

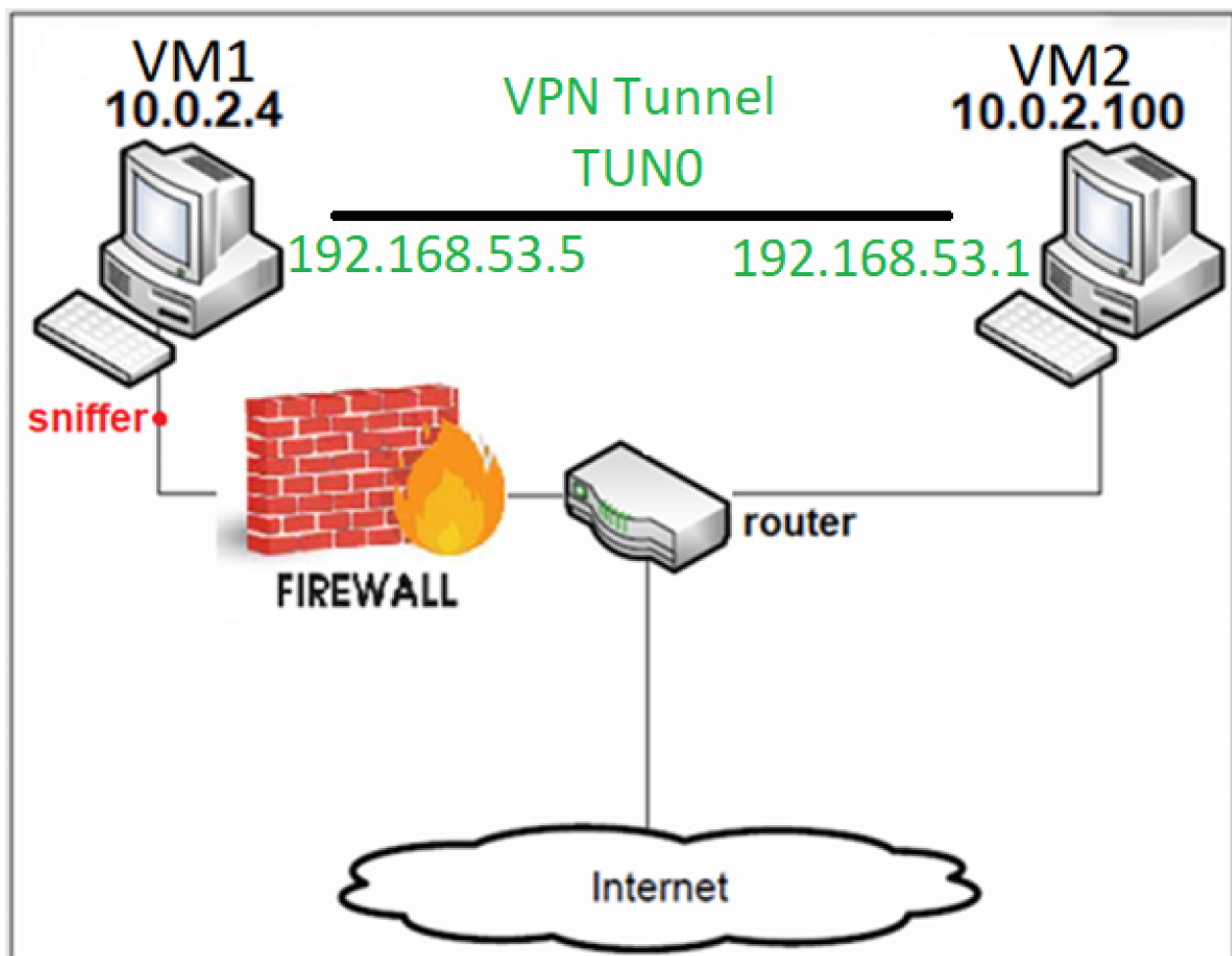
## Linux Firewall Evasion Lab

### כתובות IP לכל מחשב

Name	IP	TUN0
VM1	10.0.2.4	192.168.53.5
VM2	10.0.2.100	192.168.53.1

### Firewall

התמונה מטה מייצגת את מבנה הרשת



## Task 1: VM Setup

מבוא:

תיאור

הגדרת המכונות הוירטואליות ברשת NAT NETWORK

מטרה

ליצור רשת המכילה 2 מכונות וירטואליות שעל אחת מהן ירוץ הfirewall

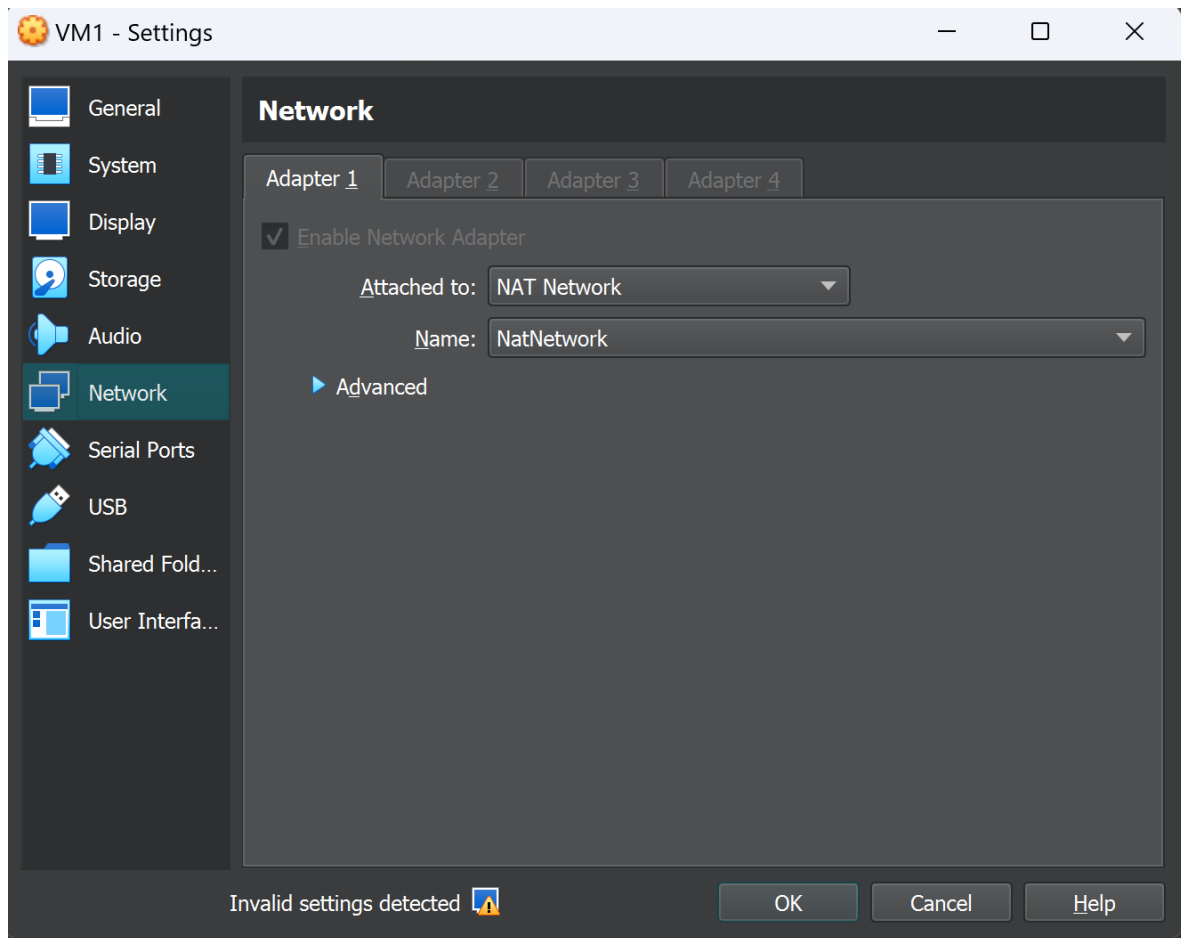
תוצאה מצופה

2 מכונות וירטואליות עובדות

## ביצוע המשימה

נראה שיש לנו 2 מכונות וירטואליות VM1 ו VM2.

