Task 1 Report: Advanced Exploitation Lab

Target: Metasploitable2 Linux VM
Tester: Mohit Relekar

Executive Summary

A successful penetration test was conducted against the target host, focusing on advanced exploitation techniques. The assessment identified a critical remote code execution vulnerability in the Java RMI Registry service. This vulnerability was exploited to gain initial access, leading to a complete compromise of the target system. The exploit chain demonstrates a lack of security hardening around legacy Java services.

Exploit Chain Log

Exploit ID	Description	Target IP	Status	Payload
007	Java RMI Registry Command Execution	192.168.56.104	Success	cmd/unix/reverse

Technical Findings

1. Reconnaissance & Service Discovery

Initial reconnaissance via network scanning identified the target host and enumerated its running services. The Java RMI Registry service on port 1099 was identified as a primary attack vector due to its known susceptibility to command injection attacks.

Evidence: 01_Nmap-Scan.png

2. Exploitation

The `exploit/multi/misc/java_rmi_server` module was successfully executed against the target. The exploit works by tricking the RMI registry into loading a malicious Java class from a remote attacker-controlled server, resulting in remote command execution.

Evidence: 02_Java_RMI_Exploit_Success.png

3. Post-Exploitation & Proof of Access

Following successful exploitation, a command shell session was established. System reconnaissance commands were executed to confirm access and identify the underlying operating system. The ability to read the `/etc/passwd` file was demonstrated, proving unauthorized access to sensitive system information.

Evidence: 03_Proof_Of_Access.png

Custom PoC Development Summary

A public Python exploit for a buffer overflow vulnerability was modified to enhance its reliability. The return address was updated to a JMP ESP instruction specific to the target binary's memory space (`0x625011AF`). A 32-byte NOP sled was prepended to the payload to increase the probability of successful execution, and the reverse shell payload was parameterized to accept configurable LHOST and LPORT values.

ASLR Bypass Technique Summary

Address Space Layout Randomization was bypassed using a Return-Oriented Programming technique. Gadgets including `pop rdi; ret` were chained to leak a Libc function address through the puts() function. This address was used to calculate the base address of Libc, allowing for the successful execution of system('/bin/sh') despite runtime memory randomization.

Remediation Recommendations

1. Immediate Action: Disable the Java RMI Registry service on all non-essential systems.
2. Network Segmentation: Implement firewall rules to restrict access to port 1099/tcp from untrusted networks.
3. Patch Management: Apply the latest security patches to all Java Runtime Environments and remove outdated software.
4. Compensating Controls: Deploy intrusion detection system (IDS) rules to detect malicious RMI traffic and exploit attempts.

Conclusion

The Java RMI Registry service presented a critical vulnerability that allowed complete system compromise. The successful demonstration of this exploit chain underscores the importance of rigorous patch management and network security controls to protect against known vulnerabilities in legacy services.