Security Report

# W4KEY Lab Penetration Test

Report Version: 20

Report ID: a3f850ed-9e26-4f78-9cee-11d3657d178a

Date: 14/02/2024 - 15/02/2024

CONFIDENTIAL

# Table of contents


Scope

Statistics and Risk

Results (10)

Critical File Upload Vulnerability

Medium Weak Hashing Algorithm

Low TLS 1.0 / 1.1 Protocol Detected

Low Directory listing

Low LUCKY13

Low BEAST

Low Weak Diffie-Hellman - Logjam Attack

Low OCSP Stapling Not Configured

Low Web Server HTTP Header Information Disclosure

Low [HSTS] Strict transport security not enforced

Report summary comment

Report authors

Changelog

# Scope

10.0.0.0/24

# Methodology and Standards:

- OSTTMM(Open Source Security Testing Methodology Manual)
- OWASP(Open Web Application Security Project)
- ISSAF(Information Systems Security Assessment Framework)
- WASC-TC(Web Application Security Consortium Threat Classification)
- PTF(Penetration Testing Framework)
- OISSG(Information Systems Security Assessment Framework)
- NIST SP800-115(Technical Guide to Information Security Testing and Assessment)

# Statistics and Risk

| Severity | Number |
|---|---|
| Critical | 1 |
| High | 0 |
| Medium | 1 |
| Low | 8 |
| Info | 0 |

The risk of application security vulnerabilities discovered during an assessment will be rated according to a custom-tailored version the OWASP Risk Rating Methodology. Risk severity is determined based on the estimated technical and business impact of the vulnerability, and on the estimated likelihood of the vulnerability being exploited:

| | | Overall Risk Severity | | |
|---|---|---|---|---|
| Impact | HIGH | Medium | High | Critical |
| | MEDIUM | Low | Medium | High |
| | LOW | Info | Low | Medium |
| | | LOW | MEDIUM | HIGH |
| | | Likelihood | | |

Our Risk rating is based on this calculation: **Risk = Likelihood * Impact**.