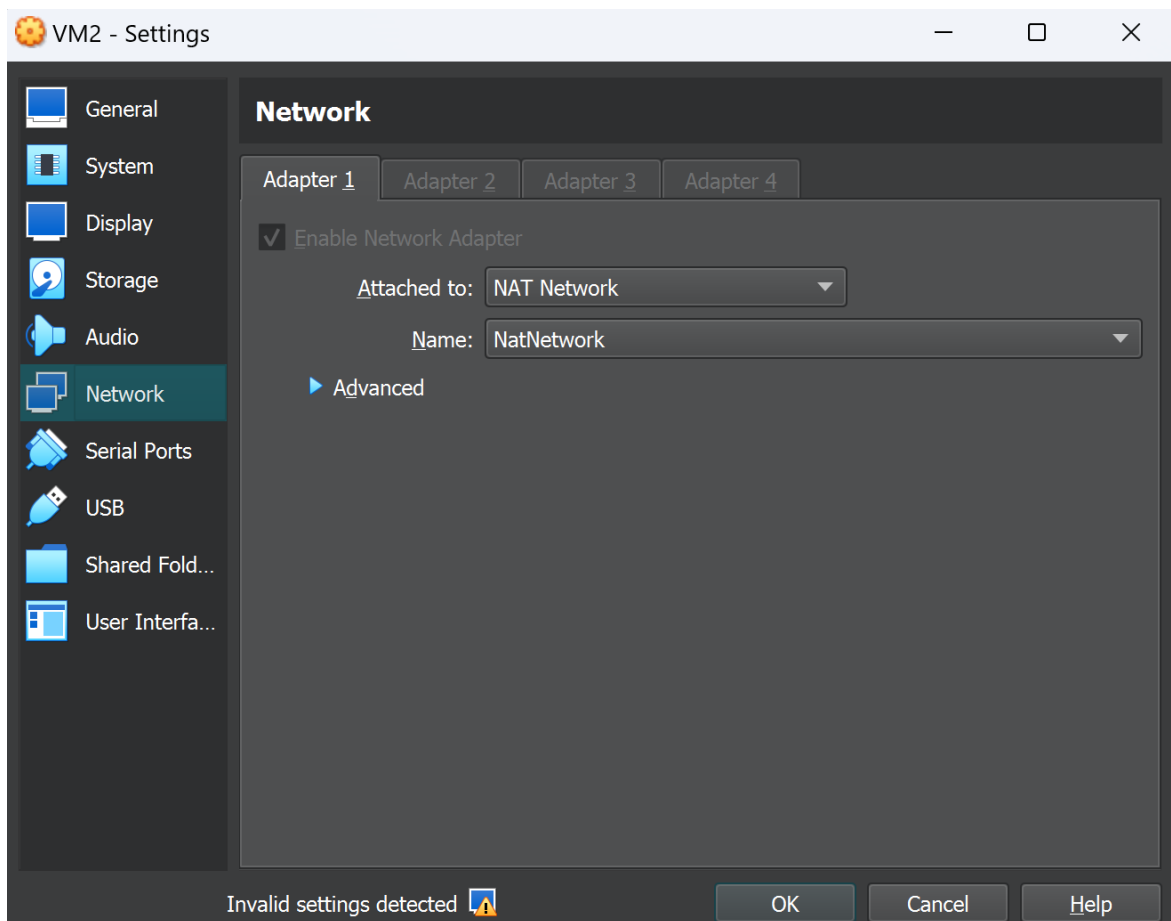
כעת נראה את VM1:



ניתן לראות שיצרנו מכונה וירטואלית המחוברת לאינטרנט.

```
[Tue May 23 12:52:59] VM1:~$ ifconfig
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:36:00:22
        inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
        inet6 addr: fe80::a00:27ff:fe36:22/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:16531 errors:0 dropped:0 overruns:0 frame:0
        TX packets:4825 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:23080819 (23.0 MB)  TX bytes:562548 (562.5 KB)
```

כעת נראה את VM2:



```
[Tue May 23 12:16:38] VM2:~$ ifconfig
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:34:c5:51
        inet addr:10.0.2.100  Bcast:10.0.2.255  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:8324 errors:0 dropped:0 overruns:0 frame:0
        TX packets:2467 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:11004016 (11.0 MB)  TX bytes:467424 (467.4 KB)
```

ניתן לראות שיצרו מכונה וירטואלית המחוברת לאינטרנט.

### סיכום המשימה

הצלחנו לבצע את המשימה, ניתן לראות שיצרנו 2 מכונות וירטואליות המחוברות לאינטרנט עם כתובות IP שונות. התוצאות התאימו למצופה ולא נתקלנו בבעיות במהלך ביצוע המשימה.

## Task 2: Set up Firewall

### מבוא:

#### תיאור

הפעלת חומת אש, וקינפוג חוקים חדשים המונעים גישה לאתר  
sdarot.tw

#### מטרה

לחסום גישה מ-VM1 לשליחת פאקטות לשרתים של sdarot

#### תוצאה מצופה

כישלון בעת ניסיון גישה לשרתים של sdarot

## ביצוע המשימה

תחילה נרצה לבדוק שאנחנו אכן מצליחים לגשת לאתר סדרות באמצעות פקודת PING לדומיין של סדרות:

```
[Tue May 23 12:55:24] VM1:~$ ping sdarot.tw
PING sdarot.tw (185.224.81.69) 56(84) bytes of data.
64 bytes from abelohost-69.81.224.185.dedicated-ip.abelons.com (185.224.81.69):
icmp_seq=1 ttl=50 time=69.9 ms
64 bytes from abelohost-69.81.224.185.dedicated-ip.abelons.com (185.224.81.69):
icmp_seq=2 ttl=50 time=70.3 ms
64 bytes from abelohost-69.81.224.185.dedicated-ip.abelons.com (185.224.81.69):
icmp_seq=3 ttl=50 time=70.8 ms
```

ניתן לראות כי ניתן לגשת לאתר זה ping עבר בהצלחה.

כעת נרצה לבדוק את כל כתובות הדומיין הקשורים לאתר sdarot:

```
[Tue May 23 12:56:57] VM1:~$ host www.sdarot.tw
www.sdarot.tw has address 37.221.65.66
www.sdarot.tw has address 79.133.51.206
www.sdarot.tw has address 185.224.81.69
```

נראה כי לסדרות יש 3 כתובות IP שנצטרך לחסום.

לאחר מכן, נשתמש ב[ufw](#) (נותן גישה פשוטה וידידותית יותר למשתמש להגדיר חוקים בחומת האש):

תחילה נפעיל את חומת האש

```
[Tue May 23 12:55:33] VM1:~$ sudo ufw enable
Firewall is active and enabled on system startup
```

נראה שחומת האש הופעלה.

