צור אליהו, 300647799

סדנה באבטחת מידע – תרגיל 4

הערה: כל ההנחות שלי ורפרנסים שהשתמשתי בהם מהרשת(stackoverflow וכו') נמצאים בסוף מסמך זה, כולל הסבר על שימוש של אותם רפרנסים בפונקציות הרלוונטיות שהשתמשתי בהן בקוד. כמו כן הוספתי את הלינקים עצמם גם בהערות בקוד.
(סימנתי הנחות כלליות בטקסט בצבע צהוב)

המבנה הכללי של התוכנית:

קבצי הModule:
fw.c, fw.h, hookfuncs.c, hookfuncs.h, log.c, log.h, stateless_funcs.c stateless_funcs.h, stateful_funcs.c, stateful_funcs.h ,Makefile
קבצי הInterface:
main.c, Makefile (ועוד קבצי חוקים וhosts שקיבלנו ושמתי בתיקייה הנ"ל, סתם לנוחות, לא חובה להשתמש.)

התנהלות המערכת הכללית:

אני כמובן מדלג פה על ההסבר של התרגיל הקודם. בתמציתו, אני ממשיך מיד מהשלב שבו יש טבלת חיבורים סטטית, והפקטות קודם כל עוברות דרכה.
לאחר מכן, במידה וזוהי פקטת TCP עם ack כבוי – היא פותחת חיבור חדש בטבלת החיבורים. וכל פעם שמגיעה פקטה שקשורה לחיבור הנ"ל, הפקטה עוברת דרך הטבלה הדינמית(ולא דרך הסטטית) ובעזרת מכונת מצבים פשוטה, מעדכנת את הסטטוס של אותו חיבור – בהתאם למה שהיא ראתה. אם הגיעה פקטה לא קשורה, שלא תואמת למצב המכונה הנוכחי – הפקטה מיד נזרקת (הלוג מתעדכן כמובן)

כעת אפרט על 2 ההתקנים החדשים, ועל התנהלות הfirewall מול ההתקנים הללו.

ההתקן fw_conn_tab

ההתקן הנ"ל אחראי להציג את החיבורים הקיימים במערכת.
כמו שהוגדר לנו, בשלב הhandshake כל חיבור חי למשך 25 שניות. אם לא התבצע שינוי(מצב החיבור לא השתנה לestablished, על מכונת המצבים יפורט בהמשך), החיבור נסגר.
אם כן התבצע שינוי (הסתיימה לחיצת היד המשולשת), החיבור עובר למצב established (עוד על החיבורים בהמשך)
גישה להתקן היא על ידי קריאה וכתיבה ל /dev/fw_conn_tab, שכן התקן זה יכול להיות גדול במיוחד ולהכיל הרבה חיבורים, מימשתי אותו כהתקן רגיל ללא מגבלת גודל.
המבנה שמחזיק את החיבורים הוא מבנה **דינמי**, רשימה מקושרת (דומה למבנה הלוג) שמעדכנת כל פעם את החיבור שהשתנה.
כל שדות המבנה נפלטים בפקודת קריאה להתקן, למעט 2 שדות שאמורים לבחון אם קיבלתי פקטות כפולות(אפשרי לחלוטין ברשת, גם כשלא נתפסים על 2 HOOKS), ולקבל אותן אוטומטית אך לא לשנות את מצב החיבור. (השתמשתי פה בoffset ובid של כל פקטה)
פקודת cat תפלוט את המידע הנוכחי של החיבור, אבל הוא לא קריא במיוחד. כדי להבין אותו – רצוי להשתמש בתוכנית המשתמש המצורפת, שתתרגם את הכל בצורה קריאה.
הנתונים והסבר עליהם:
<Timestamp> <Source_IP> <Dest_IP> <Source_port> <Dest_port> <protocol/state> <Type>
Timestamp - מתי התבצע העידכון האחרון בחיבור
Source\dest IP\port – ברור מאליו
Protocol/state:

לשדה יש יש כמה אפשרויות:

tcp_SYN_SENT_WAIT_SYN_ACK
tcp_SYN_ACK_SENT_WAIT_ACK
tcp_ESTABLISHED
tcp_END

כל שלב ברור – איזה מצב החיבור נמצא כרגע. הוא מחכה לack, או אולי שלח syn או שלח syn ack וכו'

במקביל, ישנו את השדה האחרון:

