הטכניון – מכון טכנולוגי לישראל

הפקולטה להנדסת חשמל
Networked Software Systems Laboratory

ספר פרויקט
שיפור אבטחת מידע באינטרנט

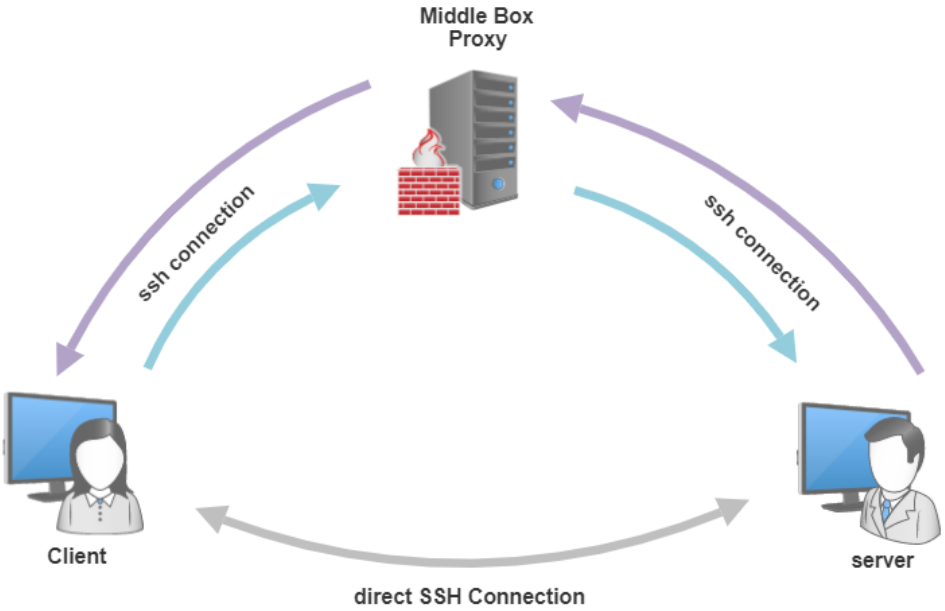
חורף תשע"ח

מאת : עמיד שבלי ואשרף יאסין

מנחה: שי ורגפטיק

Table of Contents

Project Schema



איור 1: סכימת הפריוקט.

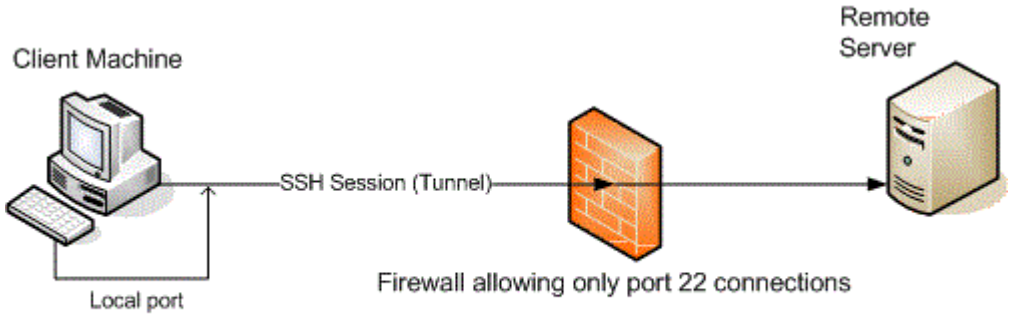
הלקוח רוצה להתחבר לשרת בעזרת SSH. מסופקת לו עכשיו האופציה להתחבר דרך Proxy המאפשר קשר מוצפן ומאובטח.

מבוא

כיום, העברת מידע בין מחשבים (לדוגמה, בין שרת ולקוח) מתבצעת באופן מאובטח ע"י פרוטוקולים כגון SSH, המצפין את המידע ובכך מונע התקפות, כגון man in the middle, בו לגורם לא מאושר יש את היכולת לקרוא את המידע.

הבעיה בפרוטוקול זה היא שהלקוח והשרת מסתמכים אחד על השני שהם לא יכניסו תכונות זדוניות בתוכן המועבר ובמקרים. מכאן מגיע הצורך להקים חומת אש, שניתן לסמוך עליה, בין השרת והלקוח שתפענח את המידע המועבר ביניהם.

עד לפני הפרויקט הזה לא היינו מודעים למה זה SSH, FIREWALL וכדומה, והיום אנו מוצאים שיש בכלים אלו שימוש רב. אלו תוכנות מאוד שימושיות ולדעתנו כל אוהבי המחשבים יכולים להפיק מהם תועלת. זוהי הסיבה שבחרנו בפרויקט הזה, על מנת לסקור כמה מהתכונות השימושיות יותר של SSH, ולהבין איך ה-FIREWALL מתנהג בין שני חיבורים ומה התפקיד שלו. בפרויקט זה לא נדרש רקע קודם בנושא אלא במהלך הפרויקט צוברים את הידע הדרוש שמאפשר להבין את האופן בו הדברים עובדים מאחורי הקלעים בהתחברות לשרתי חוץ או כל מחשב אחר שפונים אליו על מנת לקבל שירות כלשהו.



איור 2: חיבור יכול לעבור דרך Firewall שמחליט אם לאשר את תוכנו או לא.

כדי להתרשם מהעצמה של הפרוטוקול נקדים ונאמר שניתן לנהל מחשב אישי או שרת כמעט לחלוטין בעזרת SSH. לרוב השרתים אין בכלל ממשק גרפי והטיפול השוטף בהם נעשה מרחוק בעזרת SSH. זה יכול להיות מסוכן לשרת וגם ללקוח כאשר אין פיקוח על המידע המועבר ביניהם.

מטרת הפרויקט

בניית חומת אש אשר מאזינה לחיבור ה-SSH בין לקוח לשרת בו מפוענח המידע המועבר בין שני הצדדים, וזאת על מנת להבטיח שהתעבורה המוצפנת אינה מסוכנת .

SSH, או בשמו המלא **Sh**ell **S**ecured, הוא פרוטוקול מאובטח ומהווה את הדרך הנפוצה ביותר לניהול מאובטח של שרתים מרחוק. באמצעות מספר שיטות הצפנה, SSH מספק מנגנון להקמת חיבור מאובטח בין שני הצדדים, אימות של זהות שני הצדדים, והעברת מידע מוצפן ביניהם .

אחת הבעיות המרכזיות בחיבור היא סיכון בהעברת מידע מוצפן שלא ניתן לפקח עליו, למשל ע"י חומת-אש.

כאשר לקוח מנסה להתחבר לשרת באמצעות TCP ,השרת מציג את פרוטוקולי ההצפנה ואת הגרסאות המתאימות שהוא תומך בהם. אם ללקוח יש זוג תואם דומה של פרוטוקול וגרסה, הצדדים יגיעו להסכם ואז החיבור יתחיל.

ברגע שהקשר הוקם, שני הצדדים משתמשים באלגוריתם Diffie-Hellma Key Exchange (אלגוריתם שמאפשר יצירת סוד משותף באופן מאובטח בין שני צדדים) כדי ליצור מפתחות הצפנה סימטריים משותפים אשר ישמשו מעתה כדי להצפין את כל ה-session הנוכחי.

הצפנה סימטרית משתמשת במפתח סודי יחיד להצפנה ופענוח של מידע ע"י הלקוח והשרת. למעשה, כל אחד עם גישה למפתח זה יכול לפענח את המידע המועבר.

חומת האש משתמשת במפתח הסימטרי לפענוח המידע המועבר בין שני הצדדים כאשר בכל session מיוצר מפתח חדש. היתרון בשיטה זו הוא שחומת האש לא תוכל לפענח כל מידע שמועבר אלא רק בהסכמת אחד הצדדים והמפתח יהיה תקף ל session אחד בלבד.

רקע תאורטי

מבנה החבילה :

מודל השכבות הוא מודל סטנדרטי המאפשר לחלק את פרוטוקולי התקשורת לקבוצות או שכבות, כאשר לכל שכבה תפקיד משלה. המבנה של החבילה מורכב מחמש שכבות.

שכבה אפליקציה :

משתמשת בפרוטוקולי transport לתקשורת עם מחשבים מרוחקים.