ואז נוסיף את החוקים שחוסמים את כתובות הIP

```
[Tue May 23 12:55:54] VM1:~$ sudo ufw deny out on enp0s3 to 79.133.51.206
Rule added
[Tue May 23 12:55:59] VM1:~$ sudo ufw deny out on enp0s3 to 185.224.81.69
Rule added
[Tue May 23 12:56:03] VM1:~$ sudo ufw deny out on enp0s3 to 37.221.65.66
Rule added
```

ניתן לראות שהחוקים נוספו בהצלחה.

ולבסוף נבדוק את טבלת חומת האש המעודכנת:

```
[Tue May 23 12:56:07] VM1:~$ sudo ufw status
Status: active

To Action From
--
79.133.51.206 DENY OUT Anywhere on enp0s3
185.224.81.69 DENY OUT Anywhere on enp0s3
37.221.65.66 DENY OUT Anywhere on enp0s3
```

ניתן לראות שחומת האש מופעלת עם החוקים שהוגדרו לפני כן בהצלחה.

זה עתה, נרצה לבדוק אם חומת האש חוסמת את הגישה אל sdarot:

```
[Tue May 23 12:56:17] VM1:~$ ping sdarot.tw
PING sdarot.tw (79.133.51.206) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
```

ניתן לראות כי החסימה בוצעה בהצלחה ולא ניתן לגשת אל השרתים של sdarot.



### סיכום המשימה

הצלחנו לבצע את המשימה.

ניתן לראות שהצלחנו ליצור חוקים אשר חוסמים גישה אל כתובות ה IP של sdarot ולאחר יצירת החוקים לא היה ניתן לגשת ולבצע שליחת ping. התוצאות התאימו למצופה מאחר והצלחנו לבצע חסימת גישה לשרתי sdarot ולא נתקלנו בבעיות במהלך ביצוע המשימה.

## Task 3: Bypassing Firewall using VPN

### מבוא:

#### תיאור

יצירת חיבור VPN לצורך מעקף חוקי חומת האש.

#### מטרה

נרצה ליצור חיבור VPN בין הלקוח לשרת ולהעביר את ה-packets דרך החיבור הנ"ל וע"י כך לעקוף את החסימה המוגדרת בחומת האש לשרתים ספציפיים.

#### תוצאה מצופה

נצליח להעביר packets בין המחשב לשרת ולגשת אל אתר סדרות דרך חיבור ה-VPN שיצרנו, למרות ההגבלות שהגדרנו בחומת האש.

## ביצוע המשימה

לשם יצירת חיבור הVPN, נצטרך לבצע את השלבים הבאים:

### Step 1: Run VPN Server

תחילה, נבדוק איזה interfaces קיימים לנו ברשת של מחשב VM2:

```
[Tue May 23 13:39:29] VM2:~$ ifconfig -a
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:34:c5:51
        inet addr:10.0.2.100  Bcast:10.0.2.255  Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:13189 errors:0 dropped:0 overruns:0 frame:0
        TX packets:5356 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:16750681 (16.7 MB)  TX bytes:889279 (889.2 KB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:561 errors:0 dropped:0 overruns:0 frame:0
        TX packets:561 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1
        RX bytes:39677 (39.6 KB)  TX bytes:39677 (39.6 KB)
```

ניתן לראות ממשק אחד בשם enp0s3.

במעבדה קיבלנו קבצים מצורפים לטובת יצירת חיבור VPN בין השרת ללקוח בשפת C.

Vpnserver.c

```
C vpnserver.c
1  #include <fcntl.h>
2  #include <stdio.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include <arpa/inet.h>
6  #include <linux/if.h>
7  #include <linux/if_tun.h>
8  #include <sys/ioctl.h>
9
10 #define PORT_NUMBER 55555 // The same port number with the client
11 #define BUFF_SIZE 2000    // The buffer size
12
13 struct sockaddr_in peerAddr; // The address of the peer
```

בפורט 55555 נאזין ל packets בפרוטוקול UDP.

הbuff\_size זה הגודל של הבאפר לקריאה ושליחה של נתונים. peerAddr – מבנה שמכיל את הכתובת של הלקוח.

```

15 // Create a tun device and return its file descriptor.
16 // tun device is a virtual network device that looks like a network device
17 // to the user but does not send packets anywhere but to the user space program that created it.
18 int createTunDevice() {
19     int tunfd;
20     struct ifreq ifr;
21     memset(&ifr, 0, sizeof(ifr)); // Clear ifr structure, ifr is pointer to struct ifreq
22
23     ifr.ifr_flags = IFF_TUN | IFF_NO_PI; // IFF_TUN: TUN device, IFF_NO_PI: Do not provide packet information
24
25     tunfd = open("/dev/net/tun", O_RDWR); // Open the tun device
26     ioctl(tunfd, TUNSETIFF, &ifr); // Setup tun device
27
28     return tunfd; // Return the file descriptor of the tun device,
29     // file descriptor is an integer number that uniquely identifies an open file of the process
30 }

```

הפונקציה אחראית ליצירת מכשיר רשת וירטואלי בשם tun ומחזיר המיקום של הקובץ קונפיגורציה של interface.

Tun – interface הפועל בשכבת הרשת במודל הOS, משמש למטרות ניתוב packets ומאפשר שליחה וקבלה של packets ברשת באופן וירטואלי מתהליכים הרצים במחשב (מדמה רכיב רשת) המתחבר לשרת VPN ומעביר את הpackets ליעד.

lfr – מבנה המוגדר בקרנל של לינוקס ומשמש לקינפוג ממשקי רשת.

```

32 // Create a UDP server socket and return its file descriptor.
33 int initUDPServer() {
34     int sockfd;
35     struct sockaddr_in server; // The address of the server
36     char buff[100]; // The buffer to store the message from the client
37
38     memset(&server, 0, sizeof(server)); // Clear server structure
39     server.sin_family = AF_INET; // IPv4
40     server.sin_addr.s_addr = htonl(INADDR_ANY); // INADDR_ANY: Listen to all interfaces
41     server.sin_port = htons(PORT_NUMBER); // The port number
42
43     sockfd = socket(AF_INET, SOCK_DGRAM, 0); // Create a UDP socket
44     bind(sockfd, (struct sockaddr*) &server, sizeof(server)); // Bind the socket to the address of the server
45
46     // Wait for the VPN client to "connect".
47     bzero(buff, 100); // Clear buff
48     int peerAddrLen = sizeof(struct sockaddr_in); // The length of the address of the peer
49     int len = recvfrom(sockfd, buff, 100, 0, (struct sockaddr *) &peerAddr, &peerAddrLen); // Receive the message from the client
50
51     printf("Connected with the client: %s\n", buff); // Print the message from the client
52     return sockfd; // Return the file descriptor of the socket
53 }
54

```

הפונקציה יוצרת socket בפרוטוקול UDP ומחכה לחיבור.

```

56 // Handle tun device
57 void tunSelected(int tunfd, int sockfd){
58     int len; // The length of the message
59     char buff[BUFF_SIZE]; // The buffer to store the message
60
61     printf("Got a packet from TUN\n"); // Print the message
62
63     bzero(buff, BUFF_SIZE); // Clear buff
64     len = read(tunfd, buff, BUFF_SIZE); // Read the message from the tun device
65     sendto(sockfd, buff, len, 0, (struct sockaddr *) &peerAddr,
66     sizeof(peerAddr)); // Send the message to the client
67 }

