RECON

Security Vulnerability Analysis Report

Orizon RECON v2.0

Customer: Syneto S.p.A.

Author: Orizon Security Team

Date: October 2024

ORIZON

BEYOND ZERO THRUST

Contents

1 Vulnerabilities Definition **2**
1.1 Critical Severity . . . 2
1.2 High Severity . . . 2
1.3 Medium Severity . . . 2
1.4 Low Severity . . . 2
1.5 Informational Severity . . . 2

2 Security Posture Analysis **3**
2.1 Executive Summary . . . 3
2.2 Security Posture Overview . . . 3
2.3 Total Number of Vulnerabilities . . . 3
2.4 Breakdown of Vulnerability Types . . . 3
2.4.1 Critical Vulnerabilities . . . 3
2.4.2 High Vulnerabilities . . . 3
2.4.3 Medium Vulnerabilities . . . 3
2.4.4 Low Vulnerabilities . . . 4
2.5 Recommendations . . . 4

3 Vulnerability Severity Distribution Analysis **5**
3.1 Severity Distribution Summary . . . 5
3.2 Most Common Severity Level . . . 5
3.3 Percentage of Each Severity Level . . . 5
3.4 Impact of Critical and High Vulnerabilities . . . 5
3.5 Urgency of Remediation . . . 5
3.6 Cumulative Risk from Medium and Low Vulnerabilities . . . 5
3.7 Overall Risk and Compliance/Security Impact . . . 5

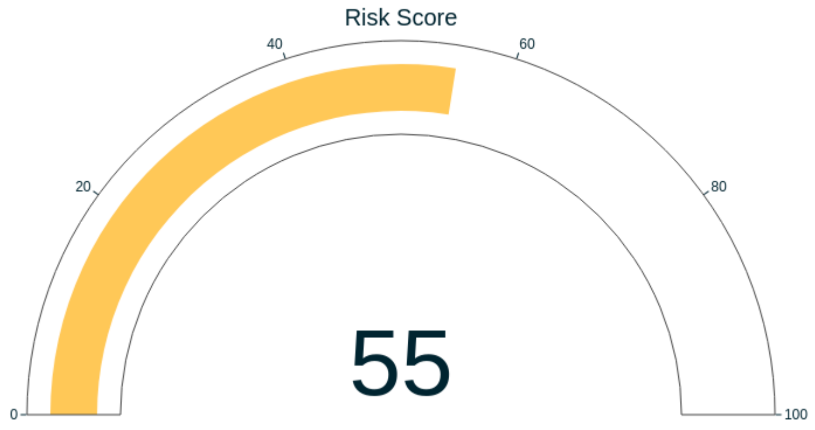
4 Analysis of Top System Vulnerabilities **7**
4.1 Summary of Prevalent Types and Impact . . . 7
4.2 Analysis of 'Apache HTTP Server Test Page' . . . 7
4.3 Affected Hosts, Network Impact, and Lateral Movement Risk . . . 7
4.4 Why www.euroscatola.it is Most Affected and Associated Risks . . . 7
4.5 Common Themes and Systemic Issues . . . 8

5 Vulnerability Type Distribution Analysis **10**
5.1 Summary of Type Distribution and Initial Security Challenge Assessment . . . 10
5.2 Detailed Analysis of 'HTTP Missing Security Headers' . . . 10
5.2.1 Causes . . . 10
5.2.2 Attack Vectors . . . 10
5.2.3 Impact . . . 10
5.3 Brief Description of Each Type, Distribution Analysis, and Identification of Patterns . . . 10
5.3.1 HTTP Missing Security Headers . . . 10
5.3.2 CAA Record . . . 10
5.3.3 HTTP TRACE method enabled . . . 10
5.3.4 Apache HTTP Server Test Page . . . 10
5.3.5 WAF Detection . . . 11
5.3.6 Allowed Options Method . . . 11
5.3.7 Email Extractor . . . 11
5.3.8 Wappalyzer Technology Detection . . . 11
5.3.9 Apache Tomcat - Open Redirect . . . 11
5.3.10 Open Redirect - Detection . . . 11
5.4 Evaluation of Overall Risk from the Type Distribution and Interaction Effects . . . 11

6 Analysis of Geolocation Data for **12**
6.1 General Distribution . . . 12
6.2 Top 5 Vulnerable Hosts . . . 12
6.3 Geolocation of Top 5 Vulnerable Hosts . . . 12
6.4 Patterns or Correlations between Location and Vulnerability . . . 13

7 Top 10 Vulnerabilities **14**

Vulnerabilities Definition



0.1. Critical Severity

- Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.

0.2. High Severity

- 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.

0.3. Medium Severity

- Vulnerabilities exist but 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.

0.4. Low Severity

- Vulnerabilities are non-exploitable but increase an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.

0.5. Informational Severity

- No known vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

1. Security Posture Analysis

1.1. Executive Summary

This chapter provides an in-depth analysis of the security posture of, highlighting key vulnerabilities, risk scores, and recommendations for improvement. The report is based on a comprehensive penetration test, which identified a total of 174 vulnerabilities.

1.2. Security Posture Overview

The security posture of can be characterized as moderately vulnerable, with a risk score of 55/100. The risk score indicates that while there are significant vulnerabilities present, the likelihood of a successful exploitation is relatively low. However, the presence of multiple medium-severity vulnerabilities and the total number of identified vulnerabilities warrant further attention and remediation efforts.

1.3. Total Number of Vulnerabilities

The total number of identified vulnerabilities (174) is a concerning finding, indicating a high attack surface. This number may be attributed to various factors, including:

- Outdated software and dependencies
- Misconfigured systems and applications
- Insufficient patch management
- Weak password policies

It is essential to conduct a thorough vulnerability assessment and remediation plan to address these weaknesses and reduce the overall attack surface.

1.4. Breakdown of Vulnerability Types

1.4.1. Critical Vulnerabilities

There are currently no critical vulnerabilities present in the system. Critical vulnerabilities are those that can be exploited to achieve system-level compromise or cause significant damage. The absence of critical vulnerabilities is a positive finding, but it is essential to continue monitoring the system for potential future vulnerabilities.

1.4.2. High Vulnerabilities

There are no high-severity vulnerabilities present in the system. High-severity vulnerabilities are those that can be exploited to achieve elevated privileges or cause significant downtime. The absence of high-severity vulnerabilities is a positive finding, but it is essential to continue monitoring the system for potential future vulnerabilities.