# Results (10)

## Critical File Upload Vulnerability

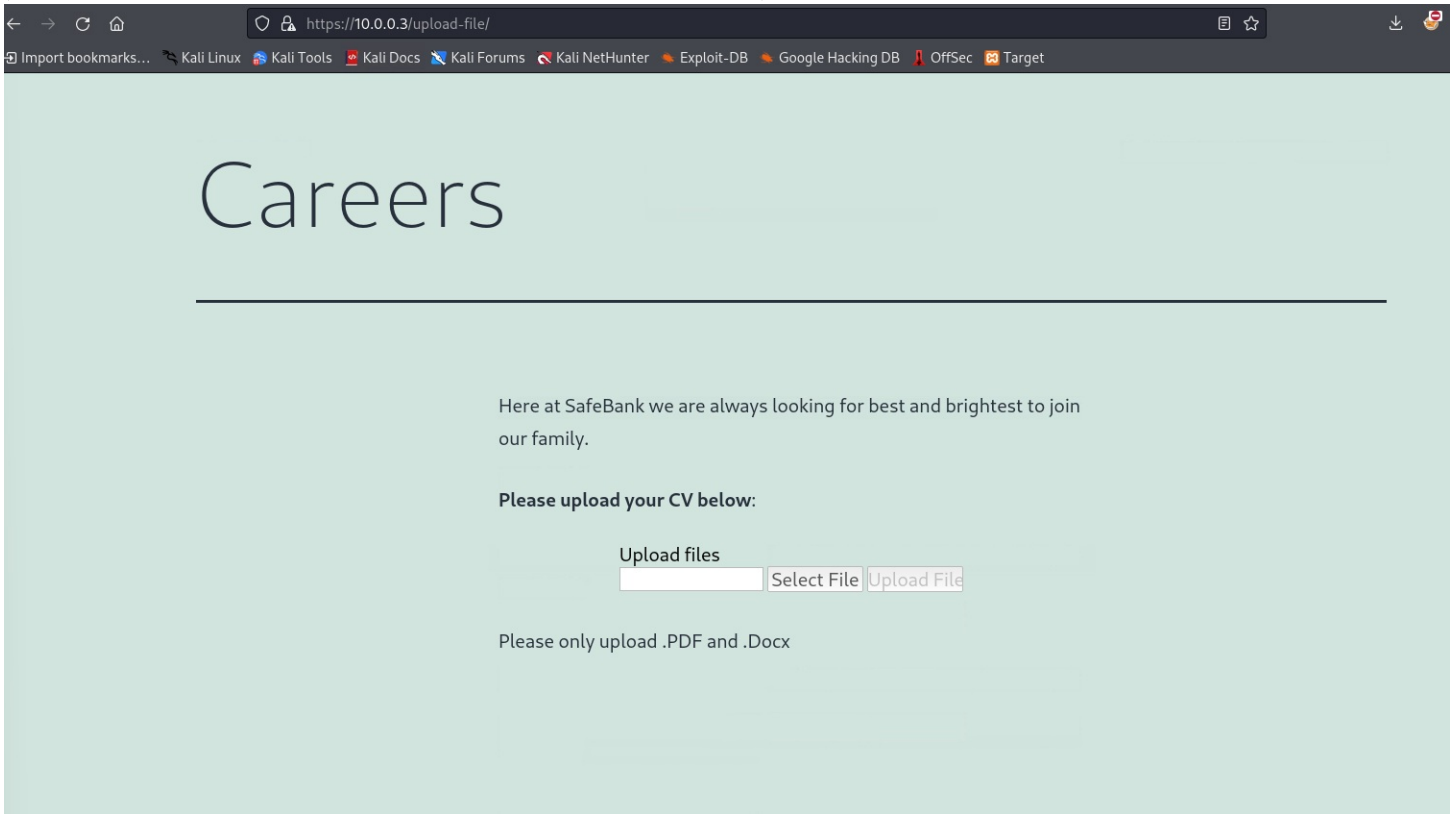
**Status:**
Open (Waiting for review)

**Description:**
CWE-434 describes: “Unrestricted Upload of File with Dangerous Type” a system with this weakness may authenticate the upload function but fail to verify or restrict the file to the type intended by the software developer. For example uploading a malware executable instead of a picture file to a photograph sharing website.

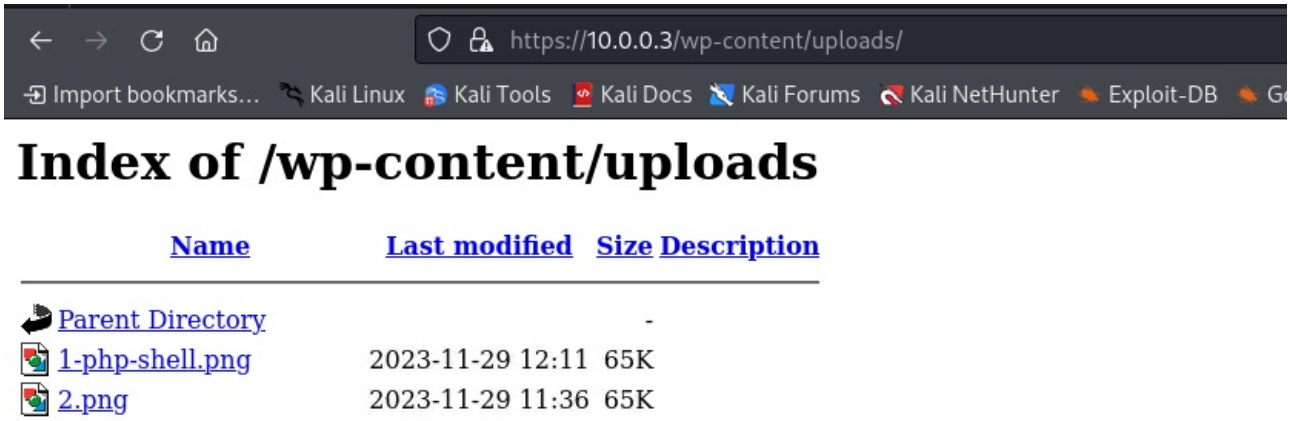
**Proof of Concept:**
It was possible to upload PDF and docx files to the webserver. This could be combined with the "Directory Listings Enabled" vulnerability to identify previously uploaded files.

