

SHARKZ4HIRE

SECURITY WITH TEETH

Pentest Report 2025 - Crew Members Hub

Prepared for: CVSS Bonsecours

Tiger Shark

2025-05-02

Contents

Document Revisions	4
Executive Summary	5
Introduction	5
Findings	5
Recommendations	5
Scope and Methodology	6
Scope	6
Architecture Diagram	7
Methodology	7
Findings and Recommendations	8
S4H-001: Service Account Weak Password	9
Assets Impacted	9
Description	9
Proof	9
Recommendation	10
References	10
S4H-002: Dangerous Sudoer Rule for www-data	11
Assets Impacted	11
Description	11
Proof	11
Recommendation	11
References	12
S4H-003: Outdated Dependency (CVE-2022-21797)	13
Assets Impacted	13
Description	13
Proof	13
Recommendation	13
References	13
S4H-004: Anonymous FTP File Upload	14
Assets Impacted	14
Description	14
Proof	15
Recommendation	15
References	16

S4H-005: Prototype Pollution in jQuery (CVE-2019-11358)	17
Assets Impacted	17
Description	17
Proof	18
Recommendation	18
References	18
S4H-006: Open Ports	19
Assets Impacted	19
Description	19
Proof	20
Recommendation	20
References	21
S4H-007: Exposed Directory Listing	22
Assets Impacted	22
Description	22
Proof	23
Recommendation	23
References	23
Tools and Technologies	24
Observations and Additional Recommendations	24
Observations	24
Reliance on IPv6 Addresses	24
Standard Port Usage	24
Publicly Accessible Test Files	25
Additional Recommendations	25
Enhance IPv6 Security Awareness and Tooling	25
Consider Non-Standard Ports (with Caution)	25
Implement Regular Security Audits of Web Content	25
Strengthen Logging and Monitoring	25
Adopt a “Principle of Least Privilege” More Granularly	26
Implement Security Headers	26
Conduct Regular Vulnerability Scanning	26
Enhance Password Complexity Requirements	26
Implement Network Segmentation	26
Appendix	27
Appendix A - Log Application	27

Appendix B - SSH Conf 30

Document Revisions

Version	Date	Amendment	Author
0.1	2025-04-18	Initial version	Tiger Shark
0.2	2025-04-20	Fix typos	Tiger Shark
0.3	2025-05-01	Review	Thresher Shark
1.0	2025-05-02	Final version	Dusky Shark

Executive Summary

Introduction

This report summarizes the findings of a penetration test conducted on a crew members hub applications and associated server infrastructure. The assessment aimed to identify security vulnerabilities that could compromise the confidentiality, integrity, or availability of the application and its data.

Findings

The penetration test identified several critical vulnerabilities, including an outdated jQuery library, open ports and weak password policies across all servers.

ID	Title	Severity	CVSS Score
S4H-001	Service Account Weak Password	Critical	10.0
S4H-002	Dangerous Sudoer Rule for www-data	Critical	9.9
S4H-003	Outdated Dependency (CVE-2022-21797)	Critical	9.8
S4H-004	Anonymous FTP File Upload	Critical	9.5
S4H-005	Prototype Pollution in jQuery (CVE-2019-11358)	Critical	9.3
S4H-006	Open Ports on sea-shanty.ctf	Critical	9.1
S4H-007	Exposed Directory Listing	Informational	N/A

Recommendations

Immediate remediation actions are required to address the identified vulnerabilities. These include patching the libraries, closing open ports, and enforcing strong password policies.

To fortify your digital defenses against evolving threats, an annual penetration test is an indispensable investment in your organization’s security posture.

Scope and Methodology

This section details the boundaries and the technical approach taken during the penetration test. It ensures transparency about what was assessed and how the assessment was performed.

Scope

The scope of this penetration test was carefully defined to focus on the security posture of the server infrastructure responsible for hosting and managing these applications. This section explicitly outlines the specific systems, applications, and components that were included within the boundaries of this assessment, providing clarity on what was actively tested. Conversely, it also delineates any elements that were explicitly excluded from this engagement to ensure a clear understanding of the assessment's limitations and focus.

In Scope:

- Passwords strength test
- Code review python logging application
- SSH Configuration
- Servers:
 - sea-shanty.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:281)
 - crew-quarters.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:280)
 - ocean-diary.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:279)

Out of Scope:

- Any server not included the architecture diagram.
- Physical security of the server locations.
- Anchor.