שכבת transport :

מאפשרת תקשורת ברמה של תהליכים

TCP הוא oriented connection, הוא יוצר session בין שני תהליכים. כך שמובטחים שני דברים:

סדר - החבילות יגיעו ליעדן בסדר שבו הן נשלחו

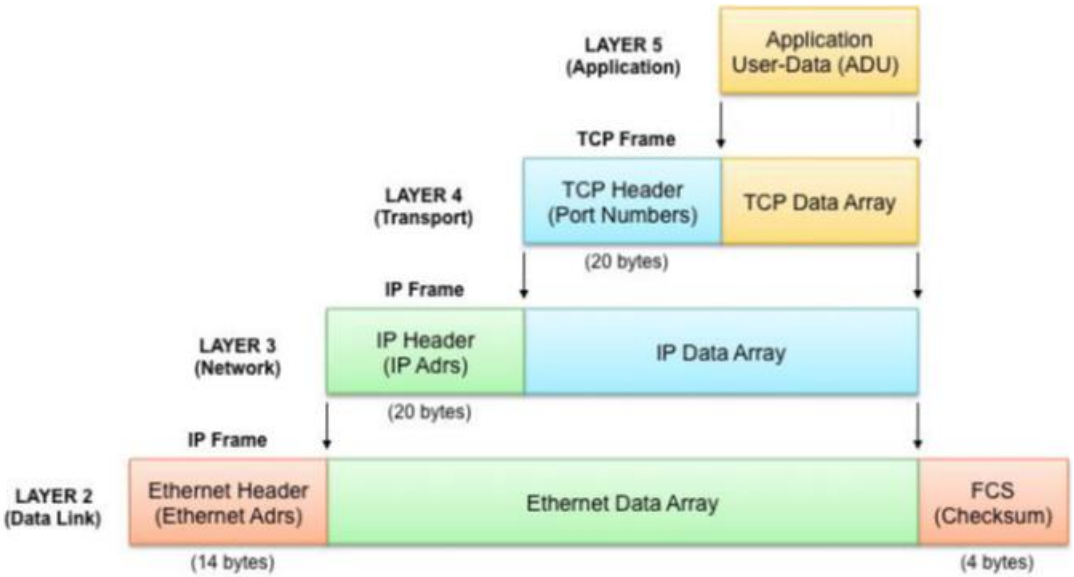
אמינות - כל החבילות יגיעו .

שכבת internet(network) :

ניתוב חבילות מתחנת המקור לתחנת היעד. מאפשר ניתוב חבילות בין רשתות שונות אבל לא מובטח שחבילה תגיע אל יעדה. הפרוטוקול רק מחליט איך לנתב כל חבילה.

שכבת Data Link :

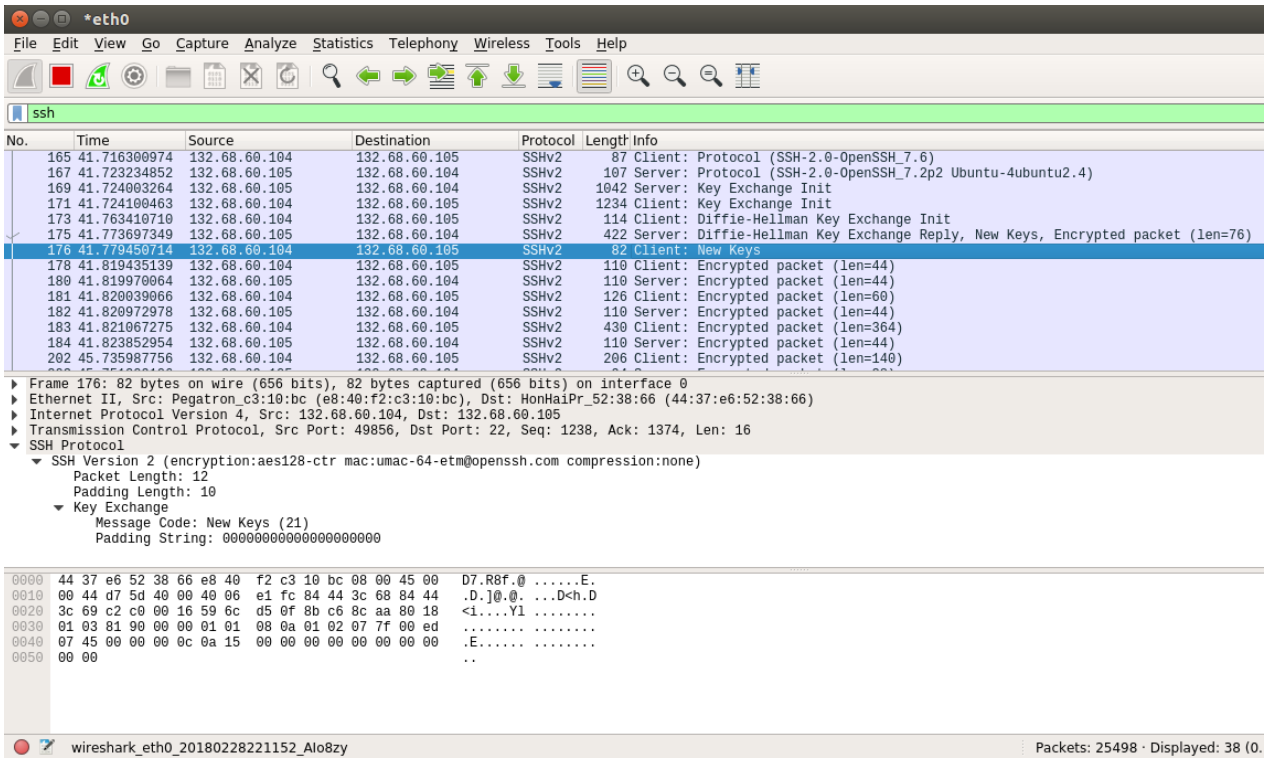
מעבירה חבילה מתחנה נוכחית לתחנה שכנה באותה רשת. מטפלת בפרטים הקשורים לחומרה ולממשק הפיזי .



איור 3 : מבנה חבילת ה- SSH

Wireshark :

תוכנה המיישמת ממשק גרפי לחבילות המועברות בין חיבור רשת. במילים אחרות, מאפשרת ניתוח התעבורה ברשת.



איור 4 : רצף הודעות שעובר בין הלקוח והשרת בעת הקמת הקשר. מודגשת החבילה שממנה ואילך החבילות מוצפנות.

עבור כל חבילה, אפשר לקבל מידע כגון: מה כתובת היעד אליה נשלחת החבילה וממי היא נשלחת.. זה מקל על ניתוח חבילות על מנת לשייך אותן ללקוח או לשרת ובפרט לדעת באיזה שלב מתחיל חיבור SSH ביניהם.

בעת התחלת חיבור ה-SSH המידע המועבר מוצפן והתוכנה מזהה את זה על ידי הוספת מילה encrypted. כך יודעים איזה מידע רלוונטי לפענוח וגם את אורך המידע המוצפן .

```
180 41.819970064  132.68.60.105   132.68.60.104   SSHv2   110 Server: Encrypted packet (len=44)
181 41.820039066  132.68.60.104   132.68.60.105   SSHv2   126 Client: Encrypted packet (len=60)
182 41.820972978  132.68.60.105   132.68.60.104   SSHv2   110 Server: Encrypted packet (len=44)
183 41.821067275  132.68.60.104   132.68.60.105   SSHv2   430 Client: Encrypted packet (len=364)
184 41.823852954  132.68.60.105   132.68.60.104   SSHv2   110 Server: Encrypted packet (len=44)

▸ Frame 180: 110 bytes on wire (880 bits), 110 bytes captured (880 bits) on interface 0
▸ Ethernet II, Src: HonHaiPr_52:38:66 (44:37:e6:52:38:66), Dst: Pegatron_c3:10:bc (e8:40:f2:c3:10:bc)
▸ Internet Protocol Version 4, Src: 132.68.60.105, Dst: 132.68.60.104
▸ Transmission Control Protocol, Src Port: 22, Dst Port: 49856, Seq: 1374, Ack: 1298, Len: 44
▾ SSH Protocol
  ▾ SSH Version 2 (encryption:aes128-ctr mac:umac-64-etm@openssh.com compression:none)
      Packet Length: 32
      Encrypted Packet: a1910edf9d6b6834f9b2b4ac672d9508ebe9243ec6819f14...
      MAC: c74a5d97ba3fa093

0000  e8 40 f2 c3 10 bc 44 37  e6 52 38 66 08 00 45 00   .@....D7 .R8f..E.
0010  00 60 80 82 40 00 40 06  38 bc 84 44 3c 69 84 44   .`..@.@. 8..D<i.D
0020  3c 68 00 16 c2 c0 8b c6  8c aa 59 6c d5 4b 80 18   <h...... ..Yl.K..
0030  00 f9 32 d6 00 00 01 01  08 0a 00 ed 07 51 01 02   ..2..... .....Q..
0040  07 89 00 00 00 20 a1 91  0e df 9d 6b 68 34 f9 b2   ..... .. ...kh4..
0050  b4 ac 67 2d 95 08 eb e9  24 3e c6 81 9f 14 3a 84   ..g-.... $>....:.
0060  6c 5d 12 1b 97 60 c7 4a  5d 97 ba 3f a0 93         l]...`.J ]..?..
```

איור 5 : מודגש החלק המוצפן בחבילה, 4 בתים לפני זה אורך החבילה, שכאן זה $(20)_{16}$=Bytes(32)$_{10}$. ו8 הבתים בסוף משמשים לחתימת ה-MAC.

חיבור SSH :

SSH הוא פרוטוקול לתקשורת מחשבים בחיבור מאובטח. בעת התקנתו ואתחולו על שני מחשבים מרוחקים ניתן לבצע :

- התחברות למחשב מרחוק
- הרצת פקודות על המחשב מרחוק
- העברת קבצים בין בין המחשבים בעזרת פקודת SCP (File transfers using SCP are fully enciphered) דוגמה ' scp 132.68.60.104:Desktop/iv1.txt ~/Desktop '
- מספק חיבור מאובטח
- המידע המועבר בין שני המחשבים מוצפן
- אימות חזק בין הלקוח לשרת כך שלא ניתן לזייף חבילות.

