Logotipo

Descripción generada automáticamente

CVE-2021-42013

Security Assessment  
Findings Report

Business Confidential

* Date of the project: 2024-12-16
* Project name: Demo for Master Thesis (TFM)

# Table of Contents

[Table of Contents 3](#_Toc143269467)

[Confidentiality Statement 4](#_Toc143269468)

[Disclaimer 4](#_Toc143269469)

[Contact Information 4](#_Toc143269470)

[Assesment overview 5](#_Toc143269471)

[Assessment Components 5](#_Toc143269472)

[Internal Penetration Test 5](#_Toc143269473)

[Finding severity ratings 6](#_Toc143269474)

[Risk factors 6](#_Toc143269475)

[Likelihood 6](#_Toc143269476)

[Impact 6](#_Toc143269477)

[Scope 7](#_Toc143269478)

[Hosts analyzed 7](#_Toc143269479)

[Scope Exclusions 7](#_Toc143269480)

[Client Allowances 7](#_Toc143269481)

[Executive summary 8](#_Toc143269482)

[Scoping and time limitations 8](#_Toc143269483)

[Testing summary 8](#_Toc143269484)

[Tester Notes and Recommendations 8](#_Toc143269485)

[Vulnerability summary & report card 9](#_Toc143269486)

[Internal Penetration Test Findings 9](#_Toc143269488)

# Confidentiality Statement

This document is the exclusive property of CVE-2021-42013 and C15 (C15).

This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires consent of both CVE-2021-42013 and C15).

CVE-2021-42013 may share this document with auditors under non-disclosure agreements to demonstrate penetration test requirement compliance.

# Disclaimer

A penetration test is considered a snapshot in time. The findings and recommendations reflect the information gathered during the assessment and not any changes or modifications made outside of that period.

Time-limited engagements do not allow for a full evaluation of all security controls. C15 prioritized the assessment to identify the weakest security controls an attacker would exploit. C15 recommends conducting similar assessments on an annual basis by internal or third-party assessors to ensure the continued success of the controls.

# Contact Information

|  |  |  |
| --- | --- | --- |
| **Name** | **Title** | **Contact Information** |
| **CVE-2021-42013** | | |
| **UIT** | Student | Email: |
| **C15** | | |
| **C15** | Penetration Tester | Email: 22520645@gm.uit.edu.vn |

# Assesment overview

From to 2024-12-16, CVE-2021-42013 engaged C15, from now on, C15, to evaluate the security posture of its infrastructure compared to current industry best practices that included an internal network penetration test. All testing performed is based on the NIST SP 800-115 Technical Guide to Information Security Testing and Assessment, OWASP Testing Guide (v4), and customized testing frameworks.

Phases of penetration testing activities include the following:

* **Planning** – Customer goals are gathered and rules of engagement obtained.
* **Discovery** – Perform **scanning** and **enumeration** to identify potential vulnerabilities, weak areas, and exploits.
* **Attack** – Confirm potential vulnerabilities through exploitation and perform additional discovery upon new access.
* **Reporting** – **Document** all found vulnerabilities and exploits, failed attempts, and company strengths and weaknesses.

# Assessment Components

## Internal Penetration Test

An internal penetration test emulates the role of an attacker from inside the network. An engineer will scan the network to identify potential host vulnerabilities and perform common and advanced internal network attacks, such as: LLMNR/NBT-NS poisoning and other man- in-the-middle attacks, token impersonation, kerberoasting, pass-the-hash, golden ticket, and more. The engineer will seek to gain access to hosts through lateral movement, compromise domain user and admin accounts, and exfiltrate sensitive data.

# Finding severity ratings

The following table defines levels of severity and corresponding CVSS score range that are used throughout the document to assess vulnerability and risk impact.

|  |  |  |
| --- | --- | --- |
| **Severity** | **CVSS V3**  **Score Range** | **Definition** |
| **Critical** | 9.0–10.0 | Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately. |
| **High** | 7.0 – 8.9 | Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible. |
| **Moderate** | 4.0 – 6.9 | Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved. |
| **Low** | 0.1 – 3.9 | Vulnerabilities are non-exploitable but would reduce an organization’s attack surface. It is advised to form a plan of action and patch during the next maintenance window. |
| **Informational** | N/A (Informational data) | No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation. |

# Risk factors

Risk is measured by two factors: Likelihood and impact.

## Likelihood

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, attacker skill level, and client environment.

## Impact

Impact measures the potential vulnerability’s effect on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.

# Scope

## Hosts analyzed

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Host | Information about the host | | | |
| Ports | Service | Version | CVEs |
| 10.10.114.139 | 8080 | http | 2.4.49 | CVE-2024-38476 CVE-2024-38474 CVE-2023-25690 CVE-2022-31813 CVE-2022-23943 CVE-2022-22720 CVE-2021-44790 CVE-2021-42013 CVE-2024-38475 CVE-2022-28615 CVE-2022-22721 CVE-2022-36760 CVE-2021-44224 CVE-2024-38473 CVE-2024-40898 CVE-2024-39573 CVE-2024-38477 CVE-2024-38472 CVE-2024-27316 CVE-2023-31122 CVE-2023-27522 CVE-2022-30556 CVE-2022-30522 CVE-2022-29404 CVE-2022-26377 CVE-2022-22719 CVE-2021-41773 CVE-2021-41524 CVE-2006-20001 CVE-2023-38709 CVE-2024-24795 CVE-2024-39884 CVE-2023-45802 CVE-2022-37436 CVE-2022-28614 CVE-2022-28330 |

## Scope Exclusions

Per client request, C15 did not perform any of the following attacks during testing:

* Denial of service (DoS)
* Phishing/Social Engineering

All other attacks not specified above were permitted by CVE-2021-42013.

## Client Allowances

CVE-2021-42013 provided C15 the following allowances:

* Internal access to network via TBD.

# Executive summary

C15 evaluated CVE-2021-42013’s internal security posture through penetration testing from to 2024-12-16. The following sections provide a high-level overview of vulnerabilities discovered, successful and unsuccessful attempts, and strengths and weaknesses.

Executive Summary:  
  
The penetration testing on the host with IP address 10.10.114.139 revealed critical vulnerabilities in the Apache HTTP Server version 2.4.49. The following CVEs were identified with their respective descriptions and severity levels:  
  
1. CVE-2024-38476 (Critical): Vulnerability in the core of Apache HTTP Server 2.4.59 and earlier allows for information disclosure, SSRF, or local script execution via backend applications with malicious response headers. Users are recommended to upgrade to version 2.4.60 to fix this issue.  
  
2. CVE-2024-38474 (Critical): Substitution encoding issue in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier allows attackers to execute scripts in directories permitted by the configuration but not directly reachable by any URL. Users are recommended to upgrade to version 2.4.60 to address this vulnerability.  
  
3. CVE-2023-25690 (Critical): Some mod\_proxy configurations on Apache HTTP Server versions 2.4.0 through 2.4.55 allow for a HTTP Request Smuggling attack. Users are recommended to update to at least version 2.4.56 to mitigate this risk.  
  
4. CVE-2022-31813 (Critical): Apache HTTP Server 2.4.53 and earlier may not send the X-Forwarded-\* headers to the origin server based on the client-side Connection header hop-by-hop mechanism, potentially bypassing IP-based authentication on the origin server/application.  
  
5. CVE-2022-22720 (Critical): Apache HTTP Server 2.4.52 and earlier fails to close inbound connections when errors are encountered, exposing the server to HTTP Request Smuggling.  
  
6. CVE-2021-44790 (Critical): A carefully crafted request body can cause a buffer overflow in the mod\_lua multipart parser, potentially leading to code execution. Users are advised to update to the latest version to address this vulnerability.  
  
7. CVE-2021-42013 (Critical): The fix for CVE-2021-41773 in Apache HTTP Server 2.4.50 was found to be insufficient, allowing for a path traversal attack to map URLs to files outside the configured directories. This could lead to remote code execution.  
  
In conclusion, the Apache HTTP Server on the host 10.10.114.139 is vulnerable to critical security issues that could result in information disclosure, remote code execution, and HTTP Request Smuggling attacks. It is imperative for the organization to upgrade to the recommended versions and apply necessary patches to mitigate these vulnerabilities effectively. Regular security updates and monitoring are essential to maintain a secure web server environment.

## Scoping and time limitations

Scoping during the engagement did not permit denial of service or social engineering across all testing components.

Time limitations were in place for testing. Internal network penetration testing was permitted for days.

## Testing summary

The network assessment evaluated CVE-2021-42013’s internal security posture. From an internal perspective, the C15 performed vulnerability scanning against the IP addresses provided by CVE-2021-42013 to evaluate the overall patching health of the network.

TBD

## Tester Notes and Recommendations

TBD overall security (good/regular/bad).

TBD constants that stood out in the process

TBD reccomendations.

We recommend that the CVE-2021-42013 team reviews the patching recommendations made in the TBD section of the report along with reviewing the provided scans for a full overview of the items to be patched. We also recommend that Demo corp improve their patch management policies and procedures to help prevent potential attacks within their network.

TBD alerts triggered.

Overall, the CVE-2021-42013 network performed as expected for the penetration test. We recommend that the CVE-2021-42013 team thoroughly review the recommendations made in this report, patch the findings, and re-test annually to improve their overall security posture.

# Vulnerability summary & report card

The following tables illustrate the vulnerabilities found by impact and recommended remediations:

## Internal Penetration Test Findings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | 18 | 6 |  | 0 |
| **Critical** | **High** | **Moderate** | **Low** | **Informational** |

### Finding 1

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-38476**
* Severity score: **9.8** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Apache HTTP Server Outdated Version (Critical)  
  
#### Severity Rationale:  
The vulnerability in the core of Apache HTTP Server 2.4.59 and earlier versions is classified as critical due to the potential for information disclosure, Server-Side Request Forgery (SSRF), or local script execution through backend applications with malicious or exploitable response headers.   
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could allow attackers to gain unauthorized access to sensitive information, manipulate server-side requests, or execute malicious scripts on the server. Exploiting this vulnerability could lead to severe consequences for the organization's confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Information Disclosure: Attackers could access sensitive information stored on the server, including user credentials, financial data, or proprietary information.  
  
2. Server-Side Request Forgery (SSRF): Attackers could manipulate server-side requests to access internal resources, pivot to other systems, or perform further attacks within the network.  
  
3. Local Script Execution: Attackers could execute malicious scripts on the server, leading to unauthorized access, data manipulation, or disruption of services.  
  
4. Regulatory Compliance Violations: Failure to address this vulnerability could result in non-compliance with industry regulations and data protection laws, leading to legal consequences and financial penalties.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the vulnerability in the Apache HTTP Server core, attackers could potentially gain unauthorized access to sensitive information, manipulate server-side requests, or execute malicious scripts on the server. This could result in data breaches, unauthorized access, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to upgrade to version 2.4.60 to mitigate the risk associated with this critical vulnerability.

#### Public exploits related to this finding

## Description of the finding

Vulnerability in core of Apache HTTP Server 2.4.59 and earlier are vulnerably to information disclosure, SSRF or local script execution via backend applications whose response headers are malicious or exploitable.  
  
Users are recommended to upgrade to version 2.4.60, which fixes this issue.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.59 and earlier (CVE-2024-38476) that allows for information disclosure, SSRF, or local script execution, users are recommended to upgrade to version 2.4.60.   
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.60.  
2. Ensure that all backend applications are secure and do not have malicious or exploitable response headers.  
3. Regularly monitor and audit the server for any suspicious activity or unauthorized access attempts.

### Finding 2

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-38474**
* Severity score: **9.8** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Substitution Encoding Issue in mod\_rewrite (Critical)  
  
#### Severity Rationale:  
The vulnerability in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier versions, which allows attackers to execute scripts in directories permitted by the configuration but not directly reachable by any URL or disclose scripts meant to only be executed as CGI, is classified as critical.   
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could allow attackers to execute arbitrary scripts, disclose sensitive information, or bypass security controls. Exploiting this vulnerability could lead to severe consequences for the organization's confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Script Execution: Attackers could execute arbitrary scripts in directories permitted by the configuration, leading to unauthorized access, data manipulation, or disruption of services.  
  
