DoS and DDoS Attack Simulation Lab (Kali Linux on Oracle VirtualBox)

This project demonstrates how to simulate and analyze Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks in a controlled lab using Kali Linux on Oracle VirtualBox, focusing on the Xerxes tool, defensive controls (rate limiting, Anti-DoS/DDoS, WAF), log analysis, vulnerability identification, and security best practices.

⚠️ Legal Notice: Run these techniques only in your own lab or explicitly authorized test environments. Never attack real, production, or unauthorized systems.

1. Project Overview

This lab helps you understand and document:

- Definitions and types of DoS/DDoS attacks.
- How to use Xerxes from Kali Linux to simulate HTTP DoS/DDoS against a lab web server.
- How to observe the attack impact using basic monitoring and logs.
- How to test and document protections: rate limiting, Anti-DoS/DDoS features, and Web Application Firewalls (WAFs).
- How to identify vulnerable services, versions, and configurations from scanning and log data.
- How to align the lab with NIST Cybersecurity Framework (CSF) functions and ISO 27001 Annex A controls related to availability and DDoS protection.
- Security best practices, analysis, conclusions, and recommendations.

You will run everything in an Oracle VirtualBox lab with Kali Linux as the attacker and a vulnerable web server VM as the target.

2. Lab Architecture and Requirements

2.1 Lab Topology

- Host machine: Your main Windows/macOS/Linux system.
- Virtualization: Oracle VirtualBox.
- Attacker VM: Kali Linux.
- Target VM: Simple web server (e.g., Ubuntu Server with Apache, Metasploitable, DVWA, or a custom HTTP server).
- Network mode:
    - Option 1: Host-only network (both VMs visible only to the host).
    - Option 2: Internal network (isolated lab network; no Internet).

2.2 Tools Used

- Kali Linux (attacker):
    - `git`, `gcc` (to compile Xerxes).
    - Xerxes DoS/DDoS tool.
    - `nmap` (service and version scanning).
    - `curl`, `ab` (ApacheBench) or `hey` (optional load testing).
    - Log viewers: `less`, `grep`, `tail`.
- Target Web Server:
    - Apache/Nginx or another HTTP server.
    - Basic firewall (iptables/ufw) or cloud WAF/rate-limiting service (if you extend to cloud).

3. DoS and DDoS Concepts

3.1 Definitions

- Denial of Service (DoS): An attack that attempts to make a service unavailable by overwhelming it with traffic or exploiting resource exhaustion using a single attacking host.
- Distributed Denial of Service (DDoS): A DoS attack launched from multiple distributed systems (botnets, compromised hosts) to overwhelm the target's bandwidth, CPU, memory, or application resources.

3.2 Common DoS/DDoS Types

- Volume-based attacks: Flood bandwidth (UDP floods, ICMP floods, amplified reflection attacks).
- Protocol attacks: Abuse network/transport protocols (SYN floods, fragmented packets).
- Application-layer attacks: Target HTTP/HTTPS, APIs, or specific URLs (slow HTTP, login brute force, HTTP floods).

3.3 Why DoS/DDoS Matters

- Direct service outage, SLA violations, and business loss.
- Used as a distraction for other attacks (e.g., data exfiltration, account takeover).
- Regulatory and contractual requirements to ensure availability (e.g., ISO 27001, NIST CSF).

4. Setting Up Kali and Xerxes

4.1 Kali Linux Setup on VirtualBox

1. Download Kali Linux ISO from the official site.
2. Create a new VM in VirtualBox:
    - OS type: Linux → Debian (64-bit).
    - Resources: at least 2 GB RAM, 2 vCPUs.
3. Attach the Kali ISO and install.
4. Configure Kali network adapter to Host-only or Internal network (same as target).