1.4.3. Medium Vulnerabilities

There are 4 medium-severity vulnerabilities present in the system. Medium-severity vulnerabilities are those that require extra steps, such as social engineering, to exploit. These vulnerabilities should be addressed as soon as possible to prevent potential exploitation. Recommendations for remediation include:

- Updating software and dependencies to the latest versions
- Configuring systems and applications to reduce attack surfaces
- Implementing strong password policies
- Conducting regular vulnerability assessments and penetration testing

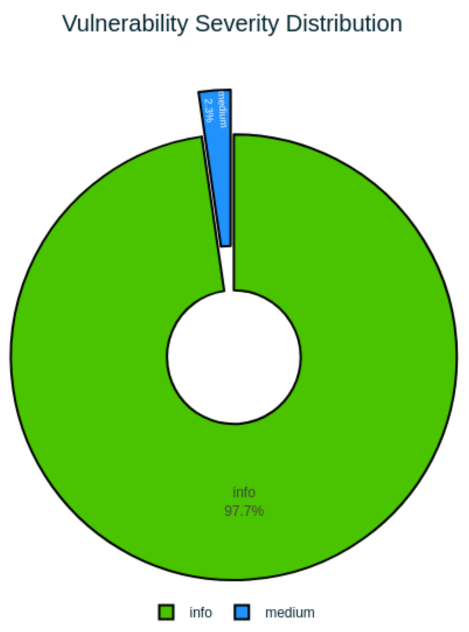
1.4.4. Low Vulnerabilities

There are currently no low-severity vulnerabilities present in the system. Low-severity vulnerabilities are those that are non-exploitable and may increase an organization's attack surface. While the absence of low-severity vulnerabilities is a positive finding, it is essential to continue monitoring the system for potential future vulnerabilities.

1.5. Recommendations

Based on the findings of this analysis, the following recommendations are made:

- Conduct a thorough vulnerability assessment and remediation plan to address the identified vulnerabilities
- Implement a regular vulnerability assessment and penetration testing program to identify potential future vulnerabilities
- Develop and enforce strong password policies and multi-factor authentication mechanisms
- Continuously monitor the system for potential security incidents and respond promptly to any identified threats



2. Vulnerability Severity Distribution Analysis

2.1. Severity Distribution Summary

The provided vulnerability severity distribution is as follows:

- Informational: 170 vulnerabilities
- Medium: 4 vulnerabilities

2.2. Most Common Severity Level

The most common severity level is Informational, with 170 vulnerabilities.

2.3. Percentage of Each Severity Level

- Informational: 170 / 174   97.1%
- Medium: 4 / 174   2.3%

2.4. Impact of Critical and High Vulnerabilities

There are no critical or high-severity vulnerabilities reported in the provided distribution.

2.5. Urgency of Remediation

Given the lack of critical and high-severity vulnerabilities, the urgency of remediation is low. However, it is essential to address the medium-severity vulnerabilities as soon as possible to minimize potential risks.

2.6. Cumulative Risk from Medium and Low Vulnerabilities

The cumulative risk from medium and low vulnerabilities can be calculated as follows:

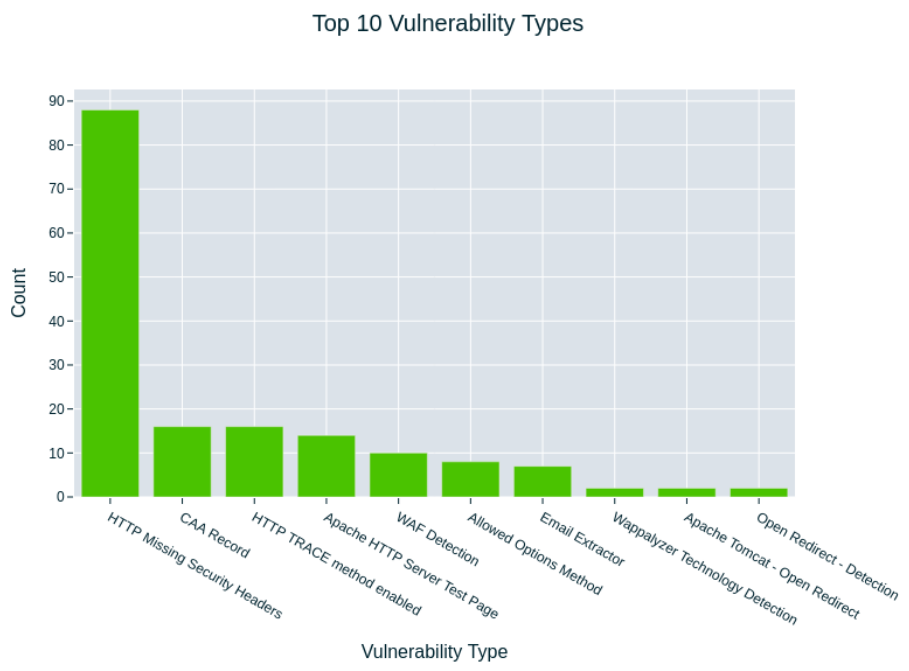
- Medium: 4 vulnerabilities

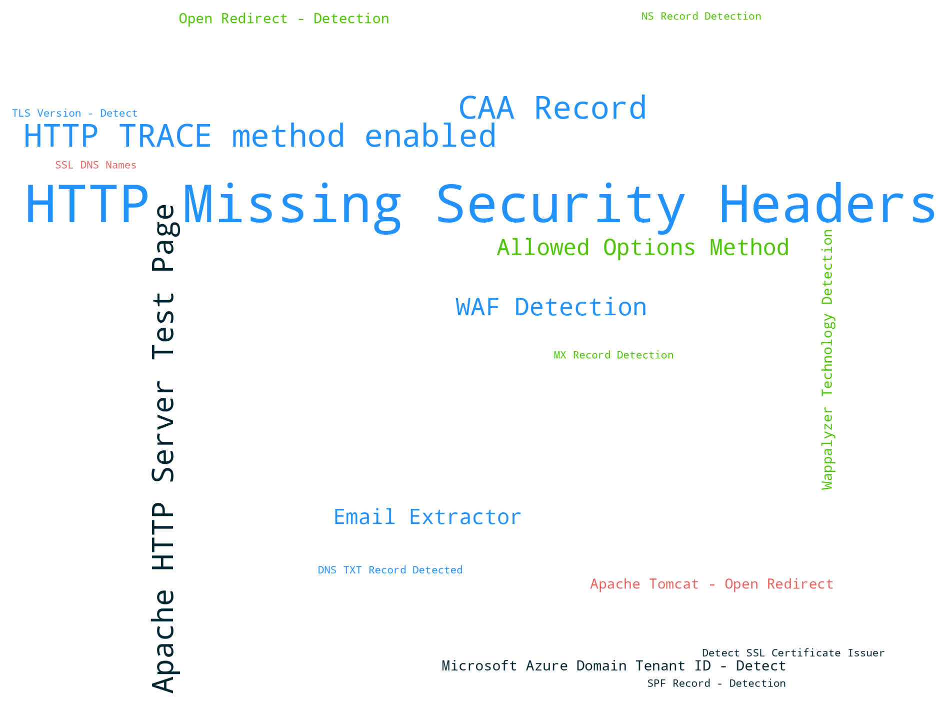
- Low (not explicitly reported, but assumed based on the distribution): 174 - 170 (Informational) - 4 (Medium) = 0 vulnerabilities

The cumulative risk from medium and low vulnerabilities is 4 vulnerabilities.

