במילי שניות. ההגשה הנכונה והיעילה ביותר תזכה את בעליה ב-0.5 נקודות בונס לציון הסופי. (אופן המדידה שלנו: הרצת התוכנית שלכם 10 פעמים על אותו הקלט, וביצוע ממוצע על הזמן נטו).

### חלק ח' – כתיבת סקריפט firewall.sh

למעשה, חומת האש שלנו לא תפלט פקטות רק לפי שדה אחד (כמו בתוכנית firewall.exe), אלא תתמוך במספר רב של שדות (פעולת AND) ובמספר רב של חוקים (פעולת OR). נרצה לכתוב סקריפט אשר מקבל ב-stdin את הפקטות, וכארגומנט שם של קובץ המכיל מספר רב של חוקים, ופולט ל-stdout את כל הפקטות שהתאמתו על לפחות אחד מהחוקים.

**מבנה חוק:** אסופה של 4 שדות שביניהם מתקיימת פעולת AND. למשל:

src-ip=253.145.84.201/32, dst-ip=189.112.138.228/32, src-port=53-53, dst-port=0-6

חוק זה יעביר פקטות שיש להם גם את כתובת המקור הרלוונטית, גם את כתובת היעד הרלוונטית, וגם את מספרי הפורטים (מקור ויעד) רלוונטיים. שימו לב: יכולים להיות רווחים בין תווי השווה (=) והפסיק (.). כמו כן, השדות יכולים להיות מעורבלים.

**מבנה קובץ חוקים:** קובץ החוקים מכיל מספר חוקים שביניהם מתקיימת פעולת OR. כל חוק נמצא בשורה חדשה. כמו כן, יכולות להיות שורות ריקות בקובץ, וכן הערות (מתחילות ב-#) בסוף שורת חוק או בשורה רגילה. לדוגמה:

```
src-ip= 253.145.84.201/32, dst-ip=189.112.138.228/32, src-port= 53-53,dst-port=0-6
# Comment, yep.
Src-ip=253.145.84.201/31, dst-ip=189.112.138.228/31,src-port=53-53, dst-port=0-6 # Comment in here

src-ip=253.145.84.201/31 , dst-ip=189.112.138.228/24,src-port = 53-700,dst-port=0-60
```

על הסקריפט לפרסר את קובץ החוקים (רמז: מומלץ להשתמש ב-sed, ב-awk ובכלי CLI נוספים), לבצע פילוט באמצעות firewall.exe, ולפלוט לפלט הסטנדרטי את כל הפקטות שעומדות בלפחות אחד מהחוקים. על הפקטות להיות ממויינות בעזרת **sort (כלי CLI)**.

**שימו לב:** שם הסקריפט צריך להיות **firewall.sh**. כמו כן, נתונים לכם שני טסטים לבדיקת נכונות: 

- test2.sh, test2-pkts.in: בודק נכונות של חוק בודד בעזרת firewall.exe. השוו את הפלט שלכם עם test2.out
- test3-pkts.in ו-test3-rules.in: נועד לבדוק נכונות של הסקריפט שלכם firewall.sh. השוו את הפלט עם test3.out

**עצרו! אם יש משהו שלא ברור בנוגע ל-firewall.sh – זה הזמן לשאול בפורום.**

חזהו. סה"כ קצר (:

### דגשים מיוחדים

1. אנחנו נשתמש ב-Makefile שלכם – כאחריותכם לוודא שהוא עובד ופועל כנדרש!
2. יש לוודא שהתוכנית פועלת ללא דליפות זיכרון בעזרת Valgrind (ראו סדנה 6). לתוכניות עם דליפות זיכרון יורדו נקודות.
3. הקוד שלכם חייב לעמוד בקונבנציות הקוד כפי שראינו בתרגולים 1 ו-2. ירדו נקודות למי שלא יעבוד לפי הקונבנציות.

### הוראות הגשה:

1. עברו היטב על הוראות ההגשה של תרגילי הבית המופיעים באתר טרם ההגשה! ודאו כי התכנית שלכם עומדת בדרישות הבאות:
  - התכנית קריאה וברורה
  - התכנית מתועדת היטב לפי דרישות התיעוד המופיעות באתר.
2. יש להגיש לינק ל-Repository המכיל את הקבצים (שימו לב לשמות הקבצים עם lower case). על ה-Repository להיות בעל הרשאות Public. בעת בדיקת התרגיל, אנו נבצע clone ל-Repository שלכם, נבצע קומפילציה בעזרת ה-makefile שתגישו ונבדוק את התוכנית. שימו לב לשם הספרייה הנדרש מכם!
3. **אין הגשות חוזרות לתרגיל! בבקשה, שימו לב שהגשתם הכל כנדרש! בפרט, יש להגיש לפי הפורמט הבא:**  

```
https://github.com/your-username/repository-name
0123456789 student_1_mail@campus.technion.ac.il first_name_1 last_name_1
0123456789 student_2_mail@campus.technion.ac.il first_name_2 last_name_2
```
4. שאלות בנוגע לתרגיל יש להפנות לפורום התרגיל ב-moodle בלבד. ניתן לשלוח שאלות במייל למתרגל האחראי על התרגיל בלבד, ורק במידה והשאלה מכילה פתרון חלקי.

### סיכום מפרט התרגיל:

libinput.so input.h	קבצים נתונים
------------------------	--------------

field.h ip.h port.h string.h	
Makefile main.cpp field.cpp ip.cpp port.cpp string.cpp	קבצים להגשה
test0.sh test1.sh test2.sh test0-pkts.in test1-pkts.in test2-pkts.in test3-pkts.in test3-rules.in test0.out test1.out test2.out test3.out	טסטים נתונים
libfirewall.so, firewall.exe, firewall.sh	שמות הספרייה / תכנית / סקריפט שיש ליצור
ע"י יצירת Public Repository ב-GitHub	אופן ההגשה
אלון רשלבר alonrs@campus.technion.ac.il	אחראי התרגיל

בהצלחה!!