```

הפונקציה מופעלת כאשר יש מידע בבאפר המוכנים לקריאת מהtun, ושולחת אותו ללקוח באמצעות הsocket שיצרנו.

```
69 // Handle socket connection
70 void socketSelected (int tunfd, int sockfd){
71     int len;
72     char buff[BUF_SIZE]; // The buffer to store the message
73
74     printf("Got a packet from the tunnel\n"); // Print the message
75
76     bzero(buff, BUF_SIZE); // Clear buff
77     len = recvfrom(sockfd, buff, BUF_SIZE, 0, NULL, NULL); // Receive the message from the client
78     write(tunfd, buff, len); // Write the message to the tun device
79
80 }
```

הפונקציה מופעלת כאשר הנתונים מוכנים לקריאה מה UDP SOCKET שהתקבלו מהלקוח ומעביר אותם לתun.

```
82 // The main function
83 int main (int argc, char * argv[]) {
84     int tunfd, sockfd;
85
86     tunfd = createTunDevice(); // Create tun device
87     sockfd = initUDPServer(); // Create UDP server socket
88
89     // Enter the main Loop
90     while (1) {
91         fd_set readFDSet;
92
93         FD_ZERO(&readFDSet); // Clear readFDSet
94         FD_SET(sockfd, &readFDSet); // Add sockfd to readFDSet
95         FD_SET(tunfd, &readFDSet); // Add tunfd to readFDSet
96         select(FD_SETSIZE, &readFDSet, NULL, NULL, NULL); // Wait for the data to be ready
97
98         if (FD_ISSET(tunfd, &readFDSet)) tunSelected(tunfd, sockfd); // If tunfd is ready, handle tun device
99         if (FD_ISSET(sockfd, &readFDSet)) socketSelected(tunfd, sockfd); // If sockfd is ready, handle socket connection
100     }
101 }
```

הפונקציה הראשית אשר יוצרת SOCKET וinterface ומחכה למעבר נתונים מאחד הצדדים.

FD\_ZERO מנקה את המצביע לקבצי הקונפיגורציה.

FD\_SET מוסיף קובץ קונפיגורציה למצביע.

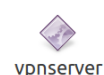
הפונקציה select היא פונקציה חוסמת אשר מחכה לקבלת או שליחת נתונים.

בהתאם לקבלת הנתונים מאחד הצדדים, נקראת הפונקציה המתאימה שמעבירה את המידע מה tun socket או מהtun socket.

כעת, נכנס אל VM2 שהוא השרת VPN שלנו, ונרצה לקמפל את הקוד בקובץ vpnserver.c ולהריץ אותו:

```
[Wed May 24 09:22:57] VM2:~$ gcc -o vpnserver vpnserver.c
[Wed May 24 09:25:09] VM2:~$ sudo ./vpnserver
```

נשים לב שלאחר הקימפול קיבלנו קובץ הרצה:



לאחר מכן נריץ את הפקודה הבאה:

```
[Wed May 24 09:25:44] VM2:~$ sudo ifconfig tun0 192.168.53.1/24 up
```

הפקודה הנ"ל משייכת את ה-IP 192.168.53.1 עם ה-subnet /24 אל ה-interface  
שהגדרנו שהוא tun0 והפרמטר up מפעיל את ה-interface.

כעת, נבדוק אילו interfaces קיימים בVM2:

```
[Wed May 24 09:26:12] VM2:~$ ifconfig -a
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:34:c5:51
          inet addr:10.0.2.100  Bcast:10.0.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:13295 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5458 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:16824764 (16.8 MB)  TX bytes:899181 (899.1 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:621 errors:0 dropped:0 overruns:0 frame:0
          TX packets:621 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:44051 (44.0 KB)  TX bytes:44051 (44.0 KB)

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
0-00
          inet addr:192.168.53.1  P-t-P:192.168.53.1  Mask:255.255.255.0
          inet6 addr: fe80::efc4:34a8:c280:1312/64 Scope:Link
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

ניתן לראות כי נוצר לנו interface חדש בשם tun0 והוא IP של 192.168.53.1  
כפי שהגדרנו לו.

כדי שVM2 יפעל כשרת, נצטרך להפוך אותו למעין gateway שמעביר את הpackets  
ולכן נפעיל לו את האופציה להעביר packets דרכו באמצעות הפקודה הבאה:

```
[Wed May 24 09:26:18] VM2:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip forward = 1
```

כעת VM2 יוכל להעביר packets דרך הצינור VPN שניצור בהמשך לרשת החיצונית  
ולהפך.

## Step 2: Run VPN Client

תחילה, נבדוק איזה interfaces קיימים לנו ברשת של מחשב VM1:

```
[Tue May 23 13:37:23] VM1:~$ ifconfig -a
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:36:00:22
        inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
        inet6 addr: fe80::a00:27ff:fe36:22/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:17224 errors:0 dropped:0 overruns:0 frame:0
        TX packets:5471 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:23576929 (23.5 MB)  TX bytes:639912 (639.9 KB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:635 errors:0 dropped:0 overruns:0 frame:0
        TX packets:635 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1
        RX bytes:58700 (58.7 KB)  TX bytes:58700 (58.7 KB)
```

ניתן לראות ממשק אחד בשם enp0s3.

במעבדה קיבלנו קבצים מצורפים לטובת יצירת חיבור VPN בין השרת ללקוח בשפת C.

בקוד בקובץ vpnclient.c שינינו את הIP שהוגדר לשרת לIP של השרת VPN שלנו שהוא 10.0.2.100:

```
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#include <linux/if.h>
#include <linux/if_tun.h>
#include <sys/ioctl.h>

#define BUFF_SIZE 2000 // The buffer size
#define PORT_NUMBER 5555 // The port number
#define SERVER_IP "10.0.2.100" // The IP address of the server
struct sockaddr_in peerAddr; // The address of the peer
```

כאן הוספנו גם את הIP של השרת VPN שלנו שהוא 10.0.2.100.



```

15 // Create a tun device and return its file descriptor.
16 int createTunDevice() {
17     int tunfd;
18     struct ifreq ifr; // ifr is pointer to struct ifreq
19     memset(&ifr, 0, sizeof(ifr)); // Clear ifr structure
20
21     ifr.ifr_flags = IFF_TUN | IFF_NO_PI; // IFF_TUN: TUN device, IFF_NO_PI: Do not provide packet information
22
23     tunfd = open("/dev/net/tun", O_RDWR); // Open the tun device
24     ioctl(tunfd, TUNSETIFF, &ifr); // Setup tun device
25
26     return tunfd;
27 }

```

הפונקציה אחראית ליצירת מכשיר רשת וירטואלי בשם tun ומחזיר המיקום של הקובץ קונפיגורציה של הinterface.

Tun – interface הפועל בשכבת הרשת במודל הOSI, משמש למטרות ניתוב packets ומאפשר שליחה וקבלה של packets ברשת באופן וירטואלי מתהליכים הרצים במחשב (מדמה רכיב רשת) המתחבר לשרת VPN ומעביר את הpackets ליעד.

lfr – מבנה המוגדר בקרנל של לינוקס ומשמש לקניפוג ממשקי רשת.

