צור אליהו, 300647799

סדנה באבטחת מידע – תרגיל 2

** **הערה חשובה**: שם המודול שלי הוא fw.ko, כפי שהמתרגל אישר בפורום לקרוא למודול בשל פיצול הקבצים. אז לאחר הקימפול, חשוב לעשות insmod למודול fw.ko ולא לשם הקובץ c הראשי**

פירוט כללי של הקוד:**Module**
הפעם לקחתי את התרגיל הקודם ופירקתי אותו כבר עכשיו לכמה קבצי C.
הרעיון היה, להקל על עצמי בשלב מאוחר יותר היות ובתרגיל הזה יכלתי להישאר עם קובץ אחד.
הפירוק היה כדלקמן:
קובץ **hookfuncs.c**, שמכיל את כל הפונקציות שקשורות לnetfilter ותפיסת ההוקים הנכונים לתפיסת הפקטות(רוב הקוד של תרגיל קודם נכנס לקובץ זה)
ועוד קובץ **hw2secws.c**, שמכיל בתוכו את הקוד שפותח ורושם את הsysfs device, כמו שראינו בכיתה.

קבצי **header**:
2 קבצים – **hookfuncs.h** שנועד להיות הheader של קובץ hookfuncs.c ולהכיל הכרזות עבורו
fw.h שנועד להיות הheader של hw2secws.c.

Makefile
כמו שנאמר בפורום, יכלנו לבחור איזה שם שרצינו למודול (קובץ ko.), אז במקרה שלנו – השתמשתי בשם fw.ko שהמתרגל נתן, וכך שיניתי גם את הMakefile בהתאם.

הפונקציות שמימשתי:
על הקובץ hookfuncs.c/h עברתי בתרגיל הקודם, לכן אין מה לפרט על הפונקציות שלו, כל הפירוט עליהם נמצא בhw1_dry.
למעט 2 שינויים:
הוספת 2 מונים - cnt_blocked, cnt_accepted
הרעיון בהוספתם היה שיוכלו לעקוב אחר כמות הפקטות כדי לשלוח מידע זה למשתמש. נשים לב שהוספתי אותם לפונקציות משבוע שעבר - block_hook_func, pass_hook_func בהתאמה, כאשר לפני שהוא מחזיר אם להפיל או להעביר את הפקטה – כל פונקציה מעלה את המונה המתאים לה ב1.

נפרט על הפונקציות של hw2secws.c:
static int __init module_init_function(void)
הפונקציה הראשית של המודול. מכילה 5 שלבים:

1. רישום הSysfs Device – נעשה כפי שראינו בכיתה.
2. יצירת sysfs Class – כפי שראינו בכיתה בעזרת הפונקציה class_create
3. יצירת הdevice עצמו, בעזרת device_create
4. יצירת הsysfs file בעזרת device_create_file
5. start hooks – רישום הhooks של תרגיל קודם בכדי לתפוס את הפקטות.

אם יש כשלון באחד מהשלבים הנ"ל – הפונקציה מבצעת ניקוי של כל מה שעשינו עד השלב שבו התרחש הכשלון.

static void __exit module_exit_function(void)

הפונקציה הראשית שיוצאת מהמודול. אחראית על ניקוי כל מה שפתחנו בinit.

ssize_t display(struct device *dev, struct device_attribute *attr, char *buf)
הפונקציה שמגיבה לבקשות READ מההתקן שלנו.
היא מחזירה הודעה גנרית על מספר הפקטות והפירוט שהתבקשנו. יכלתי לבחור להעביר רק את המשתנים, אבל החלטתי לתמוך גם בפקודת cat, ולכן השארתי את הטקסט כמו שהוא ולא החזרתי רשימת מספרים.
בתרגילים הבאים אני מניח שאצטרך להוציא הרבה מידע מההתקן, ולכן סביר להניח שאעדיף להחזיר רשימת משתנים או מחרוזת שאצטרך לפרסר כדי לקבל נתונים. אבל נכון לעכשיו – המימוש הנ"ל היה מספיק עבור צרכיי.

ssize_t modify(struct device *dev, struct device_attribute *attr, const char *buf, size_t count)
הפונקציה שמגיבה לפקודות WRITE על ההתקן שלנו. דומה למה שראינו בכיתה, רק שאם נשלח כל משהו אחר שהוא אינו 0, הפונקציה לא מגיבה ולא עושה כלום (מעין בדיקה כפולה, בפועל אינני יודע מי יקרא לפונקציה שלי בכל פעם.)
אך אם נשלח 0 – הפונקציה מאפסת את 2 המונים שהוגדרו.

פירוט של הקוד: **User**

תוכנית פשוטה לקריאה וכתיבה לתוך התקנים.
כצעד ראשון היא בודקת ארגומנטים. אם יש יותר מ2(שם התוכנית ותו) היא יוצאת
אחרת, היא פותחת את הקובץ לקריאה וכתיבה.
אם הועבר תו 0 – היא מאפסת את המונים על ידי קריאה לmodify של ההתקן
אם לא הועבר התו 0 – היא פשוט מדפיסה את המונים על ידי קריאה לdisplay של ההתקן