The consultant found that a user would also download and open the uploaded file as it was from a CV submission page. The user would open the document and could execute the macro embedded wtihin it. This could result in the user being infected with malware and resulting in compromising the internal network by giving the attacker command execution on the user's machine.

**Attachment: chrome_ieMou3XtS1.png** (Size: 89312 bytes)
(SHA256 File Checksum: fa2f640c222759a603aa54093ca805efbe67d7c7e677c8a61e932f5787fcb69d)




**Attachment: chrome_ntBPX4NWFl.png** (Size: 158552 bytes)
(SHA256 File Checksum: 8bdcc9134408c0e24f2f22c4499d0345ec265f35f89b988ab4cfa352d85ef958)



| | | |
|---|---|---|
| 3_0.png | 2023-11-29 11:35 | 61K |
| 2022/ | 2022-08-11 04:58 | - |
| 2023/ | 2023-11-03 04:43 | - |
| 2024/ | 2024-02-15 08:59 | - |
| Test-php00.pdf | 2023-11-29 10:56 | 40 |
| Test.doc | 2023-11-17 14:21 | 20 |
| forminator/ | 2023-06-21 08:31 | - |
| s-php.pdf | 2023-11-17 14:31 | 39 |
| s-php00.pdf | 2023-11-17 14:30 | 39 |
| s-php2500.pdf | 2023-11-17 14:29 | 39 |
| s-php_-pdf-php_.pdf | 2023-11-17 14:30 | 39 |
| s-php_.pdf | 2023-11-17 14:28 | 39 |
| s.pdf | 2023-11-17 14:43 | 78 |
| s_php.pdf | 2023-11-17 14:31 | 39 |
| shell.doc | 2024-02-15 09:22 | 2.2K |
| smashing-the-state-m..> | 2023-11-29 11:30 | 65K |
| test-php_.pdf | 2023-11-29 11:09 | 328 |
| test-php_.png | 2023-11-29 12:08 | 5 |
| test-phptest.pdf | 2023-11-17 14:35 | 39 |
| test.pdf | 2023-11-29 11:16 | 39 |
| test1234-php_.doc | 2023-11-17 15:02 | 39 |
| ~$270722Test.doc | 2022-07-27 08:55 | 162 |
| ~$CV.doc | 2022-07-27 05:08 | 162 |

*Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.1.6 Server at 10.0.0.3 Port 443*

**References:**
https://cwe.mitre.org/data/definitions/434.html
https://www.owasp.org/index.php/Unrestricted_File_Upload

## Medium Weak Hashing Algorithm

**Status:**
Open (Waiting for review)

**Description:**
The service was found to have used an SSL certificate chain that had been signed using a cryptographically weak hashing algorithm (e.g., MD2, MD4, MD5, or SHA1). These signature algorithms are known to be vulnerable to collision attacks. An attacker can exploit this to generate another certificate with the same digital signature, allowing an attacker to masquerade as the affected service. Note that all SSL certificate chains signed with SHA-1 that expire after January 1st, 2017, are considered vulnerable. This is in accordance with Google's gradual sunsetting of the SHA-1 cryptographic hash algorithm.

**Proof of Concept:**
The screenshot shows the certificate was signed using SHA-1.
**Attachment: hashing algo.png** (Size: 14294 bytes)
(SHA256 File Checksum: 147a668e00a666eb6208f14829f3a3a8257f59577defb55e078cebf25f8f9799)

```
Signature Algorithm        SHA1 with RSA -- besides: users will receive a strong browser WARNING
Server key size            RSA 1024 bits (exponent is 65537)
```