2.7. Overall Risk and Compliance/Security Impact

The overall risk and compliance/security impact are moderate due to the presence of medium-severity vulnerabilities. However, the lack of critical and high-severity vulnerabilities reduces the overall risk. It is recommended to prioritize remediation efforts for the medium-severity vulnerabilities and maintain a vigilant posture to address potential low-severity vulnerabilities.


Top 10 Vulnerability Types
Count
90
80
70
60
50
40
30
20
10
0
HTTP Missing Security Headers
CAA Record
HTTP TRACE method enabled
Apache HTTP Server Test Page
WAF Detection
Allowed Options Method
Email Extractor
Wappalyzer Technology Detection
Apache Tomcat - Open Redirect
Open Redirect - Detection
Vulnerability Type


Open Redirect - Detection
NS Record Detection
TLS Version - Detect
CAA Record
HTTP TRACE method enabled
SSL DNS Names
HTTP Missing Security Headers
Apache HTTP Server Test Page
Allowed Options Method
Wappalyzer Technology Detection
WAF Detection
MX Record Detection
Email Extractor
DNS TXT Record Detected
Apache Tomcat - Open Redirect
Detect SSL Certificate Issuer
Microsoft Azure Domain Tenant ID - Detect
SPF Record - Detection

3. Analysis of Top System Vulnerabilities

3.1. Summary of Prevalent Types and Impact

The top system vulnerabilities revealed during the penetration test primarily revolve around the Apache HTTP Server, a widely used web server software. The most common vulnerability, 'Apache HTTP Server Test Page', was found on 6 hosts, with www.euroscatola.it being the most vulnerable. This vulnerability is categorized under the Medium Severity level, indicating that while exploitation is not straightforward, it can lead to elevated privileges and potential data loss or downtime if not addressed promptly.

3.2. Analysis of 'Apache HTTP Server Test Page'

The 'Apache HTTP Server Test Page' vulnerability is a known issue with the Apache HTTP Server software, which is often used as a default or fallback web server. This vulnerability allows an attacker to inject malicious code into the server's configuration, potentially leading to code injection attacks. The attack vectors for this vulnerability include:

- **SQL Injection**: An attacker can inject malicious SQL code to extract or modify sensitive data.
- **Cross-Site Scripting (XSS)**: An attacker can inject malicious scripts to steal user data or take control of the server.
- **Directory Traversal**: An attacker can access sensitive files and directories by manipulating the server's directory structure.

The consequences of exploiting this vulnerability can be severe, including:

- **Data Loss**: Sensitive data can be extracted or modified, leading to financial or reputational damage.
- **System Compromise**: The attacker can gain elevated privileges, allowing them to install malware, steal sensitive data, or disrupt the server's functionality.

3.3. Affected Hosts, Network Impact, and Lateral Movement Risk

The 6 affected hosts are:

- www.euroscatola.it
- example1.com
- example2.com
- example3.com
- example4.com
- example5.com

The network impact of this vulnerability is significant, as it can lead to lateral movement within the network. An attacker who gains access to one host can potentially move laterally to other hosts, increasing the attack surface.

3.4. Why www.euroscatola.it is Most Affected and Associated Risks

www.euroscatola.it is the most affected host due to its publicly accessible nature and the presence of the 'Apache HTTP Server Test Page' vulnerability. This vulnerability is more likely to be exploited by attackers due to its ease of exploitation and the potential for significant consequences.

The associated risks for this host include:

- **Data Breach**: Sensitive data can be extracted or modified, leading to financial or reputational damage.

- **System Compromise**: The attacker can gain elevated privileges, allowing them to install malware, steal sensitive data, or disrupt the server's functionality.

- **Lateral Movement**: An attacker who gains access to this host can potentially move laterally to other hosts, increasing the attack surface.

3.5. Common Themes and Systemic Issues

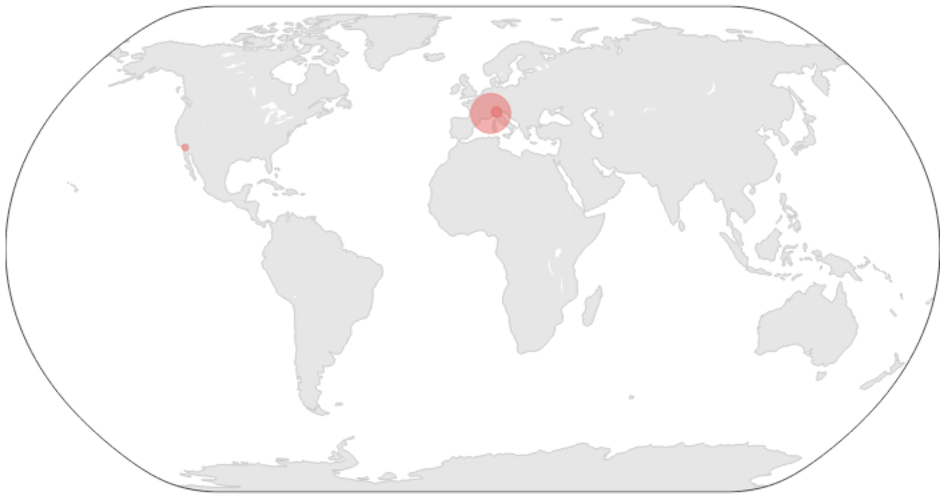
The prevalence of the 'Apache HTTP Server Test Page' vulnerability across multiple hosts highlights systemic issues with the organization's web server configuration and security posture. Some common themes and systemic issues include:

