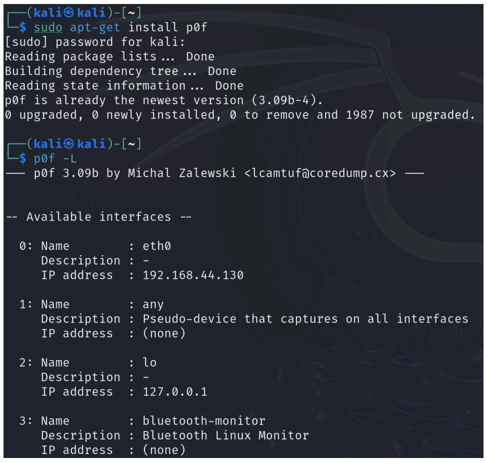
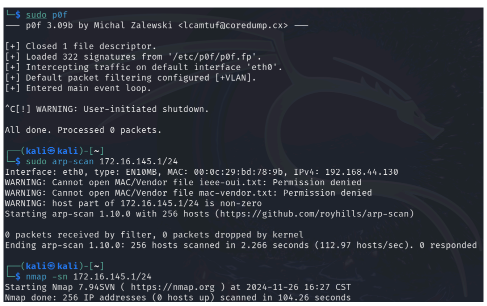
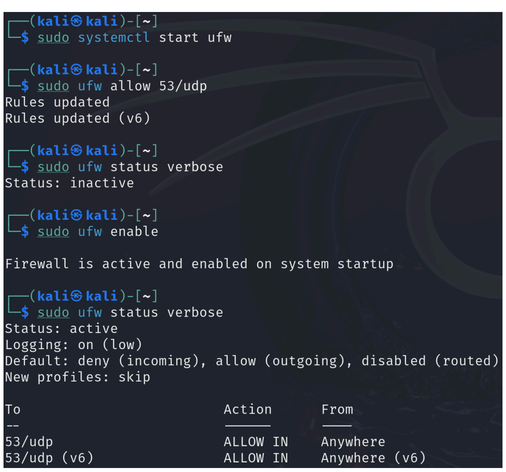


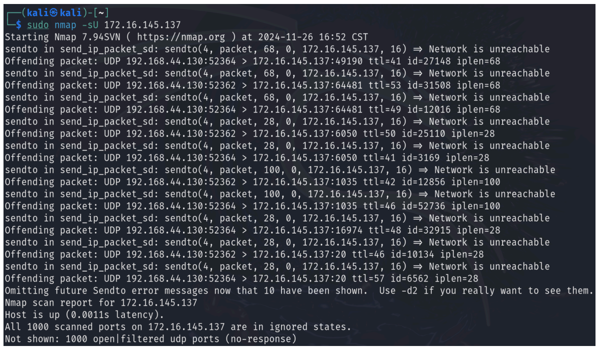
The first screenshot demonstrates the installation and use of the p0f tool for passive fingerprinting. The sudo apt-get install p0f command confirms that p0f is already installed and up-to-date. Next, the p0f -L command is used to list all available network interfaces, showing eth0 as the primary interface with an IP address of 192.168.44.130. Running sudo p0f starts the fingerprinting process, loading 322 signatures from its default configuration. However, no packets were processed during the session, possibly due to a lack of traffic on the interface. This step sets up the foundation for passively analyzing network packets and identifying OS configurations.



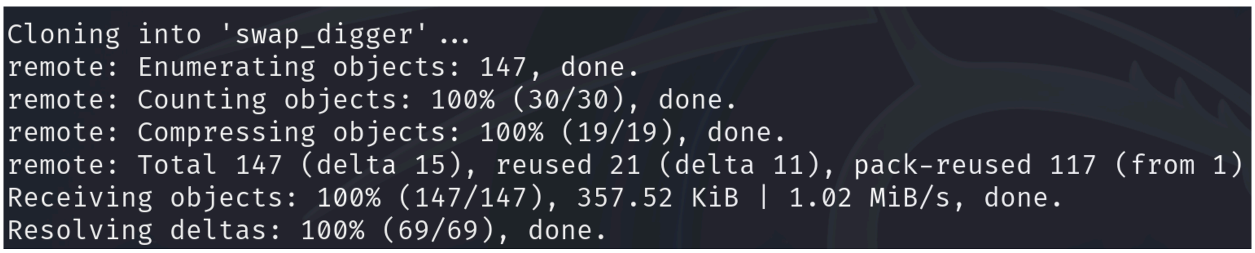
The second screenshot corresponds to Part II, where arp-scan and nmap are used to analyze the local network. The sudo arp-scan command attempts to map the network but encounters warnings about permissions (Cannot open MAC/Vendor file ieee-oui.txt) and an unreachable host. Similarly, the nmap -sn command performs a ping sweep across the 172.16.145.1/24 subnet, but no hosts respond, suggesting restricted connectivity or network misconfiguration. An additional nmap -sU UDP scan also shows Network unreachable errors. These steps reveal potential barriers in scanning and emphasize the importance of proper network and firewall configurations during forensic analysis.