**References:**
https://tools.ietf.org/html/rfc3279
https://docs.microsoft.com/en-us/security-updates/SecurityAdvisories/2008/961509
https://security.googleblog.com/2014/09/gradually-sunsetting-sha-1.html
https://shattered.io/

https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

## Low TLS 1.0 / 1.1 Protocol Detected

**Status:**
Open (Waiting for review)

**Description:**
The remote service accepts connections encrypted using TLS 1.0. TLS 1.0 has a number of cryptographic design flaws. Modern implementations of TLS 1.0 mitigate these problems, but newer versions of TLS like 1.1 and 1.2 are designed against these flaws and should be used whenever possible. PCI DSS v3.2 requires that TLS 1.0 be disabled entirely by June 30, 2018, except for POS POI terminals (and the SSL/TLS termination points to which they connect) that can be verified as not being susceptible to any known exploits.

**Proof of Concept:**
The screenshot shows the ciphers that were accepted by TLSv1.0 and TLSv1.1.
**Attachment: chrome_qMSI68DFks.png** (Size: 379174 bytes)
(SHA256 File Checksum: d82e54fa9a7050a11cd618790db0afef5710930991f7d12be261dc2eca286574)

```
Preferred TLSv1.1  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  256 bits  DHE-RSA-AES256-SHA            DHE 1024 bits
Accepted  TLSv1.1  256 bits  DHE-RSA-CAMELLIA256-SHA       DHE 1024 bits
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  128 bits  DHE-RSA-AES128-SHA            DHE 1024 bits
Accepted  TLSv1.1  128 bits  DHE-RSA-CAMELLIA128-SHA       DHE 1024 bits
Accepted  TLSv1.1  256 bits  AES256-SHA
Accepted  TLSv1.1  256 bits  CAMELLIA256-SHA
Accepted  TLSv1.1  128 bits  AES128-SHA
Accepted  TLSv1.1  128 bits  CAMELLIA128-SHA
Accepted  TLSv1.1  128 bits  TLS_DHE_RSA_WITH_SEED_CBC_SHA
Accepted  TLSv1.1  128 bits  TLS_RSA_WITH_SEED_CBC_SHA
Accepted  TLSv1.1  128 bits  TLS_RSA_WITH_IDEA_CBC_SHA
Preferred TLSv1.0  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.0  256 bits  DHE-RSA-AES256-SHA            DHE 1024 bits
Accepted  TLSv1.0  256 bits  DHE-RSA-CAMELLIA256-SHA       DHE 1024 bits
Accepted  TLSv1.0  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.0  128 bits  DHE-RSA-AES128-SHA            DHE 1024 bits
Accepted  TLSv1.0  128 bits  DHE-RSA-CAMELLIA128-SHA       DHE 1024 bits
Accepted  TLSv1.0  256 bits  AES256-SHA
Accepted  TLSv1.0  256 bits  CAMELLIA256-SHA
Accepted  TLSv1.0  128 bits  AES128-SHA
Accepted  TLSv1.0  128 bits  CAMELLIA128-SHA
Accepted  TLSv1.0  128 bits  TLS_DHE_RSA_WITH_SEED_CBC_SHA
Accepted  TLSv1.0  128 bits  TLS_RSA_WITH_SEED_CBC_SHA
Accepted  TLSv1.0  128 bits  TLS_RSA_WITH_IDEA_CBC_SHA
```

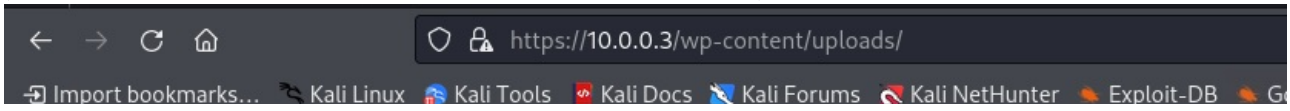
**References:**
https://www.tenable.com/plugins/nessus/104743

## Low Directory listing

**Status:**
Open (Waiting for review)

**Description:**
Web servers can be configured to automatically list the contents of directories that do not have an index page present. This can aid an attacker by enabling them to quickly identify the resources at a given path, and proceed directly to analyzing and attacking those resources. It particularly increases the exposure of sensitive files within the directory that are not intended to be accessible to users, such as temporary files and crash dumps. Directory listings themselves do not necessarily constitute a security vulnerability. Any sensitive resources within the web root should in any case be properly access-controlled, and should not be accessible by an unauthorized party who happens to know or guess the URL. Even when directory listings are disabled, an attacker may guess the location of sensitive files using automated tools.