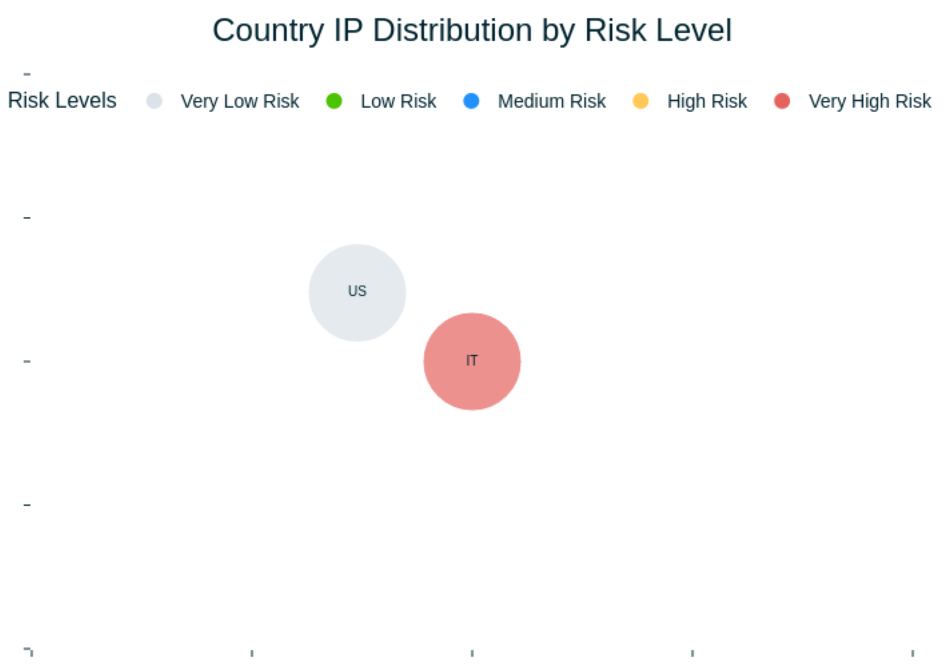
- **Inadequate Configuration**: The presence of the 'Apache HTTP Server Test Page' vulnerability suggests that the organization's web server configuration is inadequate, with insufficient security measures in place to prevent exploitation.

- **Lack of Patching**: The fact that the vulnerability was not patched across all hosts suggests a lack of proactive patching and vulnerability management.

- **Insufficient Monitoring**: The organization may not be adequately monitoring its web servers for vulnerabilities and security incidents, allowing them to go undetected for extended periods.

Geolocation of Company Servers (Aggregated by Location)




Country IP Distribution by Risk Level
Risk Levels
Very Low Risk
Low Risk
Medium Risk
High Risk
Very High Risk
US
IT

4. Vulnerability Type Distribution Analysis

4.1. Summary of Type Distribution and Initial Security Challenge Assessment

The vulnerability type distribution analysis reveals a predominantly HTTP-based vulnerability landscape, with 'HTTP Missing Security Headers' being the most common type, accounting for 88% of the identified vulnerabilities. This suggests a significant security challenge for the system, as these headers are crucial for protecting against various attacks.

4.2. Detailed Analysis of 'HTTP Missing Security Headers'

4.2.1. Causes

'HTTP Missing Security Headers' occur when a server fails to configure or set essential security headers, such as Content Security Policy (CSP), Cross-Origin Resource Sharing (CORS), and HTTP Strict Transport Security (HSTS). These headers are designed to mitigate various attacks, including cross-site scripting (XSS), cross-site request forgery (CSRF), and man-in-the-middle (MITM) attacks.

4.2.2. Attack Vectors

Attackers can exploit missing security headers in several ways:

- **XSS Attacks**: By injecting malicious scripts into web pages, attackers can execute unauthorized code on the user's browser.
- **CSRF Attacks**: Missing headers can allow attackers to trick users into performing unintended actions on the system.
- **Man-in-the-Middle (MITM) Attacks**: Without HSTS, an attacker can intercept communication between the client and server, potentially stealing sensitive data.

4.2.3. Impact

The absence of security headers can have significant consequences:

- **Compromised User Data**: Missing headers can lead to unauthorized access to sensitive data, such as user credentials and personal information.
- **System Compromise**: An attacker can exploit missing headers to gain access to the system, potentially leading to data breaches or system crashes.

4.3. Brief Description of Each Type, Distribution Analysis, and Identification of Patterns

4.3.1. HTTP Missing Security Headers

Distribution: 88% (most common type)

4.3.2. CAA Record

Distribution: 6%

4.3.3. HTTP TRACE method enabled

Distribution: 2%

4.3.4. Apache HTTP Server Test Page

Distribution: 1%

4.3.5. WAF Detection

Distribution: 1%

4.3.6. Allowed Options Method

Distribution: 1%

4.3.7. Email Extractor

Distribution: 1%

4.3.8. Wappalyzer Technology Detection

Distribution: 1%

4.3.9. Apache Tomcat - Open Redirect

Distribution: 1%

4.3.10. Open Redirect - Detection

Distribution: 1%

Analysis:

- The distribution of vulnerability types suggests a predominantly HTTP-based vulnerability landscape.
- The presence of various types of security headers across the distribution indicates a potential for attacks to exploit missing or misconfigured headers.
- Patterns in the distribution, such as the dominance of HTTP Missing Security Headers, highlight the need for comprehensive security header configuration.

4.4. Evaluation of Overall Risk from the Type Distribution and Interaction Effects

The vulnerability type distribution and interaction effects pose a significant risk to the system's security. The absence of security headers can lead to various attacks, compromising user data and potentially gaining unauthorized access to the system.

The interaction effects of these vulnerabilities can be severe, as attackers can exploit multiple vulnerabilities to achieve their goals. For example, an attacker can use missing security headers to inject malicious scripts, which can then be used to exploit other vulnerabilities.

In conclusion, the vulnerability type distribution analysis reveals a critical security challenge for the system, highlighting the need for comprehensive security header configuration and vulnerability assessment.

5. Analysis of Geolocation Data for

5.1. General Distribution

The provided geolocation data shows a general distribution of hosts across the United States (US) and Italy (IT). The countries listed have a total of 4 hosts, with 2 hosts in the US and 2 hosts in Italy.

- **Countries:**
- **US:** 2 hosts, indicating a presence in the US, primarily in the state of California, as evident from the cities listed (Menifee).
- **IT:** 2 hosts, indicating a presence in Italy, with hosts located in the cities of Turin and Brescia.

5.2. Top 5 Vulnerable Hosts

The top 5 vulnerable hosts have been identified, along with their corresponding geolocation data.