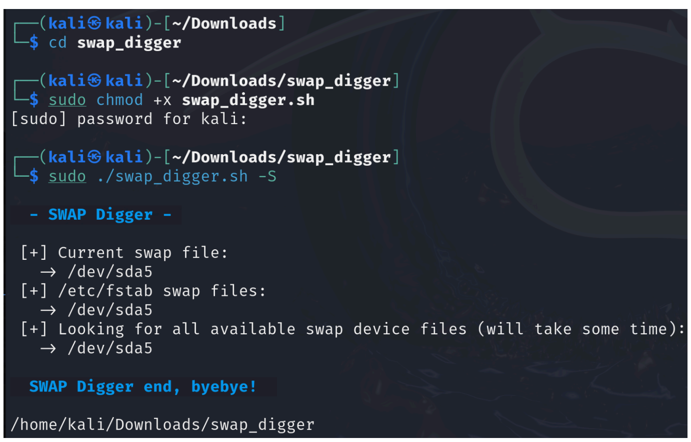


The third screenshot aligns with the firewall configuration step in Part II, Step 9, where ufw (Uncomplicated Firewall) is configured to allow UDP port 53 traffic. Initially, the firewall is inactive, as confirmed by sudo ufw status verbose. Using sudo ufw enable activates it, and rules are applied to allow incoming UDP packets on port 53. The final status output verifies the firewall is active, logging is enabled, and the new rules are in place. This step is crucial for permitting specific traffic during port scans and preparing the environment for further investigation.



The fourth screenshot shows the results of a UDP port scan using nmap, aligned with Part II, Step 9. After configuring the firewall, nmap -sU is run against the target host (172.16.145.137). Despite allowing UDP port 53 in the firewall, the scan results still indicate Network unreachable. This could be due to other network-level restrictions or incorrect configurations on the target host. The result highlights challenges in network exploration and the importance of verifying connectivity and target readiness.





These screenshots showcase the installation and execution of swap_digger, as part of Part III. In the beginning, the git clone command fails due to outdated authentication methods, which require personal access tokens instead of passwords. After resolving this, the repository is successfully cloned, the script is made executable using chmod +x, and it is run with ./swap_digger.sh -S. The output confirms the active swap file as /dev/sda5, providing insight into potential data remnants stored in swap space. This step is critical for recovering sensitive information like credentials, session data, and other forensic artifacts stored temporarily in memory.

sudo ./swap_digger.sh -a: The command is used within the context of analyzing and extracting data from the system's swap space, which contains sensitive remnants of information temporarily written from memory. By running the script, the swap_digger tool gains access to the system-level swap file for forensic analysis. The -a option performs a sweep of the swap partition, removing various types of potentially valuable artifacts, including web-entered passwords, emails, XML-based credentials, and structured data. It also searches for WiFi access points, frequently accessed HTTP/HTTPS URLs, FTP domains, opened files, and traces of IP addresses, though some results, such as IP addresses, may include false positives. The script also attempts to locate cryptographic hashes like MD5, SHA1, and SHA256. This analysis provides forensic investigators with insights into the system's activity, recovering data that could include usernames, passwords, session details, or activity traces left in swap memory. The output, such as human-readable strings and artifact details, is typically saved in a file (e.g., /tmp/swap_dig/swap_dump.txt) for further analysis. This functionality is critical in digital forensic investigations for reconstructing user activity, recovering lost credentials, or identifying potential security breaches and misuse of the system, making the -a option an essential part of the artifact recovery process.

In Step 14, the command rkhunter is used to scan the Linux system for rootkits, hidden files, and suspicious configurations. After installing the tool using sudo apt-get install rkhunter, the sudo rkhunter –check -rwo command initiates the analysis. This command checks for files or scripts that deviate from known safe versions stored in the rkhunter database. In Step 15, the command chkrootkit serves as another layer of defense for rootkit detection. After installing it with sudo apt-get install chkrootkit, running sudo chkrootkit scans for known rootkit signatures. It examines system binaries and critical configurations, verifying whether they are infected or intact. Each scanned element, like cron or date, is checked against a list of typical rootkit alterations, and the tool provides clear results on whether these elements are safe or compromised. The ascii command in step 16 is useful for displaying ASCII character information in various formats. The command ascii -s hello converts the string "hello" into its corresponding ASCII values, displaying each character's decimal, hexadecimal, and binary representations. When running ascii -x, the command outputs an extended ASCII table, listing all characters alongside their respective codes. In step 17 we use the xxd command to generate hexadecimal dumps of files, facilitating the identification of file signatures. For instance, xxd -g 1 0zapftis.rar | head outputs the first ten lines of the hexadecimal representation of a .rar file, including its signature 52 61 72, which corresponds to the Roshal Archive format. In Step 18, the strings command is utilized to extract human-readable text from binary files. By combining it with grep, we can search for specific patterns, such as jpg, within a file. For example, strings -t x terry-work-usb-2009-12-11.E01 | grep -i jpg reveals occurrences of jpg in the binary file along with their offsets, which are displayed in hexadecimal. In Step 19, the Sleuth Kit, a suite of tools for examining disk images and recovering files, is introduced. The fsstat command provides detailed filesystem info, such as type, volume info, and cluster sizes. For example, on a FAT16 image, it shows sector size and FAT structure, helping analysts understand data organization. The fls command lists files and directories in a disk image, showing names and metadata, which is crucial for identifying valuable or deleted files. Similarly, the ils command lists inode info, including creation and modification timestamps, vital for forensic timelines. The img_stat command examines the disk image structure, showing type and size, ensuring image integrity for analysis, while img_cat extracts raw data from disk images for further investigation. In Step 20, the fiwalk command is for analyzing filesystems within disk images, providing detailed file info, including metadata and hash values, which helps verify file integrity and identify tampering.