4.2 Target Web Server Setup

1. Install a Linux server VM (e.g., Ubuntu Server).
2. Configure the network adapter to the same Host-only/Internal network.
3. Install Apache:
4. `bash`

```
sudo apt update
sudo apt install apache2 -y
sudo systemctl enable apache2
sudo systemctl start apache2
```

5.
6. Confirm the web server is reachable from Kali:
7. `bash`

```
ping <TARGET_IP>
curl http://<TARGET_IP>/
```

```
┌──(bjnetwork㉿bjnetwork)-[~]
└─$ ping 10.0.2.12
PING 10.0.2.12 (10.0.2.12) 56(84) bytes of data.
64 bytes from 10.0.2.12: icmp_seq=1 ttl=64 time=3.45 ms
64 bytes from 10.0.2.12: icmp_seq=2 ttl=64 time=6.01 ms
64 bytes from 10.0.2.12: icmp_seq=3 ttl=64 time=1.33 ms
64 bytes from 10.0.2.12: icmp_seq=4 ttl=64 time=2.08 ms
64 bytes from 10.0.2.12: icmp_seq=5 ttl=64 time=1.08 ms
64 bytes from 10.0.2.12: icmp_seq=6 ttl=64 time=0.833 ms
64 bytes from 10.0.2.12: icmp_seq=7 ttl=64 time=1.70 ms
64 bytes from 10.0.2.12: icmp_seq=8 ttl=64 time=1.08 ms
64 bytes from 10.0.2.12: icmp_seq=9 ttl=64 time=0.975 ms
64 bytes from 10.0.2.12: icmp_seq=10 ttl=64 time=1.57 ms
64 bytes from 10.0.2.12: icmp_seq=11 ttl=64 time=3.12 ms
```

4.3 Installing Xerxes on Kali

1. Update Kali:
2. `bash`

```
sudo apt update
sudo apt install git gcc make -y
```

3.
4. Clone Xerxes from GitHub:
5. `bash`

```
git clone https://github.com/ngrok90/xerxes.git
cd xerxes
```

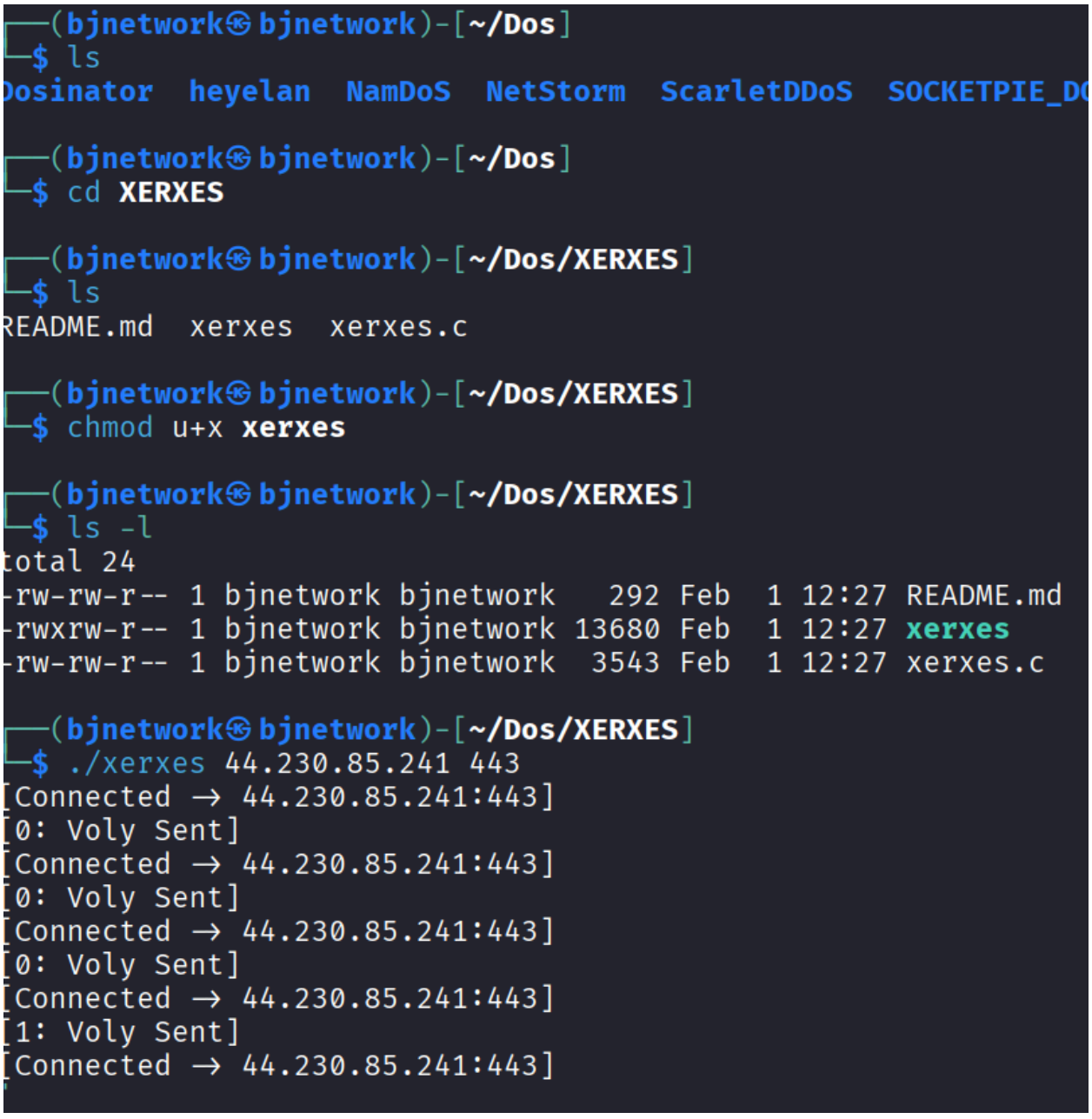
6.
3. Compile Xerxes (if needed, depending on repo structure):
4. `bash`

```
gcc xerxes.c -o xerxes
chmod +x xerxes
```