**Proof of Concept:**
The screenshot shows that directory listings were enabled.
**Attachment: chrome_ntBPX4NWFl.png** (Size: 158552 bytes)
(SHA256 File Checksum: 8bdcc9134408c0e24f2f22c4499d0345ec265f35f89b988ab4cfa352d85ef958)



# Index of /wp-content/uploads

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| 1-php-shell.png | 2023-11-29 12:11 | 65K | |
| 2.png | 2023-11-29 11:36 | 65K | |
| 3_0.png | 2023-11-29 11:35 | 61K | |
| 2022/ | 2022-08-11 04:58 | - | |
| 2023/ | 2023-11-03 04:43 | - | |
| 2024/ | 2024-02-15 08:59 | - | |
| Test-php00.pdf | 2023-11-29 10:56 | 40 | |
| Test.doc | 2023-11-17 14:21 | 20 | |
| forminator/ | 2023-06-21 08:31 | - | |
| s-php.pdf | 2023-11-17 14:31 | 39 | |
| s-php00.pdf | 2023-11-17 14:30 | 39 | |
| s-php2500.pdf | 2023-11-17 14:29 | 39 | |
| s-php_-pdf-php_.pdf | 2023-11-17 14:30 | 39 | |
| s-php_.pdf | 2023-11-17 14:28 | 39 | |
| s.pdf | 2023-11-17 14:43 | 78 | |
| s_php.pdf | 2023-11-17 14:31 | 39 | |
| shell.doc | 2024-02-15 09:22 | 2.2K | |
| smashing-the-state-m..> | 2023-11-29 11:30 | 65K | |
| test-php_.pdf | 2023-11-29 11:09 | 328 | |
| test-php_.png | 2023-11-29 12:08 | 5 | |
| test-phptest.pdf | 2023-11-17 14:35 | 39 | |
| test.pdf | 2023-11-29 11:16 | 39 | |
| test1234-php_.doc | 2023-11-17 15:02 | 39 | |
| ~$270722Test.doc | 2022-07-27 08:55 | 162 | |
| ~$CV.doc | 2022-07-27 05:08 | 162 | |

*Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.1.6 Server at 10.0.0.3 Port 443*

**References:**
https://cwe.mitre.org/data/definitions/538.html
https://cwe.mitre.org/data/definitions/548.html

## Low LUCKY13

**Status:**
Open (Waiting for review)

**Description:**
Several outdated CBC ciphers were found to be in use, potentially exposing the application to TLS attacks such as LUCKY13. While these can be difficult to exploit, it is still recommended that these ciphers are removed or disabled.