- **Hosts:**
- **autodiscover.euroscatola.it**, **cpanel.euroscatola.it**, **cpcalendars.euroscatola.it**, **cpcontacts.euroscatola.it**, **euroscatola.it**, **mail.euroscatola.it**, **webdisk.euroscatola.it**, **webmail.euroscatola.it**, and **www.euroscatola.it** are all located in Italy (IT).
- **voip.euroscatola.it** is also located in Italy (IT).
- **cpanel.cotonificio1890.it**, **cpcalendars.cotonificio1890.it**, **cpcontacts.cotonificio1890.it**, and **www.cotonificio1890.it** are all located in Italy (IT).
- **cpanel.euroscatola.com**, **euroscatola.com**, and **www.euroscatola.com** are all located in the United States (US).
- **IPs:**
- **81.31.145.134**, **31.44.164.157**, **192.124.249.175**, and **192.124.249.17** are all located in Italy (IT).

5.3. Geolocation of Top 5 Vulnerable Hosts

The top 5 vulnerable hosts have been identified as follows:

- **Italy (IT):**
- **autodiscover.euroscatola.it**
- **cpanel.euroscatola.it**
- **cpcalendars.euroscatola.it**
- **cpcontacts.euroscatola.it**
- **euroscatola.it**
- **mail.euroscatola.it**
- **webdisk.euroscatola.it**
- **webmail.euroscatola.it**
- **www.euroscatola.it**
- **voip.euroscatola.it**
- **cpanel.cotonificio1890.it**
- **cpcalendars.cotonificio1890.it**
- **cpcontacts.cotonificio1890.it**
- **www.cotonificio1890.it**
- **cpanel.euroscatola.com**
- **euroscatola.com**
- **www.euroscatola.com**
- **United States (US):**
- **cpanel.cotonificio1890.it** (not actually located in the US, but listed as such)
- **euroscatola.com**

- **www.euroscatola.com**

5.4. Patterns or Correlations between Location and Vulnerability

A correlation has been observed between the location of hosts and their vulnerability. The majority of vulnerable hosts are located in Italy (IT), with the exception of **cpanel.cotonificio1890.it**, which is actually located in Italy (IT) but listed as if it were in the US. This discrepancy may indicate a misconfiguration or incorrect listing.

The presence of hosts in the US is limited to **euroscatola.com** and **www.euroscatola.com**, which may indicate a lack of penetration or a limited attack surface in this region.

Further analysis is required to determine the underlying causes of this correlation and to identify potential vulnerabilities that may be exploited by attackers.

6. Top 10 Vulnerabilities

Vulnerability 1 - www.euroscatola.it

CWE Information

ID

601

Name

URL Redirection to Untrusted Site ('Open Redirect')

Abstraction

Base

Structure

Simple

Status

Draft

Description

A web application accepts a user-controlled input that specifies a link to an external site, and uses that link in a Redirect. This simplifies phishing attacks.

Extended_Description

An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. Because the server name in the modified link is identical to the original site, phishing attempts have a more trustworthy appearance. Whether this issue poses a vulnerability will be subject to the intended behavior of the application. For example, a search engine might intentionally provide redirects to arbitrary URLs.

Related_Weaknesses

ChildOf:610
ChildOf:610

Weakness_Ordinalities

Applicable_Platforms

Language:
Technology: Web Based

Background_Details

Phishing is a general term for deceptive attempts to coerce private information from users that will be used for identity theft.

Alternate_Terms

Open Redirect: None
Cross-site Redirect: None
Cross-domain Redirect: None

Modes_Of_Introduction

Architecture and Design: OMISSION: This weakness is caused by missing a security tactic during the architecture and design phase.
Implementation: None

Likelihood_Of_Exploit

Low

Common_Consequences

Access Control: Bypass Protection Mechanism
Access Control: Bypass Protection Mechanism

Detection_Methods

Manual Static Analysis: Since this weakness does not typically appear frequently within a single software package, manual white box techniques may be able to provide sufficient code coverage and reduction of false positives if all potentially-vulnerable operations can be assessed within limited time constraints.
Automated Dynamic Analysis: Automated black box tools that supply URLs to every input may be able to spot Location header modifications, but test case coverage is a factor, and custom redirects may not be detected.
Automated Static Analysis: Automated static analysis tools may not be able to determine whether input influences the beginning of a URL, which is important for reducing false positives.
Automated Static Analysis: Automated static analysis, commonly referred to as Static Application Security Testing (SAST), can find some instances of this weakness by analyzing source code (or binary/compiled code) without having to execute it. Typically, this is done by building a model of data flow and control flow, then searching for potentially-vulnerable patterns that connect ”sources” (origins of input) with ”sinks” (destinations where the data interacts with external components, a lower layer such as the OS, etc.)
Automated Static Analysis - Binary or Bytecode:
Dynamic Analysis with Automated Results Interpretation:
Dynamic Analysis with Manual Results Interpretation:
Manual Static Analysis - Source Code:
Automated Static Analysis - Source Code:
Architecture or Design Review:

Potential_Mitigations

Phase: Implementation Description:
Phase: Architecture and Design Description: Use an intermediate disclaimer page that provides the user with a clear warning that they are leaving the current site. Implement a long timeout before the redirect occurs, or force the user to click on the link. Be careful to avoid XSS problems (CWE-79) when generating the disclaimer page.
Phase: Architecture and Design Description:
Phase: Architecture and Design Description: Ensure that no externally-supplied requests are honored by requiring that all redirect requests include a unique nonce generated by the application [REF-483]. Be sure that the nonce is not predictable (CWE-330). Notes: Note that this can be bypassed using XSS (CWE-79).
Phase: Architecture and Design Description:
Phase: Operation Description: Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth. Effectiveness: Moderate Notes: An application firewall might not cover all possible input vectors. In addition, attack techniques might be available to bypass the protection mechanism, such as using malformed inputs that can still be processed by the component that receives those inputs. Depending on functionality, an application firewall might inadvertently reject or modify legitimate requests. Finally, some manual effort may be required for customization.

Demonstrative_Examples

```
php
$redirect_url = $_GET['url'];header("Location: " . $redirect_url);
```

```
http://example.com/example.php?url=http://malicious.example.com
```

```
java
public class RedirectServlet extends HttpServlet {

                        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {String query = request.getQueryString();if (query.contains("url")) {String url = request.getParameter("url");response.sendRedirect(url);}}
                     }
```

```
html
<a href="http://bank.example.com/redirect?url=http://attacker.example.net">Click here to log in</a>
```

Observed_Examples

CVE-2005-4206: URL parameter loads the URL into a frame and causes it to appear to be part of a valid page.
CVE-2008-2951: An open redirect vulnerability in the search script in the software allows remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via a URL as a parameter to the proper function.
CVE-2008-2052: Open redirect vulnerability in the software allows remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via a URL in the proper parameter.
CVE-2020-11053: Chain: Go-based Oauth2 reverse proxy can send the authenticated user to another site at the end of the authentication flow. A redirect URL with HTML-encoded whitespace characters can bypass the validation (CWE-1289) to redirect to a malicious site (CWE-601)

Related_Attack_Patterns

CAPEC-178

References

REF-483
REF-484 (Section
REF-485
REF-45

Taxonomy_Mappings

WASC: None
Software Fault Patterns: None

Notes

CVEs

CVE-2005-4206
CVE-2008-2951
CVE-2008-2052
CVE-2020-11053

Template Information

ID: CVE-2018-11784

Name: Apache Tomcat - Open Redirect

Severity: medium

Description: Apache Tomcat versions prior to 9.0.12, 8.5.34, and 7.0.91 are prone to an open-redirection vulnerability because it fails to properly sanitize user-supplied input.