2. Information Disclosure: Attackers could disclose scripts meant to be executed as CGI, exposing sensitive information or proprietary code to unauthorized parties.  
  
3. Bypassing Security Controls: Attackers could bypass security controls and gain unauthorized access to restricted directories or resources on the server.  
  
4. Regulatory Compliance Violations: Failure to address this vulnerability could result in non-compliance with industry regulations and data protection laws, leading to legal consequences and financial penalties.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the substitution encoding issue in mod\_rewrite, attackers could potentially execute arbitrary scripts, disclose sensitive information, or bypass security controls on the server. This could result in unauthorized access, data breaches, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to upgrade to version 2.4.60 and use the "UnsafeAllow3F" rewrite flag to mitigate the risk associated with this critical vulnerability.

#### Public exploits related to this finding

## Description of the finding

Substitution encoding issue in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier allows attacker to execute scripts in  
directories permitted by the configuration but not directly reachable by any URL or source disclosure of scripts meant to only to be executed as CGI.  
  
Users are recommended to upgrade to version 2.4.60, which fixes this issue.  
  
Some RewriteRules that capture and substitute unsafely will now fail unless rewrite flag "UnsafeAllow3F" is specified.

## Mitigation steps

To mitigate the substitution encoding issue in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier (CVE-2024-38474) that allows an attacker to execute scripts in directories permitted by the configuration but not directly reachable by any URL, users are recommended to upgrade to version 2.4.60.  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.60.  
2. Review and update RewriteRules to ensure they do not capture and substitute unsafely.  
3. If RewriteRules capture and substitute unsafely, add the rewrite flag "UnsafeAllow3F" to allow them to continue functioning as intended.  
4. Regularly review and monitor the mod\_rewrite configuration for any suspicious or unsafe rules that could be exploited by attackers.

### Finding 3

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2023-25690**
* Severity score: **9.8** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: HTTP Request Smuggling Attack in mod\_proxy (Critical)  
  
#### Severity Rationale:  
The vulnerability in some mod\_proxy configurations on Apache HTTP Server versions 2.4.0 through 2.4.55, which allows for HTTP Request Smuggling attacks, is classified as critical.   
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to request splitting/smuggling, bypass of access controls, proxying unintended URLs to origin servers, and cache poisoning. Exploiting this vulnerability could have severe consequences for the organization's confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Bypass of Access Controls: Attackers could bypass access controls in the proxy server, gaining unauthorized access to sensitive resources or data.  
  
2. Proxying Unintended URLs: Attackers could proxy unintended URLs to existing origin servers, potentially exposing sensitive information or executing malicious actions.  
  
3. Cache Poisoning: Attackers could manipulate the cache content, leading to the serving of malicious content to users or disruption of services.  
  
4. Data Integrity Compromise: The integrity of data transmitted through the proxy server could be compromised, leading to unauthorized modifications or disclosure of sensitive information.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the HTTP Request Smuggling attack in mod\_proxy configurations, attackers could potentially bypass access controls, proxy unintended URLs, manipulate cache content, and compromise data integrity on the server. This could result in unauthorized access, data breaches, service disruptions, reputational damage, and legal consequences for the organization. It is crucial for users to update to at least version 2.4.56 of Apache HTTP Server to mitigate the risk associated with this critical vulnerability.

#### Public exploits related to this finding

## Description of the finding

Some mod\_proxy configurations on Apache HTTP Server versions 2.4.0 through 2.4.55 allow a HTTP Request Smuggling attack.  
  
  
  
  
Configurations are affected when mod\_proxy is enabled along with some form of RewriteRule  
 or ProxyPassMatch in which a non-specific pattern matches  
 some portion of the user-supplied request-target (URL) data and is then  
 re-inserted into the proxied request-target using variable   
substitution. For example, something like:  
  
  
  
  
RewriteEngine on  
RewriteRule "^/here/(.\*)" "http://example.com:8080/elsewhere?$1"; [P]  
ProxyPassReverse /here/ http://example.com:8080/  
  
  
Request splitting/smuggling could result in bypass of access controls in the proxy server, proxying unintended URLs to existing origin servers, and cache poisoning. Users are recommended to update to at least version 2.4.56 of Apache HTTP Server.

## Mitigation steps

To mitigate the HTTP Request Smuggling attack vulnerability in mod\_proxy configurations on Apache HTTP Server versions 2.4.0 through 2.4.55 (CVE-2023-25690), users are recommended to update to at least version 2.4.56 of Apache HTTP Server.  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.56 or later.  
2. Review and modify mod\_proxy configurations to ensure that RewriteRule or ProxyPassMatch patterns do not match user-supplied request-target data in a non-specific manner.  
3. Avoid re-inserting user-supplied request-target data into the proxied request-target using variable substitution.  
4. Regularly monitor and audit the mod\_proxy configurations for any potential vulnerabilities that could lead to HTTP Request Smuggling attacks.

### Finding 4

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-31813**
* Severity score: **9.8** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: X-Forwarded-\* Headers Issue in Apache HTTP Server (Critical)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server 2.4.53 and earlier versions, which may not send the X-Forwarded-\* headers to the origin server based on the client-side Connection header hop-by-hop mechanism, is classified as critical.  
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could be exploited to bypass IP-based authentication on the origin server or application. Without the X-Forwarded-\* headers being properly forwarded, the origin server may not be able to accurately identify the client's IP address or other relevant information, potentially leading to unauthorized access or misuse of resources.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Bypass of IP-based Authentication: Attackers could exploit this issue to bypass IP-based authentication mechanisms on the origin server or application, gaining unauthorized access to sensitive resources.  
  
2. Misuse of Resources: Without the accurate forwarding of X-Forwarded-\* headers, the origin server may not be able to apply proper access controls or logging, leading to potential misuse of resources or data.  
  
3. Security Misconfigurations: Failure to send X-Forwarded-\* headers could result in security misconfigurations that expose the organization to additional risks and vulnerabilities.  
  
4. Regulatory Compliance Violations: Non-compliance with security best practices for header forwarding could lead to regulatory violations and legal consequences.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the X-Forwarded-\* headers issue in Apache HTTP Server, attackers could potentially bypass IP-based authentication mechanisms, gain unauthorized access to sensitive resources, and misuse the organization's resources. This could result in unauthorized access, data breaches, security misconfigurations, reputational damage, and legal consequences for the organization. It is crucial for users to address this vulnerability promptly to mitigate the risk associated with this critical issue.

#### Public exploits related to this finding

## Description of the finding

Apache HTTP Server 2.4.53 and earlier may not send the X-Forwarded-\* headers to the origin server based on client side Connection header hop-by-hop mechanism. This may be used to bypass IP based authentication on the origin server/application.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.53 and earlier (CVE-2022-31813) that may not send the X-Forwarded-\* headers to the origin server based on the client-side Connection header hop-by-hop mechanism, users can take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this issue.  
2. Configure the server to explicitly forward the X-Forwarded-\* headers to the origin server, regardless of the client-side Connection header settings.  
3. Implement additional authentication mechanisms on the origin server/application to mitigate the risk of bypassing IP-based authentication.  
4. Regularly monitor and review the server configuration to ensure that X-Forwarded-\* headers are being forwarded correctly to prevent potential bypasses of IP-based authentication.

### Finding 5

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-23943**
* Severity score: **None** (**None** severity)

## Finding severity rationale

The vulnerability does not have a severity attached to.

#### Public exploits related to this finding

## Description of the finding

None

## Mitigation steps

To provide accurate mitigation steps for the CVE-2022-23943 vulnerability, I would need more specific information about the vulnerability. The Common Vulnerability Scoring System (CVSS) score of 9.8 indicates a critical severity level, but additional details about the nature of the vulnerability are necessary to recommend appropriate mitigation measures. Please provide a description or additional context for CVE-2022-23943 so that I can offer tailored mitigation steps.

### Finding 6

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-22720**
* Severity score: **9.8** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Incomplete Connection Closure in Apache HTTP Server (Critical)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server 2.4.52 and earlier versions, which fails to close inbound connections when errors are encountered while discarding the request body, exposing the server to HTTP Request Smuggling, is classified as critical.  
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could be exploited to conduct HTTP Request Smuggling attacks. Incomplete closure of inbound connections can lead to request manipulation, cache poisoning, and unauthorized access to sensitive resources, potentially compromising the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. HTTP Request Smuggling: Attackers could manipulate requests to bypass security controls, access unauthorized resources, or poison caches, leading to data breaches or service disruptions.  
  
2. Unauthorized Access: Incomplete closure of connections could allow attackers to maintain persistent connections and exploit vulnerabilities to gain unauthorized access to the server or sensitive data.  
  
3. Data Integrity Compromise: Manipulation of inbound connections could compromise the integrity of data transmitted through the server, leading to unauthorized modifications or disclosure of sensitive information.  
  
4. Regulatory Compliance Violations: Failure to address this vulnerability could result in non-compliance with industry regulations and data protection laws, leading to legal consequences and financial penalties.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the incomplete connection closure vulnerability in Apache HTTP Server, attackers could potentially conduct HTTP Request Smuggling attacks, gain unauthorized access to sensitive resources, compromise data integrity, and violate regulatory compliance requirements. This could result in data breaches, service disruptions, reputational damage, and legal consequences for the organization. It is crucial for users to address this critical vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

Apache HTTP Server 2.4.52 and earlier fails to close inbound connection when errors are encountered discarding the request body, exposing the server to HTTP Request Smuggling

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.52 and earlier (CVE-2022-22720) that fails to close inbound connections when errors are encountered, potentially exposing the server to HTTP Request Smuggling, users can take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this issue.  
2. Configure the server to properly close inbound connections when errors occur, especially when discarding the request body.  
3. Implement additional security measures to detect and prevent HTTP Request Smuggling attacks, such as using a Web Application Firewall (WAF) or security proxy.  
4. Regularly monitor server logs and network traffic for any signs of abnormal or suspicious activity that could indicate an ongoing HTTP Request Smuggling attack.

### Finding 7

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2021-44790**
* Severity score: **9.8** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Buffer Overflow in mod\_lua Multipart Parser (Critical)  
  
#### Severity Rationale:  
The vulnerability in the mod\_lua multipart parser in Apache HTTP Server 2.4.51 and earlier versions, which can be exploited by a carefully crafted request body to cause a buffer overflow (specifically in r:parsebody() called from Lua scripts), is classified as critical.  
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to a buffer overflow condition. Exploiting this vulnerability could potentially allow attackers to execute arbitrary code, crash the server, or gain unauthorized access to sensitive information, compromising the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Arbitrary Code Execution: Attackers could exploit the buffer overflow to execute arbitrary code within the context of the server, leading to unauthorized access, data manipulation, or disruption of services.  
  
2. Server Crash: A successful exploitation of the buffer overflow could crash the server, resulting in downtime, service disruptions, and potential data loss.  
  
3. Unauthorized Access: Attackers could leverage the buffer overflow to gain unauthorized access to sensitive information or resources on the server, compromising data confidentiality.  
  
4. Regulatory Compliance Violations: Failure to address this vulnerability could result in non-compliance with industry regulations and data protection laws, leading to legal consequences and financial penalties.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the buffer overflow in the mod\_lua multipart parser, attackers could potentially execute arbitrary code, crash the server, gain unauthorized access to sensitive information, and violate regulatory compliance requirements. This could result in data breaches, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this critical vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

#### Exploit ID 51193.

* Description: Apache 2.4.x - Buffer Overflow
* Download URL: https://www.exploit-db.com/exploits/51193

## Description of the finding

A carefully crafted request body can cause a buffer overflow in the mod\_lua multipart parser (r:parsebody() called from Lua scripts). The Apache httpd team is not aware of an exploit for the vulnerabilty though it might be possible to craft one. This issue affects Apache HTTP Server 2.4.51 and earlier.