5.

6. Basic usage syntax:
7. `bash`

```
./xerxes <TARGET_DOMAIN_OR_IP> <PORT>
 Example (HTTP on port 80):
./xerxes 192.168.56.10 80
```



5. Pre-Attack Recon and Scanning

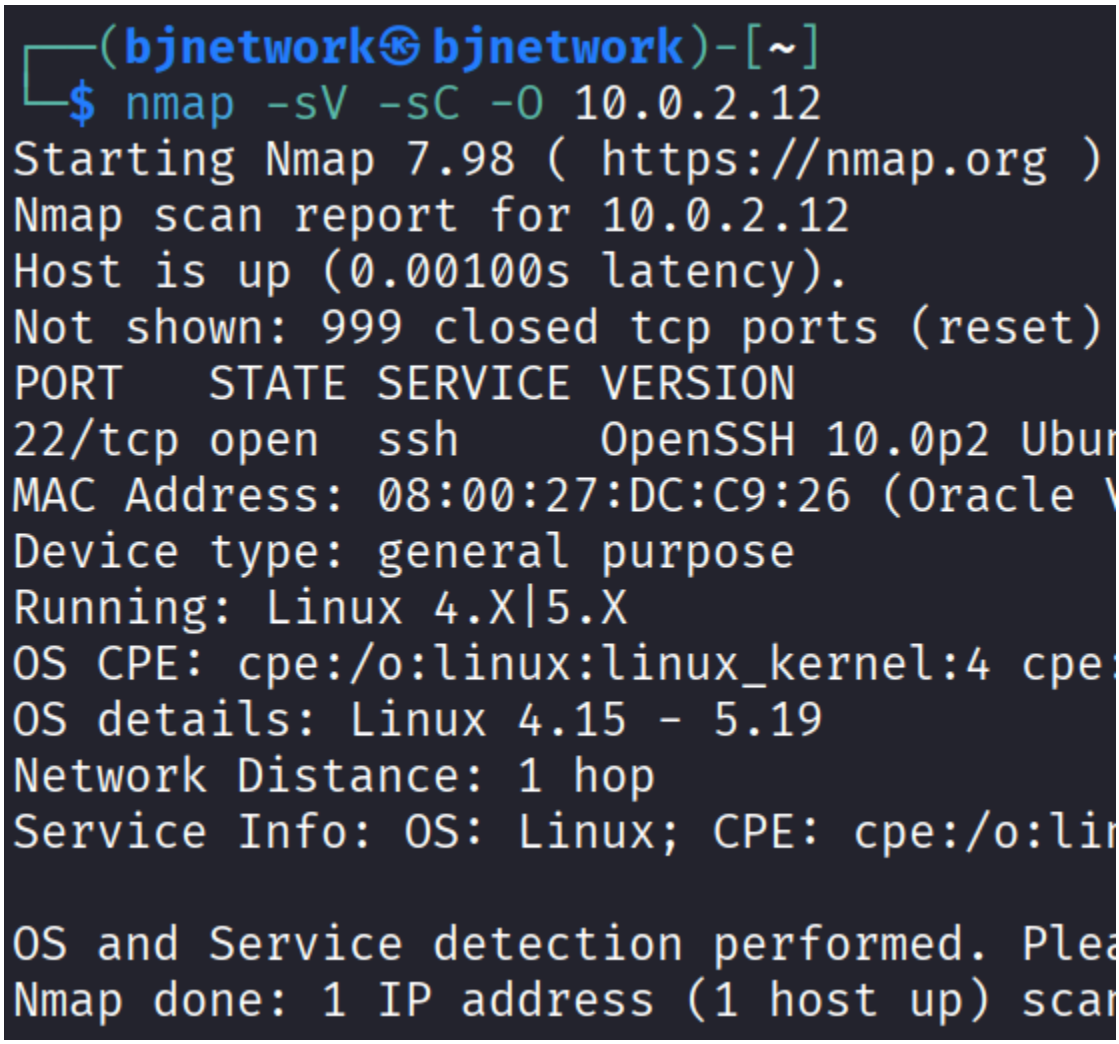
Before simulating DoS, you should understand the target surface.