**Proof of Concept:**
The screenshot shows the list of ciphers, specifically including the CBC ciphers that were accepted causing LUCKY13.
**Attachment: chrome_qMSI68DFks.png** (Size: 379174 bytes)
(SHA256 File Checksum: d82e54fa9a7050a11cd618790db0afef5710930991f7d12be261dc2eca286574)

```
Preferred TLSv1.1  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.1  256 bits  DHE-RSA-AES256-SHA            DHE 1024 bits
Accepted  TLSv1.1  256 bits  DHE-RSA-CAMELLIA256-SHA       DHE 1024 bits
Accepted  TLSv1.1  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
```

```
Accepted  TLSv1.1  128 bits  DHE-RSA-AES128-SHA            DHE 1024 bits
Accepted  TLSv1.1  128 bits  DHE-RSA-CAMELLIA128-SHA       DHE 1024 bits
Accepted  TLSv1.1  256 bits  AES256-SHA
Accepted  TLSv1.1  256 bits  CAMELLIA256-SHA
Accepted  TLSv1.1  128 bits  AES128-SHA
Accepted  TLSv1.1  128 bits  CAMELLIA128-SHA
Accepted  TLSv1.1  128 bits  TLS_DHE_RSA_WITH_SEED_CBC_SHA
Accepted  TLSv1.1  128 bits  TLS_RSA_WITH_SEED_CBC_SHA
Accepted  TLSv1.1  128 bits  TLS_RSA_WITH_IDEA_CBC_SHA
Preferred TLSv1.0  256 bits  ECDHE-RSA-AES256-SHA          Curve 25519 DHE 253
Accepted  TLSv1.0  256 bits  DHE-RSA-AES256-SHA            DHE 1024 bits
Accepted  TLSv1.0  256 bits  DHE-RSA-CAMELLIA256-SHA       DHE 1024 bits
Accepted  TLSv1.0  128 bits  ECDHE-RSA-AES128-SHA          Curve 25519 DHE 253
Accepted  TLSv1.0  128 bits  DHE-RSA-AES128-SHA            DHE 1024 bits
Accepted  TLSv1.0  128 bits  DHE-RSA-CAMELLIA128-SHA       DHE 1024 bits
Accepted  TLSv1.0  256 bits  AES256-SHA
Accepted  TLSv1.0  256 bits  CAMELLIA256-SHA
Accepted  TLSv1.0  128 bits  AES128-SHA
Accepted  TLSv1.0  128 bits  CAMELLIA128-SHA
Accepted  TLSv1.0  128 bits  TLS_DHE_RSA_WITH_SEED_CBC_SHA
Accepted  TLSv1.0  128 bits  TLS_RSA_WITH_SEED_CBC_SHA
Accepted  TLSv1.0  128 bits  TLS_RSA_WITH_IDEA_CBC_SHA
```

**References:**
http://www.isg.rhul.ac.uk/tls/Lucky13.html
https://docs.secureauth.com/pages/viewpage.action?pageId=14778519

## Low BEAST

**Status:**
Open (Waiting for review)

**Description:**
Several outdated CBC ciphers were found to be in use, potentially exposing the application to TLS attacks such as BEAST. While these can be difficult to exploit, it is still recommended that these ciphers are removed or disabled.

**Proof of Concept:**
The screenshot shows the list of ciphers, specifically including the CBC ciphers that were accepted causing BEAST.
**Attachment: BEAST.png** (Size: 41730 bytes)
(SHA256 File Checksum: 108665921ab17b37bbfe173121f41c455022238d3e4878947d542f4ce6142423)

```
BEAST (CVE-2011-3389)                    TLS1: ECDHE-RSA-AES256-SHA DHE-RSA-AES256-SHA DHE-RSA-CAMELLIA256-SHA ECDHE-RSA-AES128-SHA DHE-RSA-AES128-SHA DHE-RSA-CAMELLIA128-SHA AES256-SHA CAMELLIA256-SHA
                                                 AES128-SHA CAMELLIA128-SHA DHE-RSA-SEED-SHA SEED-SHA IDEA-CBC-SHA
```

**References:**
https://docs.secureauth.com/pages/viewpage.action?pageId=14778519

## Low Weak Diffie-Hellman - Logjam Attack

**Status:**
Open (Waiting for review)

**Description:**
Diffie-Hellman key exchange is a popular cryptographic algorithm that allows Internet protocols to agree on a shared key and negotiate a secure connection. It is fundamental to many protocols including HTTPS, SSH, IPsec, SMTPS, and protocols that rely on TLS. Researchers have uncovered several weaknesses in how Diffie-Hellman key exchange has been deployed: The Logjam attack allows a man-in-the-middle attacker to downgrade vulnerable TLS connections to 512-bit export-grade cryptography. This allows the attacker to read and modify any data passed over the connection. The attack is reminiscent of the freak but is due to a flaw in the TLS protocol rather than an implementation vulnerability and attacks a Diffie-Hellman key exchange rather than an RSA key exchange. The attack affects any server that supports DHE Export ciphers and affects all modern web browsers.