SSH

SSH, או בשמו המלא Secured Shell, פרוטוקול לתקשורת מוצפנת בין שני מחשבים המאפשר ביצוע פעולות ושירותים אחרים בין שני מחשבים המחוברים לרשת דרך ערוץ מאובטח ברשת לא מאובטחת המבוסס על מודל שרת / לקוח.
כדי להשתמש ב-SSH דרושה התקנת והפעלת תוכנת שרת SSH במחשב המרוחק ע"י הפקודה :

```
sudo apt install openssh-server
```

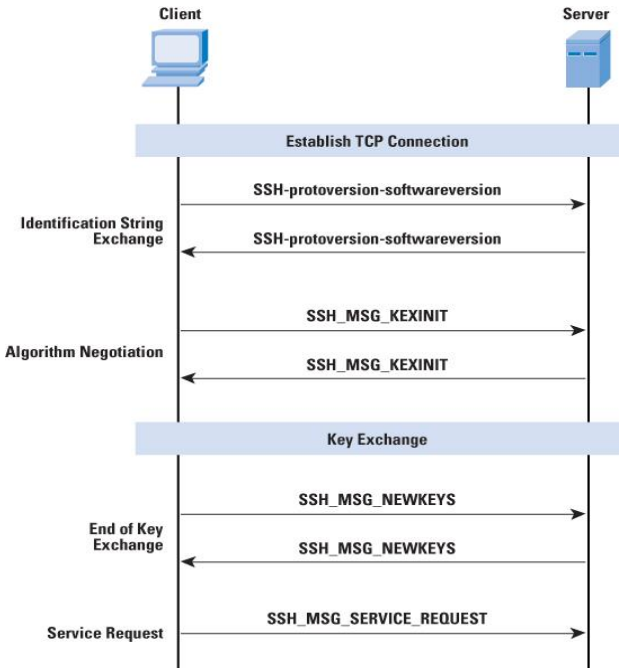
ותוכנת לקוח SSH במחשב המקומי ע"י הפקודה:

```
sudo apt install openssh-client
```

וכדי להתחבר ממחשב הלקוח לשרת משתמשים בפקודה:

```
ssh <<destination_ip>>
```

כאשר <<destination_ip>> היא כתובת ה-ip של השרת.



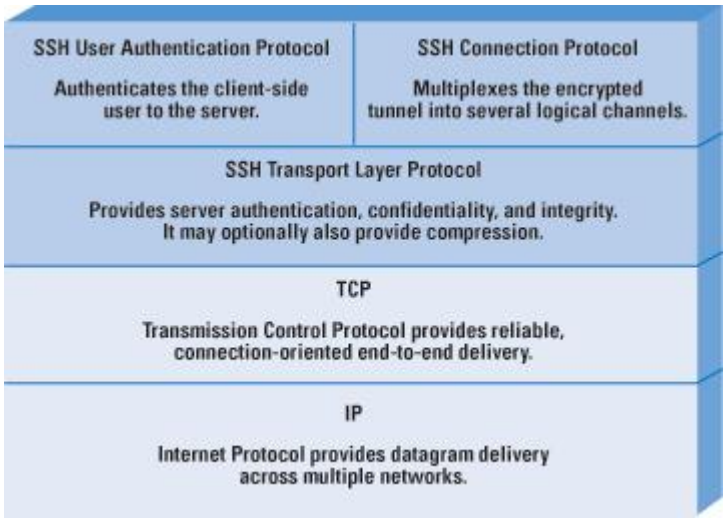
איור 6 : המחשה לסוגי ההודעות שמועברות בעת הקמת קשר.

בהתחלה מתבצעת הקמת קשר בין הלקוח והשרת בו הם מסכימים על אלגוריתם הצפנה משותף שיעבדו איתו. כל אחד עובר על רשימת האלגוריתמים של הצד השני והראשון המשותף נבחר. יש אפשרות לאלץ את השרת לבחור באלגוריתם ספציפי כל עוד השרת תומך בו.
אחרי הקמת הקשר נוצרים המפתחות הסודיים של ה session הנוכחי, "במקרה שלנו אנחנו בחרנו באלגוריתם aes-ctr 128", אשר עובד עם שני מפתחות, אחד key

ייחודי, ו-IV שגדל באחד עבור כל שליחת חבילה. בהמשך נפרט גם על כל מפתח ואיך האלגוריתם מספק את האבטחה הנדרשת.
כפי שנאמר לעיל, הפרוטוקול הזה עובד מעל שכבת ה-TCP כדי למנוע אובדן חבילות ושיגיעו בסדר הנכון, וזה בגלל אופן עבודת האלגוריתם שדורש סדר בין החבילות מכיוון שבכל שליחת חבילה הלקוח משתמש בערך מסוים של IV ובכל חבילה הוא מגדיל אותו באחד. מהצד השני השרת עובד עם אותו IV ולכן הוא חייב לפענח את החבילה עם ה IV שבו הלקוח הצפין. לכן גם הוא מקדם אותו בכל חבילה שמגיעה כדי שיהיה עקבי עם הלקוח. כלומר אם מאבדים חבילה אז השרת יאבד הסנכרון בינו לבין הלקוח ואז לא ניתן לפענח נכון. בנוסף, כאשר החבילות מגיעות בסדר לא נכון אז השרת יפענח חבילות אלה עם מונים לא תואמים ולכן לא יצליח לפענח החבילה! מכאן הצורך בעבודה עם פרוטוקול אמין כמו ה- TCP הוא הכרחי.

מבנה חבילות

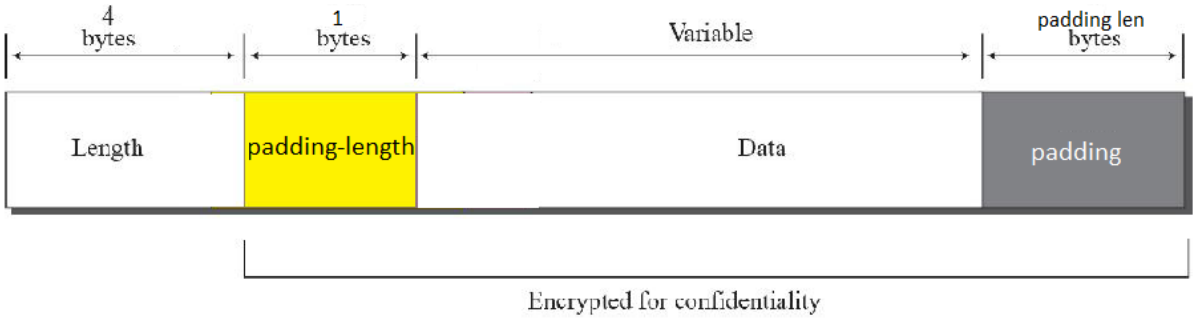
כפי שהוזכר, פרוטוקול ה- SSH עובד מעל פרוטוקול ה TCP אשר מבנה החבילה מורכב מ:



איור 7 : שכבת ה-SSH מעל שכבת ה-TCP.

כל המידע עובר דרך פרוטוקול הTCP כדי למנוע איבוד חבילות או הגעה בסדר לא נכון ובך הפענוח יתבצע בצורה נכונה.

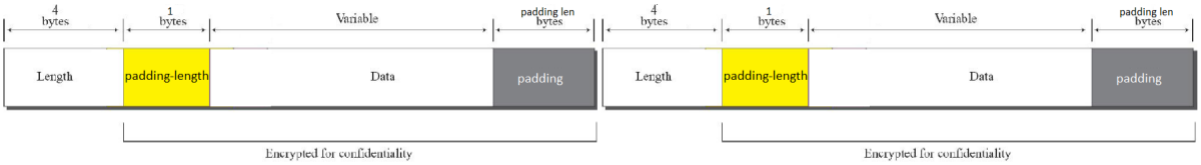
SSH Packet Format



איור 8: מבנה חבילת SSH בשכבה העליונה.

בשכבת האפליקציה מתווסף לחבילה ה-lenght שהוא אורך החבילה שתוצפן בהמשך בגודל 4 בתים, ומתווסף אורך הריפוד padding-length בגודל בית אחד כאשר את הריפוד מוסיפים בסוף החבילה על מנת לרפד את המידע לגודל בלוק הצפנה (שבאלגוריתם aes128-ctr זה 16 בית). הריפוד יכול להיות רנדומלי.
החבילה הנ"ל תוצפן בלי שדה ה-Length ונשלחת מעל שכבת ה-TCP.

בפועל החבילה הנשלחת יכולה להיות מורכבת מיותר מחבילת SSH אחת, שזה מאפשר ניצולת יותר גבוהה של הערוץ. דבר זה מתואר באיור הבא:



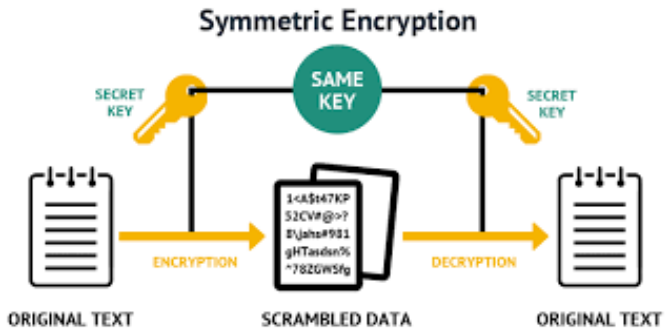
איור 9: החבילה יכולה להיות משורשרת מכמה חבילות מוצפנות.

