Discovering vulnerabilities quickly with targeted scanning(essentials)

Executive Summary

This assessment identified a critical arbitrary file read vulnerability within the target web application. By strategically applying targeted scanning techniques, we were able to retrieve sensitive server files, specifically the contents of `/etc/passwd`. The issue poses a **high severity risk**, as it allows unauthorized access to system-level information, potentially enabling privilege escalation or reconnaissance for further exploitation.

Introduction

The objective of this engagement was to assess the effectiveness of targeted scanning in identifying high-impact vulnerabilities within strict time constraints. The focus was on leveraging Burp Scanner's precision scanning capabilities to rapidly detect and exploit any file disclosure issues.

Methodology

1. Used Burp Suite's scanner to perform a targeted scan on specific endpoints instead of full application scanning to stay within the time limit.

2. Manually inspected application behaviour and guessed potentially vulnerable routes (e.g., download, view, or export endpoints).

3. Launched an active scan on a suspicious endpoint with user-controllable parameters.

4. Analyzed the scanner's findings to confirm a local file inclusion (LFI) vulnerability.

5. Manually crafted a payload to exploit the vulnerability and retrieve system file contents.

Vulnerability Findings

- **Type:** Local File Inclusion (LFI)

- **Location:** Parameter in a vulnerable endpoint (e.g., `?file=...`)

- **Severity:** High

Description

The application exposes a parameter that allows user-controlled input to be passed directly to the file system access logic without proper validation or sanitization. By manipulating the parameter to include file traversal sequences (`../../`), it was possible to access and read arbitrary files on the server.

Proof of Concept (PoC)

1. Identified a parameter suspected of accessing local files:

   `GET /download?file=report.pdf`

2. Modified the request to target `/etc/passwd`:

   `GET /download?file=../../../../etc/passwd`

3. Received a successful response containing contents of the system password file:

   ruby

   CopyEdit

   `root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin ...`

4. Verified that the vulnerability was exploitable without authentication and responded quickly enough to solve the lab within the time constraint.

Impact Assessment

Exploitation of this vulnerability allows attackers to:

- Read arbitrary files on the server, including configuration files, logs, and credentials
- Conduct system enumeration and planning for lateral movement or privilege escalation.
- Potentially retrieve application source code or sensitive user data.

Recommendations

1. Implement strict input validation on all user-supplied file names and paths.
2. Use whitelisting to limit file access to specific directories and file types.
3. Avoid directly passing user input into file system functions.

4. Perform routine vulnerability scans with targeted focus on dynamic and file-based endpoints.

5. Educate developers on secure coding practices for file handling.

Conclusion

This assessment highlights the effectiveness of targeted scanning in high-pressure scenarios. The identified LFI vulnerability underscores the need for robust input sanitization and access controls in file management functionality. Proactively addressing such flaws will significantly strengthen the application's security posture.