Architecture Diagram

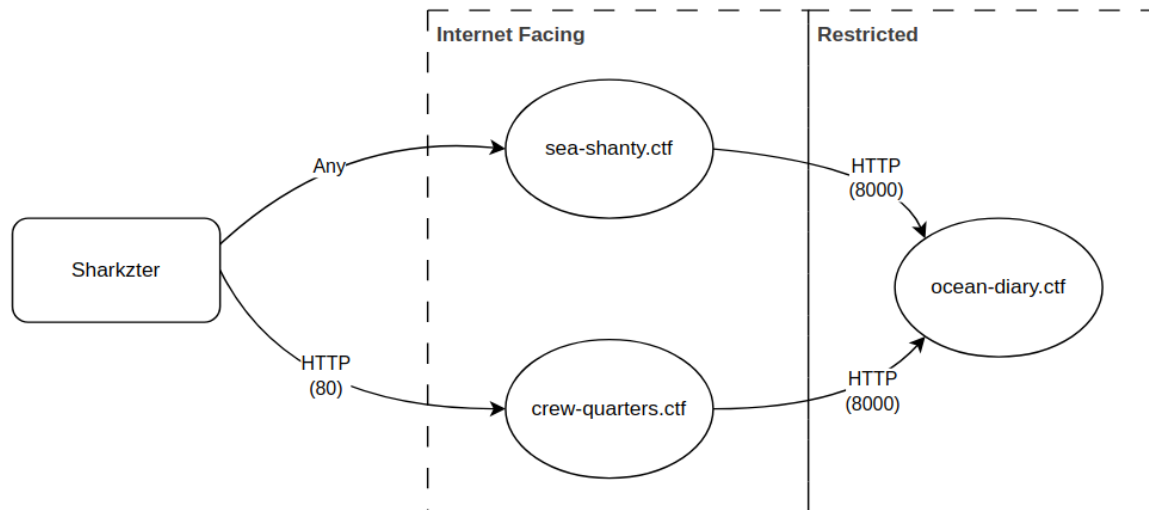


Figure 1: Crew Members Hub Architecture supplied by CVSS Bonsecours.

Methodology

During this penetration test, a multi-faceted approach was employed, commencing with Information Gathering. This phase involved leveraging Open Source Intelligence (OSINT) techniques to gather publicly available information about the target organization and its infrastructure, complemented by active DNS enumeration to map out the network footprint. Subsequently, Vulnerability Scanning was conducted utilizing industry-standard automated tools to identify potential security weaknesses across the defined scope. To validate the findings of automated scans and uncover more complex vulnerabilities, rigorous Manual Testing was performed. This included in-depth web application testing, meticulous network protocol analysis, and targeted examination of identified potential weaknesses. Where explicitly agreed upon beforehand, the Exploitation phase involved attempting to leverage identified vulnerabilities in a controlled manner to demonstrate potential impact. If within the agreed scope, Social Engineering tactics were employed to evaluate human susceptibility to manipulation. Throughout the engagement, any provided or obtained Credential Usage was carefully documented and utilized solely for authorized testing purposes. Several assumptions were made during this assessment, including the accuracy of the provided network diagrams and the stability of the target environment. Known limitations included the inherent constraints of time-boxed testing and the potential for automated tools to miss certain nuanced vulnerabilities detectable only through manual analysis.

Findings and Recommendations

The subsequent sections of this report present the critical findings uncovered during the penetration test, offering a detailed examination of the specific security vulnerabilities identified within the assessed environment. For each finding, a comprehensive analysis is provided, outlining the potential impact and risk level associated with the weakness. Crucially, this section also delivers clear, actionable recommendations designed to effectively remediate the identified vulnerabilities and enhance the overall security posture of the organization. Prioritization of these recommendations is included to guide immediate and strategic security improvements.

S4H-001: Service Account Weak Password

Severity	Score	CVSS Vector
Critical	10.0	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Assets Impacted

- sea-shanty.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:281)
- crew-quarters.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:280)
- ocean-diary.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:279)

Description

A weak password for the service account `sa` was successfully cracked using bruteforce techniques, granting unauthorized access which could potentially be leveraged to escalate privileges or compromise sensitive system resources.