## Mitigation steps

To mitigate the buffer overflow vulnerability in the mod\_lua multipart parser (CVE-2021-44790) that can be triggered by a carefully crafted request body, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Disable mod\_lua if it is not required for the server's functionality to reduce the attack surface.  
3. Implement input validation mechanisms to sanitize and validate request bodies before processing them, especially in Lua scripts that utilize r:parsebody().  
4. Regularly monitor server logs for any signs of unusual or malicious request patterns that could indicate attempts to exploit the buffer overflow vulnerability.  
5. Consider implementing additional security measures such as WAF rules to detect and block potentially malicious requests that could trigger the buffer overflow.

### Finding 8

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2021-42013**
* Severity score: **9.8** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Path Traversal Attack in Apache HTTP Server (Critical)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.49 and 2.4.50, which allows for a path traversal attack to map URLs to files outside the directories configured by Alias-like directives, is classified as critical.  
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could be exploited to access files outside of the intended directories and potentially lead to remote code execution. Exploiting this vulnerability could result in unauthorized access to sensitive information, data manipulation, or compromise of the server's integrity and availability.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Unauthorized Access: Attackers could use path traversal techniques to access files outside of the configured directories, potentially exposing sensitive data or configuration files.  
  
2. Data Manipulation: By accessing files outside of the intended directories, attackers could manipulate data, inject malicious code, or compromise the integrity of the server.  
  
3. Remote Code Execution: If CGI scripts are enabled for the aliased paths, attackers could exploit the vulnerability to achieve remote code execution, allowing them to execute arbitrary commands on the server.  
  
4. Security Misconfigurations: Failure to properly configure access controls or protect against path traversal attacks could lead to security misconfigurations and unauthorized access to critical resources.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the path traversal attack in Apache HTTP Server versions 2.4.49 and 2.4.50, attackers could potentially access sensitive files, manipulate data, and even achieve remote code execution on the server. This could result in data breaches, unauthorized access, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this critical vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

#### Exploit ID 50406.

* Description: Apache HTTP Server 2.4.50 - Path Traversal &amp; Remote Code Execution (RCE)
* Download URL: https://www.exploit-db.com/exploits/50406

#### Exploit ID 50446.

* Description: Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (2)
* Download URL: https://www.exploit-db.com/exploits/50446

#### Exploit ID 50512.

* Description: Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (3)
* Download URL: https://www.exploit-db.com/exploits/50512

#### Exploit ID 50552.

* Description: Apache HTTP Server 2.4.50 Path Traversal and Code Execution - Paper
* Download URL: https://www.exploit-db.com/exploits/50552

## Description of the finding

It was found that the fix for CVE-2021-41773 in Apache HTTP Server 2.4.50 was insufficient. An attacker could use a path traversal attack to map URLs to files outside the directories configured by Alias-like directives. If files outside of these directories are not protected by the usual default configuration "require all denied", these requests can succeed. If CGI scripts are also enabled for these aliased pathes, this could allow for remote code execution. This issue only affects Apache 2.4.49 and Apache 2.4.50 and not earlier versions.

## Mitigation steps

To mitigate the path traversal and remote code execution vulnerability in Apache HTTP Server 2.4.49 and 2.4.50 (CVE-2021-42013) that allows an attacker to map URLs to files outside configured directories by Alias-like directives, potentially leading to remote code execution, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that includes a comprehensive fix for CVE-2021-42013.  
2. Review and update Alias-like directives to ensure that they do not allow path traversal outside of intended directories.  
3. Implement proper access controls and restrictions, such as "require all denied" for files outside configured directories, to prevent unauthorized access.  
4. Disable CGI scripts for aliased paths to mitigate the risk of remote code execution.  
5. Regularly audit and monitor server configurations and access logs for any suspicious activity that could indicate exploitation attempts of path traversal and remote code execution vulnerabilities.  
6. Consider implementing additional security measures, such as web application firewalls, to detect and block malicious requests targeting path traversal and remote code execution vulnerabilities.

### Finding 9

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-38475**
* Severity score: **9.1** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Improper Escaping of Output in mod\_rewrite (Critical)  
  
#### Severity Rationale:  
The vulnerability in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier versions, which allows an attacker to map URLs to filesystem locations that are permitted to be served by the server but not directly reachable by any URL, resulting in code execution or source code disclosure, is classified as critical.  
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could be exploited to execute arbitrary code or disclose sensitive source code. Improper escaping of output in mod\_rewrite can lead to unauthorized access to server resources, data manipulation, or compromise of the server's integrity and confidentiality.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Code Execution: Attackers could exploit the vulnerability to map URLs to filesystem locations and execute arbitrary code on the server, leading to unauthorized access, data manipulation, or disruption of services.  
  
2. Source Code Disclosure: Improper escaping of output could result in the disclosure of sensitive source code or configuration files that are not intended to be directly reachable by URLs.  
  
3. Unauthorized Access: Attackers could gain unauthorized access to restricted server resources or directories by exploiting the improper escaping of output in mod\_rewrite.  
  
4. Security Misconfigurations: Failure to properly constrain substitutions in server context could lead to security misconfigurations and unauthorized access to critical resources.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the improper escaping of output in mod\_rewrite, attackers could potentially map URLs to filesystem locations, execute arbitrary code, or disclose sensitive source code on the server. This could result in unauthorized access, data breaches, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this critical vulnerability promptly by ensuring proper constraints on substitutions and using the "UnsafePrefixStat" rewrite flag cautiously to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

Improper escaping of output in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier allows an attacker to map URLs to filesystem locations that are permitted to be served by the server but are not intentionally/directly reachable by any URL, resulting in code execution or source code disclosure.   
  
Substitutions in server context that use a backreferences or variables as the first segment of the substitution are affected.  Some unsafe RewiteRules will be broken by this change and the rewrite flag "UnsafePrefixStat" can be used to opt back in once ensuring the substitution is appropriately constrained.

## Mitigation steps

To mitigate the vulnerability in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier (CVE-2024-38475) that allows an attacker to map URLs to filesystem locations not directly reachable by any URL, potentially leading to code execution or source code disclosure, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Review and update RewriteRules to ensure proper escaping of output and prevent mapping URLs to unintended filesystem locations.  
3. Avoid using backreferences or variables as the first segment of substitutions in server context to mitigate the risk of improper escaping.  
4. Some unsafe RewriteRules may be affected by these changes. Ensure that RewriteRules are appropriately constrained and consider using the rewrite flag "UnsafePrefixStat" to opt back in after confirming the substitution is secure.  
5. Regularly review and monitor the mod\_rewrite configuration for any potentially unsafe rules that could be exploited by attackers to map URLs to unauthorized filesystem locations.

### Finding 10

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-28615**
* Severity score: **9.1** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Read Beyond Bounds in ap\_strcmp\_match() in Apache HTTP Server (Critical)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.53 and earlier, which may crash or disclose information due to a read beyond bounds in ap\_strcmp\_match() when provided with an extremely large input buffer, is classified as critical.  
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to server crashes or information disclosure. While the server itself cannot be coerced into such a call, third-party modules or Lua scripts that utilize ap\_strcmp\_match() may be hypothetically affected, potentially compromising the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Server Crashes: Exploiting the read beyond bounds in ap\_strcmp\_match() could lead to server crashes, resulting in downtime, service disruptions, and potential data loss.  
  
2. Information Disclosure: Attackers could potentially disclose sensitive information due to the read beyond bounds, compromising data confidentiality and exposing critical data to unauthorized parties.  
  
3. Third-Party Module Vulnerabilities: Third-party modules or Lua scripts that utilize ap\_strcmp\_match() may be affected, leading to additional security risks and potential exploitation of the vulnerability.  
  
4. Operational Disruptions: Server crashes or information disclosure could disrupt operations, impact service availability, and lead to financial losses for the organization.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the read beyond bounds in ap\_strcmp\_match() in Apache HTTP Server, attackers could potentially cause server crashes, disclose sensitive information, or exploit vulnerabilities in third-party modules or Lua scripts. This could result in operational disruptions, data breaches, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this critical vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

Apache HTTP Server 2.4.53 and earlier may crash or disclose information due to a read beyond bounds in ap\_strcmp\_match() when provided with an extremely large input buffer. While no code distributed with the server can be coerced into such a call, third-party modules or lua scripts that use ap\_strcmp\_match() may hypothetically be affected.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.53 and earlier (CVE-2022-28615) that may lead to a crash or information disclosure due to a read beyond bounds in ap\_strcmp\_match() when provided with an extremely large input buffer, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Monitor and review third-party modules or Lua scripts that utilize ap\_strcmp\_match() for potential impacts of the vulnerability.  
3. Implement input validation mechanisms to limit the size of input buffers passed to ap\_strcmp\_match() to prevent read beyond bounds.  
4. Regularly review server logs and monitor for any signs of crashes or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider restricting access to sensitive information or critical server functions that could be exposed in the event of a crash or information disclosure.

### Finding 11

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-22721**
* Severity score: **9.1** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Integer Overflow in LimitXMLRequestBody in Apache HTTP Server (Critical)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.52 and earlier, where an integer overflow occurs if LimitXMLRequestBody is set to allow request bodies larger than 350MB on 32-bit systems, leading to out-of-bounds writes, is classified as critical.  
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could result in out-of-bounds writes and potentially lead to server crashes, data corruption, or unauthorized access to sensitive information. Exploiting this vulnerability could compromise the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Server Crashes: The integer overflow leading to out-of-bounds writes could cause server crashes, resulting in downtime, service disruptions, and potential data loss.  
  
2. Data Corruption: Out-of-bounds writes could corrupt data stored in memory or on disk, leading to data integrity issues and potential loss of critical information.  
  
3. Unauthorized Access: Exploiting the vulnerability could potentially allow attackers to gain unauthorized access to sensitive information or resources on the server.  
  
4. Operational Disruptions: Server crashes, data corruption, or unauthorized access could disrupt operations, impact service availability, and lead to financial losses for the organization.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the integer overflow in LimitXMLRequestBody in Apache HTTP Server, attackers could potentially cause server crashes, corrupt data, or gain unauthorized access to sensitive information. This could result in operational disruptions, data breaches, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this critical vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

If LimitXMLRequestBody is set to allow request bodies larger than 350MB (defaults to 1M) on 32 bit systems an integer overflow happens which later causes out of bounds writes. This issue affects Apache HTTP Server 2.4.52 and earlier.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.52 and earlier (CVE-2022-22721) that results in an integer overflow and out-of-bounds writes when LimitXMLRequestBody is set to allow request bodies larger than 350MB on 32-bit systems, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Review and adjust the LimitXMLRequestBody directive to ensure that it does not allow request bodies larger than the system can handle without triggering an integer overflow.  
3. Consider setting reasonable limits on request body sizes to prevent potential integer overflow issues.  
4. Regularly monitor server logs and performance metrics to detect any unusual behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Implement additional security measures, such as request body size limits and input validation, to prevent potential out-of-bounds writes and protect the server from malicious requests.

### Finding 12

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-36760**
* Severity score: **9.0** (**CRITICAL** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: HTTP Request Smuggling in mod\_proxy\_ajp (Critical)  
  
#### Severity Rationale:  
The vulnerability in mod\_proxy\_ajp of Apache HTTP Server versions 2.4.54 and prior, which allows for inconsistent interpretation of HTTP requests (HTTP Request Smuggling), enabling an attacker to smuggle requests to the AJP server it forwards requests to, is classified as critical.  
  
The critical severity level indicates that this vulnerability poses a significant risk to the organization's security as it could be exploited to manipulate or smuggle requests, potentially leading to unauthorized access, data manipulation, or service disruptions. Exploiting this vulnerability could compromise the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Request Manipulation: Attackers could manipulate HTTP requests to bypass security controls, access unauthorized resources, or perform malicious actions on the server.  
  