Type

לשדה זה יש הרבה יותר אפשרויות לפרט את מצב החיבור:

FTP_HANDSHAKE – זוהה חיבור מסוג זה, והוא בשלב לחיצת היד המשולשת
FTP_ESTABLISHED – שלב לחיצת היד נגמר, כעת החיבור בוצע
FTP_CONNECTED - הלקוח התחבר ונמצא בשלב שהוא יכול להעביר נתונים
FTP_TRANSFER – חיבור ארביטרי שנפתח כל פעם שמתקבלת בקשה להעברת קובץ. יכולים להיפתח כמה חיבורים כאלה. ככלל, החיבורים הללו(אם לא עודכנו) מתנקים לאחר 25 שניות שהם ללא שימוש, כדי למנוע פירצה.
FTP_END - הכיוון הנ"ל שלח פקודת סיום

HTTP_HANDSHAKE - זוהה חיבור מסוג זה, והוא בשלב לחיצת היד המשולשת
HTTP_ESTABLISHED - שלב לחיצת היד נגמר, כעת החיבור בוצע
HTTP_CONNECTED – זוהתה פקודה והידר מהסוג הנ"ל
HTTP_END - הכיוון הנ"ל שלח פקודת סיום

TCP_GEN_HANDSHAKE – חיבור כללי (לא פרוטוקולים שקבענו) בשלב לחיצת היד המשולשלת
TCP_GEN_ESTABLISHED - שלב לחיצת היד נגמר, כעת החיבור בוצע
TCP_GEN_END - הכיוון הנ"ל שלח פקודת סיום

נשים לב שהטבלה מבצעת את הדברים הללו (כולל הנחות שאני הנחתי):

- הטבלה מנקה כל חיבור במצב handshake לאחר לכל היותר 25 שניות ללא שינוי(אם חיבור נתקע במצב הזה)
- הטבלה מנקה כל חיבור שהגיע למצב tcp_END לאחר לכל היותר 25 שניות(לשם קריאות הטבלה והפחתת העומס, אין סיבה לזכור חיבורים – לא קיבלנו הוראה מפורשת.)
- הטבלה מנקה כל חיבור במצב IDLE לאחר 9 דקות (ויודעת לפתוח מחדש אם אחרי 9 דקות החיבור מתחדש), גם כאן כדי לחסוך בעומס בטבלה – אין סיבה לזכור חיבורים באמת.

ישנן הערות נוספות שאפשר לראות בטבלה עצמה (היא מפרטת קצת על ערכים מסויימים)

ההתקן hosts

לפי מה שכתוב בפורום, בחרתי לממש את התקן זה כהתקן sysfs.
ההתקן יושב בכתובת /sys/class/fw/hosts וה attribute שאליו ניתן לכתוב ולקרוא כדי לעדכן את הרשימה הזאת נמצא ב/sys/class/fw/hosts/hosts.
דיבור עם ההתקן הזה הוא פשוט – ההתקן מקבלה רשימה של הוסטים כפי שראינו בקובץ הדוגמה, ומכניס אותם למבנה דינמי בזיכרון. אם אחת הפקטות מנסה לשלוח מידע לאחד מההוסטים הללו, היא נתפסת על ידי הטבלה הדינמית, ובעזרת מכונצ המצבים היא נחסמת(עוד מידע על כך בהמשך).

קריאה וכתיבה להתקן זה היא כמו לsysfs רגיל, וכמובן הגישה הכי נוחה היא גישה דרך קובץ המשתמש שסופק.

פירוט על הקבצים:

fw.c:

קובץ זה כמעט ולא השתנה מתרגיל קודם, למעט 2 התקנים חדשים שנפתחו בו. שהם ההתקנים שהרחבתי עליהם קודם לכן. האחד כהתקן sysfs והשני כהתקן רגיל.