```

29 // Create a UDP socket and return its file descriptor.
30 int connectToUDPServer(){
31     int sockfd;
32     char *hello="Hello"; // The message to send to the server
33
34     memset(&peerAddr, 0, sizeof(peerAddr)); // Clear peerAddr structure
35     peerAddr.sin_family = AF_INET; // IPv4
36     peerAddr.sin_port = htons(PORT_NUMBER); // The port number
37     peerAddr.sin_addr.s_addr = inet_addr(SERVER_IP); // The IP address
38
39     sockfd = socket(AF_INET, SOCK_DGRAM, 0); // Create a UDP socket
40
41     // Send a hello message to "connect" with the VPN server
42     sendto(sockfd, hello, strlen(hello), 0,
43           (struct sockaddr *) &peerAddr, sizeof(peerAddr)); // Send the message to the server
44
45     return sockfd;
46 }

```

הפונקציה מחברת את הלקוח אל הsocket שנוצר בפרוטוקול UDP אצל השרת VPN ושולחת הודעת Hello המסמנת יצירת חיבור מוצלח.

```

49 // Handle tun device packets.
50 void tunSelected(int tunfd, int sockfd){
51     int len;
52     char buff[BUFF_SIZE]; // The buffer to receive data from the tun device
53
54     printf("Got a packet from TUN\n"); // Debug message
55
56     bzero(buff, BUFF_SIZE); // Clear the buffer
57     len = read(tunfd, buff, BUFF_SIZE); // Read data from the tun device
58     sendto(sockfd, buff, len, 0, (struct sockaddr *) &peerAddr,
59           sizeof(peerAddr)); // Send data to the VPN server
60 }

```

הפונקציה מופעלת כאשר יש מידע בבאפר המוכנים לקריאת מהtun, ושולחת אותו ללקוח באמצעות הsocket שיצרנו.

```

62 // Handle VPN server packets.
63 void socketSelected (int tunfd, int sockfd){
64     int len;
65     char buff[BUF_SIZE]; // The buffer to receive data from the VPN server
66
67     printf("Got a packet from the tunnel\n"); // Debug message
68
69     bzero(buff, BUF_SIZE); // Clear the buffer
70     len = recvfrom(sockfd, buff, BUF_SIZE, 0, NULL, NULL); // Read data from the VPN server
71     write(tunfd, buff, len); // Write data to the tun device
72 }

```

הפונקציה מופעלת כאשר הנתונים מוכנים לקריאה מה UDP SOCKET שהתקבלו מהלקוח ומעביר אותם לתun.

```

74 // The main function
75 int main (int argc, char * argv[]) {
76     int tunfd, sockfd;
77
78     tunfd = createTunDevice(); // Create a tun device
79     sockfd = connectToUDPServer(); // Create a UDP socket and connect to the VPN server
80
81     // Enter the main loop
82     while (1) {
83         fd_set readFDSet;
84
85         FD_ZERO(&readFDSet); // Clear the readFDSet
86         FD_SET(sockfd, &readFDSet); // Add the socket to the readFDSet
87         FD_SET(tunfd, &readFDSet); // Add the tun device to the readFDSet
88         select(FD_SETSIZE, &readFDSet, NULL, NULL, NULL); // Wait for an activity on one of the file descriptors
89
90         if (FD_ISSET(tunfd, &readFDSet)) tunSelected(tunfd, sockfd); // The tun device has received data
91         if (FD_ISSET(sockfd, &readFDSet)) socketSelected(tunfd, sockfd); // The socket has received data
92     }
93 }

```

הפונקציה הראשית יוצרת interface ברשת הלקוח וחיבור לsocket בשרת VPN.

FD\_ZERO מנקה את המצביע לקבצי הקונפיגורציה.

FD\_SET מוסיף קובץ קונפיגורציה למצביע.

Fdset הוא המבנה שמכיל את קבצי הקונפיגורציה של הממשק והsocket.

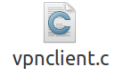
הפונקציה select היא פונקציה חוסמת אשר מחכה לקבלת או שליחת נתונים.

בהתאם לקבלת הנתונים מאחד הצדדים, נקראת הפונקציה המתאימה שמעבירה את המידע מה tun לsocket או מהsocket לתun.

נכנס אל VM1 שהוא הלקוח שלנו, ונרצה לקמפל את הקוד בקובץ vpnclient.c ולהריץ אותו:

```
[Wed May 24 09:38:54] VM1:~$ gcc -o vpnclient vpnclient.c
[Wed May 24 09:39:24] VM1:~$ sudo ./vpnclient 10.0.2.100
```

נשים לב שלאחר הקימפול קיבלנו קובץ הרצה:



לאחר מכן נריץ את הפקודה הבאה:

```
[Wed May 24 09:43:23] VM1:~$ sudo ifconfig tun0 192.168.53.5/24 up
```

הפקודה הנ"ל משייכת את הIP 192.168.53.5 עם subnet /24 אל interface tun0 שהגדרנו שהוא tun0 והפרמטר up מפעיל את ה interface.

נבדוק שאכן הגדרנו כנדרש וניתן לראות את tun0 ברשימת הממשקים:

```
[Wed May 24 09:43:41] VM1:~$ ifconfig -a
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:36:00:22
        inet addr:10.0.2.4  Bcast:10.0.2.255  Mask:255.255.255.0
        inet6 addr: fe80::a00:27ff:fe36:22/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:17452 errors:0 dropped:0 overruns:0 frame:0
        TX packets:5727 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:23716641 (23.7 MB)  TX bytes:663882 (663.8 KB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:855 errors:0 dropped:0 overruns:0 frame:0
        TX packets:855 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1
        RX bytes:74653 (74.6 KB)  TX bytes:74653 (74.6 KB)

tun0    Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
        inet addr:192.168.53.5  P-t-P:192.168.53.5  Mask:255.255.255.0
        inet6 addr: fe80::d33e:4b0d:5406:86ad/64 Scope:Link
        UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:500
        RX bytes:0 (0.0 B)  TX bytes:144 (144.0 B)
```

נראה כי התווסף בהצלחה ממשק חדש בשם tun0 עם הIP 192.168.53.5.

כעת נבדוק אם החיבור נוצר:

```
[Thu May 25 06:24:34] VM2:~$ gcc -o vpnserver vpnserver.c
[Thu May 25 06:24:46] VM2:~$ sudo ./vpnserver
Connected with the client: Hello
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

ניתן לראות שהחיבור נוצר בהצלחה בין ה-VM1 ו-VM2 בכך שקיבלנו הודעות פלט מתאימות בטרמינל שמאשרות שנוצר חיבור וירטואלי ב-tun.

### Step 3: Set Up Routing on Client and Server VMs

בשלב נרצה לנתב מחדש בטבלת הניתוב של מחשב VM1 את ה packets לשרתי סדרות שיעברו דרך הממשק הוירטואלי tun0 שבנינו ולבסוף נבדוק שהתווסף בהצלחה לטבלה:

```
[Thu May 25 06:59:18] VM1:~$ sudo route add -host 37.221.65.66 tun0
[Thu May 25 07:01:21] VM1:~$ sudo route add -host 79.133.51.206 tun0
[Thu May 25 07:01:42] VM1:~$ sudo route add -host 185.224.81.69 tun0
[Thu May 25 07:02:03] VM1:~$ route -n
```

Kernel IP routing table							
Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	10.0.2.1	0.0.0.0	UG	0	0	0	enp0s3
10.0.2.0	0.0.0.0	255.255.255.0	U	0	0	0	enp0s3
→ 37.221.65.66	0.0.0.0	255.255.255.255	UH	0	0	0	tun0
→ 79.133.51.206	0.0.0.0	255.255.255.255	UH	0	0	0	tun0
→ 169.254.0.0	0.0.0.0	255.255.0.0	U	1000	0	0	enp0s3
→ 185.224.81.69	0.0.0.0	255.255.255.255	UH	0	0	0	tun0
192.168.53.0	0.0.0.0	255.255.255.0	U	0	0	0	tun0

ניתן לראות בטבלת הניתוב שהתווספו בהצלחה 3 כתובות IP חדשות המציינות את שרתי סדרות.