2. Unauthorized Access: Exploiting the vulnerability could lead to unauthorized access to sensitive information or resources on the AJP server, compromising data confidentiality.  
  
3. Data Manipulation: Attackers could manipulate requests to modify data, inject malicious content, or disrupt services, leading to data integrity issues.  
  
4. Service Disruptions: Inconsistent interpretation of HTTP requests could result in service disruptions, downtime, and impact on the availability of the server.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the HTTP Request Smuggling vulnerability in mod\_proxy\_ajp of Apache HTTP Server, attackers could potentially manipulate requests, gain unauthorized access to sensitive resources, manipulate data, or disrupt services. This could result in data breaches, unauthorized access, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this critical vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling') vulnerability in mod\_proxy\_ajp of Apache HTTP Server allows an attacker to smuggle requests to the AJP server it forwards requests to. This issue affects Apache HTTP Server Apache HTTP Server 2.4 version 2.4.54 and prior versions.

## Mitigation steps

To mitigate the 'HTTP Request Smuggling' vulnerability in mod\_proxy\_ajp of Apache HTTP Server versions 2.4.54 and prior (CVE-2022-36760) that allows an attacker to smuggle requests to the AJP server it forwards requests to, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Configure mod\_proxy\_ajp to ensure consistent interpretation of HTTP requests and prevent request smuggling attacks.  
3. Implement request normalization techniques to detect and block malicious requests that could be used for smuggling attacks.  
4. Regularly monitor server logs and network traffic for any signs of request smuggling attempts or unusual request patterns.  
5. Consider implementing additional security measures, such as using a Web Application Firewall (WAF) or security proxy, to detect and mitigate HTTP Request Smuggling attacks targeting mod\_proxy\_ajp.

### Finding 13

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2021-44224**
* Severity score: **8.2** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Server Side Request Forgery (SSRF) in Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.7 up to 2.4.51, where a crafted URI sent to httpd configured as a forward proxy with ProxyRequests on can cause a crash (NULL pointer dereference) or, for configurations mixing forward and reverse proxy declarations, can allow requests to be directed to a declared Unix Domain Socket endpoint, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to server crashes, Server Side Request Forgery (SSRF) attacks, or unauthorized access to Unix Domain Socket endpoints. Exploiting this vulnerability could compromise the availability, integrity, and confidentiality of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Server Crashes: A crafted URI could cause a crash (NULL pointer dereference) in the Apache HTTP Server, leading to downtime, service disruptions, and potential data loss.  
  
2. Server Side Request Forgery (SSRF): Attackers could manipulate requests to direct traffic to declared Unix Domain Socket endpoints, potentially accessing sensitive resources or services on the server.  
  
3. Unauthorized Access: Exploiting the vulnerability could allow attackers to bypass security controls and access Unix Domain Socket endpoints, leading to unauthorized access to critical resources.  
  
4. Data Exposure: SSRF attacks could result in the exposure of sensitive data or services that should not be directly accessible from external requests.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the Server Side Request Forgery (SSRF) vulnerability in Apache HTTP Server, attackers could potentially cause server crashes, direct requests to unauthorized Unix Domain Socket endpoints, access sensitive resources, or expose confidential data. This could result in service disruptions, unauthorized access, data breaches, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

A crafted URI sent to httpd configured as a forward proxy (ProxyRequests on) can cause a crash (NULL pointer dereference) or, for configurations mixing forward and reverse proxy declarations, can allow for requests to be directed to a declared Unix Domain Socket endpoint (Server Side Request Forgery). This issue affects Apache HTTP Server 2.4.7 up to 2.4.51 (included).

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server versions 2.4.7 up to 2.4.51 (CVE-2021-44224) that allows for a crash (NULL pointer dereference) or Server Side Request Forgery (SSRF) when a crafted URI is sent to httpd configured as a forward proxy (ProxyRequests on), users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Disable the ProxyRequests directive if it is not required for the server's functionality to reduce the attack surface.  
3. Implement input validation mechanisms to sanitize and validate URIs before processing them in the forward proxy configuration.  
4. Regularly monitor server logs and performance metrics for any signs of crashes or unusual behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Review and adjust the server configuration to prevent mixing forward and reverse proxy declarations that could lead to SSRF vulnerabilities.

### Finding 14

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-38473**
* Severity score: **8.1** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Encoding Problem in mod\_proxy (High)  
  
#### Severity Rationale:  
The vulnerability in mod\_proxy in Apache HTTP Server versions 2.4.59 and earlier, which allows request URLs with incorrect encoding to be sent to backend services, potentially bypassing authentication via crafted requests, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could be exploited to bypass authentication mechanisms and potentially gain unauthorized access to backend services. Exploiting this vulnerability could compromise the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Authentication Bypass: Attackers could craft requests with incorrect encoding to bypass authentication mechanisms and gain unauthorized access to backend services.  
  
2. Unauthorized Access: Exploiting the encoding problem in mod\_proxy could allow attackers to access sensitive data or resources without proper authentication, leading to data breaches or unauthorized actions.  
  
3. Data Exposure: Crafted requests with incorrect encoding could expose sensitive information or functionality in backend services that should be protected.  
  
4. Security Misconfigurations: Failure to handle encoding properly in mod\_proxy could lead to security misconfigurations and vulnerabilities that could be exploited by attackers.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the encoding problem in mod\_proxy in Apache HTTP Server, attackers could potentially bypass authentication mechanisms, gain unauthorized access to backend services, and expose sensitive data. This could result in unauthorized access, data breaches, security misconfigurations, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to upgrade to version 2.4.60 to address this High severity vulnerability promptly and mitigate the risk of exploitation.

#### Public exploits related to this finding

## Description of the finding

Encoding problem in mod\_proxy in Apache HTTP Server 2.4.59 and earlier allows request URLs with incorrect encoding to be sent to backend services, potentially bypassing authentication via crafted requests.  
Users are recommended to upgrade to version 2.4.60, which fixes this issue.

## Mitigation steps

To mitigate the encoding problem in mod\_proxy in Apache HTTP Server 2.4.59 and earlier (CVE-2024-38473) that allows request URLs with incorrect encoding to be sent to backend services, potentially bypassing authentication via crafted requests, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.60 or later to address this vulnerability.  
2. Configure mod\_proxy to enforce proper URL encoding standards to prevent the bypassing of authentication mechanisms.  
3. Implement input validation mechanisms to ensure that request URLs are correctly encoded before being forwarded to backend services.  
4. Regularly monitor and audit the server logs for any suspicious activity or unauthorized access attempts that could indicate exploitation of this vulnerability.  
5. Consider implementing additional security measures, such as WAF rules or request filtering, to detect and block crafted requests that attempt to exploit the encoding problem in mod\_proxy.

### Finding 15

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-40898**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Server Side Request Forgery (SSRF) in Apache HTTP Server on Windows (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server on Windows with mod\_rewrite in server/vhost context, which allows for Server Side Request Forgery (SSRF) attacks that potentially leak NTLM hashes to a malicious server via SSRF and malicious requests, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to the exposure of sensitive NTLM hashes to malicious actors. Exploiting this vulnerability could compromise authentication mechanisms, leak sensitive information, and potentially lead to unauthorized access to systems or resources.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. NTLM Hash Exposure: Attackers could leverage SSRF attacks to leak NTLM hashes to a malicious server, potentially compromising user credentials and authentication mechanisms.  
  
2. Unauthorized Access: Exploiting the vulnerability could lead to unauthorized access to sensitive systems or resources by leveraging leaked NTLM hashes.  
  
3. Data Leakage: NTLM hash exposure could result in the leakage of sensitive information that could be used for further attacks or unauthorized access.  
  
4. Security Risks: SSRF attacks could introduce security risks and vulnerabilities that could be exploited by malicious actors to compromise the organization's systems.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the SSRF vulnerability in Apache HTTP Server on Windows with mod\_rewrite in server/vhost context, attackers could potentially leak NTLM hashes to a malicious server, compromising authentication mechanisms and potentially gaining unauthorized access to systems or resources. This could result in data breaches, unauthorized access, security risks, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to upgrade to version 2.4.62 to address this High severity vulnerability promptly and mitigate the risk of exploitation.

#### Public exploits related to this finding

## Description of the finding

SSRF in Apache HTTP Server on Windows with mod\_rewrite in server/vhost context, allows to potentially leak NTML hashes to a malicious server via SSRF and malicious requests.  
  
Users are recommended to upgrade to version 2.4.62 which fixes this issue.

## Mitigation steps

To mitigate the Server-Side Request Forgery (SSRF) vulnerability in Apache HTTP Server on Windows with mod\_rewrite in server/vhost context (CVE-2024-40898) that allows potential leakage of NTLM hashes to a malicious server via SSRF and malicious requests, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.62 or later to address this vulnerability.  
2. Review and secure the mod\_rewrite configuration in server/vhost context to prevent SSRF attacks and unauthorized access to sensitive information.  
3. Implement input validation and sanitization mechanisms to validate and restrict the URLs that can be accessed via SSRF.  
4. Consider implementing network segmentation and access controls to limit the exposure of sensitive information, such as NTLM hashes, to external servers.  
5. Regularly monitor server logs and network traffic for any signs of SSRF activity or attempts to leak NTLM hashes.  
6. Educate users and administrators about the risks of SSRF and the importance of secure server configurations to prevent such vulnerabilities.

### Finding 16

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-39573**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Server Side Request Forgery (SSRF) in mod\_rewrite (High)  
  
#### Severity Rationale:  
The vulnerability in mod\_rewrite in Apache HTTP Server versions 2.4.59 and earlier, which allows an attacker to potentially exploit Server Side Request Forgery (SSRF) by causing unsafe RewriteRules to unexpectedly set up URLs to be handled by mod\_proxy, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to SSRF attacks, potentially allowing attackers to manipulate URLs and direct requests to unintended destinations via mod\_proxy. Exploiting this vulnerability could result in unauthorized access, data exposure, or service disruptions.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Server Side Request Forgery (SSRF): Attackers could manipulate RewriteRules to set up URLs to be handled by mod\_proxy, potentially leading to SSRF attacks and unauthorized access to internal resources.  
  
2. Unauthorized Access: Exploiting the vulnerability could allow attackers to direct requests to unintended destinations, bypassing security controls and gaining unauthorized access to sensitive data or services.  
  
3. Data Exposure: SSRF attacks could result in the exposure of sensitive information or resources that should not be directly accessible from external requests.  
  
4. Service Disruptions: Improper handling of RewriteRules could lead to service disruptions, downtime, and impact on the availability of the server.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the SSRF vulnerability in mod\_rewrite in Apache HTTP Server, attackers could potentially manipulate RewriteRules to set up URLs to be handled by mod\_proxy, leading to SSRF attacks and unauthorized access to internal resources. This could result in data breaches, unauthorized access, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to upgrade to version 2.4.60 to address this High severity vulnerability promptly and mitigate the risk of exploitation.

#### Public exploits related to this finding

## Description of the finding

Potential SSRF in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier allows an attacker to cause unsafe RewriteRules to unexpectedly setup URL's to be handled by mod\_proxy.  
Users are recommended to upgrade to version 2.4.60, which fixes this issue.

## Mitigation steps

To mitigate the potential Server-Side Request Forgery (SSRF) vulnerability in mod\_rewrite in Apache HTTP Server 2.4.59 and earlier (CVE-2024-39573) that allows an attacker to cause unsafe RewriteRules to unexpectedly set up URLs to be handled by mod\_proxy, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.60 or later to address this vulnerability.  
2. Review and update RewriteRules in the mod\_rewrite configuration to ensure they are safe and do not inadvertently set up URLs to be handled by mod\_proxy.  
3. Implement input validation and sanitization mechanisms to validate and restrict the URLs processed by mod\_rewrite to prevent SSRF attacks.  
4. Regularly audit and monitor the mod\_rewrite configuration for any unsafe rules that could be exploited by attackers to trigger SSRF vulnerabilities.  
5. Consider implementing additional security measures, such as WAF rules or request filtering, to detect and block malicious requests that attempt to exploit the SSRF potential in mod\_rewrite.  
6. Educate users and administrators about the risks of SSRF and the importance of secure server configurations to prevent such vulnerabilities.

