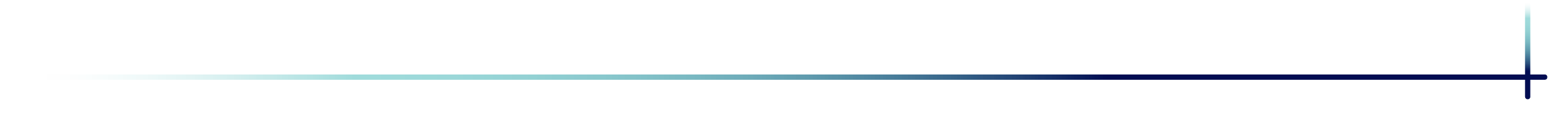
**מעבדה ב -Sniffing ו -Spoofing**

**של חבילות**

**הקדמה**

"רחרוח" (Sniffing) וזיוף (Spoofing) של חבילות הם שני מושגים חשובים בתחום אבטחת הרשת; הם שני איומים מרכזיים בתקשורת רשת. היכולת להבין את שני האיומים הללו חיונית להבנת אמצעי האבטחה ברשת. קיימים כלים רבים ל-Sniffing ול-Spoofing – לרחרוח ולזיוף של מנות: Wireshark, Tcpdump, Netwox, Scapy וכדומה. מומחי אבטחה וגם תוקפים עושים שימוש נרחב בחלק מהכלים הללו, והידע כיצד להשתמש בכלים אלה חשוב מאוד, אלא שחשוב יותר להבין כיצד כלים אלה עובדים, כלומר כיצד Sniffing ו-Spoofing של חבילות מיושמים בתוכנה.

**סביבת העבודה:** מעבדה זו נבדקה על סביבת המכונה הווירטואלית של פרויקט SEED המריצה את מערכת ההפעלה Ubuntu 16.4. ניתן להורידה מהאתר של SEED.

**כלים לביצועSniffing ו – Snoofing של חבילות**

ניתן להשתמש בכלים רבים לביצוע Sniffing ו-Spoofing, אך לרובם יש פונקציונליות קבועה ומוגבלת. Scapy שונה: הוא יכול לשמש לא רק ככלי, אלא גם כאבן בניין לבניית כלי Sniffing ו-Spoofing אחרים, כלומר אנחנו יכולים לשלב את פונקציות ה- Scapy בתוכנית שלנו. בחלק זה של המעבדה נשתמש ב- Scapy לכל משימה.

כדי להשתמש ב- Scapy, נכתוב תוכנית Python ונריץ אותה. ראו את הדוגמה שלהלן. עלינו להריץ את Python בהרשאת root. הרשאה זו דרושה כדי לבצע sniffing של חבילות.

בתחילת התוכנית (שורה ❶) עלינו לייבא את כל המודולים של Scapy.

$ view mycode.py

#!/bin/bin/python

from scapy.all import \* ❶

a = IP()

a.show()

$sudo python mycode.py

###[ IP ]###

version = 4

ihl = None

...

אנחנו יכולים גם להיכנס למצב האינטראקטיבי של Python ואז להפעיל את התוכנית שלנו, שורה אחת בכל פעם, בסביבת Python. צורה זו של עבודה נוחה כשאנחנו עורכים ניסויים שבמסגרתם אנחנו משנים את הקוד לעיתים תכופות. לדוגמה:

$ sudo python

>>> from scapy.all import \*

>>> a = IP()

>>> a.show()

###[ IP ]###

version = 4

ihl = None

...

**משימה ראשונה: Sniffing של חבילות**

Wireshark הוא כלי ה-Sniffing הפופולרי ביותר, והוא קל לשימוש. נשתמש בו מאוחר יותר. עם זאת, קשה להשתמש ב- Wireshark כאבן בניין לבניית כלים אחרים. למטרה זו נשתמש ב- Scapy. מטרת משימה זו היא ללמוד כיצד להשתמש ב- Scapy כדי לבצע sniffing של חבילות ב-Python. להלן קוד לדוגמה:

#!/usr/bin/python

from scapy.all import \*

def print\_pkt(pkt):

pkt.show()

pkt = sniff(filter=’icmp’,prn=print\_pkt)

**חלק I:** התוכנית שלעיל מבצעת sniffing של החבילות. הפונקציה print\_pkt() תיקרא בכל פעם שה-sniffer יזהה חבילה. פונקציה זו תדפיס חלק מהמידע על החבילה. הריצו את התוכנית עם הרשאת root והדגימו שאתם אכן יכולים לזהות חבילות. לאחר מכן, הריצו שוב את התוכנית, אך הפעם בלי להשתמש בהרשאת root. תארו את התצפיות שלכם והסבירו אותן.