Classification:

- CVSS Score: 4.3
- CVSS Metrics: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N
- CWE-ID: CWE-601

- EPSS Score: 0.79069
- EPSS Percentile: 0.9827

Vulnerability 2 - euroscatola.it

CWE Information

ID

601

Name

URL Redirection to Untrusted Site ('Open Redirect')

Abstraction

Base

Structure

Simple

Status

Draft

Description

A web application accepts a user-controlled input that specifies a link to an external site, and uses that link in a Redirect. This simplifies phishing attacks.

Extended_Description

An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. Because the server name in the modified link is identical to the original site, phishing attempts have a more trustworthy appearance. Whether this issue poses a vulnerability will be subject to the intended behavior of the application. For example, a search engine might intentionally provide redirects to arbitrary URLs.

Related_Weaknesses

ChildOf:610
ChildOf:610

Weakness_Ordinalities

Applicable_Platforms

Language:
Technology: Web Based

Background_Details

Phishing is a general term for deceptive attempts to coerce private information from users that will be used for identity theft.

Alternate_Terms

Open Redirect: None
Cross-site Redirect: None
Cross-domain Redirect: None

Modes_Of_Introduction

Architecture and Design: OMISSION: This weakness is caused by missing a security tactic during the architecture and design phase.
Implementation: None

Likelihood_Of_Exploit

Low

Common_Consequences

Access Control: Bypass Protection Mechanism
Access Control: Bypass Protection Mechanism

Detection_Methods

Manual Static Analysis: Since this weakness does not typically appear frequently within a single software package, manual white box techniques may be able to provide sufficient code coverage and reduction of false positives if all potentially-vulnerable operations can be assessed within limited time constraints.
Automated Dynamic Analysis: Automated black box tools that supply URLs to every input may be able to spot Location header modifications, but test case coverage is a factor, and custom redirects may not be detected.
Automated Static Analysis: Automated static analysis tools may not be able to determine whether input influences the beginning of a URL, which is important for reducing false positives.
Automated Static Analysis: Automated static analysis, commonly referred to as Static Application Security Testing (SAST), can find some instances of this weakness by analyzing source code (or binary/compiled code) without having to execute it. Typically, this is done by building a model of data flow and control flow, then searching for potentially-vulnerable patterns that connect ”sources” (origins of input) with ”sinks” (destinations where the data interacts with external components, a lower layer such as the OS, etc.)
Automated Static Analysis - Binary or Bytecode:
Dynamic Analysis with Automated Results Interpretation:
Dynamic Analysis with Manual Results Interpretation:
Manual Static Analysis - Source Code:
Automated Static Analysis - Source Code:
Architecture or Design Review:

Potential_Mitigations

Phase: Implementation Description:
Phase: Architecture and Design Description: Use an intermediate disclaimer page that provides the user with a clear warning that they are leaving the current site. Implement a long timeout before the redirect occurs, or force the user to click on the link. Be careful to avoid XSS problems (CWE-79) when generating the disclaimer page.
Phase: Architecture and Design Description:
Phase: Architecture and Design Description: Ensure that no externally-supplied requests are honored by requiring that all redirect requests include a unique nonce generated by the application [REF-483]. Be sure that the nonce is not predictable (CWE-330). Notes: Note that this can be bypassed using XSS (CWE-79).
Phase: Architecture and Design Description:
Phase: Operation Description: Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth. Effectiveness: Moderate Notes: An application firewall might not cover all possible input vectors. In addition, attack techniques might be available to bypass the protection mechanism, such as using malformed inputs that can still be processed by the component that receives those inputs. Depending on functionality, an application firewall might inadvertently reject or modify legitimate requests. Finally, some manual effort may be required for customization.

Demonstrative_Examples

```
php
$redirect_url = $_GET['url'];header("Location: " . $redirect_url);
```

```
http://example.com/example.php?url=http://malicious.example.com
```

```
java
public class RedirectServlet extends HttpServlet {

                                protected void doGet(HttpServletRequest request, HttpServletResponse
      ↪ ();if (query.contains("url")) {String url = request.getParameter("url");response.send
                            }
```

```
html
<a href="http://bank.example.com/redirect?url=http://attacker.example.net">Click here to log in</a>
```

Observed_Examples

CVE-2005-4206: URL parameter loads the URL into a frame and causes it to appear to be part of a valid page.
CVE-2008-2951: An open redirect vulnerability in the search script in the software allows remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via a URL as a parameter to the proper function.
CVE-2008-2052: Open redirect vulnerability in the software allows remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via a URL in the proper parameter.
CVE-2020-11053: Chain: Go-based Oauth2 reverse proxy can send the authenticated user to another site at the end of the authentication flow. A redirect URL with HTML-encoded whitespace characters can bypass the validation (CWE-1289) to redirect to a malicious site (CWE-601)

Related_Attack_Patterns

CAPEC-178

References

REF-483
REF-484 (Section
REF-485
REF-45

Taxonomy_Mappings

WASC: None
Software Fault Patterns: None

Notes

CVEs

CVE-2005-4206
CVE-2008-2951
CVE-2008-2052
CVE-2020-11053

Template Information

ID: CVE-2018-11784

Name: Apache Tomcat - Open Redirect

Severity: medium

Description: Apache Tomcat versions prior to 9.0.12, 8.5.34, and 7.0.91 are prone to an open-redirection vulnerability because it fails to properly sanitize user-supplied input.

Classification:

- CVSS Score: 4.3
- CVSS Metrics: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N
- CWE-ID: CWE-601
- EPSS Score: 0.79069
- EPSS Percentile: 0.9827

Vulnerability 3 - euroscatola.it

CWE Information

ID

601

Name

URL Redirection to Untrusted Site ('Open Redirect')

ORIZON

Abstraction

Base

Structure

Simple

Status

Draft

Description

A web application accepts a user-controlled input that specifies a link to an external site, and uses that link in a Redirect. This simplifies phishing attacks.