### Finding 17

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-38477**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Null Pointer Dereference in mod\_proxy (High)  
  
#### Severity Rationale:  
The vulnerability in mod\_proxy in Apache HTTP Server versions 2.4.59 and earlier, which allows an attacker to crash the server via a malicious request due to a null pointer dereference, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to server crashes and service disruptions. Exploiting this vulnerability could impact the availability and reliability of the server, potentially leading to downtime and operational issues.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Server Crashes: Attackers could exploit the null pointer dereference in mod\_proxy to crash the server, resulting in downtime, service disruptions, and potential data loss.  
  
2. Service Disruptions: Server crashes caused by the vulnerability could lead to service disruptions, impacting the availability of the server and potentially affecting users or customers.  
  
3. Operational Impact: The vulnerability could disrupt normal operations, leading to delays, loss of productivity, and potential financial losses for the organization.  
  
4. Security Risks: Server crashes could introduce security risks and vulnerabilities that could be exploited by attackers for further malicious activities.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the null pointer dereference in mod\_proxy in Apache HTTP Server, attackers could potentially crash the server via a malicious request, leading to service disruptions and operational impact. This could result in downtime, loss of productivity, financial losses, reputational damage, and potential legal consequences for the organization. It is crucial for users to upgrade to version 2.4.60 to address this High severity vulnerability promptly and mitigate the risk of exploitation.

#### Public exploits related to this finding

## Description of the finding

null pointer dereference in mod\_proxy in Apache HTTP Server 2.4.59 and earlier allows an attacker to crash the server via a malicious request.  
Users are recommended to upgrade to version 2.4.60, which fixes this issue.

## Mitigation steps

To mitigate the null pointer dereference vulnerability in mod\_proxy in Apache HTTP Server 2.4.59 and earlier (CVE-2024-38477) that allows an attacker to crash the server via a malicious request, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.60 or later to address this vulnerability.  
2. Regularly monitor server logs and performance metrics for any signs of crashes or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
3. Implement input validation mechanisms to sanitize and validate incoming requests to prevent null pointer dereference issues.  
4. Consider implementing additional security measures, such as request filtering and rate limiting, to protect the server from malicious requests that could trigger the null pointer dereference vulnerability.  
5. Educate users and administrators about the importance of keeping the server software up to date to mitigate known vulnerabilities and enhance overall security posture.

### Finding 18

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-38472**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Server Side Request Forgery (SSRF) in Apache HTTP Server on Windows (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server on Windows, which allows for Server Side Request Forgery (SSRF) attacks that potentially leak NTLM hashes to a malicious server via SSRF and malicious requests or content, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to the exposure of sensitive NTLM hashes to malicious actors. Exploiting this vulnerability could compromise authentication mechanisms, leak sensitive information, and potentially lead to unauthorized access to systems or resources.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. NTLM Hash Exposure: Attackers could leverage SSRF attacks to leak NTLM hashes to a malicious server, potentially compromising user credentials and authentication mechanisms.  
  
2. Unauthorized Access: Exploiting the vulnerability could lead to unauthorized access to sensitive systems or resources by leveraging leaked NTLM hashes.  
  
3. Data Leakage: NTLM hash exposure could result in the leakage of sensitive information that could be used for further attacks or unauthorized access.  
  
4. Security Risks: SSRF attacks could introduce security risks and vulnerabilities that could be exploited by malicious actors to compromise the organization's systems.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the SSRF vulnerability in Apache HTTP Server on Windows, attackers could potentially leak NTLM hashes to a malicious server, compromising authentication mechanisms and potentially gaining unauthorized access to systems or resources. This could result in data breaches, unauthorized access, security risks, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to upgrade to version 2.4.60 and configure the "UNCList" directive to address this High severity vulnerability promptly and mitigate the risk of exploitation.

#### Public exploits related to this finding

## Description of the finding

SSRF in Apache HTTP Server on Windows allows to potentially leak NTLM hashes to a malicious server via SSRF and malicious requests or content   
Users are recommended to upgrade to version 2.4.60 which fixes this issue.  Note: Existing configurations that access UNC paths will have to configure new directive "UNCList" to allow access during request processing.

## Mitigation steps

To mitigate the Server-Side Request Forgery (SSRF) vulnerability in Apache HTTP Server on Windows (CVE-2024-38472) that allows potential leakage of NTLM hashes to a malicious server via SSRF and malicious requests or content, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.60 or later to address this vulnerability.  
2. Review and secure the server configuration to prevent SSRF attacks and unauthorized access to sensitive information, such as NTLM hashes.  
3. Implement input validation and sanitization mechanisms to validate and restrict the URLs and content processed by the server to prevent SSRF vulnerabilities.  
4. Configure the new directive "UNCList" for existing configurations that access UNC paths to allow access during request processing.  
5. Regularly monitor server logs and network traffic for any signs of SSRF activity or attempts to leak NTLM hashes.  
6. Educate users and administrators about the risks of SSRF and the importance of secure server configurations to prevent such vulnerabilities.

### Finding 19

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-27316**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Memory Exhaustion in nghttp2 due to HTTP/2 Header Limit (High)  
  
#### Severity Rationale:  
The vulnerability in nghttp2, where incoming HTTP/2 headers exceeding the limit are temporarily buffered to generate an informative HTTP 413 response, leading to memory exhaustion if a client does not stop sending headers, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to memory exhaustion, impacting the availability and performance of the server. Exploiting this vulnerability could result in denial of service, service disruptions, and potential data loss.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Denial of Service: Memory exhaustion due to incoming HTTP/2 headers exceeding the limit could lead to denial of service, causing service disruptions and impacting user experience.  
  
2. Service Degradation: Memory exhaustion could degrade the performance of the server, leading to slow response times, timeouts, and potential loss of customers.  
  
3. Data Loss: In severe cases of memory exhaustion, data loss could occur, impacting the integrity and availability of critical information.  
  
4. Operational Impact: The vulnerability could disrupt normal operations, leading to delays, loss of productivity, and potential financial losses for the organization.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the memory exhaustion vulnerability in nghttp2 due to HTTP/2 header limits, attackers could potentially exhaust server memory, leading to denial of service and service degradation. This could result in service disruptions, slow response times, loss of customers, financial losses, reputational damage, and potential legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

HTTP/2 incoming headers exceeding the limit are temporarily buffered in nghttp2 in order to generate an informative HTTP 413 response. If a client does not stop sending headers, this leads to memory exhaustion.

## Mitigation steps

To mitigate the vulnerability in nghttp2 related to HTTP/2 incoming headers exceeding the limit, leading to memory exhaustion (CVE-2024-27316), users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update nghttp2 to a version that addresses this vulnerability.  
2. Configure nghttp2 to enforce limits on incoming headers to prevent memory exhaustion.  
3. Monitor server performance and memory usage to detect any abnormal behavior that could indicate memory exhaustion due to incoming headers.  
4. Implement rate limiting or request buffering mechanisms to handle excessive incoming headers and prevent memory exhaustion.  
5. Regularly review and adjust the configuration of nghttp2 to ensure that it can handle incoming headers within safe limits.  
6. Educate users and administrators about the risks of exceeding header limits in HTTP/2 and the importance of maintaining proper configurations to prevent memory exhaustion vulnerabilities.

### Finding 20

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2023-31122**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Out-of-Bounds Read in mod\_macro (High)  
  
#### Severity Rationale:  
The vulnerability in mod\_macro of Apache HTTP Server, which allows for an out-of-bounds read, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to unauthorized access to sensitive data, data leakage, or potential service disruptions. Exploiting this vulnerability could compromise the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Data Exposure: An out-of-bounds read could potentially expose sensitive data or information stored in memory, leading to data exposure and confidentiality breaches.  
  
2. Unauthorized Access: Exploiting the vulnerability could allow attackers to access data beyond the intended boundaries, potentially leading to unauthorized access to critical resources.  
  
3. Service Disruptions: The vulnerability could result in service disruptions, downtime, or instability in the Apache HTTP Server, impacting the availability and reliability of services.  
  
4. Data Integrity Compromise: Unauthorized access or manipulation of data through an out-of-bounds read could compromise the integrity of data stored on the server.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the out-of-bounds read vulnerability in mod\_macro of Apache HTTP Server, attackers could potentially access sensitive data, compromise data integrity, or disrupt services. This could result in data breaches, unauthorized access, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

Out-of-bounds Read vulnerability in mod\_macro of Apache HTTP Server.This issue affects Apache HTTP Server: through 2.4.57.

## Mitigation steps

To mitigate the Out-of-bounds Read vulnerability in mod\_macro of Apache HTTP Server (CVE-2023-31122) that affects versions through 2.4.57, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Disable or restrict the use of mod\_macro if it is not essential for the server's functionality to reduce the attack surface.  
3. Regularly monitor server logs and performance metrics for any signs of out-of-bounds read attempts or abnormal behavior that could indicate exploitation of this vulnerability.  
4. Implement input validation mechanisms to prevent out-of-bounds read vulnerabilities in mod\_macro.  
5. Consider implementing additional security measures, such as WAF rules or request filtering, to detect and block malicious requests that could trigger out-of-bounds read vulnerabilities.  
6. Educate users and administrators about the risks of out-of-bounds read vulnerabilities and the importance of keeping server software up to date to mitigate such security issues.

### Finding 21

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2023-27522**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: HTTP Response Smuggling in mod\_proxy\_uwsgi (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server via mod\_proxy\_uwsgi, which allows for HTTP Response Smuggling by special characters in the origin response header that can truncate or split the response forwarded to the client, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to response manipulation, data exposure, or potential security misconfigurations. Exploiting this vulnerability could compromise the integrity and confidentiality of data transmitted to clients.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Response Manipulation: Attackers could manipulate responses by using special characters in the origin response header, potentially altering the content or structure of the response forwarded to clients.  
  
2. Data Exposure: The vulnerability could lead to data exposure if responses are truncated or split, potentially exposing sensitive information to unauthorized parties.  
  
3. Security Misconfigurations: Response splitting or truncation could introduce security misconfigurations that could be exploited by attackers to bypass security controls or manipulate responses.  
  
4. Client-side Vulnerabilities: Manipulated responses could lead to client-side vulnerabilities, such as cross-site scripting (XSS) or injection attacks, impacting the security of client applications.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the HTTP Response Smuggling vulnerability in Apache HTTP Server via mod\_proxy\_uwsgi, attackers could potentially manipulate responses, expose sensitive data, or introduce security misconfigurations. This could result in data breaches, client-side vulnerabilities, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

HTTP Response Smuggling vulnerability in Apache HTTP Server via mod\_proxy\_uwsgi. This issue affects Apache HTTP Server: from 2.4.30 through 2.4.55.  
  
Special characters in the origin response header can truncate/split the response forwarded to the client.

## Mitigation steps

To mitigate the HTTP Response Smuggling vulnerability in Apache HTTP Server via mod\_proxy\_uwsgi (CVE-2023-27522) that affects versions from 2.4.30 through 2.4.55, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Configure mod\_proxy\_uwsgi to handle special characters in the origin response header properly to prevent response truncation or splitting.  
3. Implement input validation mechanisms to sanitize and validate response headers to prevent HTTP Response Smuggling attacks.  
4. Regularly monitor server logs and network traffic for any signs of response manipulation or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as WAF rules or response header validation, to detect and block malicious responses that could trigger HTTP Response Smuggling vulnerabilities.  
6. Educate users and administrators about the risks of HTTP Response Smuggling and the importance of secure server configurations to prevent such vulnerabilities.