Hookfuncs.c:

גם קובץ זה כמעט ולא השתנה מהתרגיל הקודם. למעט שינוי בלוגיקה של תפיסת הפקטות. כעת, כאשר אנו בפרוטוקול TCP והack כבוי, מסלול הפקטה הוא שונה.
קודם כל, חילוץ header הTCP השתנה (הרחבה בסוף המסמך, כולל לינק ושאלה שפתחתי בעצמי בstackoverflow ובעקבותיה הצלחתי לתפוס נכונה את הTCP header.).
אם ה ack כבוי, נפתח חיבור חדש בטבלת החיבורים. לאחר מכן, בהתאם לתנועה, החיבור מתעדכן.
כאשר מגיעה תגובת ה syn ack מהצד השני, נפתח חיבור שמסמל גם את הצד השני(כפי שראינו בכיתה) והם ביחד מתעדכנים ככל ששלבי לחיצת היד ושיחת הTCP מתקדמים.
מתוך קובץ זה נקראות פונקציות מהקובץ stateful_funcs (פירוט בהמשך) ולפי ערכי ההחזרה שלהם, פונקציות בקובץ זה מתנהגות אחרת (למשל אם החיבור לא מתאים, אז לפי ערך ההחזרה הפונקציה parse_packet יודעת אם לדחות או לקבל פקטה, ולרשום על כך בלוג עם הסיבה המתאימה)

Stateful_funcs.h:

בקובץ זה מוגדרים כל המבנים שאיתם אני עובד.
מוגדר המבנה connection, שהוא בעצם מה שמייצג חיבור (כל שורה בטבלת החיבורים).
בנוסף, מוגדר המבנה connection node, שהאובייקט שלו הוא כמובן connection ויש לו מצביע לאיבר הבא ברשימה, next.

Stateful_funcs.c:

הקובץ המרכזי בשלב הstateful.
הסבר קצר על הפונקציות:

```
int check_statful_inspection(rule_t packet, struct tcphdr *tcphd, struct iphdr *iphd
,unsigned int hooknum, unsigned char *tail, struct sk_buff *skb)
```

זוהי הפונקציה שכל פקטת TCP עוברת בה קודם לכן ובודקת מה היא. אם החיבור לא קיים – היא מתריאה על כך (על ידי בחינת הack והשוואת שדות שונים), ולאחר מכן נפתח חיבור חדש. אם הפקטה שייכת לחיבור שמגיע מצד שני של חיבור קיים(הdst==src וכדומה), היא מודיעה על כך ואז נפתח חיבור לצד השני.
אם הפקטה היא duplicate(אפשרי לחלוטין ברשת, ביחוד כשאנחנו יושבים על 2 hooks) היא מתריאה על כך ואז האפליקציה יודעת לא לבצע שינויים בטבלת החיבורים.
אם הפקטה שגויה (למשל חיבור שממתין לsyn ack ומקבל רק ack) היא תופסת אותה, מתריאה ואז האפליקציה זורקת את הפקטה וכותב ללוג.
אם החיבור קיים – היא שולחת את הפקטה לעדכן את רשימת החיבורים(פונקציה בהמשך)

```
connection_node* create_new_connection(rule_t packet, struct iphdr *iphd, int ack_state,
int syn_state)
```

פונקציה זו אחראית ליצירת חיבור חדש. היא קוראת ל2 פונקציות:

```
connection_node* create_new_connection_node(rule_t packet, struct iphdr *iphd ,int
ack_state, int syn_state)
```

שיוצרת node מהטייפ המתאים לרשימה, על ידי מעבר על השדות השונים וסיווג הפקטה (למשל אם היא רואה פורט 21 או 80, היא מסווגת אותם כחיבור מתאים HTTP או FTP וכו')

לאחר מכן, היא קוראת לפונקציה
connection_node* insert_new_connection(connection_node* curr_conn)
שאחראית רק להכניס חיבור לרשימת החיבורים הנוכחית (הרשימה מודפסת מהסוף להתחלה כי אני מתייחס לראש כזנב. לכן בהדפסות החיבור החדש ביותר יהיה למעלה. לכן הוספתי גם ניקיון של רשימת החיבורים לאחר זמן מסויים)

int update_connection(rule_t packet, struct tcphdr* tcphd,struct iphdr *iphd
,connection_node *curr_conn, unsigned char *tail, struct sk_buff *skb)
פונקציה זאת מעדכנת כל חיבור.

אם החיבור הוא חיבור **HTTP**, היא מעדכנת אותו על ידי שליחה לפונקציה
int update_http_connection(rule_t packet, struct tcphdr* tcphd, struct iphdr *iphd
,connection_node *curr_conn, unsigned char *tail, struct sk_buff *skb)
שכמובן מעדכנת אותו לפי סטטוס נוכחי.
אם הוא במצב established וראינו פקודת GET, היא מיד משנה את הסטטוס שלו למצב connected ומטפלת בו (חוסמת HOST אם יש צורך).
כמו שכתוב, התעלמתי מפקודות POST ולא עשיתי כלום במידה וראיתי אחת כזאת.