Proof

```
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: REDACTED
Hash.Target.....: REDACTED
Time.Started.....: Tue Apr 15 12:17:55 2025 (220 secs)
Time.Estimated...: Tue Apr 15 12:18:17 2025 (0 secs)
Kernel.Feature....: Pure Kernel
Guess.Base.....: File (rockyou-75.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 278 H/s (0.477ms) @ Accel:256 Loops:256 Thr:1 Vec:4
Recovered.....: 1/1 (100.00%) Digests
Progress.....: REDACTED
Rejected.....: 0/48896 (0.00%)
Restore.Point....: REDACTED
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4864-5000
Candidate.Engine.: Device Generator
Hardware.Mon.#1..: Temp: 376c Util: 292%
```

Recommendation

The password for the *sa* service account must be immediately changed to a strong, unique password adhering to industry best practices, including a minimum length of 12-15 characters, a combination of uppercase and lowercase letters, numbers, and special symbols. Consideration should also be given to the principle of least privilege, ensuring the service account has only the necessary permissions to perform its intended functions, thereby limiting the potential impact of any future unauthorized access.

References

- [Create and use strong passwords - Microsoft Support](#)
- [Formulate Strong Passwords and PIN Codes - CISA](#)
- [Password Policy Best Practices for Strong Security in AD - Netwrix](#)

S4H-002: Dangerous Sudoer Rule for www-data

Severity	Score	CVSS Vector
Critical	9.9	CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

Assets Impacted

- sea-shanty.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:281)

Description

A critical sudoer misconfiguration grants the www-data user the ability to execute tar with root privileges on its home directory. This severe vulnerability may allow potential attackers who compromise the web server to gain full system control without a password, leading to high-impact security breaches.

Proof

```
www-data@sea-shanty:/$ sudo -l
Matching Defaults entries for www-data on sea-shanty:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin
    \:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User www-data may run the following commands on sea-shanty:
    (ALL) NOPASSWD: /usr/bin/tar ^cz?v?h?f /backups/[a-zA-Z0-9_]+.tar.gz /
    var/www/html$
```

Recommendation

The sudoer rule for the www-data user must be immediately modified to restrict the use of the tar command. Instead of allowing unrestricted execution, the rule should be narrowly defined to permit only specific, necessary tar operations with explicitly defined parameters and target directories, if absolutely required. Furthermore, the necessity of granting any sudo privileges to the www-data user should be re-evaluated, and the principle of least privilege strictly enforced. If tar operations are required, consider alternative solutions that do not involve granting direct sudo access, such as using dedicated scripts executed by a different, less privileged user or leveraging application-level functionalities. All sudoer rule changes should be thoroughly reviewed and tested in a non-production environment before implementation.

References

- [Sudo - Debian](#)
- [Sudoers - Ubuntu](#)

S4H-003: Outdated Dependency (CVE-2022-21797)

Severity	Score	CVSS Vector
Critical	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

Assets Impacted

- ocean-diary.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:279)

Description

The project uses a version of the `joblib` library prior to 1.2.0, which is vulnerable to Arbitrary Code Execution (ACE). This critical security flaw exists within the `Parallel()` class due to the unsafe use of the `eval()` function when processing the `pre_dispatch` flag.

Proof

requirements.txt:

```
flask==3.1.0
joblib==1.1.0
unicorn==23.0.0
```

Recommendation

Immediately upgrade the `joblib` dependency to version **1.2.0** or a later secure version. This release contains the necessary fix to prevent the execution of arbitrary code through the `pre_dispatch` flag.

References

- [CVE-2022-21797 - NIST](#)
- [CWE-94: Improper Control of Generation of Code - MITRE](#)

S4H-004: Anonymous FTP File Upload

Severity	Score	CVSS Vector
Critical	9.5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/RC:R

Assets Impacted

- sea-shanty.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:281)

Description

Anonymous FTP file upload is enabled on the server. This configuration allows any unauthenticated user to upload files to a designated directory on the FTP server without requiring a username or password. Enabling anonymous FTP file uploads poses significant security risks, including:

- **Malware Uploads:** Attackers can upload malicious files (e.g., viruses, worms, trojans) that could potentially infect the server or be distributed to other users.
- **Illegal Content Hosting:** The server could be used to host illegal or inappropriate content, potentially leading to legal and reputational damage.
- **Denial of Service (DoS):** Attackers could flood the server with a large number of files, consuming disk space and potentially causing the server to crash or become unresponsive.
- **Data Exfiltration (Indirectly):** While direct downloading might be the primary concern with anonymous FTP, attackers could potentially stage malicious scripts or tools for later use in data exfiltration.
- **Server Resource Abuse:** Uploaded files consume server resources (disk space, bandwidth), which could impact the performance of legitimate services.