**Proof of Concept:**
The screenshot shows that the server was affected by LOGJAM.
**Attachment: logjam.png** (Size: 9194 bytes)
(SHA256 File Checksum: 4f582894777900a53d68835b3dc2367592c10919fb41866827c44af52602c329)

```
LUCKY13 (CVE-2013-0169), experimental      potentially VULNERABLE, uses cipher block chaining (CBC) ciphers with TLS. Check patches
```

**References:**
https://security.stackexchange.com/questions/89773/how-to-check-if-a-server-is-not-vulnerable-to-logjam

## Low OCSP Stapling Not Configured

**Status:**
Open (Waiting for review)

**Description:**
Although encrypted web traffic protects users, it requires additional steps to verify integrity and confidentiality. One of these steps is determining whether the SSL certificate used to secure a website is still valid, and if it has expired or been revoked. The only way to verify the current status of a certificate is to ask its vendor. Since the vendor maintains key information about the certificate, including expiration date and any actions performed on the certificate, browsers ensure user safety by requesting certificate information from the vendor instead of from the web server. The Online Certificate Status Protocol (OCSP) is a way for a web browser to determine the validity of an SSL certificate by verifying with the vendor of the certificate. While OCSP improves security, it causes websites to load slower since the browser has to communicate with the web server and the vendor. With OCSP stapling, the web server regularly downloads a cryptographically verifiable time-limited copy of the vendor's response, which it can deliver directly to the browser. Because the browser can verify the validity of this response without contacting the certificate vendor, the outbound connectivity overheads for the client are reduced and site performance and reliability is increased.

**Proof of Concept:**
The web server was found to not implement OCSP stapling in server responses.
**Attachment: chrome_09ArNTaTMS.png** (Size: 10179 bytes)
(SHA256 File Checksum: ece632c87e5eac67e6254b02e0db074c941f0d777394f54006684858cbfc01de)

```
NOT ok -- neither CRL nor OCSP URI provided
OCSP stapling                  not offered
OCSP must staple extension     --
```

## Low Web Server HTTP Header Information Disclosure

**Status:**
Open (Waiting for review)

**Description:**
By analysing the HTTP headers transferred between the server and the client, it was possible to identify the technology the application was using. This information can aid attackers and worms searching the internet for vulnerable technology should an exploit be discovered, and try more specific attacks on the server, reducing the probability of their detection and increasing their chances of success.

**Proof of Concept:**
The screenshot shows that it was possible to gather the following information from the HTTP headers returned by the web server
**Attachment: chrome_sbLGdXA1Ax.png** (Size: 28742 bytes)
(SHA256 File Checksum: 95b80fcd76c11f8e282c47a78d992a29be1d2675e051dfa92a2b1a5bf115af7b)

```
Strict Transport Security      not offered
Public Key Pinning             --
Server banner                  Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.1.6
Application banner             X-Powered-By: PHP/8.1.6
Cookie(s)                      1 issued: NOT secure, NOT HttpOnly
Security headers               --
Reverse Proxy banner           --
```

## Low [HSTS] Strict transport security not enforced

**Status:**
Open (Waiting for review)

**Description:**

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process. To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic.This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

**Proof of Concept:**

The screenshot shows the lack of HTTP Strict Transport Security.

**Attachment: chrome_sbLGdXA1Ax.png** (Size: 28742 bytes)

(SHA256 File Checksum: 95b80fcd76c11f8e282c47a78d992a29be1d2675e051dfa92a2b1a5bf115af7b)

**References:**

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
https://moxie.org/software/sslstrip/
https://hstspreload.org/
https://cwe.mitre.org/data/definitions/523.html

# Report authors

Richard Clifford (Mantis)

— E-Mail: rclifford@cybershade.org
— Social: twitter.com/MantisSTS

# Report summary comment

Mantis was engaged by W4KEY to perform security testing of the internal infrastructure. Testing was completed by Mantis on {DATE} by Mantis. The assessment was performed over 3 days with the aim of identifying security risks and vulnerabilities present within the internal infrastructure and associated server infrastructure that could negatively impact on the confidentiality, availability or integrity of the solution as well as W4KEY's network, the business data contained, and its users. This document provides identified vulnerabilities acspnd recommendations to help mitigate risk and to ensure future information security and assurance.