### Finding 22

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-30556**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Buffer Over-read in Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.53 and earlier, where the server may return lengths to applications calling r:wsread() that point past the end of the storage allocated for the buffer, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to buffer over-reads, potentially exposing sensitive data or causing application crashes. Exploiting this vulnerability could compromise the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Data Exposure: Buffer over-reads could potentially expose sensitive data stored in memory to applications, leading to data exposure and confidentiality breaches.  
  
2. Application Crashes: Returning lengths pointing past the allocated buffer storage could cause application crashes or instability, impacting the availability and reliability of services.  
  
3. Unauthorized Access: Exploiting the vulnerability could potentially allow attackers to access or manipulate data beyond the intended boundaries, leading to unauthorized access to critical resources.  
  
4. Data Integrity Compromise: Buffer over-reads could compromise the integrity of data stored in memory, potentially leading to data corruption or unauthorized modifications.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the buffer over-read vulnerability in Apache HTTP Server, attackers could potentially expose sensitive data, cause application crashes, or compromise data integrity. This could result in data breaches, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

Apache HTTP Server 2.4.53 and earlier may return lengths to applications calling r:wsread() that point past the end of the storage allocated for the buffer.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.53 and earlier (CVE-2022-30556) that may return lengths to applications calling r:wsread() that point past the end of the storage allocated for the buffer, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Review and update applications that call r:wsread() to handle buffer lengths properly and prevent accessing storage beyond the allocated buffer.  
3. Implement input validation mechanisms to ensure that buffer lengths are within the allocated storage limits to prevent potential exploitation of this vulnerability.  
4. Regularly monitor application logs and performance metrics for any signs of buffer length issues or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as input validation and buffer overflow protections, to enhance the security of applications using r:wsread() in Apache HTTP Server.  
6. Educate developers and administrators about secure coding practices and the importance of handling buffer lengths correctly to prevent vulnerabilities like this from being exploited.

### Finding 23

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-30522**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Excessive Memory Allocation in mod\_sed of Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server version 2.4.53, where mod\_sed may make excessively large memory allocations and trigger an abort when configured to do transformations with very large input, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to excessive memory consumption, server crashes, or service disruptions. Exploiting this vulnerability could impact the availability and reliability of the server.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Excessive Memory Consumption: The vulnerability could lead to excessively large memory allocations by mod\_sed, consuming system resources and potentially causing memory exhaustion.  
  
2. Server Crashes: Triggering an abort due to excessive memory allocations could result in server crashes, leading to downtime, service disruptions, and potential data loss.  
  
3. Service Disruptions: Excessive memory consumption or server crashes could disrupt services, impact user experience, and lead to operational issues.  
  
4. Operational Impact: The vulnerability could impact the availability and reliability of the server, affecting business operations and potentially causing financial losses.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the excessive memory allocation vulnerability in mod\_sed of Apache HTTP Server, attackers could potentially consume excessive memory, trigger server crashes, or disrupt services. This could result in service disruptions, downtime, loss of productivity, financial losses, reputational damage, and potential legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

If Apache HTTP Server 2.4.53 is configured to do transformations with mod\_sed in contexts where the input to mod\_sed may be very large, mod\_sed may make excessively large memory allocations and trigger an abort.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.53 (CVE-2022-30522) related to mod\_sed triggering excessively large memory allocations and potentially causing an abort, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Review and adjust the configuration of mod\_sed to ensure that it is not used in contexts where the input may be very large, potentially leading to excessive memory allocations.  
3. Implement input validation mechanisms to limit the size of input data processed by mod\_sed to prevent triggering large memory allocations.  
4. Regularly monitor server logs and performance metrics for any signs of memory allocation issues or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as limiting the memory allocation size for mod\_sed or using alternative methods for transformations to mitigate the risk of excessive memory usage and potential aborts.  
6. Educate users and administrators about the risks associated with large memory allocations and the importance of configuring mod\_sed securely to prevent such vulnerabilities.

### Finding 24

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-29404**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Denial of Service in Lua Script of Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.53 and earlier, where a malicious request to a Lua script that calls r:parsebody(0) may cause a denial of service due to no default limit on possible input size, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to denial of service, impacting the availability and performance of the server. Exploiting this vulnerability could disrupt services, cause downtime, and affect user experience.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Denial of Service: A malicious request to a Lua script could cause a denial of service by consuming excessive resources or triggering server crashes, leading to service disruptions.  
  
2. Service Disruptions: Denial of service resulting from the vulnerability could disrupt services, impact user access, and lead to operational issues.  
  
3. Operational Impact: The vulnerability could impact the availability and reliability of the server, affecting business operations and potentially causing financial losses.  
  
4. Resource Exhaustion: No default limit on input size could lead to resource exhaustion, consuming system resources and impacting the performance of the server.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the denial of service vulnerability in Lua scripts of Apache HTTP Server, attackers could potentially cause service disruptions, server crashes, or resource exhaustion. This could result in downtime, loss of productivity, financial losses, reputational damage, and potential legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

In Apache HTTP Server 2.4.53 and earlier, a malicious request to a lua script that calls r:parsebody(0) may cause a denial of service due to no default limit on possible input size.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.53 and earlier (CVE-2022-29404) related to a denial of service caused by a malicious request to a Lua script that calls r:parsebody(0) with no default limit on possible input size, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Review and update Lua scripts that utilize r:parsebody(0) to implement input size limits and validation mechanisms to prevent denial of service attacks.  
3. Configure default limits on possible input sizes for Lua scripts to prevent excessive resource consumption and potential denial of service.  
4. Regularly monitor server logs and performance metrics for any signs of resource exhaustion or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as request filtering and rate limiting, to protect against malicious requests that could trigger denial of service conditions.  
6. Educate developers and administrators about secure coding practices and the importance of implementing input size limits to prevent denial of service vulnerabilities in Lua scripts used in Apache HTTP Server.

### Finding 25

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-26377**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: HTTP Request Smuggling in mod\_proxy\_ajp (High)  
  
#### Severity Rationale:  
The vulnerability in mod\_proxy\_ajp of Apache HTTP Server versions 2.4.53 and prior, which allows for inconsistent interpretation of HTTP requests (HTTP Request Smuggling), enabling an attacker to smuggle requests to the AJP server it forwards requests to, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could be exploited to manipulate or smuggle requests, potentially leading to unauthorized access, data manipulation, or service disruptions. Exploiting this vulnerability could compromise the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Request Manipulation: Attackers could manipulate HTTP requests to bypass security controls, access unauthorized resources, or perform malicious actions on the server.  
  
2. Unauthorized Access: Exploiting the vulnerability could lead to unauthorized access to sensitive information or resources on the AJP server, compromising data confidentiality.  
  
3. Data Manipulation: Attackers could manipulate requests to modify data, inject malicious content, or disrupt services, leading to data integrity issues.  
  
4. Service Disruptions: Inconsistent interpretation of HTTP requests could result in service disruptions, downtime, and impact on the availability of the server.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the HTTP Request Smuggling vulnerability in mod\_proxy\_ajp of Apache HTTP Server, attackers could potentially manipulate requests, gain unauthorized access to sensitive resources, manipulate data, or disrupt services. This could result in data breaches, unauthorized access, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

Inconsistent Interpretation of HTTP Requests ('HTTP Request Smuggling') vulnerability in mod\_proxy\_ajp of Apache HTTP Server allows an attacker to smuggle requests to the AJP server it forwards requests to. This issue affects Apache HTTP Server Apache HTTP Server 2.4 version 2.4.53 and prior versions.

## Mitigation steps

To mitigate the 'HTTP Request Smuggling' vulnerability in mod\_proxy\_ajp of Apache HTTP Server versions 2.4.53 and prior (CVE-2022-26377) that allows an attacker to smuggle requests to the AJP server it forwards requests to, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Configure mod\_proxy\_ajp to ensure consistent interpretation of HTTP requests and prevent request smuggling attacks.  
3. Implement request normalization techniques to detect and block malicious requests that could be used for smuggling attacks.  
4. Regularly monitor server logs and network traffic for any signs of request smuggling attempts or unusual request patterns.  
5. Consider implementing additional security measures, such as using a Web Application Firewall (WAF) or security proxy, to detect and mitigate HTTP Request Smuggling attacks targeting mod\_proxy\_ajp.

### Finding 26

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-22719**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Memory Corruption in Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.52 and earlier, where a carefully crafted request body can cause a read to a random memory area, potentially leading to memory corruption and crashing the process, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to memory corruption, process crashes, and potential service disruptions. Exploiting this vulnerability could impact the availability and reliability of the server.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Memory Corruption: A read to a random memory area could lead to memory corruption, potentially compromising the integrity of data and system stability.  
  
2. Process Crashes: Exploiting the vulnerability could cause the process to crash, resulting in downtime, service disruptions, and potential data loss.  
  
3. Service Disruptions: Memory corruption or process crashes could disrupt services, impact user access, and lead to operational issues.  
  
4. Data Integrity Compromise: Memory corruption could compromise the integrity of data stored in memory, potentially leading to data loss or unauthorized modifications.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the memory corruption vulnerability in Apache HTTP Server, attackers could potentially cause process crashes, memory corruption, or service disruptions. This could result in downtime, loss of productivity, financial losses, reputational damage, and potential legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

A carefully crafted request body can cause a read to a random memory area which could cause the process to crash. This issue affects Apache HTTP Server 2.4.52 and earlier.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.52 and earlier (CVE-2022-22719) that can cause a process to crash due to a read to a random memory area from a carefully crafted request body, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Implement input validation mechanisms to sanitize and validate request bodies to prevent crafted requests that could trigger random memory reads.  
3. Regularly monitor server logs and performance metrics for any signs of crashes or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
4. Consider implementing additional security measures, such as request filtering and input validation, to protect the server from malicious requests that could lead to process crashes.  
5. Educate users and administrators about the risks associated with crafted request bodies and the importance of keeping server software up to date to mitigate known vulnerabilities.

### Finding 27

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2021-41773**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Path Traversal and Remote Code Execution in Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server version 2.4.49, which allows for path traversal attacks and potential remote code execution due to a flaw in path normalization, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to unauthorized access, data manipulation, or remote code execution. Exploiting this vulnerability could compromise the confidentiality, integrity, and availability of data and systems.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face severe consequences such as:  
  
1. Unauthorized Access: Attackers could exploit path traversal to access files outside configured directories, potentially exposing sensitive information or resources.  
  
2. Remote Code Execution: The vulnerability could allow for remote code execution if CGI scripts are enabled for the aliased paths, leading to unauthorized execution of arbitrary code on the server.  
  
3. Data Manipulation: Attackers could manipulate data, inject malicious code, or compromise the integrity of the server by exploiting the path traversal vulnerability.  
  
4. Operational Disruptions: Exploiting the vulnerability could lead to service disruptions, downtime, and potential data loss, impacting the organization's operations.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the path traversal and remote code execution vulnerability in Apache HTTP Server version 2.4.49, attackers could potentially gain unauthorized access, execute arbitrary code, manipulate data, or disrupt services. This could result in data breaches, unauthorized access, service disruptions, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

#### Exploit ID 50383.

* Description: Apache HTTP Server 2.4.49 - Path Traversal &amp; Remote Code Execution (RCE)
* Download URL: https://www.exploit-db.com/exploits/50383

#### Exploit ID 50512.

* Description: Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (3)
* Download URL: https://www.exploit-db.com/exploits/50512

## Description of the finding

A flaw was found in a change made to path normalization in Apache HTTP Server 2.4.49. An attacker could use a path traversal attack to map URLs to files outside the directories configured by Alias-like directives. If files outside of these directories are not protected by the usual default configuration "require all denied", these requests can succeed. If CGI scripts are also enabled for these aliased pathes, this could allow for remote code execution. This issue is known to be exploited in the wild. This issue only affects Apache 2.4.49 and not earlier versions. The fix in Apache HTTP Server 2.4.50 was found to be incomplete, see CVE-2021-42013.