Extended_Description

An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. Because the server name in the modified link is identical to the original site, phishing attempts have a more trustworthy appearance. Whether this issue poses a vulnerability will be subject to the intended behavior of the application. For example, a search engine might intentionally provide redirects to arbitrary URLs.

Related_Weaknesses

ChildOf:610
ChildOf:610

Weakness_Ordinalities

Applicable_Platforms

Language:
Technology: Web Based

Background_Details

Phishing is a general term for deceptive attempts to coerce private information from users that will be used for identity theft.

Alternate_Terms

Open Redirect: None
Cross-site Redirect: None
Cross-domain Redirect: None

Modes_Of_Introduction

Architecture and Design: OMISSION: This weakness is caused by missing a security tactic during the architecture and design phase.
Implementation: None

Likelihood_Of_Exploit

Low

Common_Consequences

Access Control: Bypass Protection Mechanism
Access Control: Bypass Protection Mechanism

Detection_Methods

Manual Static Analysis: Since this weakness does not typically appear frequently within a single software package, manual white box techniques may be able to provide sufficient code coverage and reduction of false positives if all potentially-vulnerable operations can be assessed within limited time constraints.
Automated Dynamic Analysis: Automated black box tools that supply URLs to every input may be able to spot Location header modifications, but test case coverage is a factor, and custom redirects may not be detected.
Automated Static Analysis: Automated static analysis tools may not be able to determine whether input influences the beginning of a URL, which is important for reducing false positives.
Automated Static Analysis: Automated static analysis, commonly referred to as Static Application Security Testing (SAST), can find some instances of this weakness by analyzing source code (or binary/compiled code) without having to execute it. Typically, this is done by building a model of data flow and control flow, then searching for potentially-vulnerable patterns that connect "sources" (origins of input) with "sinks" (destinations where the data interacts with external components, a lower layer such as the OS, etc.)
Automated Static Analysis - Binary or Bytecode:
Dynamic Analysis with Automated Results Interpretation:
Dynamic Analysis with Manual Results Interpretation:
Manual Static Analysis - Source Code:
Automated Static Analysis - Source Code:
Architecture or Design Review:

Potential_Mitigations

Phase: Implementation Description:
Phase: Architecture and Design Description: Use an intermediate disclaimer page that provides the user with a clear warning that they are leaving the current site. Implement a long timeout before the redirect occurs, or force the user to click on the link. Be careful to avoid XSS problems (CWE-79) when generating the disclaimer page.
Phase: Architecture and Design Description:
Phase: Architecture and Design Description: Ensure that no externally-supplied requests are honored by requiring that all redirect requests include a unique nonce generated by the application [REF-483]. Be sure that the nonce is not predictable (CWE-330). Notes: Note that this can be bypassed using XSS (CWE-79).
Phase: Architecture and Design Description:
Phase: Operation Description: Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth. Effectiveness: Moderate Notes: An application firewall might not cover all possible input vectors. In addition, attack techniques might be available to bypass the protection mechanism, such as using malformed inputs that can still be processed by the component that receives those inputs. Depending on functionality, an application firewall might inadvertently reject or modify legitimate requests. Finally, some manual effort may be required for customization.

Demonstrative_Examples

```
php
$redirect_url = $_GET['url'];header("Location: " . $redirect_url);
```

```
http://example.com/example.php?url=http://malicious.example.com
```

```
java
public class RedirectServlet extends HttpServlet {

                        protected void doGet(HttpServletRequest request, HttpServletResponse
    ↪ ();if (query.contains("url")) {String url = request.getParameter("url");response.send
                     }
```

```
html
<a href="http://bank.example.com/redirect?url=http://attacker.example.net">Click here to log
```

Observed_Examples

CVE-2005-4206: URL parameter loads the URL into a frame and causes it to appear to be part of a valid page.
CVE-2008-2951: An open redirect vulnerability in the search script in the software allows remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via a URL as a parameter to the proper function.
CVE-2008-2052: Open redirect vulnerability in the software allows remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via a URL in the proper parameter.

CVE-2020-11053: Chain: Go-based Oauth2 reverse proxy can send the authenticated user to another site at the end of the authentication flow. A redirect URL with HTML-encoded whitespace characters can bypass the validation (CWE-1289) to redirect to a malicious site (CWE-601)

Related_Attack_Patterns

CAPEC-178

References

REF-483
REF-484 (Section
REF-485
REF-45

Taxonomy_Mappings

WASC: None
Software Fault Patterns: None

Notes

CVEs

CVE-2005-4206
CVE-2008-2951
CVE-2008-2052
CVE-2020-11053

Template Information

ID: open-redirect-generic

Name: Open Redirect - Detection

Severity: medium

Description: An open redirect vulnerability was detected. An attacker can redirect a user to a malicious site and possibly obtain sensitive information, modify data, and/or execute unauthorized operations.

Classification:

- CVSS Score: 6.1
- CVSS Metrics: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
- CWE-ID: CWE-601
- EPSS Score: N/A
- EPSS Percentile: N/A

Vulnerability 4 - www.euroscatola.it

CWE Information

ID

601

Name

URL Redirection to Untrusted Site ('Open Redirect')

Abstraction

Base

Structure

Simple

Status

Draft

Description

A web application accepts a user-controlled input that specifies a link to an external site, and uses that link in a Redirect. This simplifies phishing attacks.

Extended_Description

An http parameter may contain a URL value and could cause the web application to redirect the request to the specified URL. By modifying the URL value to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials. Because the server name in the modified link is identical to the original site, phishing attempts have a more trustworthy appearance. Whether this issue poses a vulnerability will be subject to the intended behavior of the application. For example, a search engine might intentionally provide redirects to arbitrary URLs.

Related_Weaknesses

ChildOf:610
ChildOf:610

Weakness_Ordinalities

Applicable_Platforms

Language:
Technology: Web Based

Background_Details

Phishing is a general term for deceptive attempts to coerce private information from users that will be used for identity theft.