הצפנה

ישנם שני סוגים של הצפנה :

1. <u>הצפנה סימטרית</u> :

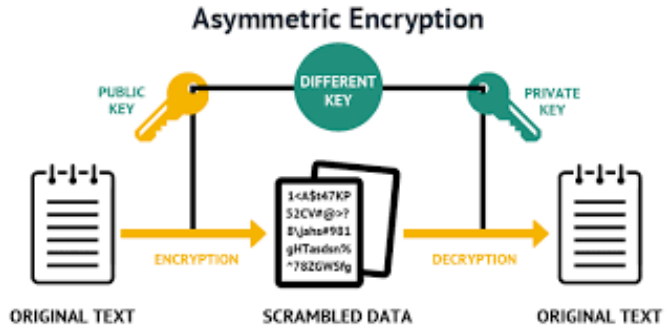
זו שיטת הצפנה שבה משתמשים במפתח הצפנה יחיד גם להצפנה של המידע וגם לפענוח של המידע המוצפן. בפועל המפתח הוא סוד משותף לשניים או יותר משתתפים .



איור 10: ההצפנה ופענוח המידע מתבצע ע"י אותו מפתח סימטרי.

2. <u>הצפנה אסימטרית</u> :

היא שיטת הצפנה שבה מפתח ההצפנה שונה ממפתח הפענוח. כלומר, כל משתמש מכין לעצמו זוג מפתחות: מפתח ציבורי (Public key) שהוא מפתח הצפנה הנגיש לכל ומפתח פרטי (Private key) , הנשמר בסוד ומשמש לפענוח. כך שלכל מפתח ציבורי קיים אך ורק מפתח פרטי יחיד המתאים לו. כדי להצפין מסר בשיטה זו על המצפין להשתמש במפתח הציבורי של המקבל, שבעזרתו הוא מצפין ושולח לו את המסר. רק המקבל מסוגל לשחזר את הטקסט המוצפן בעזרת המפתח הפרטי המתאים שברשותו. החוזק בשיטה הזו שרק המחשב שיצר את המפתח הפרטי יכול לפענח את המידע והמפתח הזה לא מועבר דרך הרשת וגם את אי התלות עם המפתח הציבורי שהוא נגיש לכל אחד. וזו בניגוד לשיטת הצפנה סימטרית, שבה מפתח הפענוח זהה למפתח ההצפנה.



איור 11 : ההצפנה מתבצעת ע"י המפתח הפומבי ופענוח המידע מתבצע ע"י המפתח הפרטי.

ניתן לחלק את הצפנים הסימטריים לשני סוגים עיקריים:

צופן בלוקים **(Block cipher)**

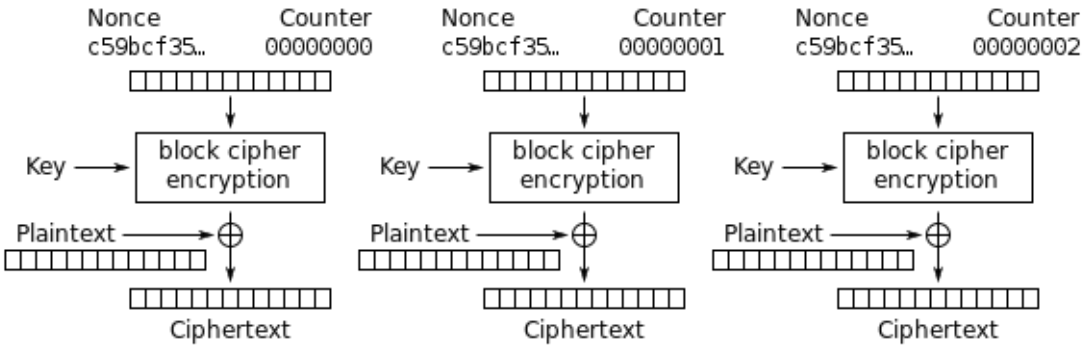
אלגוריתם המכיל פרמוטציה קבועה, ללא זיכרון, שמקבל מחרוזת סמלים מתוך האלף-בית וממירם לרצף אחר של סמלים מאותו אלף-בית, בדרך שנקבעת לפי מפתח ההצפנה. רצף הסמלים נקרא "בלוק". כל מפתח מניב תמורה אחרת מתוך כל התמורות האפשריות. לדוגמה בצופן AES הבלוק הוא באורך של 128 סיביות ומפתח באורך של לפחות 128 סיביות. משום כך האלף-בית שלו מכיל 2^128 תמורות אפשריות בהתאם למספר המפתחות האפשריים. אף על פי שסך כל התמורות האפשריות הוא !(2^128) כיוון שגודל הבלוק גם הוא 128 סיביות.

צופן זרם **(Stream cipher)**

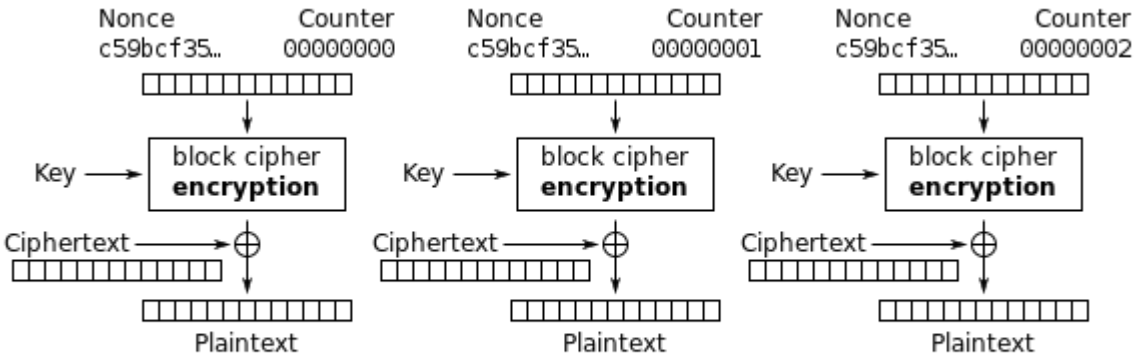
אלגוריתם עם זיכרון פנימי, שמקבל רצף סמלים בזה אחר זה ומצפין אותם באמצעות מפתח הצפנה לרצף סמלים אחר מאותו אלף-בית, עם טרנספורמציה המשתנה במהלך ההצפנה בהתאם לתוצאת הצפנה קודמת. בצופן זרם הסמלים יכולים להיות בגודל סיבית אחת, בית אחד או מילה אחת. בדרך כלל צופן זרם מהיר יותר מצופן בלוקים ומתאים במיוחד בתקשורת אלחוטית או כאשר אורך המידע המיועד להצפנה לא ידוע מראש.

יש מגוון רב של אלגוריתמי הצפנת ה session ואנו השתמשנו באלגוריתם הסימטרי מסוג הצפנת בלוקים AES-CTR128 (Advanced Encryption Standard - Counter 128)

התמונות למטה ממחישות איך נעשית הצפנת המידע עבור הבלוקים וגם הפענוח של המידע המוצפן.



Counter (CTR) mode encryption



Counter (CTR) mode decryption

האלגוריתם משתמש במפתח ומונה בגודל 128 bits להצפנת ופענוח המידע כך שהמפתח יוגרל מחדש בכל session, והמונה מורכב מחלק קבוע מוגרל בכל הקמת session וחלק נוסף הגדל באחד עבור כל בלוק מידע מוצפן כך שאם שולחים הודעה זהה מספר פעמים ברצף המידע המוצפן יהיה שונה לגמרי ולא ניתן להסיק מה ערכו של המונה או המפתחות. זו שיטה חזקה מכוון שהמפתח לא מספיק כדי לפענח מידע אלא גם צריך מעקב אחרי כל החבילות הנשלחות.