ח- – מציין שנקבל את כתובות ה IP במספר במקום הדומיין.

#### Step 4: Set Up NAT on Server VM

ננקה את טבלאות החוקים של חומת האש במחשב VM2 כדי להבטיח שלא יהיו לנו חוקים מוגדרים שעלולים להוות חסימה עם הפקודה הבאה:

```
[Thu May 25 07:41:19] VM2:~$ sudo iptables -F
```

בפקודה הבאה ננקה גם את כל החוקים בטבלת הNAT:

```
[Thu May 25 07:41:29] VM2:~$ sudo iptables -t nat -F
```

-F – flush – מציין ניקוי של הטבלה בזיכרון.

ביצענו זאת כדי לנקות את הנתונים המוגדרים על המחשבים שהעבירו packets ברשת.

בפקודה הבאה נגדיר בVM2 בטבלת הNAT חוק חדש:

```
[Thu May 25 07:41:49] VM2:~$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o enp0s3
```

החוק הנ"ל מגדיר שכל הpackets אשר יוצאות לרשת החיצונית יזוהו (התחזות) כאילו עברו דרך interface enp0s3 וכך הpackets שעוברות דרך tun0 יועברו כביכול דרך enp0s3 אל הרשת החיצונית.

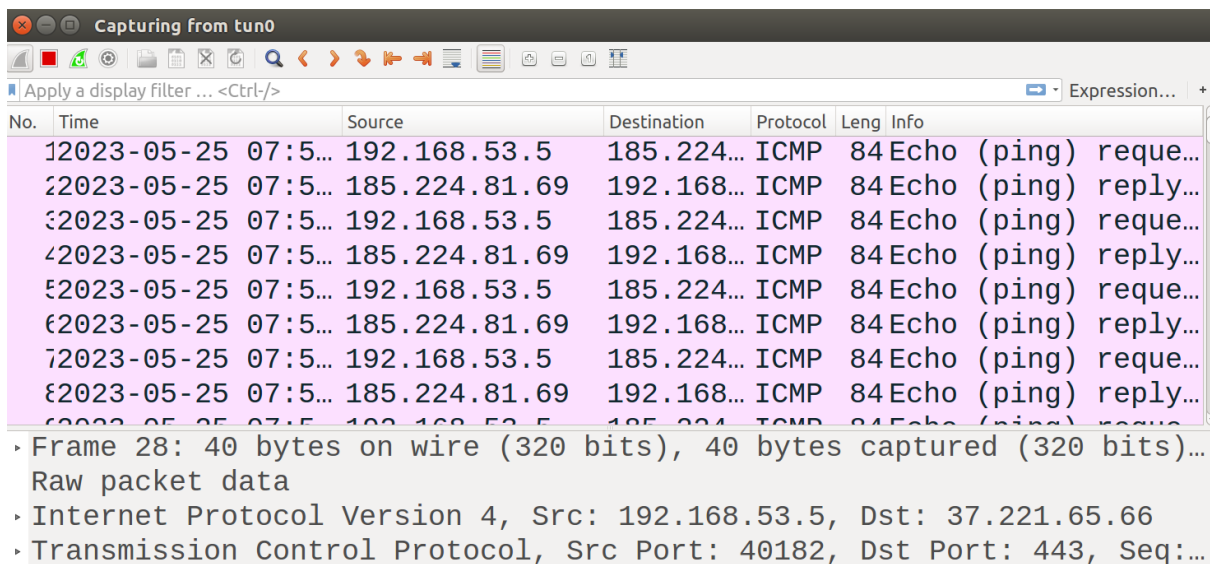
ביצענו את כל השלבים הנדרשים לטובת הקמת חיבור VPN TUNNEL בין VM1 ו-VM2 וכעת נרצה להדגים שניתן לגשת אל שרתי סדרות.

בפקודה הבאה נשלח ping אל שרת סדרות:

```
[Thu May 25 07:45:51] VM1:~$ ping sdarot.tw
PING sdarot.tw (185.224.81.69) 56(84) bytes of data.
64 bytes from abelohost-69.81.224.185.dedicated-ip.abelons.com (185.224.81.69):
icmp_seq=1 ttl=49 time=72.6 ms
64 bytes from abelohost-69.81.224.185.dedicated-ip.abelons.com (185.224.81.69):
icmp_seq=2 ttl=49 time=72.1 ms
64 bytes from abelohost-69.81.224.185.dedicated-ip.abelons.com (185.224.81.69):
icmp_seq=3 ttl=49 time=78.7 ms
64 bytes from abelohost-69.81.224.185.dedicated-ip.abelons.com (185.224.81.69):
icmp_seq=4 ttl=49 time=71.9 ms
64 bytes from abelohost-69.81.224.185.dedicated-ip.abelons.com (185.224.81.69):
icmp_seq=5 ttl=49 time=71.3 ms
```

ניתן לראות כי הפינג עבר בהצלחה וקיבלנו גם echo reply.

נרצה לוודא זאת גם באמצעות wireshark שאכן הצלחנו להעביר את ה-packets דרך ה-VPN TUNNEL שיצרנו:



The image shows a Wireshark packet capture window titled "Capturing from tun0". The packet list table displays several ICMP Echo (ping) requests and replies. The selected packet (No. 28) is expanded, showing its raw packet data: Internet Protocol Version 4 (Src: 192.168.53.5, Dst: 37.221.65.66) and Transmission Control Protocol (Src Port: 40182, Dst Port: 443, Seq: ...).

No.	Time	Source	Destination	Protocol	Leng	Info
1	2023-05-25 07:5...	192.168.53.5	185.224...	ICMP	84	Echo (ping) reque...
2	2023-05-25 07:5...	185.224.81.69	192.168...	ICMP	84	Echo (ping) reply...
3	2023-05-25 07:5...	192.168.53.5	185.224...	ICMP	84	Echo (ping) reque...
4	2023-05-25 07:5...	185.224.81.69	192.168...	ICMP	84	Echo (ping) reply...
5	2023-05-25 07:5...	192.168.53.5	185.224...	ICMP	84	Echo (ping) reque...
6	2023-05-25 07:5...	185.224.81.69	192.168...	ICMP	84	Echo (ping) reply...
7	2023-05-25 07:5...	192.168.53.5	185.224...	ICMP	84	Echo (ping) reque...
8	2023-05-25 07:5...	185.224.81.69	192.168...	ICMP	84	Echo (ping) reply...
28	2023-05-25 07:5...	192.168.53.5	37.221.65.66	ICMP	84	Echo (ping) reque...

Frame 28: 40 bytes on wire (320 bits), 40 bytes captured (320 bits) on tun0  
Raw packet data  
Internet Protocol Version 4, Src: 192.168.53.5, Dst: 37.221.65.66  
Transmission Control Protocol, Src Port: 40182, Dst Port: 443, Seq: ...

ניתן לראות שאכן ה-packets עוברים דרך ה-VPN TUNNEL שיצרנו, מאחר והם נשלחו עם ה-IP של VM1 אשר הוגדר ב-interface של tun0 שהוא 192.168.53.5.