אם החיבור הוא חיבור **FTP**, היא מעדכנת אותו על ידי שליחה לפונקציה
int update_ftp_connection(rule_t packet, struct tcphdr* tcphd, struct iphdr *iphd
,connection_node *curr_conn, unsigned char *tail)
ופה הסיווג הוא קצת שונה.
קודם כל, השתמשתי בבאפר דינמי שמתמלא כל הזמן עד שמקבל פקודה שלמה (גם כשהיא מתפצלת לכמה פאקטות. הלינק לאיך עשיתי את זה נמצא בסוף מסמך זה) ואז קיבלתי את הפקודה והקודים שהFTP מחזיר.
השתמשתי בcodes של FTP כמו בלינק שקיבלנו. ואם קיבלתי קוד "230", אז זיהיתי שהתחבר משתמש ועדכנתי את החיבור מestablished ל connected.
לאחר מכן המשכתי לעקוב אחר הפקודות.
אם קיבלתי פקודת PORT, מיד פתחתי חיבור מתאים (בשני הכיוונים) כך שאוכל לעקוף את הטבלה הסטטית. בדקתי את זה על ידי חסימה של פורט 20 למשל, ואז העברת קבצים. על כל פקודת פורט שמבקשת להעביר DATA, נפתחים 2 חיבורים שמתוייגים כ FTP_TRANSFER.
כמו שציינתי, חיבורים אלו מתנקים לאחר 25 שניות אם לא נעשה שום עידכון או שינוי בהם.
לאחר מכן, אם זיהיתי פקודת QUIT אני כמובן מעביר את החיבור לEND, ומצפה לקבל FIN ו FIN ACK משני הצדדים כראוי. אם לא קיבלתי מאחד הצדדים FIN (כאשר אני כן מצפה לקבל), אני מיד זורק את הפקטה וסוגר את החיבור בשל חשש לניצול של הפורט הפתוח.

נשים לב שאם החיבור הוא חיבור גנרי של TCP, יצרתי פונקציה שמעדכנת אותו אבל היא ריקה ומיד מחזירה 1.
int update_gen_connection(rule_t packet, struct tcphdr* tcphd, connection_node *curr_conn)
עשיתי זאת בשביל לשמור על מודולריות הקוד. אם מחר כן נצטרך להוסיף בדיוק לכל חיבור גנרי – התשתית במקרה הזה כבר מוכנה (שינוי פעוט ובכל זאת שינוי נח והופך את הקוד ליותר ברור.

כמו כן הוספתי מגוון פונקציות עזר.

כמו למשל is_connection_exists, שבודקת האם קיים חיבור, find_oppposite_connection שמוצאת חיבור הפוך, פונקציות שמעדכנות סטטוס מsyn_sent ל ack_sent וכדומה, ועוד פונקציות עזר שעזרו לי לגשת למבני הנתונים שלי ולשחק איתם.

הנחות נוספות והערות כלליות:

כך למדתי איך לחלוץ נכון את המידע מהskb ולקבל את הheader המתאים:
שאלתי שאלה בstackoverflow:
http://stackoverflow.com/questions/37468119/why-skb-buffer-from-both-hooks-needs-20-bytes-to-skip-and-not-just-the-input-hoo
הפנו אותי לכאן:

http://stackoverflow.com/questions/29656012/netfilter-like-kernel-module-to-get-source-and-destination-address

וכך הבנתי איך להשתמש נכון ולשלוף את הheader נכון(TCP או UDP למשל)

איך לשלוף נכון את המידע כדי לקבל את הhttp headers:

http://vger.kernel.org/~davem/skb_data.html
נקודתית הבנתי שאני צריך להשתמש בskb_copy_bits בעזרת זה:
http://stackoverflow.com/questions/13588639/whats-the-correct-way-to-process-all-the-payload-of-a-sk-buff-packet-in-linux

הקודים שקיבלנו:
https://en.wikipedia.org/wiki/List_of_FTP_server_return_codes

איך לפרסר את פקודות ה FTP כמו שצריך, בעזרת זה:
http://stackoverflow.com/questions/29553990/print-tcp-packet-data