להלן תמונה של חבילה מוצפנת ועוד אחת אחרי הפענוח בסקריפט הפענוח כך שהחבילה מכילה את התוכן של הפקודה "ll" שהלקוח בצע על השרת:

החבילה המוצפנת :

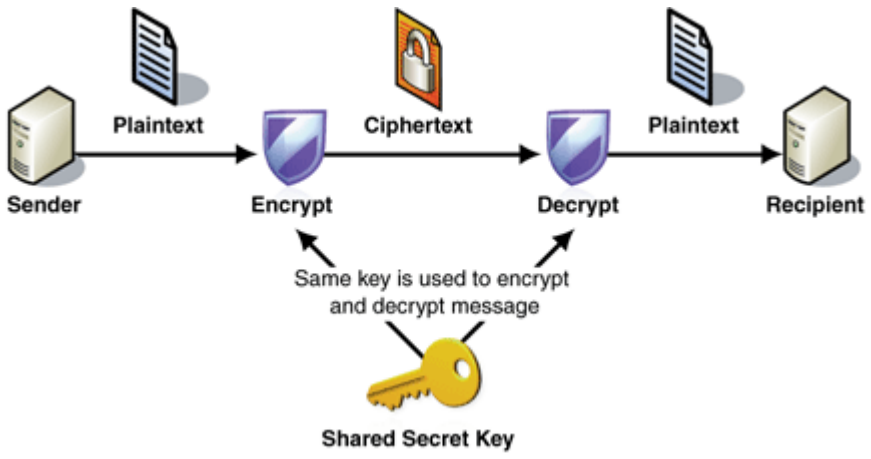
```
|')R\xfej\xa79\x86hm\xc0a0\x0b\xbf\xcee*=R3\xf1\x00\x11H\xa9d\x8a\x19c\xc3k\x92\xe2\x80\x14\xa1Z\xe2
\x9c\xd7\xaf\x92\xbf\xe4O\xa4\x80\x19:(x\xc3`\x7fR\xbe\x1c\xa8\xe2\xe2\xf7x\x1ar'"\x1f\x03\xe2*\xcc
\xea#\x9c\x11,\xab\xd29@8\x00\xd6\xab\xa0b\xe0\xd6l#\x9a\x0fd\|;-s'"'\x96e\x98\xba\x0bEQ\x96\x89\x88
\x05Q\xd0\xb6\x9f\xfb\xd8|:\tn)\x16p\x9d=\x87\x19\xa8+\xb3s'"$;w\xff\xc9\xe3\xb0J\xc8\x8a\x14j\x13\xb3
\x92&\x90>4)\x1d\x03\xe8X\x05\x9e\xaf\xdd\x96\xe8\x12\xcc\xa4\xbac\x0c\x81\xd8\x008c\x9dZoj.\xc8A\xed
\x1d]\x9f\xee\x95E\xb9\xa3\x7f\xbb\xadX\xf1M\xa4^]\x82\x80}\xb4\xb8\xdf\x9f\xad\xc2\xb9\x19Vg]\xc4
\x9e`&\x04\x82\xc9\xb9\xf6\x7f@{\x8c*\xfa\x88\xfc\xb3\xa9\xf5\xf6DH6\xb7\xd9P\x1bP(\xe9f<\xefF=0\x8c
\x1a\x8f\x0b\x13\x89\xbc\xabs\xca\xf0\x8d[\xf0b\x13\t\xcbZ\xd2C8\xf0\x11T\xd4\x82Z\x8f\x04\xa2\xb5
\x16O\xc9pC\x012\xc0(\xd5|\xd8\x06U\x94\||\xb4o\xa2d\xbd:\x8f\xd0z\xb8!=H\x11kZa~\xee\x02\x8d\x9d\x860]
\xef\x98\xf0A@?e7\xe8\xabZS\x11\xe0\xa9\xa3\xeb\x9cP\xc9'm\xe7\xf0\xe6\xc9\x00\xfaO\xe6\x99&}\x1bM
\x9c-\xc5z {\x1b\xd5\xc5\xe6\xc5\xb7\xb5*\xa5\xf1\xd9\xca\xa0\xc5\xda\x93ti/\r\xfb\xa2\x84\x14\xda\x1a
\x97zz\xf5E\xa2\xa0\x190\xf1i\x13\xder\xf2\|`/l\x97\xa3\x0f\x8e\x08\x9f\xafB\x13!\x02\xae\x06\xde\x11
\xb8\x111\x8e\xd1\xf9\xd7#V\x0b?\xe9cB\x94\xb9\xf9\x9f|v\x14\xb0p\x85!H\x01\xf6\xf2"'\x80\x1dZ\xd1:D
\xa5\x88\xaf\x13}xB\xcb\xce\xff''sI\x93\x84\xfdb\x13\x1c\xce\'>\xde\xd52\xcfq\xc4\x95\xbbWJ3\t\xbb\x9d
\x8ac\xdcQ\xab\xc4\xad\x07o\xfd@G\xfb\xa8\x9c\xdc\xb5s\x0b\xcdU\xe1\xd3\x83U\x98\xdb\xf2\xbb,h\xeb
\xe5"\xf2\x9d\x9b\xecQ\xb8S\xb2\x80\x9f\xb4t\x12"{dE\x90}\xadU_\xe9\xd7\xe4\n\x95&\xd8\xc66\x99\x83
\xba\xe4w\xf7=\xa4\xda\x0cGF%+\xa6\xd8\xaa\nH\xa1%\xd1\xf1\xc5\x0f\x14l.\x9e\x05\x9f\xf3c\xc0\x1fW
\xf3t\xd5|\xcem\x055\xbe\xf2q\xb9<\x92&\xb0\xda\x84Z\xf7<\xa8\xca%\xd9\x95\x16\xd7\xef\x8c\xe7\xaf\x0f
\xe5\x8c\x10H\xa2\xf5"\x02\x16h\xba\xad)\x125n\xef\xa3\xcaup/\xc6\xe7\xc9w\xe2n\xc2\xfc\xe70\xf70\xcc
\x08\xbb\'\xbbu\xa6:\xac8n\xa0c\xbc\x15\xa1\x99\xf3lb\xf5\x06F\x0e\xef\x10> \xa6\x88\xe8y\xcd\x88U#
\xb9]\x06\xdc\xb9=\xe8w\xd2]\xb8\x93\xc7\x18\xcd\x03\x86C\xfe\x1b1\x9c\xe3\x93\xc2}\xa5w(xI\xbd\xe6
\xb0\x8e\xbb<\|zY\'\x97O:&Z\x08i\x14\x15)\xbf\xe5lr;\xc0Y\xbc\xc3\x85\x0b\xe4\xec\xfbNH\x9b~[\x1b\xd5
\xb1\xb8\xc4b\xd0A^\xca\x11U\xd9\x1d\xc2\xf5w\xc319\xa8\xba\x00|\xe4$.@m\xcd\x1etX!\x10*\x97\xae\xee\t
\xb3\xcb]=\xa9\x19\xfd\xf5\x1f\x97b\xba\x13\x00\xd7\x96\xe9@\xef\x16\xe3-\xc1\x8d\x8c[\x85\x139o\x13o-
\xbd\x89\x06\xad\xa8\xb0\x85\xcdv\xbb\xb5;\x83\xb2F\xf5s\xbcjP\xefs\x0b\xcb\xee\x08\xb0]\x90\xb3\xda<I
{\xd5\x88_jVC\x8bvFS(l=\x9c\x85Q\x038\xfb\x8c o}\|7\xd9\x1b\xdd\xdan\x97KD\x867\x89\xf3\xb1\xfb\xbe4
\x1d\x178ae\xc0\xd5\xc56\r\x8bAi\xe9\xe4\xf5u9_SQ]\xee\x9c\x95h\xbe\xdeo\xb9\xbe\x9b\xa9\x99m\x80\xdd
\xd5\xf4)Hxu\xbf\x0f\x83\x08q\xaeFbt \xc0\xcd6d\xa3\xb5@\x9dnVu\xdat\xa93/\xb9\x15{\xbe(\x00\xf2\xf5
\xf7\x91\xa5\x18\|\xa2j\xd3GLx?\xbf\xbf\x7fo>\x8cS\xe9\x86B\xd2\xd4=\xc1\xedj<5\xd2\xef \xae\x07\xf1
\xec\xaen\xa9\x13\xc0\x90\xd5\xd8\xdf\xe3\x95Hf\xfa\xc4\x02\xe2\x7f\xc1\xe6;\xfc&5\x90\x01m\x10\x19
\x8c.=u\x97d\x1b_\xcd\xf2\xc0\x1d\x93\x89\xadTk\x88\x89\xca\x14\xf5G\x83\x81\x10\xbf^Z\xd2\xifb,\xc8
\x11\x84(uG\xd9\xa5XT\xf7#A\x94\xe6\xc8\xffj\xa8t\xa7\xca\xb3\xd585,\xfb{iY\xbb\x8c\xb0\xe6\x1c\x87E3
\xb8\xd5\xf7\xbe\x88\x1fj\xac\xc9\x06/\xe0\r\x16\xfc\xb3T\xa2\xf8\x9c1\xec\xd9)\x88{q\xf6\xa2\xf4y_w
\x08\xc5Ie\xecZt\xd7:cz\x97Sd\xca}AF\x92\xcev \xd7\xcdQ\x04y?\x7f)M\xa8\x02\x04\xae\xb1\x86\x817r\xee
\xc3\x8a\n\x14o\x01\xe6\x91\xc2#\x91\x87r\x0fs\x9f\xa8\xb7\x83\xd0x\xe8\xec\xfa\x87\xa8o|\xf9:\xe6\xff
\xd9\xf3\xd4&\xce\xe1O@\xd6\xd7\xf2\x0e\xecg\x97\'\xd5\'\xf3\xa4\xa2\xe8>\xa6I\x151\xd9}\x10\x98D\xa9
\xa1\x8d\xd3\xcf6e6\xb4K\xf8\x10PaD\x93\x10\x9b\x8bp\xba\xfe}\x1dk\xc1\x92g\xbc}\x9c\xcf \x15\x93\xaa!
#\xaet\xe9\xe1?\x93\nU;\x9e\xac\x81\xfb\xb2\xa1(W\x8cW5Y*.\xee%E\xae;\x9dh\xfe\xa6\x9e\xcc\xf85\x1b
\x1a jl\x92\x156\xee0\x01. \xcc\xb4;l\x81\x84\x0f\x1d\xeb\xd6\'#M\xab\xb7\xfe0\x86\xf7\xbc\xa0\xb9
\xfc-\xbb\x9b\xc7\xe5W\xa4\\l\xa3C\xfa\x99\xd02F\xc3\xdb''\\\xb3\x1a\xe9\x1b4\x05)\xd7\xb2k\x97\xad
\x08\xb9\x90|\xe3\x0e\x83L\xcexV\xd4\xf6\xdd\xe3\x8d\x97L\xbf''X\x1a#\xe8\xa2\x16[ Zm\x90b
\x86;A^''\x82\xb0\xf6\r\xa4\x89\x1a\x9d\x90\xe7\xb9\xe2\x16\x97C\xa9''6\xf5\xe1-\xffpc)\xb4\x1c-
\x04l"^\x7f''k\xdb|aKi\x1b\xa3\xbd\xe2fA\xc0\xf3Ad\x9dJ\x9d\xda\xf2?4\x0bd\xf4ob\x02\xd4\xb4"W\xb8\xd4
\xc5\x1f\xf0&FKl\xf8\x85\'o\x9eJ\xc9\xf1R\xbeTU\xd3\xa2\x96\xf1\xf5\x98\xf2\xba\xa0\x91\xd9\x1ba4\xc3n
\xac\xf4ez\xf3\x1c\x7f\xddp\xa9\xffv\xd3|-\x14RM46S\xc5\x02\x15\xbc\'\x00\x12Mf\xbf\xcf\x10\xecu\xe7O
\xed\xfc\xea|\xac)z\xc5\n\xea\xdc\x8e\x16\xfc\x18\xb9&\x92>s)F\x00\x08}\xca\\\x88\xc4\x02\xb8\xeb\xca
\xce&\xc5!$\xa6\xe0P\xdaGbd\xb0{+\x8fk\x8bM|\xc3k\xb0H\xc8\xe0\xd5\x10\xd7\xc9U\x19\x9e\xd1\x1e\x8b
\xac`\xb8\'\x1f\xe7\x9av1\xfb\xe7\x93\xb6r:&\x1e\xa5\x8fI\x9b\x91q\x0ek\xa8\xed\xc1\x8bKA\xf2*\x01
\xbed\xcf\xa7\x7f\x18\xaf\xc6\xc6\xf4\x8b\x92`\x08\x1f\x13\x84\xfdT\x87/+\xf1\xd3\xcd\xe55\xa3\xe9)
\x06v\x0c\xfe^\xd5\xc3\xd9\x03\x00\x9d"\xd7\xc7#\n\xc4\x00Z\x1e\x0e\'\x877\xe6\xc4X\x12\xb0\xb9\x08Q
\xccP\xa6\xefB \x9c,\xe3A3\xe2\xa9DIRq\n\x9d\xdc\xac\xf8Z\x19\x8a\xd6\x06\xa9\xf9A\xe2\x80\xc3\x96\x06
\n\xe2\xae\xe0M\x96MG\x87\x924\x98\x91p\xcc\x82\xff\xaa\xaf\x1d\xe5g\'?\t\xbb/\x9a \x91\xcc\xa9\xae
\x1a\xb1\x90\x8e\x1f\xb2}\xcdl\xc1c\xa3''\xbe\xeew\xda\t\x89\x13\xfb(\xf3\xce\xef\xf4\x9e\x11,\x06\x1d
\xd0\xfe\x84"\x85\n\xf0>^\xce3Y (\xc3\x1adX\xae\xae\xc0D\x82KBG\xe4\xff\'\x18d~\xec>\xbc\tXb\x03D^\x1e
SD\xc4\xf2\x9d\x03=\xc7K\x12=\x16\xd5\x80f\x1b\xd1\x14=\xe3\xd7\x11\xeb!X\xee\xcf\xf25\x99.\xc1
```