בתמונה רואים כי ה-packets עוברים בהצלחה מ-VM1 אל שרת סדרות ובחזרה.

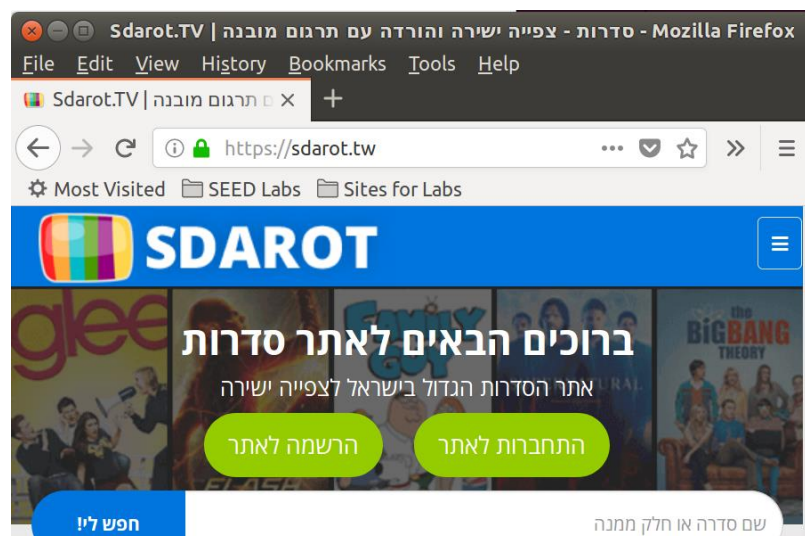


נרצה לראות בטרמינל שאכן מתקבלות הודעות פלט מתאימות על מעבר packets בtunnel.

```
[Thu May 25 06:41:47] VM1:~$ gcc -o vpnclient vpnclient.c
[Thu May 25 06:43:02] VM1:~$ sudo ./vpnclient
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
```

ניתן לראות כי יש תעבורה בtunnel שיצרנו.

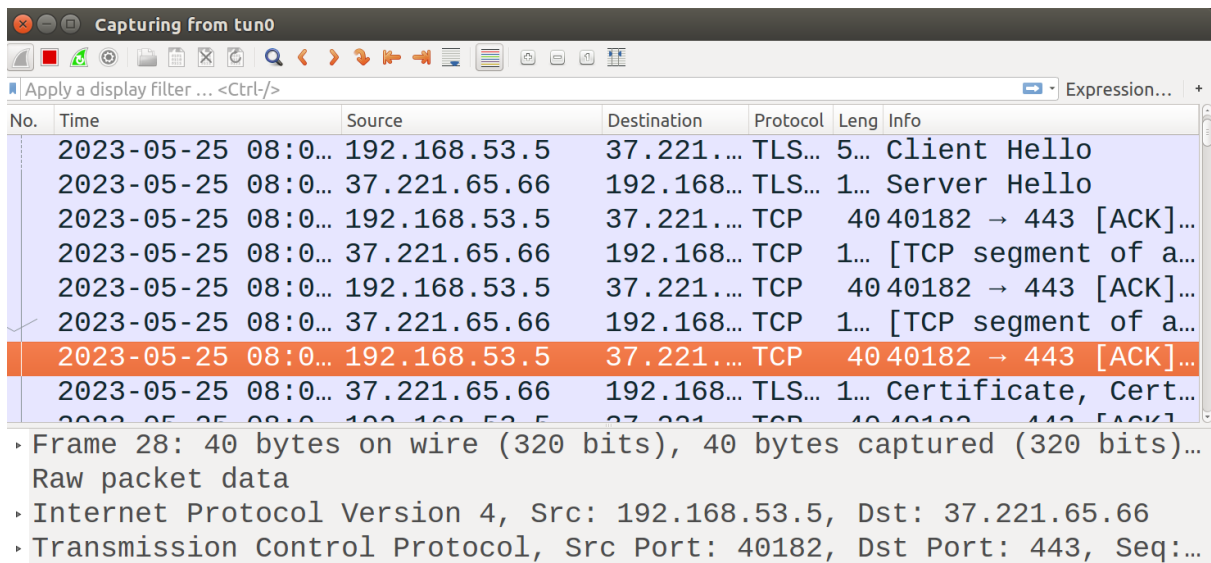
נרצה לראות שגם ניתן לגשת אל אתר סדרות:



ניתן לראות שהדף נטען בהצלחה.



נרצה לראות גם את התעבורה בwireshark:



No.	Time	Source	Destination	Protocol	Leng	Info
	2023-05-25 08:0...	192.168.53.5	37.221.65.66	TLS...	5...	Client Hello
	2023-05-25 08:0...	37.221.65.66	192.168.53.5	TLS...	1...	Server Hello
	2023-05-25 08:0...	192.168.53.5	37.221.65.66	TCP	40	40182 → 443 [ACK]...
	2023-05-25 08:0...	37.221.65.66	192.168.53.5	TCP	1...	[TCP segment of a...
	2023-05-25 08:0...	192.168.53.5	37.221.65.66	TCP	40	40182 → 443 [ACK]...
✓ 28	2023-05-25 08:0...	192.168.53.5	37.221.65.66	TCP	40	40182 → 443 [ACK]...
	2023-05-25 08:0...	37.221.65.66	192.168.53.5	TLS...	1...	Certificate, Cert...
	2023-05-25 08:0...	192.168.53.5	37.221.65.66	TCP	40	40182 → 443 [ACK]...

Frame 28: 40 bytes on wire (320 bits), 40 bytes captured (320 bits) on interface tun0  
Raw packet data

- Internet Protocol Version 4, Src: 192.168.53.5, Dst: 37.221.65.66
- Transmission Control Protocol, Src Port: 40182, Dst Port: 443, Seq:...

נראה כי יש חיבור TCP אל שרתי סדרות מהIP של מחשב VM1 אשר הוגדר ב interface כ192.168.53.5.

## סיכום המשימה

הצלחנו לבצע את המשימה.

תחילה הגדרנו שרת VPN ב VM2 כדי שנוכל להתחבר אליו וליצור חיבור VPN TUNNEL שבעזרתו נוכל לעקוף את הגבלות חומת האש.

לאחר מכן, יצרנו חיבור VPN ב VM1 וחיברנו אותו אל שרת ה VPN של VM2.

ניתבנו ב VM1 שכל התעבורה שלו תעבור דרך ה VPN TUNNEL שבנינו והגדרנו במחשב השרת שכל התעבורה תצא לרשת במקום מ interface tun0 מ interface enp0s3.

לבסוף, ניסינו להעביר packets מ VM1 אל שרתי סדרות והראינו שה packets אכן עברו בהצלחה דרך חיבור ה VPN TUNNEL שיצרנו.

גילינו כיצד ליצור חיבור VPN TUNNEL בין 2 מחשבים ולמדנו שבעזרתו ניתן לעקוף חוקי חומת אש שהוגדרו.

התוצאות התאימו למצופה מאחר והצלחנו לעקוף את חוקי חומת האש ולגשת אל שרתי סדרות למרות ההגבלות וזה נעשה בעזרת חיבור ה VPN TUNNEL שיצרנו.

לא נתקלנו בבעיות במהלך ביצוע המשימה.

## סיכום כללי למעבדה

ראשית נדרשנו ליצור 2 מכונות וירטואליות המחוברות לאינטרנט עם כתובות IP שונות.