## Mitigation steps

To mitigate the path traversal and remote code execution vulnerability in Apache HTTP Server 2.4.49 (CVE-2021-41773) that allows an attacker to map URLs to files outside configured directories and potentially execute remote code, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that fully addresses this vulnerability, as the fix in version 2.4.50 was found to be incomplete.  
2. Review and update Alias-like directives to ensure they do not allow path traversal outside of intended directories.  
3. Implement proper access controls and restrictions, such as "require all denied" for files outside configured directories, to prevent unauthorized access.  
4. Disable CGI scripts for aliased paths to mitigate the risk of remote code execution.  
5. Regularly monitor server logs for any signs of unusual or malicious request patterns that could indicate attempts to exploit the path traversal and remote code execution vulnerability.  
6. Consider implementing additional security measures such as WAF rules to detect and block potentially malicious requests that could exploit this vulnerability.

### Finding 28

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2021-41524**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Null Pointer Dereference in Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server version 2.4.49, where a null pointer dereference was detected during HTTP/2 request processing, allowing an external source to cause a Denial of Service (DoS) attack on the server with a specially crafted request, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to service disruptions, downtime, and impact the availability of the server. Exploiting this vulnerability could result in denial of service and affect the organization's operations.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Denial of Service: Attackers could exploit the null pointer dereference to cause a DoS attack on the server, leading to service disruptions and impacting user access.  
  
2. Service Disruptions: The vulnerability could result in service disruptions, downtime, and potential loss of availability for critical services.  
  
3. Operational Impact: DoS attacks could impact the organization's operations, productivity, and potentially lead to financial losses.  
  
4. Reputation Damage: Service disruptions and downtime caused by the vulnerability could damage the organization's reputation and trust with customers.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the null pointer dereference vulnerability in Apache HTTP Server version 2.4.49, attackers could potentially cause a Denial of Service attack, leading to service disruptions and downtime. This could result in loss of productivity, financial losses, reputational damage, and potential legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

While fuzzing the 2.4.49 httpd, a new null pointer dereference was detected during HTTP/2 request processing, allowing an external source to DoS the server. This requires a specially crafted request. The vulnerability was recently introduced in version 2.4.49. No exploit is known to the project.

## Mitigation steps

To mitigate the null pointer dereference vulnerability in Apache HTTP Server version 2.4.49 (CVE-2021-41524) that allows an external source to perform a Denial of Service (DoS) attack on the server through specially crafted HTTP/2 requests, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Configure the server to handle HTTP/2 requests securely and validate incoming requests to prevent null pointer dereference issues.  
3. Implement input validation mechanisms to sanitize and validate HTTP/2 requests to prevent specially crafted requests that could trigger the vulnerability.  
4. Regularly monitor server logs and performance metrics for any signs of DoS attacks or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as rate limiting or request filtering, to protect the server from malicious HTTP/2 requests that could lead to a DoS condition.  
6. Educate users and administrators about the risks of null pointer dereference vulnerabilities and the importance of keeping server software up to date to mitigate known vulnerabilities.

### Finding 29

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2006-20001**
* Severity score: **7.5** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Memory Read/Write in Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.54 and earlier, where a carefully crafted If: request header can cause a memory read or write of a single zero byte in a pool memory location beyond the header value sent, potentially leading to a process crash, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to memory corruption, process crashes, and potential service disruptions. Exploiting this vulnerability could impact the availability and reliability of the server.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Memory Corruption: A memory read or write beyond the header value could lead to memory corruption, potentially compromising the integrity of data and system stability.  
  
2. Process Crashes: The vulnerability could cause the process to crash, resulting in downtime, service disruptions, and potential data loss.  
  
3. Service Disruptions: Memory corruption or process crashes could disrupt services, impact user access, and lead to operational issues.  
  
4. Data Integrity Compromise: Memory corruption could compromise the integrity of data stored in memory, potentially leading to data loss or unauthorized modifications.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the memory read/write vulnerability in Apache HTTP Server, attackers could potentially cause memory corruption, process crashes, or service disruptions. This could result in downtime, loss of productivity, financial losses, reputational damage, and potential legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

A carefully crafted If: request header can cause a memory read, or write of a single zero byte, in a pool (heap) memory location beyond the header value sent. This could cause the process to crash.  
  
This issue affects Apache HTTP Server 2.4.54 and earlier.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.54 and earlier (CVE-2006-20001) that can cause a memory read or write beyond the header value sent, potentially leading to a process crash, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Implement input validation mechanisms to sanitize and validate If: request headers to prevent memory read or write beyond the intended values.  
3. Regularly monitor server logs and performance metrics for any signs of memory-related issues or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
4. Consider implementing additional security measures, such as request filtering and input validation, to protect the server from malicious If: request headers that could trigger memory-related vulnerabilities.  
5. Educate users and administrators about the risks associated with carefully crafted request headers and the importance of keeping server software up to date to mitigate known vulnerabilities.

### Finding 30

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2023-38709**
* Severity score: **7.3** (**HIGH** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: HTTP Response Splitting in Apache HTTP Server (High)  
  
#### Severity Rationale:  
The vulnerability in the core of Apache HTTP Server, where faulty input validation allows malicious or exploitable backend/content generators to split HTTP responses, is classified as High severity.  
  
The High severity level indicates that this vulnerability poses a significant risk to the organization's security as it could lead to HTTP response splitting, potentially enabling attackers to manipulate responses and conduct various attacks. Exploiting this vulnerability could compromise the integrity and confidentiality of data transmitted over HTTP.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face significant consequences such as:  
  
1. Response Manipulation: Attackers could manipulate HTTP responses by splitting them, potentially injecting malicious content or altering the intended response structure.  
  
2. Security Risks: HTTP response splitting could introduce security risks, such as cross-site scripting (XSS) or injection attacks, impacting the security of client applications.  
  
3. Data Exposure: Split HTTP responses could expose sensitive information or resources to unauthorized parties, leading to data breaches or unauthorized access.  
  
4. Client-side Vulnerabilities: Manipulated responses could lead to client-side vulnerabilities, affecting the security and integrity of client applications.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the HTTP response splitting vulnerability in Apache HTTP Server, attackers could potentially manipulate responses, inject malicious content, or expose sensitive data. This could result in security breaches, data exposure, client-side vulnerabilities, financial losses, reputational damage, and legal consequences for the organization. It is crucial for users to address this High severity vulnerability promptly to mitigate the risk of exploitation and protect the organization's assets.

#### Public exploits related to this finding

## Description of the finding

Faulty input validation in the core of Apache allows malicious or exploitable backend/content generators to split HTTP responses.  
  
This issue affects Apache HTTP Server: through 2.4.58.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server versions through 2.4.58 (CVE-2023-38709) related to faulty input validation allowing malicious or exploitable backend/content generators to split HTTP responses, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Review and secure the backend/content generators to ensure they do not produce malicious or exploitable responses that could lead to response splitting.  
3. Implement proper input validation mechanisms to sanitize and validate HTTP responses to prevent response splitting attacks.  
4. Regularly monitor server logs and network traffic for any signs of response splitting attempts or unusual response patterns that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as response header validation and content inspection, to detect and block malicious responses that could trigger response splitting vulnerabilities.  
6. Educate users and administrators about the risks associated with response splitting vulnerabilities and the importance of maintaining secure configurations to prevent such exploits.

### Finding 31

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-24795**
* Severity score: **6.3** (**MEDIUM** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: HTTP Response Splitting in Apache HTTP Server (Medium)  
  
#### Severity Rationale:  
The vulnerability in multiple modules of Apache HTTP Server, allowing an attacker to inject malicious response headers into backend applications and cause an HTTP desynchronization attack, is classified as Medium severity.  
  
The Medium severity level indicates that this vulnerability poses a moderate risk to the organization's security as it could lead to HTTP response splitting and potential desynchronization attacks. While the impact is not as severe as high-level vulnerabilities, it still requires prompt attention to mitigate the risk.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face consequences such as:  
  
1. HTTP Response Splitting: Attackers could inject malicious response headers, potentially leading to HTTP response splitting and manipulation of client-side behavior.  
  
2. HTTP Desynchronization: The vulnerability could enable desynchronization attacks, causing inconsistencies in HTTP communication and potentially leading to security risks.  
  
3. Client-side Vulnerabilities: Manipulated responses could introduce client-side vulnerabilities, such as cross-site scripting (XSS) or injection attacks, impacting the security of client applications.  
  
4. Data Exposure: HTTP response splitting could expose sensitive information or resources to unauthorized parties, leading to data breaches or unauthorized access.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the HTTP response splitting vulnerability in Apache HTTP Server, attackers could potentially manipulate responses, inject malicious content, or cause desynchronization attacks. This could result in security risks, client-side vulnerabilities, data exposure, financial losses, reputational damage, and legal consequences for the organization. It is important for users to upgrade to version 2.4.59 promptly to address this Medium severity vulnerability and mitigate the risk of exploitation.

#### Public exploits related to this finding

## Description of the finding

HTTP Response splitting in multiple modules in Apache HTTP Server allows an attacker that can inject malicious response headers into backend applications to cause an HTTP desynchronization attack.  
  
Users are recommended to upgrade to version 2.4.59, which fixes this issue.

## Mitigation steps

To mitigate the HTTP Response splitting vulnerability in multiple modules of Apache HTTP Server (CVE-2024-24795) that allows an attacker to inject malicious response headers into backend applications and cause an HTTP desynchronization attack, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.59 or later to address this vulnerability.  
2. Review and secure the backend applications to prevent injection of malicious response headers that could lead to HTTP desynchronization attacks.  
3. Implement input validation mechanisms to sanitize and validate response headers to prevent HTTP Response splitting attacks.  
4. Regularly monitor server logs and network traffic for any signs of response header injection or abnormal response patterns that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as response header validation and content inspection, to detect and block malicious responses that could trigger HTTP desynchronization attacks.  
6. Educate users and administrators about the risks associated with HTTP Response splitting vulnerabilities and the importance of keeping server software up to date to mitigate known security issues.

### Finding 32

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2024-39884**
* Severity score: **6.2** (**MEDIUM** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Source Code Disclosure in Apache HTTP Server (Medium)  
  
#### Severity Rationale:  
The vulnerability in the core of Apache HTTP Server version 2.4.60, where a regression ignores some use of the legacy content-type based configuration of handlers, resulting in source code disclosure of local content under certain circumstances, is classified as Medium severity.  
  
The Medium severity level indicates that this vulnerability poses a moderate risk to the organization's security as it could lead to source code disclosure and potential exposure of sensitive information. While the impact is not as severe as high-level vulnerabilities, it still requires attention to prevent data exposure.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face consequences such as:  
  
1. Source Code Disclosure: The vulnerability could lead to the disclosure of source code or sensitive information stored in local content, potentially exposing proprietary data or intellectual property.  
  
2. Data Exposure: Source code disclosure could expose sensitive information, configuration details, or credentials, leading to data breaches or unauthorized access.  
  
3. Security Risks: Exposing source code could introduce security risks, such as unauthorized access, data manipulation, or exploitation of vulnerabilities in the code.  
  
4. Compliance Violations: Source code disclosure may violate data protection regulations or industry standards, leading to compliance issues and potential legal consequences.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the source code disclosure vulnerability in Apache HTTP Server, attackers could potentially access and disclose sensitive source code or information stored in local content. This could result in data breaches, unauthorized access, security risks, reputational damage, and legal consequences for the organization. It is important for users to upgrade to version 2.4.61 promptly to address this Medium severity vulnerability and mitigate the risk of source code exposure.

#### Public exploits related to this finding

## Description of the finding

A regression in the core of Apache HTTP Server 2.4.60 ignores some use of the legacy content-type based configuration of handlers.   "AddType" and similar configuration, under some circumstances where files are requested indirectly, result in source code disclosure of local content. For example, PHP scripts may be served instead of interpreted.  
  
Users are recommended to upgrade to version 2.4.61, which fixes this issue.