אחרי הפענוח:

```
total 156
drwxr-xr-x 26 sshproj sshproj 4096 22:07 14 ינו ./
drwxr-xr-x  5 root    root    4096 04:00 9 ינו ../
-rw-rw-r--  1 sshproj sshproj    0 22:08 14 ינו a.txt
-rw-------  1 sshproj sshproj 3042 21:52 14 ינו .bash_history
-rw-r--r--  1 sshproj sshproj  220 19:12 12 נוב .bash_logout
-rw-r--r--  1 sshproj sshproj 3771 19:12 12 נוב .bashrc
drwx------ 15 sshproj sshproj 4096 20:51 17 דצמ .cache/
drwx------  3 sshproj sshproj 4096 19:15 12 נוב .compiz/
drwx------ 20 sshproj sshproj 4096 19:24 22 נוב .config/
drwx------  3 root    root    4096 17:58 16 נוב .dbus/
drwxr-xr-x  4 sshproj sshproj 4096 14:31 1 ינו Desktop/
-rw-r--r--  1 sshproj sshproj   25 19:12 12 נוב .dmrc
drwxr-xr-x  2 sshproj sshproj 4096 19:12 12 נוב Documents/
drwxr-xr-x  3 sshproj sshproj 4096 11:20 17 דצמ Downloads/
drwxrwxr-x  3 sshproj sshproj 4096 17:56 16 נוב .eclipse/
-rw-r--r--  1 sshproj sshproj 8980 19:12 12 נוב examples.desktop
drwx------  2 sshproj sshproj 4096 10:49 17 דצמ .gconf/
drwx------  3 sshproj sshproj 4096 19:24 22 נוב .gnome/
drwx------  3 sshproj sshproj 4096 21:15 14 ינו .gnupg/
-rw-------  1 sshproj sshproj 6092 21:15 14 ינו .ICEauthority
drwx------  3 sshproj sshproj 4096 19:12 12 נוב .local/
drwx------  5 sshproj sshproj 4096 17:42 16 נוב .mozilla/
drwxr-xr-x  2 sshproj sshproj 4096 19:12 12 נוב Music/
drwxr-xr-x  2 sshproj sshproj 4096 19:12 12 נוב Pictures/
drwx------  3 sshproj sshproj 4096 19:24 22 נוב .pki/
-rw-r--r--  1 sshproj sshproj  655 19:12 12 נוב .profile
drwxrwxr-x  3 sshproj sshproj 4096 21:39 22 נוב projA/
drwxr-xr-x  2 sshproj sshproj 4096 19:12 12 נוב Public/
drwxr-xr-x  2 sshproj sshproj 4096 20:56 22 נוב .rpmdb/
drwx------  2 sshproj sshproj 4096 22:38 22 נוב .ssh/
-rw-r--r--  1 sshproj sshproj    0 20:19 15 נוב .sudo_as_admin_successful
drwxrwxr-x  3 sshproj sshproj 4096 17:56 16 נוב .swt/
drwxr-xr-x  2 sshproj sshproj 4096 19:12 12 נוב Templates/
drwxr-xr-x  2 sshproj sshproj 4096 19:12 12 נוב Videos/
drwxrwxr-x  5 sshproj sshproj 4096 10:52 17 דצמ workspace/
-rw-------  1 sshproj sshproj   56 21:15 14 ינו .Xauthority
-rw-------  1 sshproj sshproj   82 21:15 14 ינו .xsession-errors
-rw-------  1 sshproj sshproj 1250 16:54 14 ינו .xsession-errors.old
```

Encryption Keys

באלגוריתם ההצפנה בפרויקט הזה משתמשים במפתח משותף לשני הצדדים כך שהשולח מצפין את המידע בו והמקבל מפענח אותו בעזרת אותו מפתח. קיימים שני מפתחות ייחודים המיועדים להצפנת ופענוח המידע **Key**,**IV**. אשר הלקוח והשרת מייצרים בעת הקמת הקשר בצורה מאובטחת וסודית.

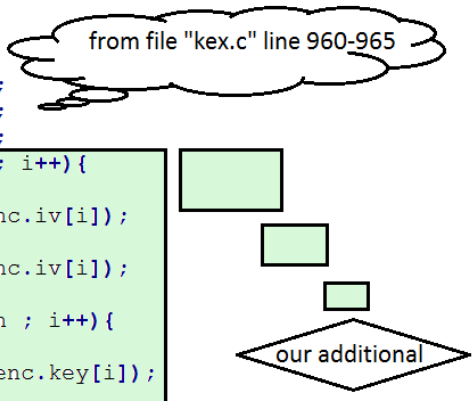


איור 12 : באלגוריתם aes_ctr128 משתמשים במפתח סימטרי.

לכל צד מיוצרים שני זוגות של מפתחות:
(**Key1**, **IV1**) אשר משתמשים בו להצפנה ופענוח מידע של הלקוח.
(**Key2**, **IV2**) אשר משתמשים בו להצפנה ופענוח מידע של השרת .