5.1 Discover Services and Versions (Nmap)

Run an nmap scan from Kali:

bash

```
nmap -sV -sC -O 192.168.56.10
# -sV: service/version detection
# -sC: default scripts
# -O : OS detection
```



Key things to look for:

- Open ports: e.g., 80 (HTTP), 443 (HTTPS), 22 (SSH).
- Service names and versions: e.g., `Apache httpd 2.4.41`, `OpenSSH 8.4`.
- OS fingerprint: Linux version family.

How to use this information:

- Identify vulnerable versions (e.g., old Apache, outdated SSH).
- Decide which service to stress-test (e.g., HTTP on port 80).
- Map to potential CVEs later using vulnerability databases.

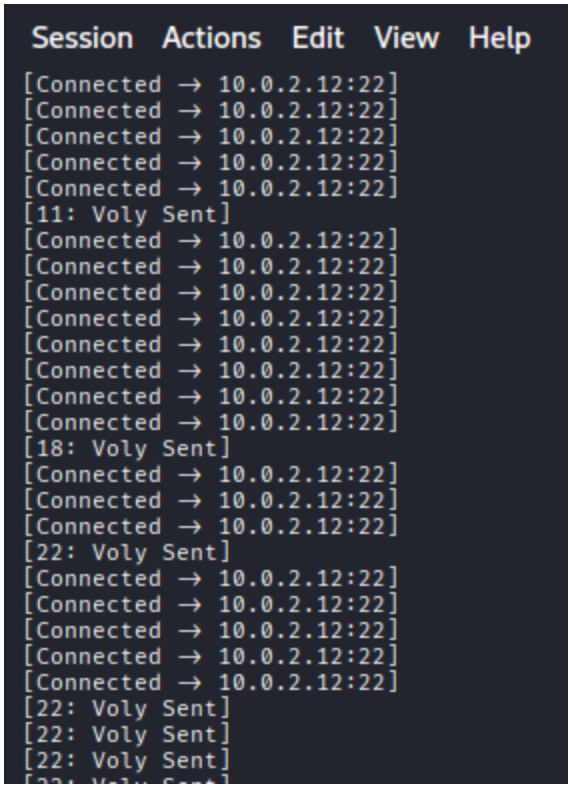
5.2 Important Information to Capture

During scanning and recon, record:

- IP addresses and hostnames of target.
- Open ports and services.
- Service versions and banners.
- Any rate-limiting or error messages from the application when sending multiple requests.

You will reuse this data for:

- Selecting the target endpoint for Xerxes.
- Identifying vulnerabilities (e.g., outdated software).
- Designing defensive rules (e.g., WAF patterns, rate limits for specific URIs).



6. Running DoS/DDoS Simulation with Xerxes

6.1 Baseline: Normal Traffic

1. From Kali, generate light traffic to measure normal behavior:
2. `bash`

```
curl -I http://192.168.56.10/
ab -n 100 -c 10 http://192.168.56.10/
# or another simple HTTP benchmark tool
```

3.
4. Observe: response time, HTTP status codes (200 OK), server load (CPU/RAM) on target.