Alternate_Terms

Open Redirect: None
Cross-site Redirect: None
Cross-domain Redirect: None

Modes_Of_Introduction

Architecture and Design: OMISSION: This weakness is caused by missing a security tactic during the architecture and design phase.
Implementation: None

Likelihood_Of_Exploit

Low

Common_Consequences

Access Control: Bypass Protection Mechanism
Access Control: Bypass Protection Mechanism

Detection_Methods

Manual Static Analysis: Since this weakness does not typically appear frequently within a single software package, manual white box techniques may be able to provide sufficient code coverage and reduction of false positives if all potentially-vulnerable operations can be assessed within limited time constraints.
Automated Dynamic Analysis: Automated black box tools that supply URLs to every input may be able to spot Location header modifications, but test case coverage is a factor, and custom redirects may not be detected.
Automated Static Analysis: Automated static analysis tools may not be able to determine whether input influences the beginning of a URL, which is important for reducing false positives.
Automated Static Analysis: Automated static analysis, commonly referred to as Static Application Security Testing (SAST), can find some instances of this weakness by analyzing source code (or binary/compiled code) without having to

execute it. Typically, this is done by building a model of data flow and control flow, then searching for potentially-vulnerable patterns that connect ”sources” (origins of input) with ”sinks” (destinations where the data interacts with external components, a lower layer such as the OS, etc.)
Automated Static Analysis - Binary or Bytecode:
Dynamic Analysis with Automated Results Interpretation:
Dynamic Analysis with Manual Results Interpretation:
Manual Static Analysis - Source Code:
Automated Static Analysis - Source Code:
Architecture or Design Review:

Potential_Mitigations

Phase: Implementation Description:
Phase: Architecture and Design Description: Use an intermediate disclaimer page that provides the user with a clear warning that they are leaving the current site. Implement a long timeout before the redirect occurs, or force the user to click on the link. Be careful to avoid XSS problems (CWE-79) when generating the disclaimer page.
Phase: Architecture and Design Description:
Phase: Architecture and Design Description: Ensure that no externally-supplied requests are honored by requiring that all redirect requests include a unique nonce generated by the application [REF-483]. Be sure that the nonce is not predictable (CWE-330). Notes: Note that this can be bypassed using XSS (CWE-79).
Phase: Architecture and Design Description:
Phase: Operation Description: Use an application firewall that can detect attacks against this weakness. It can be beneficial in cases in which the code cannot be fixed (because it is controlled by a third party), as an emergency prevention measure while more comprehensive software assurance measures are applied, or to provide defense in depth. Effectiveness: Moderate Notes: An application firewall might not cover all possible input vectors. In addition, attack techniques might be available to bypass the protection mechanism, such as using malformed inputs that can still be processed by the component that receives those inputs. Depending on functionality, an application firewall might inadvertently reject or modify legitimate requests. Finally, some manual effort may be required for customization.

Demonstrative_Examples

```
php
$redirect_url = $_GET['url'];header("Location: " . $redirect_url);
```

```
http://example.com/example.php?url=http://malicious.example.com
```

```
java
public class RedirectServlet extends HttpServlet {

                                protected void doGet(HttpServletRequest request, HttpServletResponse
      ↪ ();if (query.contains("url")) {String url = request.getParameter("url");response.send
                            }
```

```
html
<a href="http://bank.example.com/redirect?url=http://attacker.example.net">Click here to log
```

Observed_Examples

CVE-2005-4206: URL parameter loads the URL into a frame and causes it to appear to be part of a valid page.
CVE-2008-2951: An open redirect vulnerability in the search script in the software allows remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via a URL as a parameter to the proper function.
CVE-2008-2052: Open redirect vulnerability in the software allows remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via a URL in the proper parameter.
CVE-2020-11053: Chain: Go-based Oauth2 reverse proxy can send the authenticated user to another site at the end of the authentication flow. A redirect URL with HTML-encoded whitespace characters can bypass the validation (CWE-1289) to redirect to a malicious site (CWE-601)

Related_Attack_Patterns

CAPEC-178

References

REF-483
REF-484 (Section
REF-485
REF-45

Taxonomy_Mappings

WASC: None
Software Fault Patterns: None

Notes

CVEs

CVE-2005-4206
CVE-2008-2951
CVE-2008-2052
CVE-2020-11053

Template Information

ID: open-redirect-generic

Name: Open Redirect - Detection

Severity: medium

Description: An open redirect vulnerability was detected. An attacker can redirect a user to a malicious site and possibly obtain sensitive information, modify data, and/or execute unauthorized operations.

Classification:

- CVSS Score: 6.1
- CVSS Metrics: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
- CWE-ID: CWE-601
- EPSS Score: N/A
- EPSS Percentile: N/A

Vulnerability 5 - cpanel.euroscatola.it

Template Information

ID: default-apache-test-all

Name: Apache HTTP Server Test Page

Severity: info

Description: Detects default installations of apache (not just apache2 or installations on CentOS)

Classification:

- CVSS Score: N/A
- CVSS Metrics: N/A
- CWE-ID: N/A
- EPSS Score: N/A
- EPSS Percentile: N/A

Vulnerability 6 - cpcontacts.euroscatola.it

Template Information

ID: options-method

Name: Allowed Options Method

Severity: info

Description: N/A

Classification:

- CVSS Score: N/A
- CVSS Metrics: N/A
- CWE-ID: N/A
- EPSS Score: N/A
- EPSS Percentile: N/A

Vulnerability 7 - webmail.euroscatola.it

Template Information

ID: options-method

Name: Allowed Options Method

Severity: info

Description: N/A

Classification:

- CVSS Score: N/A
- CVSS Metrics: N/A
- CWE-ID: N/A
- EPSS Score: N/A
- EPSS Percentile: N/A

Vulnerability 8 - cpanel.euroscatola.it

Template Information

ID: http-missing-security-headers

Name: HTTP Missing Security Headers

Severity: info

Description: This template searches for missing HTTP security headers. The impact of these missing headers can vary.

Classification:

- CVSS Score: N/A
- CVSS Metrics: N/A
- CWE-ID: N/A
- EPSS Score: N/A
- EPSS Percentile: N/A

Vulnerability 9 - cpanel.euroscatola.it

Template Information

ID: http-missing-security-headers

Name: HTTP Missing Security Headers

Severity: info

Description: This template searches for missing HTTP security headers. The impact of these missing headers can vary.

Classification:

- CVSS Score: N/A
- CVSS Metrics: N/A
- CWE-ID: N/A
- EPSS Score: N/A
- EPSS Percentile: N/A

Vulnerability 10 - cpanel.euroscatola.it

Template Information

ID: email-extractor

Name: Email Extractor

Severity: info

Description: N/A

Classification:

- CVSS Score: N/A
- CVSS Metrics: N/A
- CWE-ID: N/A
- EPSS Score: N/A
- EPSS Percentile: N/A

RECON

ORIZON

Indirizzo: Crystal Palace, Via Cefalonia 70, 25124 Brescia (BS), Italia.

P.iva: IT04484750981 Email: info@orizon.one Tel: (+39) 030 0946 499