## Mitigation steps

To mitigate the regression vulnerability in Apache HTTP Server 2.4.60 (CVE-2024-39884) that results in source code disclosure of local content due to ignoring some use of the legacy content-type based configuration of handlers, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.61 or later to address this regression vulnerability.  
2. Review and update the configuration related to "AddType" and similar directives to ensure proper handling of content types and prevent source code disclosure.  
3. Implement access controls and restrictions to prevent unauthorized access to sensitive files and directories.  
4. Regularly monitor server logs and file access for any signs of unauthorized disclosure or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as file permission settings and content security policies, to protect against source code disclosure and unauthorized access to local content.  
6. Educate users and administrators about the risks associated with misconfigured content-type handlers and the importance of maintaining secure server configurations to prevent such vulnerabilities.

### Finding 33

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2023-45802**
* Severity score: **5.9** (**MEDIUM** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Memory Leak in HTTP/2 Stream Reset Handling in Apache HTTP Server (Medium)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server, where a memory leak occurs in the handling of HTTP/2 stream resets by clients, leading to a potential increase in memory footprint and the risk of running out of memory if multiple requests and resets are sent to keep the connection open, is classified as Medium severity.  
  
The Medium severity level indicates that while the probability of exploitation is low during normal HTTP/2 use, the potential impact of memory leaks and resource exhaustion can still pose risks to the server's performance and stability.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face consequences such as:  
  
1. Memory Leak: The vulnerability could lead to a gradual increase in memory consumption due to deferred memory deallocation, potentially impacting server performance.  
  
2. Resource Exhaustion: Continuous sending of requests and resets to keep connections open could exhaust memory resources, leading to potential out-of-memory conditions.  
  
3. Performance Degradation: The memory leak and resource exhaustion could result in performance degradation, slow response times, and service disruptions.  
  
4. Operational Impact: Running out of memory could impact the availability and reliability of the server, affecting business operations.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the memory leak in HTTP/2 stream reset handling in Apache HTTP Server, attackers could potentially exhaust memory resources, leading to performance issues, resource exhaustion, and potential service disruptions. While the probability of exploitation is low during normal use, the impact of memory leaks and resource exhaustion can still pose risks to the server's stability. It is important for users to upgrade to version 2.4.58 promptly to address this Medium severity vulnerability and mitigate the risk of memory leaks and resource exhaustion.

#### Public exploits related to this finding

## Description of the finding

When a HTTP/2 stream was reset (RST frame) by a client, there was a time window were the request's memory resources were not reclaimed immediately. Instead, de-allocation was deferred to connection close. A client could send new requests and resets, keeping the connection busy and open and causing the memory footprint to keep on growing. On connection close, all resources were reclaimed, but the process might run out of memory before that.  
  
This was found by the reporter during testing of CVE-2023-44487 (HTTP/2 Rapid Reset Exploit) with their own test client. During "normal" HTTP/2 use, the probability to hit this bug is very low. The kept memory would not become noticeable before the connection closes or times out.  
  
Users are recommended to upgrade to version 2.4.58, which fixes the issue.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server related to HTTP/2 stream resets (CVE-2023-45802) that could lead to a memory exhaustion issue due to deferred de-allocation of resources when a client resets streams, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.58 or later to address this vulnerability.  
2. Monitor and manage HTTP/2 connections to ensure that memory resources are properly reclaimed after stream resets.  
3. Implement mechanisms to handle stream resets and memory de-allocation efficiently to prevent memory exhaustion issues.  
4. Regularly monitor server memory usage and performance metrics to detect any abnormal memory growth that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing connection timeout settings or limits to prevent clients from keeping connections open indefinitely and causing memory footprint growth.  
6. Educate users and administrators about the risks associated with HTTP/2 stream resets and memory management issues, and the importance of keeping server software up to date to mitigate known vulnerabilities.

### Finding 34

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-37436**
* Severity score: **5.3** (**MEDIUM** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Response Header Truncation in Apache HTTP Server (Medium)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions prior to 2.4.55, where a malicious backend can cause response headers to be truncated early, resulting in some headers being incorporated into the response body and potentially impacting the interpretation of security-related headers by the client, is classified as Medium severity.  
  
The Medium severity level indicates that while the vulnerability may not directly lead to critical security breaches, it still poses a risk to the proper interpretation of response headers and could impact security measures implemented in the headers.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face consequences such as:  
  
1. Header Truncation: Response headers being truncated early could lead to misinterpretation of headers by the client, potentially affecting security measures or intended functionality.  
  
2. Security Implications: If security-related headers are affected by the truncation, it could impact security controls, authentication mechanisms, or other security measures implemented in the headers.  
  
3. Client-side Behavior: Misinterpreted headers could lead to unexpected client-side behavior, potential vulnerabilities, or security risks.  
  
4. Data Integrity: Truncated headers could impact the integrity of data transmitted in the response, potentially leading to data corruption or misinterpretation.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the response header truncation vulnerability in Apache HTTP Server, attackers could potentially manipulate response headers to be truncated early, impacting the interpretation of headers by the client. While the severity is medium and may not lead to critical security breaches, it could still affect security measures and client-side behavior. It is important for users to be aware of this vulnerability and consider upgrading to a version beyond 2.4.55 to mitigate the risk of header truncation issues.

#### Public exploits related to this finding

## Description of the finding

Prior to Apache HTTP Server 2.4.55, a malicious backend can cause the response headers to be truncated early, resulting in some headers being incorporated into the response body. If the later headers have any security purpose, they will not be interpreted by the client.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server prior to version 2.4.55 (CVE-2022-37436) that allows a malicious backend to cause response headers to be truncated early, resulting in some headers being incorporated into the response body and potentially impacting the interpretation of security-related headers by the client, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to version 2.4.55 or later to address this vulnerability.  
2. Review and secure the backend applications to prevent malicious manipulation of response headers that could lead to truncation and misinterpretation by clients.  
3. Implement proper input validation mechanisms to ensure that response headers are correctly formatted and not susceptible to truncation attacks.  
4. Regularly monitor server logs and network traffic for any signs of response header manipulation or abnormal response behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as response header validation and content inspection, to detect and prevent malicious responses that could impact the interpretation of security-related headers.  
6. Educate users and administrators about the risks associated with truncated response headers and the importance of maintaining secure server configurations to prevent such vulnerabilities.

### Finding 35

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-28614**
* Severity score: **5.3** (**MEDIUM** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Memory Read in Apache HTTP Server (Medium)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.53 and earlier, where the ap\_rwrite() function may read unintended memory if an attacker can cause the server to reflect very large input using ap\_rwrite() or ap\_rputs(), is classified as Medium severity.  
  
The Medium severity level indicates that this vulnerability poses a moderate risk to the organization's security as it could lead to unintended memory reads and potential information disclosure. While the impact is not as severe as high-level vulnerabilities, it still requires attention to prevent potential memory-related issues.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face consequences such as:  
  
1. Memory Read: The vulnerability could lead to unintended memory reads, potentially exposing sensitive information or data stored in memory.  
  
2. Information Disclosure: Unintended memory reads could result in information disclosure, exposing internal data or server configurations.  
  
3. Data Integrity: Memory reads could impact the integrity of data stored in memory, potentially leading to data corruption or unauthorized access.  
  
4. Security Risks: Information disclosure through unintended memory reads could introduce security risks and vulnerabilities that could be exploited by attackers.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the memory read vulnerability in Apache HTTP Server, attackers could potentially read unintended memory, leading to information disclosure and potential security risks. While the severity is medium and the impact may not be as severe as high-level vulnerabilities, it is important for users to address this vulnerability to prevent potential memory-related issues and protect sensitive information. It is recommended to compile modules against current headers to resolve the issue and mitigate the risk of unintended memory reads.

#### Public exploits related to this finding

## Description of the finding

The ap\_rwrite() function in Apache HTTP Server 2.4.53 and earlier may read unintended memory if an attacker can cause the server to reflect very large input using ap\_rwrite() or ap\_rputs(), such as with mod\_luas r:puts() function. Modules compiled and distributed separately from Apache HTTP Server that use the 'ap\_rputs' function and may pass it a very large (INT\_MAX or larger) string must be compiled against current headers to resolve the issue.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.53 and earlier (CVE-2022-28614) related to the ap\_rwrite() function reading unintended memory when reflecting very large input, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Review and secure modules that use the ap\_rwrite() or ap\_rputs() functions, especially those that reflect very large input, such as mod\_lua's r:puts() function.  
3. Implement input validation mechanisms to limit the size of input passed to ap\_rwrite() or ap\_rputs() functions to prevent unintended memory reads.  
4. Modules that use the 'ap\_rputs' function and may pass it a very large string should be recompiled against current headers to resolve the issue.  
5. Regularly monitor server logs and performance metrics for any signs of memory-related issues or abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
6. Educate developers and administrators about the risks associated with reflecting very large input and the importance of secure coding practices to prevent unintended memory reads in Apache HTTP Server modules.

### Finding 36

* Affected IP: **10.10.114.139**
* Affected port and service: **8080** (service **http**)
* Related CVE: **CVE-2022-28330**
* Severity score: **5.3** (**MEDIUM** severity)

## Finding severity rationale

### Vulnerability Severity Rationale  
  
#### Vulnerability: Out-of-Bounds Read in Apache HTTP Server (Medium)  
  
#### Severity Rationale:  
The vulnerability in Apache HTTP Server versions 2.4.53 and earlier on Windows, where the server may read beyond bounds when configured to process requests with the mod\_isapi module, is classified as Medium severity.  
  
The Medium severity level indicates that this vulnerability poses a moderate risk to the organization's security as it could lead to out-of-bounds reads and potential information exposure. While the impact is not as severe as high-level vulnerabilities, it still requires attention to prevent potential data leakage.  
  
#### Impact on the Organization:  
If this vulnerability is exploited, the organization could face consequences such as:  
  
1. Out-of-Bounds Read: The vulnerability could lead to out-of-bounds reads, potentially exposing sensitive information or data beyond the intended boundaries.  
  
2. Information Exposure: Out-of-bounds reads could result in information exposure, potentially leaking internal data or server configurations.  
  
3. Data Integrity: Reading beyond bounds could impact the integrity of data stored in memory, potentially leading to data corruption or unauthorized access.  
  
4. Security Risks: Information exposure through out-of-bounds reads could introduce security risks and vulnerabilities that could be exploited by attackers.  
  
#### Attacker's Result by Exploiting the Vulnerability:  
By exploiting the out-of-bounds read vulnerability in Apache HTTP Server on Windows, attackers could potentially read beyond bounds, leading to information exposure and potential security risks. While the severity is medium and the impact may not be as severe as high-level vulnerabilities, it is important for users to address this vulnerability to prevent potential data leakage and protect sensitive information. It is recommended to take necessary measures to mitigate the risk of out-of-bounds reads when processing requests with the mod\_isapi module.

#### Public exploits related to this finding

## Description of the finding

Apache HTTP Server 2.4.53 and earlier on Windows may read beyond bounds when configured to process requests with the mod\_isapi module.

## Mitigation steps

To mitigate the vulnerability in Apache HTTP Server 2.4.53 and earlier on Windows (CVE-2022-28330) that may read beyond bounds when configured to process requests with the mod\_isapi module, users are recommended to take the following steps:  
  
Mitigation Steps:  
1. Update Apache HTTP Server to a version that addresses this vulnerability.  
2. Review and secure the configuration of the mod\_isapi module to prevent reading beyond bounds.  
3. Implement input validation mechanisms to ensure that requests processed by the mod\_isapi module are within expected bounds to prevent potential exploitation.  
4. Regularly monitor server logs and performance metrics for any signs of abnormal behavior that could indicate exploitation attempts targeting this vulnerability.  
5. Consider implementing additional security measures, such as request filtering and input validation, to protect the server from malicious requests that could trigger reading beyond bounds in the mod\_isapi module.  
6. Educate users and administrators about the risks associated with this vulnerability and the importance of keeping server software up to date to mitigate known security issues.