6.2 Launching Xerxes

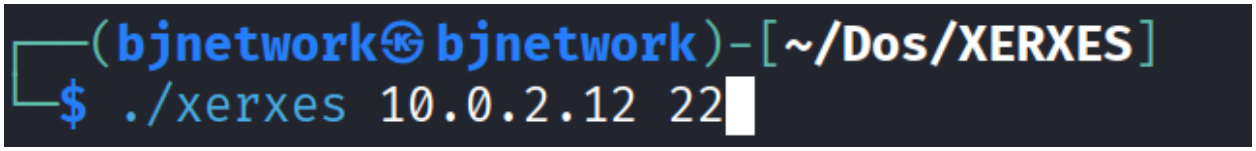
From the `xerxes` directory:

```bash
./xerxes 192.168.56.10 80
```

Xerxes maintains full TCP connections and sends repeated HTTP traffic to exhaust server resources.

Observe on the target:

- Web site becomes slow or unresponsive.
- CPU, memory, or network spikes.
- Apache/Nginx logs show many requests from the Kali IP.



```
[5: Voly Sent]
[Connected → 10.0.2.12:22]
[5: Voly Sent]
[Connected → 10.0.2.12:22]
[5: Voly Sent]
[Connected → 10.0.2.12:22]
[5: Voly Sent]
[Connected → 10.0.2.12:22]
[5: Voly Sent]
[Connected → 10.0.2.12:22]
[5: Voly Sent]
[Connected → 10.0.2.12:22]
[5: Voly Sent]
[Connected → 10.0.2.12:22]
[5: Voly Sent]
[5: Voly Sent]
[Connected → 10.0.2.12:22]
[Connected → 10.0.2.12:22]
[Connected → 10.0.2.12:22]
[Connected → 10.0.2.12:22]
[2: Voly Sent]
```

6.3 Key Things to Analyze During Attack

- Request rate: Number of requests per second (from logs or tools).
- Error rates: HTTP 500, 503, or timeouts.
- Resource usage: CPU, memory, network usage on the server.
- Application behavior: Can you still log in, browse pages, or does it hang?

These observations help you understand the impact and verify if defensive controls work.

7. Log Analysis and Post-Attack Review

7.1 Reading Web Server Logs

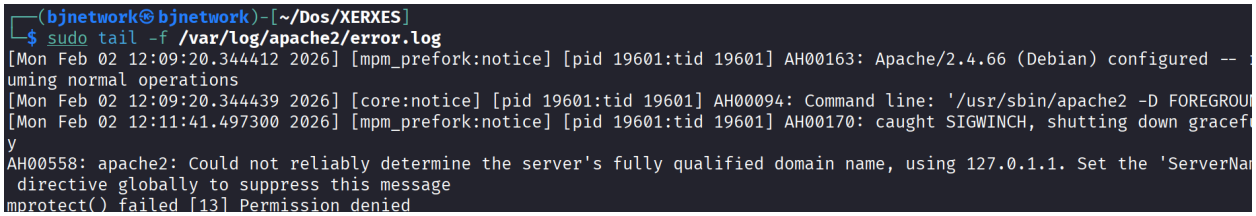
On the target (Apache example):

```
bash

sudo tail -f /var/log/apache2/access.log
sudo tail -f /var/log/apache2/error.log
```

What to look for:

- Many repeated requests from the same IP (Kali) or a small set of IPs.
- Repeated requests to the same URL or endpoint.
- Unusual HTTP methods or user agents.
- In error logs: resource exhaustion errors, timeouts, or worker process failures.



7.2 Important Items After Scanning and Attacks

After scanning and attack simulation, focus on:

- Which services were most affected? (e.g., HTTP vs SSH).
- Which versions and configurations appear weak? (e.g., old Apache, no rate limiting).
- Any misconfigurations:
    - No firewall rules limiting traffic.
    - No WAF/rate limiting at reverse proxy or edge.
- Detection signals:
    - Could you identify the attack in logs quickly?
    - Do you have metrics/alerts for high request rates?

You can use this information to prioritize hardening steps and map to NIST CSF/ISO 27001 controls.

8. Defensive Controls: Rate Limiting, Anti-DoS/DDoS, and WAF

8.1 Concept: Rate Limiting

Rate limiting restricts the number of requests a client can send within a given time window to prevent abuse or resource exhaustion.