Proof

```
tiger@shark:/$ ftp -6 9000:d37e:c40b:92cf:216:3eff:fe5a:281
Connected to 9000:d37e:c40b:92cf:216:3eff:fe5a:281.
220 Sea Shanty Collection
Name (9000:d37e:c40b:92cf:216:3eff:fe5a:281:tiger): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd draft
250 Directory successfully changed.
ftp> put 5h4rkz73r5.txt 5h4rkz73r5.txt
local: 5h4rkz73r5.txt remote: 5h4rkz73r5.txt
229 Entering Extended Passive Mode (||65504|)
150 Ok to send data.
100% |*****|    19      272.86 KiB/s    00:00 ETA
226 Transfer complete.
19 bytes sent in 00:00 (49.74 KiB/s)
```

Recommendation

Disable anonymous FTP file upload immediately. If file sharing is required, implement secure alternatives that enforce authentication and authorization, such as:

- **Authenticated FTP/SFTP:** Require users to log in with valid credentials before uploading or downloading files.
- **Secure File Sharing Platforms:** Utilize web-based file sharing applications with robust access controls and auditing capabilities.
- **Secure Cloud Storage:** Consider using secure cloud storage services that offer granular permission settings.

If anonymous uploads are deemed absolutely necessary for a specific business case, implement stringent controls, including:

- **Dedicated Upload Directory with Limited Permissions:** Restrict the permissions on the upload directory to prevent uploaded files from being executed or accessed by other processes.
- **File Size Limits:** Implement strict file size limits to prevent DoS attacks through large file uploads.
- **Regular Monitoring and Scanning:** Implement automated tools to regularly scan the upload directory for malicious content.
- **Retention Policies:** Establish clear retention policies for uploaded files and automatically remove older files.

References

- [Anonymous FTP Servers - Cobalt](#)
- [FILE TRANSFER PROTOCOL \(FTP\) - RFC](#)

S4H-005: Prototype Pollution in jQuery (CVE-2019-11358)

Severity Score	CVSS Vector
----------------	-------------

Critical 9.3	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N/E:H/RL:O/RC:C/CR:H/IR:H/MC:H/MI:H/MA:L
---------------------	---

Assets Impacted

- crew-quarters.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:280)
- Anchor

Description

The project utilizes a version of the jQuery library prior to 3.4.0. This version is vulnerable to Object.prototype pollution (CVE-2019-11358) due to improper handling of the `jQuery.extend(true, {}, ...)` function.

If an attacker can control an unsanitized source object that is merged using `jQuery.extend(true, {}, source)`, and this object contains an enumerable `__proto__` property, it can lead to the modification of the native `Object.prototype`. This can have significant security implications, including:

- **Prototype Pollution:** Modifying `Object.prototype` can lead to unexpected behavior across the entire JavaScript application, potentially affecting built-in methods and properties of all objects.
- **Denial of Service (DoS):** Malicious modifications to `Object.prototype` could cause the application to crash or become unresponsive.
- **Security Bypass:** In some scenarios, prototype pollution vulnerabilities can be leveraged to bypass security checks or introduce unintended functionality. For example, an attacker might be able to inject properties that are unexpectedly accessed by application logic.
- **Cross-Site Scripting (XSS) (Indirectly):** While not a direct XSS vulnerability in jQuery itself, prototype pollution can sometimes be a prerequisite or contributing factor to other vulnerabilities that could lead to XSS.

Proof

```
tiger@shark:/$ curl -s http://[9000:d37e:c40b:92cf:216:3eff:fe5a:280]/js/  
jquery-3.3.1.min.js | head -c 20  
/!* jQuery v3.3.1 |
```

Recommendation

Immediately upgrade the jQuery library to version **3.4.0** or a later secure version. This version includes a fix that properly handles the `__proto__` property and prevents Object.prototype pollution.

