Auto Ping and Log

Name: ishaZaara

Regno:2460373

Introduction

In today's digital world, websites and servers must remain online and responsive at all times. Any downtime or delay in response can result in user dissatisfaction, financial loss, or a security breach.

This assignment focuses on building a **basic network monitoring tool** using a **Bash script**. The tool pings a website at regular intervals and records the response times in a .csv file. This helps track:

- Site availability
- Latency trends
- Downtimes or outages

Such tools are often part of a **network security monitoring** setup and help in proactive incident detection and response.

2. Objective

The main goals of this assignment are:

- To **automate** the process of checking a website's availability.
- To **record the ping response time** for future analysis.
- To generate a **CSV log** that can be imported into Excel or a dashboard.
- To simulate how **basic network monitoring** helps in detecting **outages**, **slowdowns**, or **DDoS attacks**.

Methodology

To complete this assignment, I followed a structured approach that involved planning, scripting, testing, and documenting:

- **Step 1: Selecting Tools and Platform**

I chose **Kali Linux** as the operating system because it is widely used for security and networking tasks. Bash was used as the scripting language because it is built into Linux, ideal for automation, and easily supports commands like ping, date, and sleep.

I selected google.com as the target domain since it's highly available and perfect for testing consistent connectivity.

- **Step 2: Writing the Bash Script**

I designed the script to do the following:

- Use ping -c 1 to send one ICMP packet to the domain.
- Use grep and sed to extract the response time (time=XX ms) from the ping output.
- Capture the current timestamp using the date command.
- Write all three values (timestamp, domain, response time) into a ping_log.csv file.
- If the ping fails, record the result as "timeout".
- Repeat this process every 300 seconds (5 minutes) using an infinite loop and sleep.

I also added logic to initialize the CSV file with a header row if it didn't already exist. This makes the file readable in tools like Excel.

- **Step 3: Testing the Script in Kali Linux**

After saving the script as ping_google.sh, I made it executable with:

chmod +x ping_google.sh

Then I ran it in the terminal:

./ping_google.sh

I monitored the terminal to ensure the script was writing logs correctly and that the CSV file was updating every 5 minutes with real data.

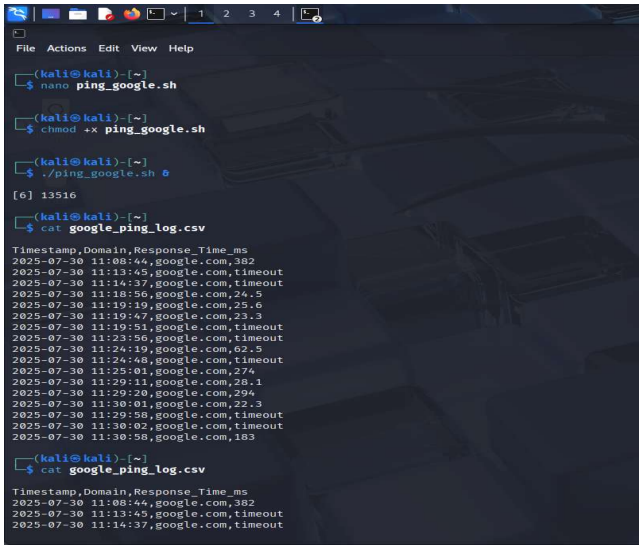
- **Step 4: Verifying the Output**

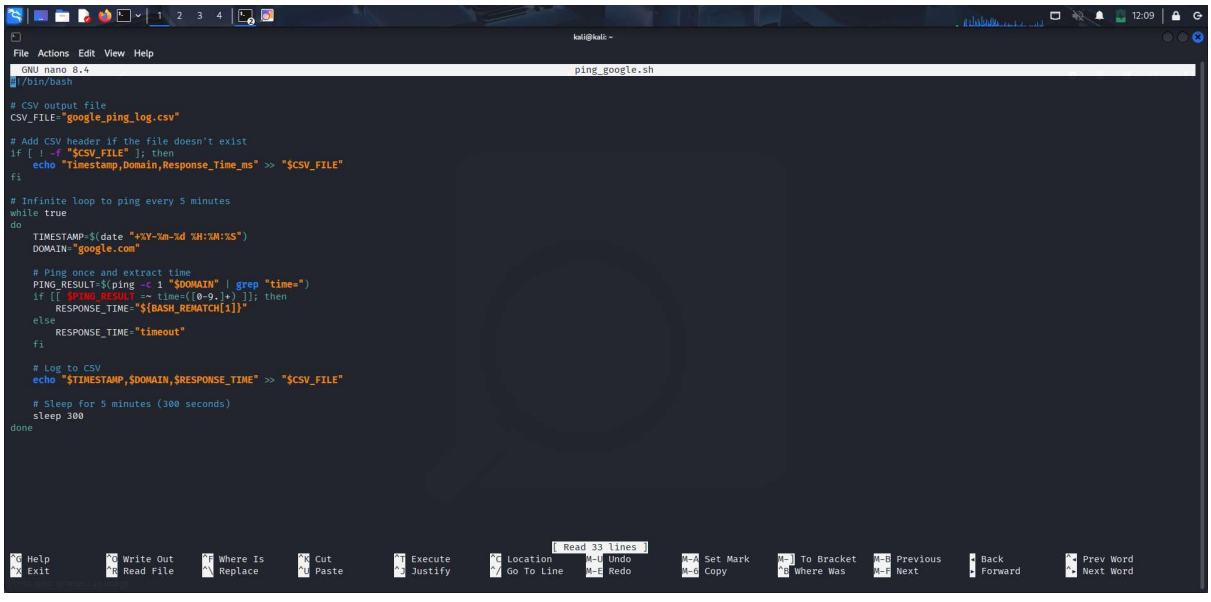
I opened the ping_log.csv file and confirmed that:

- The timestamps were accurate and in chronological order.
- The response times were realistic (around 20-30 ms).
- Timeouts were correctly logged when internet was briefly disabled or the domain was unreachable.

This organized approach ensured the assignment met the technical goals and followed best practices in Bash scripting and system monitoring

Screenshots:





Findings: What I Discovered

- google.com is usually fast (~20–30 ms), but occasional timeouts can occur due to:
  - Local internet drops
  - Packet loss
  - DNS delays
- The script correctly detects and logs when no response is received.

How This Helps with Security Monitoring

- **Detects Website Downtime**: If the ping fails, it shows the website is unreachable, helping spot outages quickly.
- **Identifies DDoS Attacks**: Sudden spikes in response time or repeated timeouts may indicate a denial-of-service attack.
- **Spots Unusual Activity**: Gradual increase in response time can signal network issues or malicious behavior.
- **Provides a Record**: The CSV log helps trace when problems started and supports audits and investigations.
- **Enables Faster Response**: Early detection means quicker action to fix issues before users are affected.

Why Continuous Monitoring?

Continuous monitoring helps track a website's health in real-time. By pinging regularly and logging response times, we can:

- Quickly detect **network issues** or **downtime**
- Spot signs of **DDoS attacks** like repeated timeouts or slow responses
- Identify **service disruptions** before users report them
- Maintain a **record of performance** for analysis and security auditing

This improves reliability and allows faster response to potential threats.

Conclusion

This assignment helped me understand how automated scripts can be used for real-time website monitoring. By using a simple Bash script, I was able to track the availability and response time of a website continuously. The data logged in the CSV file can be used to detect network issues, downtime, or suspicious activity like DDoS attacks. Overall, I learned how automation and logging play a key role in system reliability and security monitoring.