Examples:

- Limit login attempts per IP to protect against brute-force and DDoS on auth endpoints.
- Limit API calls per API key or IP to protect backend services.

Advantages:

- Reduces impact of HTTP floods and automated abuse.
- Simple, widely supported in load balancers, reverse proxies, and WAFs.

Disadvantages:

- Can block or throttle legitimate high-traffic users.
- Needs tuning and monitoring to avoid false positives.

8.2 Implementing Simple Rate Limiting (Example: Reverse Proxy or WAF Concept)

In your lab, you can simulate basic rate limiting using:

- Web server modules (e.g., Apache `mod_ratelimit` or Nginx `limit_req`).
- Cloud WAFs (AWS WAF, Cloudflare) if you extend the lab to the cloud.

Example (concept from AWS WAF rate-based rule):

- Set a threshold (e.g., 500 requests in 5 minutes per IP).
- When exceeded, block or challenge the IP (403 response).
- WAF/Reverse proxy configuration page (if you have one) showing a rate limit rule.

8.3 Anti-DoS/DDoS Services

Anti-DoS/DDoS features often include:

- Network-level scrubbing (removing malicious traffic upstream).
- Traffic anomaly detection and auto-mitigation.
- Geo-blocking, IP reputation, and bot filtering.
- Elastic scaling to absorb volumetric attacks.

Advantages:

- Protects against large-scale DDoS beyond local server capacity.
- Often includes 24/7 monitoring and automated mitigation.

Disadvantages:

- Can be costly for small organizations.
- May require DNS changes and reliance on third-party providers.

8.4 Web Application Firewall (WAF)

A WAF sits in front of your web application and filters HTTP(S) traffic based on rules and signatures.

For DoS/DDoS, WAF can:

- Enforce rate limiting and request quotas.
- Block suspicious patterns (e.g., repeated requests to a single endpoint).
- Apply challenges (CAPTCHAs or managed challenges) to separate bots from humans.

Advantages:

- Protects application-layer endpoints and APIs.
- Rules can be tuned for specific URIs, IPs, and methods.

Disadvantages:

- Misconfigured rules can block legitimate users or degrade performance.
- Requires regular maintenance and updates to stay effective.
- Example WAF dashboard showing blocked requests / rate-limited events.

9. Exploiting Vulnerabilities and Weaknesses (Ethically)

In this lab, “exploitation” focuses on demonstrating impact, not actual data theft.

9.1 Identify Weaknesses from Scans and Logs

From `nmap` and logs:

- Outdated HTTP server with known DoS CVEs.
- No rate limiting or WAF between clients and the server.
- Single IP generating thousands of requests with no throttling.

9.2 Step-by-Step Exploitation Walkthrough

1. Select a target endpoint
    - Example: `http://192.168.56.10/` or `/login`.
2. Verify there is no rate limiting or WAF
    - Send many requests with `ab` or `curl` in a loop and see if the server still responds normally or if you get blocked.
3. Launch Xerxes
4. `bash`

```
./xerxes 192.168.56.10 80
```

5.

4. Monitor impact on target
    - Check CPU and memory: `top` or `htop`.
    - Watch `access.log` for rapid requests from Kali IP.
    - Try to browse the site as a normal user → note slowdowns or failures.
5. Document results
    - Take screenshots showing increased resource use, errors, and log patterns.
    - Note how long it takes for the service to become unusable.
6. Apply defenses and repeat
    - Enable a rate limit or WAF rule.
    - Run Xerxes again and compare the effect.
    - Document whether the service remains available and how many attacks get blocked.

10. Mapping to NIST CSF and ISO 27001 Annex A

10.1 NIST Cybersecurity Framework Functions

NIST CSF has five core functions: Identify, Protect, Detect, Respond, Recover.

NIST CSF Function	**How This Lab Relates**	**Example Activities**
Identify	Understand assets, services, and DoS/DDoS risks.	Network mapping, `nmap` scans, identifying critical web services.
Protect	Implement safeguards to limit attack impact.	Rate limiting, WAF rules, Anti-DDoS, hardening server configs.
Detect	Monitor and log abnormal traffic and errors.	Log analysis (`access.log`), alerting on high request rates.
Respond	Take action to reduce or stop ongoing attacks.	Blocking IPs, tightening rate limits, enabling WAF rules.
Recover	Restore normal service and improve resilience.	Restart services, scale resources, update runbooks, lessons learned.