// Run the program with the root privilege

$ sudo python sniffer.py

// Run the program without the root privilege

$ python sniffer.py

**חלק II:** כאשר אנחנו מבצעים sniffing, אנחנו בדרך כלל מעוניינים רק בסוגים מסוימים של חבילות. אנחנו יכולים לסנן חבילות באמצעות הגדרת מסננים. המסנן של Scapy משתמש בתחביר BPF (קיצור של: Berkeley Packet Filter). ניתן למצוא את התיעוד של BPF ואת המדריך הרלוונטי באינטרנט. הגדירו את המסננים שלפניכם והריצו שוב את תוכנית ה-sniffing שלכם. שימו לב שיש להגדיר כל מסנן בנפרד:

* מסנן הלוכד רק חבילות ICMP
* מסנן הלוכד את כל חבילות ה -TCP המגיעות מ -IP מסוים והמיועדות ל - port מס' 23
* מסנן הלוכד חבילות הנשלחות אל subnet מסוים או המגיעות ממנו. אתם יכולים לבחור כל subnet, כגון 128.230.0.0/16, אולם אסור לכם לבחור את ה -subnet שאליו מחוברת המכונה הווירטואלית שלכם.

**משימה שנייה: Spoofing של חבילות ICMP**

ככלי לזיוף מנות, Scapy מאפשר לנו לשנות את השדות של חבילת IP לערכים שרירותיים. כך מאפשר Scapy לזייף, כלומר לבצע Spoofing של חבילות. מטרת משימה זו היא לשנות חבילות IP כך שכתובת ה-IP של החבילה תהיה שונה מכתובת ה-IP של המכונה. אנחנו נזהה חבילות של בקשת ICMP echo ונשלח אותן למכונה וירטואלית אחרת באותה רשת. נשתמש ב-Wireshark כדי לבדוק אם אכן החבילות ששלחנו תתקבלנה ע"י הנמען. אם החבילות תתקבלנה, תישלחנה חבילות התשובה – חבילות ה-echo reply – לכתובת ה-IP המזויפת, ששלחנו יחד עם החבילה. הקוד שלפניכם מראה דוגמה לזיוף של חבילות ICMP.

>>> from scapy.all import \*

>>> a = IP() ❶

>>> a.dst = ’10.0.2.3’ ❷

>>> b = ICMP() ❸

>>> p = a/b ❹

>>> send(p) ❺

.

Sent 1 packets.

בדוגמת הקוד שלעיל, שורה ❶ יוצרת אובייקט IP מטיפוס מחלקת IP. מאפיין של המחלקה מוגדר בכל header של חבילת IP. אנחנו יכולים להשתמש בפקודת ls(a) או ls(IP) כדי לראות את כל השמות או הערכים של המאפיינים. את כל השמות או הערכים של המאפיינים נוכל גם לראות באמצעות פקודות a.show() או IP.show(). שורה ❷ מדגימה כיצד ניתן להגדיר את שדה כתובת ה-IP של היעד. אם שדה זה אינו מוגדר, הפקודה תעשה שימוש בערך ברירת מחדל.

שורה ❸ יוצרת אובייקט ICMP. ברירת המחדל של סוג הבקשה היא בקשת echo. בשורה ❹ אנחנו משלבים את a ו-b יחד כדי ליצור אובייקט חדש. המחלקה class מבצעת עמיסת אופרטור (operator overload) לאופרטור, והוא לא מייצג יותר פעולה של חלוקה. משמעותו הופכת להיות שילוב של b ו-a כך שנקבל אובייקט חדש, שמייצג חבילת ICMP. כעת נוכל לשלוח את החבילה בעזרת קריאה ל-send() בשורה ❺. בצעו את השינויים ההכרחיים בקוד הדוגמה, ואחר כך הראו שאתם יכולים לשלוח חבילה של בקשת ICMP echo עם כתובת IP שרירותית. זהו Spoofing.