References

- [CVE-2019-11358 - NIST](#)
- [jQuery < 3.4.0 Object Prototype Pollution Vulnerability - Tenable](#)

S4H-006: Open Ports

Severity	Score	CVSS Vector
Critical	9.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H/E:U

Assets Impacted

- sea-shanty.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:281)

Description

The following network services were found to be listening on publicly accessible ports:

- **21/tcp: ftp (File Transfer Protocol)** - An unencrypted protocol for transferring files.
- **22/tcp: ssh (Secure Shell)** - A secure protocol for remote command-line access.
- **80/tcp: http (Hypertext Transfer Protocol)** - The standard protocol for web traffic.

Exposing these ports to the internet increases the attack surface of the system and presents several potential risks:

- **Port 21/tcp (FTP):**
 - **Brute-Force Attacks:** Attackers can attempt to guess usernames and passwords to gain unauthorized access to the file system.
 - **Unencrypted Communication:** Data transmitted over standard FTP is not encrypted, making it susceptible to eavesdropping and interception of sensitive information, including credentials and transferred files.
 - **Vulnerability Exploitation:** FTP servers may have known vulnerabilities that attackers can exploit to gain unauthorized access or execute arbitrary commands.
 - **Anonymous Access:** If anonymous FTP is enabled (which wasn't explicitly determined in this scan but is a common misconfiguration), anyone can potentially upload or download files, leading to the risks outlined in a previous finding.

- **Port 22/tcp (SSH):**

- **Brute-Force Attacks:** Attackers will frequently attempt to brute-force SSH passwords to gain remote command-line access with potentially high privileges.
- **Vulnerability Exploitation:** SSH daemons can have vulnerabilities that could allow attackers to bypass authentication or execute arbitrary code.
- **Weak Key Exploitation:** If weak or default SSH keys are used, they could be compromised, granting unauthorized access.

- **Port 80/tcp (HTTP):**

- **Web Application Vulnerabilities:** If a web server is running on this port, it may be susceptible to a wide range of web application vulnerabilities (e.g., SQL injection, cross-site scripting, remote code execution) that attackers can exploit to compromise the server or its data.
- **Information Disclosure:** Misconfigured web servers can inadvertently expose sensitive information.
- **Denial of Service (DoS):** Web servers can be targeted with DoS attacks to overwhelm resources and make the service unavailable.

Proof

```
tiger@shark:/$ nmap -6 -sT 9000:d37e:c40b:92cf:216:3eff:fe5a:281 -p- -T4 -Pn
Starting Nmap 3.15BETA3 ( https://nmap.org ) at 2025-04-08 12:31 EDT
Nmap scan report for 9000:d37e:c40b:92cf:216:3eff:fe5a:281
Host is up (22s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 250 hours
```

Recommendation

A thorough review of the necessity of exposing these services to the public internet is required.

- **FTP (Port 21):** Disable FTP if it is not absolutely necessary. If file transfer is required, strongly consider using SFTP (over SSH on port 22) or a secure web-based file transfer solution that encrypts data in transit and enforces strong authentication.

- **SSH (Port 22):** If remote command-line access is required, implement strong security measures for SSH, including:
 - **Using strong, unique passwords or, preferably, SSH key-based authentication.**
 - **Disabling password-based authentication if key-based authentication is implemented.**
 - **Changing the default SSH port (though this provides security through obscurity and shouldn't be the only security measure).**
 - **Restricting SSH access to specific IP addresses or networks using firewall rules.**
 - **Keeping the SSH server software up to date.**
- **HTTP (Port 80):** If a web server is intended to be publicly accessible, ensure the web application running on it is securely developed and regularly updated to patch any known vulnerabilities. Implement security best practices, such as input validation, output encoding, and protection against common web attacks. Consider redirecting HTTP traffic to HTTPS (port 443) to ensure encrypted communication.