10.2 ISO 27001 Annex A Control Examples

Some ISO 27001 Annex A controls related to DoS/DDoS and availability:

- A.5 Information security policies: Define policies for availability and DoS/DDoS mitigation.
- A.8 Asset management: Identify critical web servers and services.
- A.12 Operations security: Capacity management, protection from malware/DoS, logging and monitoring.
- A.13 Communications security: Network security controls and segmentation to limit attack blast radius.
- A.16 Information security incident management: Procedures to detect, respond to, and learn from DoS/DDoS incidents.
- A.17 Information security aspects of business continuity: Ensure availability of critical services during disruptions.

You can add a dedicated section in your report mapping each lab activity to specific controls (e.g., log analysis → A.12; WAF configuration → A.13 and A.12).

11. Security Best Practices and Recommendations

11.1 Best Practices to Mitigate DoS/DDoS

- Use layered defenses: Combine rate limiting, WAF, firewalls, and Anti-DDoS services.
- Harden services: Keep web servers and OS patched and configured securely.
- Implement logging and monitoring: Capture detailed logs and send them to a SIEM for correlation and alerting.
- Plan capacity and redundancy: Use load balancers, auto-scaling, and redundant instances for critical services.
- Test and tune regularly: Run synthetic load/DoS simulations in controlled environments to validate detection and response.

11.2 What to Analyze After Each Lab Run

After each simulation:

- Traffic patterns: Peak requests per second, IP distribution, top endpoints.
- Server metrics: CPU, memory, network utilization, disk I/O.
- Log anomalies: Errors, timeouts, unusual user agents, spikes in 4xx/5xx codes.
- Control effectiveness:
    - Did rate limiting or WAF reduce impact?
    - Were legitimate users blocked?

Use this analysis to refine configurations and update your incident playbooks.

12. Advantages and Disadvantages Summary

12.1 DoS/DDoS Simulation (Lab Perspective)

Advantages

- Safe environment to learn attack behavior and impacts.
- Helps validate logging, monitoring, and controls.
- Builds understanding of how to map technical work to NIST and ISO requirements.

Disadvantages

- Lab traffic may not fully match complex real-world DDoS patterns.
- Requires careful isolation to avoid harming other systems.
- Can be resource-intensive (CPU/network) even in a lab.

12.2 Xerxes as a Tool

Advantages

- Easy to use with simple syntax.
- Focuses on HTTP traffic with full TCP connections, making it effective in stressing web servers.

Disadvantages

- Single-source; not as realistic as large botnet-based attacks.
- May be detected/blocked easily by well-configured WAF/rate limits.

13. How to Structure This Repository

Suggested repo structure:

```text
dos-ddos-xerxes-lab/
├─ README.md                  # This file
├─ lab-diagrams/
│  └─ network-topology.png    # Screenshot or diagram of VirtualBox network
├─ screenshots/
│  ├─ virtualbox-network.png
│  ├─ kali-xerxes-run.png
│  ├─ apache-access-log.png
│  ├─ nmap-scan-output.png
│  ├─ waf-rate-limit-config.png
│  └─ metrics-before-after.png
├─ configs/
│  ├─ apache-example.conf     # Optional: sample Apache config (no secrets)
│  └─ waf-rate-limit-rules.txt
├─ notes/
│  ├─ nist-csf-mapping.md
│  └─ iso27001-annexA-mapping.md
└─ logs-examples/
   ├─ apache-access-sample.log
   └─ apache-error-sample.log
```

14. Conclusion, Analysis, and Recommendations

From this lab you learned how a single attacker using tools like Xerxes can significantly impact a vulnerable web server, and how logs and metrics reveal the traffic patterns and service degradation. By applying rate limiting, WAF rules, and other Anti-DDoS techniques, you saw how layered defenses can reduce or block malicious traffic while attempting to preserve legitimate access.

In a real environment, organizations should integrate these technical controls into a broader cybersecurity management program aligned with NIST CSF and ISO 27001, ensuring that availability is treated as a critical security objective alongside confidentiality and integrity. Regular testing, monitoring, and updating of defenses, combined with clear incident response and recovery procedures, are essential to maintain resilience against evolving DoS and DDoS threats.