שנית, יצרנו חוקים אשר חוסמים גישה אל כתובות הIP של sdarot ולאחר יצירת החוקים לא היה ניתן לגשת אל שרתי סדרות והדגמנו זאת ע"י שליחת ping.

לסיום, רצינו לעקוף את החוקים שהגדרנו בעזרת יצירת חיבור TUNNEL VPN ועשינו זאת באופן הבא:

תחילה הגדרנו שרת VPN ב VM2 כדי שנוכל להתחבר אליו וליצור חיבור VPN TUNNEL שבעזרתו נוכל לעקוף את הגבלות חומת האש.

לאחר מכן, יצרנו חיבור VPN ב VM1 וחיברנו אותו אל שרת הVPN של VM2.

ניתבנו בVM1 שכל התעבורה שלו תעבור דרך הVPN TUNNEL שבנינו והגדרנו במחשב השרת שכל התעבורה תצא לרשת במקום מinterface tun0 מinterface enp0s3.

לבסוף, ניסינו להעביר packets מVM1 אל שרתי סדרות והראינו שה-packets אכן עברו בהצלחה דרך חיבור הVPN TUNNEL שיצרנו.


לכן ניתן לראות כי הצלחנו לבצע את המשימות במעבדה ולהקים חיבור VPN שעזר לנו לעקוף את החסימות שיצרנו בחומת האש.

## משהו חדשני – APP-ID:



PRODUCTS SOLUTIONS SERVICES PARTNERS CUSTOMERS COMPANY CAREERS CONTACT

Search:



8 Applications (Clear filters)

CATEGORY	SUBCATEGORY	TECHNOLOGY	RISK	CHARACTERISTIC
8 networking	8 encrypted-tunnel 1 ip-protocol 25 proxy 6 remote-access	6 client-server 2 peer-to-peer	8 5	8 Evasive 1 Excessive Bandwidth 8 Prone to Misuse 8 Transfers Files 8 Tunnels Other Apps 3 Used by Malware 7 Vulnerabilities 7 Widely Used

NAME	CATEGORY	SUBCATEGORY	RISK	TECHNOLOGY
droidvpn	networking	encrypted-tunnel	5	client-server
firephoenix	networking	encrypted-tunnel	5	client-server
freenet	networking	encrypted-tunnel	5	peer-to-peer
gbridge	networking	encrypted-tunnel	5	client-server
gtunnel	networking	encrypted-tunnel	5	client-server
hamachi	networking	encrypted-tunnel	5	peer-to-peer
packetix-vpn	networking	encrypted-tunnel	5	client-server
ssh-tunnel	networking	encrypted-tunnel	5	client-server

App-ID של Palo Alto Networks היא תוכנה המאפשרת לנו לזהות ולשלוט באפליקציות שונות שמעבירות מידע על גבי הרשת שלנו, כולל יישומי VPN.

אך זה עובד ומה אנחנו יכולים לעשות כדי להתגונן מפני ניסיונות עקיפה בעזרת VPN:

זיהוי יישומים: App-ID משתמש בטכניקות שונות כדי לזהות יישומים, כולל התאמת חתימות, פענוח פרוטוקול וניתוח התנהגותי. בעת ניתוח packets זה יכול לזהות במדויק מגוון רחב של יישומים ותבניות שהוגדרו מראש ע"י צוות המחקר כמו פורטים, מבנה וכו' והשוואתם מול מאגר מידע של חתימות נוספות ובדיקה אם יש התאמה, כולל פרוטוקולי VPN פופולריים ואפליקציות VPN.

זיהוי וחסימה של VPN: App-ID יכול לזהות פרוטוקולי VPN כגון OpenVPN, IPSec, PPTP ואחרים. לאחר זיהוי, אתה יכול ליצור מדיניות מעודכנת לחומת האש כדי לחסום או להגביל תעבורת VPN בהתבסס על יישומים או קטגוריות ספציפיות של יישומים. זה מונע יצירת חיבורי VPN או מאפשר רק שימוש מורשה ב-VPN אשר מוגדרים מראש על ידינו.

פענוח SSL/TLS: VPNs מסוימים משתמשים בהצפנת SSL/TLS כדי להסתיר את התעבורה שלהם. חומת האש של Palo Alto Networks מציעות יכולות פענוח SSL/TLS, המאפשרות לנו לבדוק תעבורת VPN מוצפנת ולהחיל App-ID כדי לזהות את אפליקציית ה-VPN שבשימוש.

חתימות מותאמות אישית: אם צצה אפליקציית VPN חדשה או לא מוכרת, אנחנו יכולים ליצור חתימות מותאמות אישית כדי לזהות ולחסום אותה. Palo Alto Networks מספקת גמישות בהגדרת חתימות מותאמות אישית על סמך דפוסים, התנהגויות או תכונות ספציפיות של אפליקציית ה-VPN.

סינון כתובות אתרים: בנוסף לשליטה ברמת היישום, חומות האש של Palo Alto Networks מציעה יכולות סינון כתובות אתרים. אנחנו יכולים למנף קטגוריות סינון כתובות אתרים כדי לחסום גישה לאתרים ידועים של ספקי שירותי VPN, ולמנוע ממשתמשים להוריד ולהשתמש ביישומי VPN.

שילוב User ID: חומות האש של Palo Alto Networks יכולות להשתלב עם טכנולוגיות User ID, כגון Active Directory, כדי לשייך תעבורת רשת למשתמשים ספציפיים. זה מאפשר לנו ליצור מדיניות המכוונת למשתמשים בודדים או לקבוצות, מה שמשפר עוד יותר את השליטה שלנו על השימוש ב-VPN.

כדי להתגונן מפני ניסיונות עקיפת VPN עלינו לבצע את השלבים הבאים:

א. הפעל App-ID: ודא ש-App-ID מופעל בחומת האש של Palo Alto Networks ומוגדרת לזהות ולשלוט בתעבורת VPN.

ב. עדכן חתימות יישומים: עדכן באופן קבוע את מסד הנתונים של חתימות האפליקציה כך שיכלול את יישומי ה-VPN והגרסאות העדכניות ביותר.

ג. צור מדיניות App-ID: הגדר מדיניות חומת אש המאפשרת או חוסמת באופן מפורש יישומי VPN בהתבסס על הדרישות של הארגון שלנו. יש לשקול יצירת מדיניות נפרדת לקבוצות משתמשים או לרשתות שונות.

ד. אפשר פענוח SSL/TLS: אם רלוונטי, אפשר פענוח SSL/TLS כדי לבדוק תעבורת VPN מוצפנת ולהחיל מדיניות App-ID.

ה. אפשר סינון כתובות אתרים: השתמש בסינון כתובות אתרים כדי לחסום גישה לאתרים הקשורים לספקי שירותי VPN.

ו. מעקב וביקורת: מעקב רציף אחר תעבורת הרשת שלנו, סקירת יומנים וביצוע ביקורות כדי לזהות ניסיונות שימוש ב-VPN או הפרות מדיניות.

ז. חינוך משתמשים: למד את העובדים על מדיניות השימוש ב-VPN של החברה, הסיבות מאחוריה והסיכונים הפוטנציאליים הקשורים לעקוף חומת האש. חיזוק החשיבות של עמידה במדיניות האבטחה.

חשוב לעדכן באופן קבוע את תוכנת חומת האש ומסדי נתונים של חתימות כדי להבטיח זיהוי ומניעה יעילים של יישומי VPN וטכניקות חדשות. עקוב אחר טכנולוגיות VPN מתפתחות ועדכן את מנגנוני ההגנה שלך בהתאם.