References

- [Port 21 - SANS](#)
- [Port 22 - SANS](#)
- [Port 80 - SANS](#)

S4H-007: Exposed Directory Listing

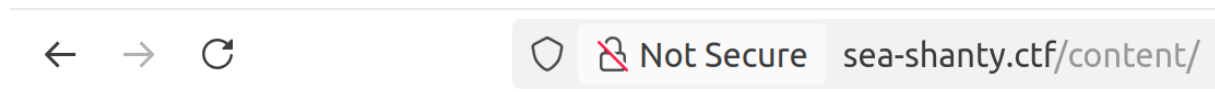
Severity	Score	CVSS Vector
Informational	N/A	N/A

Assets Impacted








- sea-shanty.ctf (9000:d37e:c40b:92cf:216:3eff:fe5a:281)

Description

It was observed that directory listing is enabled for the path /[content](#). This configuration allows anyone browsing to this URL to view a list of the files and subdirectories contained within. While this does not represent an immediate high-risk vulnerability, it can inadvertently expose sensitive information about the application's structure, file naming conventions, and potentially reveal the existence of hidden or backup files. This information could be leveraged by an attacker during reconnaissance to gain a deeper understanding of the target system and identify potential attack vectors.

Proof

Index of /content

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 ahoy.md	2025-04-24 20:26	1.6K	
 departure.md	2025-04-24 20:26	1.6K	
 draft/	2025-04-24 21:03	-	
 index.md	2025-04-24 20:26	1.1K	
 may_morning.md	2025-04-24 20:26	1.6K	
 midnight_blue.md	2025-04-24 20:26	1.6K	

Apache/2.4.58 (Ubuntu) Server at sea-shanty.ctf Port 80

Recommendation

It is recommended to disable directory listing for all web-accessible directories unless there is a specific and well-documented business need for it. Standard security practice is to prevent the exposure of internal file structures.

References

- [CWE-548: Exposure of Information Through Directory Listing](#)
- [Directory listing - PortSwigger](#)

Tools and Technologies

This section enumerates the software, utilities, and technologies leveraged throughout the penetration testing engagement. This disclosure aims to provide transparency regarding the technical resources employed during the assessment.

- **Network Discovery and Port Scanning:** Nmap (Version 3.15BETA3) was utilized to conduct comprehensive network scanning and service enumeration, identifying open ports and active services on the designated target systems.
- **Password Auditing:** hashcat was employed for offline cryptographic password analysis, specifically to evaluate the strength of identified account credentials.
- **File Transfer Protocol Interaction:** The native operating system command-line FTP client was used to interact with the FTP service, facilitating the assessment of anonymous access and file upload functionalities.
- **Web Content Retrieval:** `curl` was utilized to retrieve specific web resources, such as JavaScript libraries, from the target web server to ascertain their versions and identify potential client-side vulnerabilities.

Observations and Additional Recommendations

This section outlines noteworthy observations made during the penetration test that, while not representing direct vulnerabilities, warrant attention for enhancing the overall security posture. Additionally, it provides supplementary recommendations to further strengthen the security of the assessed environment.

Observations

Reliance on IPv6 Addresses

The infrastructure heavily relies on IPv6 addresses for server identification. While IPv6 offers numerous advantages, it can sometimes be overlooked in security configurations and monitoring tools primarily designed for IPv4. This could potentially lead to blind spots in security defenses.

Standard Port Usage

Common ports such as 21 (FTP), 22 (SSH), and 80 (HTTP) are exposed on the tested servers. While standard, their presence can attract automated attacks and increase the attack surface.

Publicly Accessible Test Files

During the assessment, some publicly accessible files with names suggesting testing or development purposes were observed on the web server. These files could potentially contain sensitive information or expose internal application details to malicious actors.

Additional Recommendations**Enhance IPv6 Security Awareness and Tooling**

Ensure that all security monitoring tools, intrusion detection/prevention systems, and firewall rules are configured to adequately support and monitor IPv6 traffic. Conduct regular training for security personnel on IPv6-related security considerations and best practices.

Consider Non-Standard Ports (with Caution)

For services like SSH, consider using non-standard ports. This can help reduce the volume of automated brute-force attacks. However, this should be implemented in conjunction with strong authentication mechanisms and not as a sole security measure. Ensure that any non-standard ports are properly documented and secured by firewall rules.

Implement Regular Security Audits of Web Content

Establish a process for regularly auditing publicly accessible web content to identify and remove any unnecessary or potentially sensitive files, such as test pages, backups, or development artifacts. Implement strict controls over the deployment of new content to prevent the accidental exposure of such files.

Strengthen Logging and Monitoring

Implement comprehensive logging for all critical systems and services, including web servers, FTP servers, and SSH. Ensure that these logs are centrally collected, securely stored, and actively monitored for suspicious activity. Implement alerting mechanisms for security-relevant events.

