Security Assessment Report

Report Title: Conduct an in-depth pentest / security audit on https://testing.pinewheel.ai

Prepared By: Ojas Gurav.

---

1. Unauthorized Access to /etc/passwd File

1. Overview

During security testing, it was observed that the contents of the /etc/passwd file were accessed using the command:

cat /etc/passwd

Severity: High

2. Risk Analysis

The presence of readable /etc/passwd file poses potential security risks, as it provides an attacker with key system information.

- Enumeration of valid usernames.
- Identification of system and service accounts.
- Potential exploitation of misconfigured accounts.

3. Key Findings

- Root Account (root:x:0:0) - The highest privilege account.
- Service Accounts (e.g., daemon, bin, sys, mail, proxy, www-data) - These are used by system services and should not have interactive login access.

4. Security Concerns

- World-readable File: If /etc/passwd has improper permissions.
- Privilege Escalation Risks: Attackers can use this information to craft targeted attacks, such as brute-force password attempts or exploiting weak configurations.

5. Recommended Mitigation

1. Restrict File Permissions:
    - Set the correct file permissions to prevent unauthorized access

2. Sensitive Data Exposure:

Overview

The **/dev** directory in Unix-like operating systems contains special files that serve as interfaces to the system's hardware and internal functionalities. These files are essential for the proper functioning of the system, as they allow user-space programs to interact with devices and system resources.

Severity: High

Methodology

Prompt Used this prompt **/dev**

Findings

console

core

fd

full

mqueue

null

ptmx

pts

random

Risk

Attackers could retrieve critical system paths and configurations.

3.Command Injection

Overview

During a recent security audit, a command injection vulnerability was identified on the target testing.pinewheel.ai The test involved appending a command ';sleep(5) to user-supplied input. The system executed the injected command, as evidenced by a delayed response.

Severity: High

Methodology

- Testing Approach: A command injection test was performed by appending ';sleep(10) to an input field.
- Observed Behavior: The system responded with a delay of 5 seconds, confirming that the injected command (;sleep(5)) was executed on the server side.
- Inference: The execution of the sleep command demonstrates that unsanitized user input is being passed to the system shell, making the site vulnerable to command injection**.**

Findings

- Vulnerability Confirmation: The delay in response confirms that the application is susceptible to command injection attacks.
- Security Implications: An attacker could exploit this vulnerability to execute arbitrary commands on the server. This may lead to unauthorized access, data breaches, or further system compromise.
- Criticality: Command injection is considered a high-risk vulnerability because it allows attackers to execute system-level commands, potentially leading to a complete takeover of the server.

Recommendations

- Input Sanitization: Implement strict input validation and sanitization to filter out potentially malicious characters (such as semicolons and other command separators).