>>> ls(a)

version : BitField (4 bits) = 4 (4)

ihl : BitField (4 bits) = None (None)

tos : XByteField = 0 (0)

len : ShortField = None (None)

id : ShortField = 1 (1)

flags : FlagsField(3 bits)= <Flag 0 ()> (<Flag 0 ()>)

frag : BitField (13 bits)= 0 (0)

ttl : ByteField = 64 (64)

proto : ByteEnumField = 0 (0)

chksum : XShortField = None (None)

src : SourceIPField = ’127.0.0.1’ (None)

dst : DestIPField = ’127.0.0.1’ (None)

options : PacketListField = [] ([])

**משימה שלישית: Traceroute**

מטרת משימה זו היא להשתמש ב- Scapy כדי להעריך את המרחק, מבחינת מספר הנתבים, בין המכונה הווירטואלית – שאתם עובדים איתה – ליעד שנבחר. הכלי שמיישם פונקציונליות זו נקרא traceroute. במשימה זו נכתוב כלי משלנו. הרעיון די פשוט: נשלח חבילה (מכל סוג) ליעד, כאשר שדה ה- (Time-To-Live) TTL שלו נקבע ל-1. הנתב הראשון ישמיט את החבילה וישלח אלינו הודעת שגיאה של ICMP. ההודעה תאמר לנו שזמן החיים עבר. כך אנחנו מקבלים את כתובת ה- IP של הנתב הראשון. לאחר מכן נגדיל את שדה ה-TTL ל-2, נשלח חבילה נוספת ונקבל את כתובת ה-IP של הנתב השני. נחזור על תהליך זה עד שהחבילה שלנו תגיע בסופו של דבר ליעד. יש לציין כי באמצעות פרוצדורה זו נקבל רק תוצאה משוערת, מכיוון שפרוטוקול IP אינו מתחייב שכל החבילות תעבורנה באותו מסלול (לא כל המנות הללו עוברות את אותו מסלול), אבל בפועל יש סיכוי גבוה שזה יקרה בפרק הזמן הקצר שבו נפעיל את הפרוצדורה. הקוד שלהלן מציג איטרציה אחת (חזרור אחד) של הפרוצדורה.

a = IP()

a.dst = ’1.2.3.4’

a.ttl = 3

b = ICMP()

send(a/b)

אם אתם מתכנתי Python מנוסים, תוכלו לכתוב את קטע קוד שיבצע את הפרוצדורה אוטומטית. אם אתה מתכנתי Python מתחילים, תוכלו לבצע את הפרוצדורה באופן ידני באמצעות שינוי ידני של שדה ה-TTL בכל איטרציה (חזרור). במקרה זה יהיה עליכם לרשום את כתובת ה- IP על סמך התצפית שלכם ב-Wireshark. תוכלו לבחור באיזו דרך שתרצו כל עוד תקבלו את התוצאה.

**משימה שלישית: Traceroute**

במשימה זו נשלב את טכניקות ה-Sniffing וה-Spoofing שלמדנו. זאת, כדי לכתוב תוכנית שתבצע Sniffing ואז Spoofing. לשם כך נזדקק לשתי מכונות וירטואליות באותו LAN. ממכונה וירטואלית A שלחו ping ל-IP שבחרתם, X. פעולה זו תיצור חבילת בקשה של ICMP echo. אם X פעיל, תוכנית ה-ping תקבל את ה-echo reply ותדפיס תגובה. תוכנית ה-Sniffing ואז Spoofing שתכתבו תפעל במכונה וירטואלית B, המנטרת את ה-LAN באמצעות פעולת sniffing של החבילות, העוברות ב-LAN. בכל פעם שהתוכנית שלכם מזהה בקשת ICMP echo (ללא קשר לכתובת ה-IP של היעד), עליה לשלוח מייד echo reply תוך שימוש ב-spoofing. לכן, תוכנית ה-ping אמורה לקבל תשובה המעידה ש-X פעיל, ללא תלות במצבו של X, בין שהוא פעיל ובין שאינו פעיל. עליכם להשתמש ב-Scapy כדי לבצע את המשימה הזו. בדו"ח שלכם אתם צריכים לספק ראיות להמחשה שהטכניקה שלכם עובדת.