כדי שה- **Middle Box** יוכל לפענח את החבילות שעוברות משני הצדדים הוא צריך לדעת את שני המפתחות האלה .
לכן כדי לספק לו אותם, הצטרכנו לגשת לקבצים של **פרוטוקול ה-SSH**, בפרט לקבצים של ייצור המפתחות של אלגוריתם ההצפנה שלנו ה-**AES-CTR-128**, וביצענו שינויים בקבצים האלה כך שבעת יצירת המפתחות אנחנו שומרים אותם בקבצים :

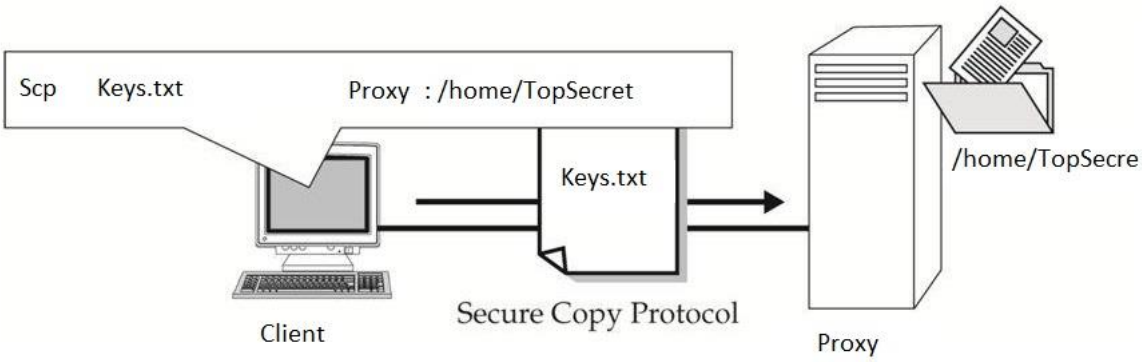
```
for (mode = 0; mode < MODE_MAX; mode++) {
    ctos = (!kex->server && mode == MODE_OUT) ||
        (kex->server && mode == MODE_IN);
    kex->newkeys[mode]->enc.iv  = keys[ctos ? 0 : 1];
    kex->newkeys[mode]->enc.key = keys[ctos ? 2 : 3];
    kex->newkeys[mode]->mac.key = keys[ctos ? 4 : 5];
    for(int i =0 ; i<kex->newkeys[mode]->enc.iv_len ; i++){
        if(mode==0)
            fprintf(iv1,"%02x",kex->newkeys[mode]->enc.iv[i]);
        else
            fprintf(iv2,"%02x",kex->newkeys[mode]->enc.iv[i]);
    }
    for(int i =0 ; i< kex->newkeys[mode]->enc.key_len ; i++){
        if(mode==0)
            fprintf(key1,"%02x",kex->newkeys[mode]->enc.key[i]);
        else
            fprintf(key2,"%02x",kex->newkeys[mode]->enc.key[i]);
    }
}
```



את המפתחות אנחנו שולחים בדרך מאובטחת ל-**Middle Box** :

ע"י הפקודה :

```
Scp <<Source_path>> <<Destination_path>>
```

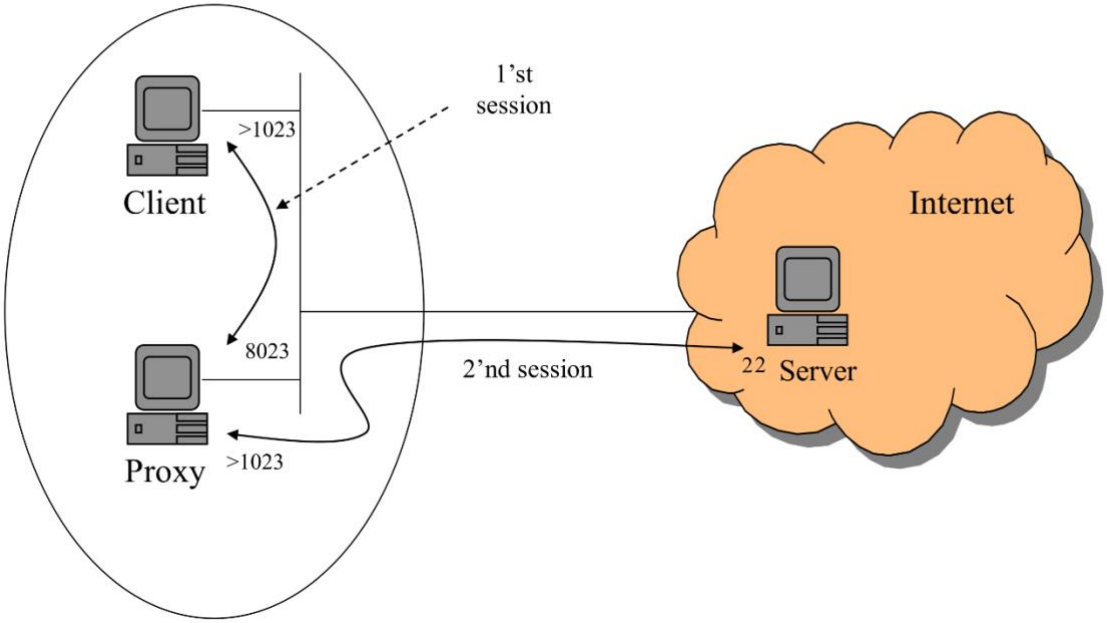


כך מעתיקים את המפתחות מאחד המחשבים ל-**Middle Box** שלנו ומשתמשים בהם בכל **Session** חדש.

Man in the middle-Proxy

ה-proxy משמש כמתווך בין השרת והלקוח והוא מקבל את כל בקשות התנועה מהלקוחות שמבקשים להתחבר ב SSH connection, אם הוא מחליט להעניק גישה, ה- proxy שולח את המידע לשרת. תשובת מחשב היעד נשלחת אל ה- proxy, אשר בודק את המידע ושולח אותו ללקוח וכך הלאה . ובאופן זה התעבורה בין השרת והלקוח מתבצעת דרך ה proxy שהוא מנהל התעבורה ומאבטח אותה.

חומות אש של proxy יכולה לבדוק תוכן המידע באופן מלא ולקבל החלטות גישה על סמך מידע ספציפי ומפורט יותר.



ה Firewall שלנו או ה Middle box מאזין לחיבור בין הלקוח והשרת בעת הקמת החיבור ומאפשרים לו לקחת את המפתחות של ה session הנוכחי. בהתאם הוא מסנן את החבילות הרלוונטיות לחיבור ביניהם ומפענח את המידע המוצפן המועבר ברשת האינטרנט ומדפיס אותו בזמן אמת כך שכל פקודה שעושה הלקוח ניתן לראות ב Middlebox. ה Middlebox רלוונטי רק עבור ה session הנוכחי כך שבחיבור אחר מפתחות ההצפנה משתנים ואז אם הלקוח לא מעביר המפתחות ה Middlebox לא יכול לפענח יותר את המידע ויתכן שיחליט להפיל את החיבור, זה יתרון מלתת את המפתחות הפרטיים של הלקוח למתווך, ואז ה Middlebox יתאפשר לו לפענח כל חבילה בלי הסכמת הלקוח וגם לא יהיה מודע שיש מתווך שמאזין על החיבור בניגוד לרצון הלקוח שלא רוצה לחשוף לאף אחד. יכול לזייף הודעות ולהעביר מידע לא נכון .

כדי לאפשר תקשורת בין הלקוח והשרת דרך ה proxy, מפנים כל בקשת התחברות מהלקוח ל proxy ומשם ה proxy מפנה החבילות לשרת שהלקוח ביקש. מה שעשינו הוא ששינינו בהגדרת מחשבי הלקוחות שכל חבילת SSH שמזוהה עם destination port 22 תופנה ל port ייחודי שהגדרנו למחשב ה proxy שהוא מאזין עליו על ידי הפקודה:

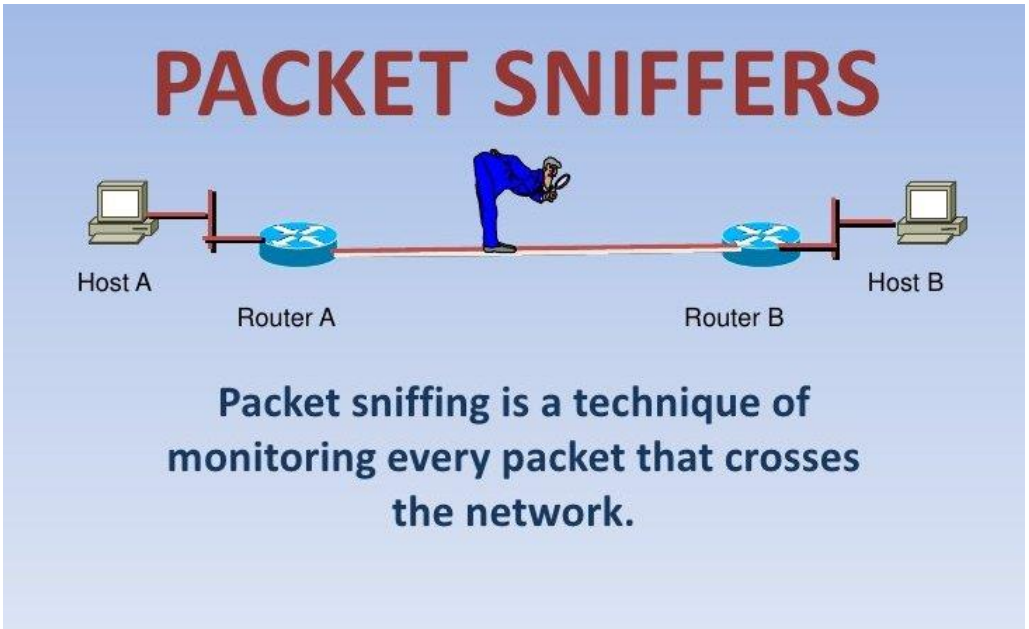
Command Line ->iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 -j REDIRECT --to-port 5000

אחרי הקמת הקשר בין הלקוח והשרת דרך ה proxy נוצרים מפתחות ל session הנוכחי, ומה שעשינו כדי לקחת את מפתחות הלקוח והשרת הוא שהעתקנו את המפתחות לקבצים, והקבצים מועתקים בצורה מאובטחת ומוצפנת על ידי שימוש בפקודה ssh copy שמעתיקה קבצים אלה ל proxy בעת הקמת הקשר ויצירת המפתחות. לאחר מכן, מקבלים תקשורת בין הלקוח והשרת שמפוקחת ע"י ה proxy וכך ע"י שימוש במפתחות שהעתקנו ה Proxy יכול לבחון את החבילות ובכך למנוע התנהגות לא חוקית משני הצדדים ולהפיל את התקשורת במידה ומפירים אותה.