Adopt a “Principle of Least Privilege” More Granularly

While the report highlights specific instances of excessive privileges, reinforce the principle of least privilege across all user accounts, service accounts, and system processes. Regularly review and refine permissions to ensure that entities only have the necessary access to perform their intended functions.

Implement Security Headers

For web applications, implement security-related HTTP headers such as Content Security Policy (CSP), HTTP Strict Transport Security (HSTS), X-Content-Type-Options, and X-Frame-Options. These headers can provide an additional layer of defense against various web-based attacks.

Conduct Regular Vulnerability Scanning

Implement a schedule for regular automated vulnerability scanning of both the application and the underlying infrastructure. Ensure that identified vulnerabilities are promptly assessed and remediated based on their severity.

Enhance Password Complexity Requirements

Beyond the immediate recommendations for the service account, review and strengthen the overall password complexity requirements for all user accounts across the infrastructure. Enforce minimum length, character diversity, and prevent the use of common or easily guessable passwords. Consider implementing multi-factor authentication (MFA) where feasible.

Implement Network Segmentation

Consider implementing network segmentation to isolate critical systems and applications. This can limit the potential impact of a security breach by preventing lateral movement to other sensitive parts of the network.

Appendix

Appendix A - Log Application

app.py:

```
from flask import Flask, request, jsonify, redirect, url_for
from joblib import Parallel, delayed
import time, csv

app = Flask(__name__)

LOG_FILE = "/opt/logger/log.txt"
PRIVS_FILE = "/opt/logservice/privs.txt"

@app.route("/")
def index():
    return redirect("/health")

@app.route("/health")
def health_check():
    """Endpoint to check the health of the application."""
    if request.args.get("verbose") == "True":
        return jsonify({"status": "ok"}), 200
    else:
        return "ok", 200

@app.route("/privs")
def privs():
    """Endpoint to check the source's logging privileges."""
    privs_str = ""

    match get_privs(request.remote_addr):
        case "r":
            privs_str = "You can read logs."
        case "w":
            privs_str = "You can write logs."
        case "rw":
            privs_str = "You can read and write logs."
        case _:
            privs_str = "You do not have any logging privileges"

    return privs_str, 200

@app.route("/log", methods=["POST"])
def log():
    """Endpoint to receive logs from remote systems."""
    privs = get_privs(request.remote_addr)
    if not "w" in get_privs(request.remote_addr):
        return "Permissions denied.", 403
```

```
try:
    form = request.form.copy()
    entry = form.pop("entry")
    write_log_to_file(request.remote_addr, entry)
    return "Entry logged.", 200
except:
    return "Unexpected error.", 500

@app.route("/logs")
def logs():
    """Endpoint to retrieve logs."""
    if not "r" in get_privs(request.remote_addr):
        return "Permissions denied.", 403

    try:
        args = request.args.copy()
        system_ip = args.pop("ip", request.remote_addr)
        logs = read_logs_from_file()
        # Parallel processing of log entries using joblib. Processing the
        # last 10 lines.
        results = Parallel(n_jobs=2, **args)(delayed(process_log_entry)(
            entry, system_ip) for entry in logs[-10:])
        results = [item for item in results if item is not None]
        return jsonify({"logs": results}), 200
    except:
        return "Unexpected error while retrieving logs.", 500

def get_privs(ip):
    """Get privileges from file."""
    privs = ""

    with open(PRIVS_FILE) as f:
        reader = csv.reader(f, delimiter=",")
        for row in reader:
            if row[0] == ip:
                privs = row[1]
                break

    return privs

def write_log_to_file(system_ip, log_data):
    """Writes a log entry to a file."""
    try:
        with open(LOG_FILE, "a") as f:
            f.write(f"{system_ip} - [{time.ctime()}] {log_data}\n")
    except FileNotFoundError:
        return []

def process_log_entry(entry, system_ip):
    """Process a log entry."""
```

```
    if entry.startswith(system_ip):
        return entry
    else:
        return None

def read_logs_from_file():
    """Reads log entries from a file."""
    try:
        with open(LOG_FILE, "r") as f:
            return f.readlines()
    except FileNotFoundError:
        return []
```

Appendix B - SSH Conf

sa.conf:

```
Match user sa
    ForceCommand /usr/bin/python3 /opt/jail.py
    PasswordAuthentication yes
    DisableForwarding yes
```