כלים לפתרון אתגרים בהם נתקלנו

Sniffer

Sniffer זוהי תוכנה או חומרה המאפשרת להאזין ולתעד תקשורת מחשבים העוברת בנקודה כלשהי ברשת. ה-sniffer קולט את חבילות המידע היוצרות את התקשורת, מנתח אותן בהתאם ומציג אותן למשתמש לאחר הניתוח.



השתמשנו ב- **sniffer script** בתחילת הדרך כדי לקחת את החבילות היוצאות ונכנסות מהמחשב שלנו , שבאמצעותו היינו מתחברים ב- **SSH connection**, כאשר האזנו לחבילות שמעל שכבת ה-**TCP** ופרקנו את החבילה ל:

1. **Destination ip address**
2. **Source ip address**
3. **Destination port**
4. **Source port**
5. **Data**

ומהמידע הזה הצלחנו לנווט בין החבילות ולדעת איך לשים להפריד בין חבילות הלקוח והשרת ולהוציא את המידע המוצפן על מנת לנסות לפענח אותו.

Sniffer Code:

```
socket.socket( socket.AF_PACKET , socket.SOCK_RAW , socket.ntohs(0x0003))

while True:
    packet = s.recvfrom(65565)

    #packet string from tuple
    packet = packet[0]

    #parse ethernet header
    eth_length = 14

    eth_header = packet[:eth_length]
    eth = unpack('!6s6sH' , eth_header)
    eth_protocol = socket.ntohs(eth[2])

    #Parse IP packets, IP Protocol number = 8
    if eth_protocol == 8 :

        #Parse IP header
        #take first 20 characters for the ip header
        ip_header = packet[eth_length:20+eth_length]

        #now unpack them
        iph = unpack('!BBHHHBBH4s4s' , ip_header)

        version_ihl = iph[0]
        version = version_ihl >> 4
        ihl = version_ihl & 0xF

        iph_length = ihl * 4

        ttl = iph[5]
        protocol = iph[6]
        s_addr = socket.inet_ntoa(iph[8]);
        d_addr = socket.inet_ntoa(iph[9]);


        #TCP protocol
        if protocol == 6 :

            t = iph_length + eth_length
            tcp_header = packet[t:t+20]

            #now unpack them
            tcph = unpack('!HHLLBBHHH' , tcp_header)

            source_port = tcph[0]
            dest_port = tcph[1]
            sequence = tcph[2]
            acknowledgement = tcph[3]
            doff_reserved = tcph[4]
            tcph_length = doff_reserved >> 4

      ... ... ... ...
      #filtering Pockets...
```

סיכום

בפרויקט זה מימשנו **middle box** אשר ממוקם בין לקוח ושרת שיש ביניהם חיבור **SSH** כך שכאשר הלקוח מתחיל להקים חיבור **SSH** לשרת ה **middle box** מבקש את המפתחות של ה **session** שנוצרים ובהסכמת הלקוח מעבירים לו את המפתחות באופן מאובטח ואז ברגע שהוא מקבל את המפתחות יש לו את היכולת לחשוף את המידע המועבר ביניהם ובכך מתקיימים שני יתרונות: שהמפתחות המועברים ל **middle box** תקפות רק ל **session** הנוכחי כך שאם הלקוח מתנתק ומתחבר שוב הוא לא יכול לפענח מידע באמצעות מפתחות שהיו לו מקודם ובזה הוא חושף מידע רק באישור מהלקוח ובזה הוא לא יכול לחשוף כל מידע מהלקוח, לזייף הודעות של הלקוח או לשלוח הודעות כאילו הוא הלקוח. היתרון השני הוא ה **middle box** חושף את המידע המועבר ובזה יש אפשרות להוספת אלגוריתם אשר בודק אם המידע מסוכן או מכיל וירוסים .

העבודה על הפרויקט התאפיינה הן ברכישת ידע תאורטי והן בהטמעה פרקטית שלו. אבטחת מידע באינטרנט נמצאת כיום בחזית התעשיה בהייטק, כלומר הפרויקט הוא מאוד עכשווי ורלוונטי.

אתגרים

- מציאת המפתחות של ה session עבור אלגוריתם ההצפנה שמבוסס עליו הפרויקט מתוך כל קבצי ה SSH שתומכים בהמון אלגוריתמי הצפנה .
- פענוח המידע המוצפן המועבר בחבילות בין שני צדדים: אתגר ראשון הוא לעקוב אחרי החבילות ולזהות איפה מתחילה ההצפנה ואתגר שני הוא לעקוב אחרי הפענוח כך ששגיאה בביט אחד תגרום לכך שהמידע לא יפוענח בצורה נכונה. .
- העברת המפתחות ל middle box בצורה מאובטחת .
- מימוש proxy שמאזין לחיבור בין שני מחשבים מרחוק .

כיווני מחקר עתידיים

1. תמיכה ביותר אלגוריתמי הצפנה.
2. תמיכה בהעברת מידע בו חבילת SSH מחולקת ליותר מחבילה אחת.
3. אימות חבילות ע"י לקוח ושרת (MAC).

נספחים

הפרויקט מבוסס על שני סקריפטים של פיתון :

1. aes-ctr128.py :

סקריפט אשר מקבל חבילות מוצפנות, מסנן אותן לפי הגודל שנמצא בראש החבילה ומפענח אותן בעזרת המפתחות שקיבל בתחילת הקמת הקשר.
הסקריפט הזה מורץ מ middle box. מריצים שתי פקודות. הראשונה :
’sudo iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 22 -j REDIRECT --to-port 5000’
על מנת לערוך טבלאות הניתוב שיגיעו אליו החבילות אשר מועברות בין הלקוח לשרת והשנייה:
‘python proxy.py 132.68.60.103 5000 132.68.60.105 22’
אשר מריצה סקריפט proxy שיאזין על החיבור כך שהכתובת שמסתימת ב 105 היא כתובת השרת ו 22 והוא פורט ה TCP והכתובת המסתימת ב 103 היא של ה middle box ו 5000 הוא הפורט שהקים עליו הלקוח החיבור .
קבלת המפתחות מהלקוח בעזרת הפקודה
'scp 132.68.60.104:Desktop/key.txt ~/Desktop/'
שמממשת העברת קבצים בין שני מחשבים בצורה אמינה ובטוחה בלי התערבות צד שלישי , 132.68.60.104 היא כתובת המחשב שאנו פונים אליו ,
הפענוח מתבצע בעזרת ספרית (built in) Crypto שמספקים לה את המפתח והמונה והיא מפענחת בבלוקים שגודל כל אחד 32 בית ומדפיסה למסך.

2. proxy.py :

בסקריפט **הProxy**- התקנו שרת שמאזין לבקשות התחברות של **SSH** מלקוחות (מאזין על פורט שרירותי שבחרנו אותו להיות מספר מעל 5000 ע"י פקודת **bind(**IP,Port**)**),ואז כל לקוח שמתחבר בהתחברות **SSH** מופנה למחשב **ה-Proxy** לפורט הנ"ל, ובכל בקשה כזאת יוצרים **thread** שיטפל בבקשת ההתחברות כך שהוא יוצר התחברות לשרת המבוקש ע"י הקצאת **TCP Socket** ומעבירים את החבילות ממחשב הלקוח למחשב השרת דרך שתי **ה Sockets**, וכל חבילה שעוברת לוקחים ממנה את המידע המוצפן הרלוונטי ושופכים אותו לתוך קובץ לצורך הפענוח.
אפשר לראות בפונקציה **Client_flush** איך לוקחים את 4 הבתים הראשונים כדי לקבל את אורך החבילה המוצפנת הבאה ולפי האורך הזה אנחנו מחלצים המידע המוצפן.
לצורך מניעת חסימות בעת קריאה מ **Socket** ריק, הגדרנו אותם **NonBlocking** ואז כאשר רצינו לקרוא מידע מאחד ה **Socket** הריקים, הוא לא נחסם ונמשיך לנסות לקרוא מה **Socket** השני.
זה מאפשר תגובתיות מאוד מהירה לעומת צריכת **CPU** גדולה יחסית.

Table of figures

ביבליוגרפיה

1. the ssh files :[https://github.com/openssh/openssh-portable]
2. network sniffer:[http://www.colasoft.com/resources/network-
   [sniffer.php
3. FireWall : [https://www.paloaltonetworks.com/cyberpedia/what-is-a-
   [firewall