Our high-level conclusion is supported by several risks identified against the business. Decisions on the treatment of the risks identified should be taken in accordance with the advice presented in this section, and formally recorded and approved by an appropriate level of management.

Key Findings:

Discovery Phase:
- Identified live IP addresses and services running on them, including a webserver on TCP ports 80 and 443.
- Discovered a company website on port 80 and a WordPress website (SafeBank) on port 443.

Company Website (Port 80):
- Found to be static with limited functionality.
- Exposed employee information such as full names, posing a potential privacy risk.

WordPress Website (Port 443 - SafeBank):
- Detected dangerous functionality allowing file uploads to a public location.
- Observed an outdated contact form plugin vulnerable to remote code execution.
- Attempts to exploit vulnerabilities were unsuccessful, although a file upload vulnerability was exploited successfully.

Endpoint Security:
- Encountered challenges due to the presence of robust antivirus software, impeding further exploitation attempts.

During the initial reconnaissance phase, Mantis discovered live IP addresses and the services running on them. Notably, a webserver was found operating on TCP ports 80 and 443, hosting a company website on port 80 and a WordPress website (SafeBank) on port 443.

The company website, hosted on port 80, was determined to be static with minimal functionality. However, it exposed certain employee information, such as full names, which could pose a potential privacy risk if exploited.

The WordPress website (SafeBank) on port 443 exhibited concerning vulnerabilities, including the ability to upload files to a public location and an outdated contact form plugin susceptible to remote code execution. Although attempts to exploit these vulnerabilities were unsuccessful, Mantis was able to exploit a file upload vulnerability by uploading a .doc file, which could potentially be used to deliver malicious payloads via macros.

Endpoint security presented challenges due to robust antivirus software installed on the testing machine, hindering further exploitation attempts and highlighting the importance of robust endpoint protection measures.

There was a number of medium and low risks also identified and detailed throughout this report.

# Changelog

| Date | Description |
|---|---|
| 15/02/2024, 19:38:35 | Create report: "W4KEY Lab Penetration Test". |
| 15/02/2024, 19:39:48 | Create issue: File Upload Vulnerability |
| 15/02/2024, 19:41:04 | Create issue: TLS 1.0 / 1.1 Protocol Detected |

| | |
|---|---|
| 15/02/2024, 19:41:11 | Create issue: Directory listing |
| 15/02/2024, 19:42:27 | Save report v.1 |
| 15/02/2024, 19:42:54 | Save report v.2 |
| 15/02/2024, 19:43:19 | Save report v.3 |
| 15/02/2024, 19:44:30 | Save report v.4 |
| 15/02/2024, 19:47:44 | Save report v.5 |
| 15/02/2024, 19:48:11 | Save report v.6 |
| 15/02/2024, 19:49:41 | Save report v.7 |
| 15/02/2024, 19:50:49 | Create issue: LUCKY13 |
| 15/02/2024, 19:50:58 | Create issue: BEAST |
| 15/02/2024, 19:51:04 | Create issue: LOGJAM |
| 15/02/2024, 19:51:45 | Create issue: OCSP Stapling Not Enabled |
| 15/02/2024, 19:51:57 | Create issue: Information Leaked in HTTP Headers |
| 15/02/2024, 19:52:10 | Create issue: Weak Hashing Algorithm |
| 15/02/2024, 19:52:49 | Create issue: [HSTS] Strict transport security not enforced |
| 15/02/2024, 19:52:54 | Save report v.8 |
| 15/02/2024, 19:53:48 | Save report v.9 |
| 15/02/2024, 19:57:47 | Save report v.10 |
| 15/02/2024, 19:58:14 | Save report v.11 |
| 15/02/2024, 19:59:34 | Save report v.12 |
| 15/02/2024, 20:01:34 | Save report v.13 |
| 15/02/2024, 20:03:20 | Save report v.14 |
| 15/02/2024, 20:12:28 | Save report v.15 |
| 15/02/2024, 20:15:26 | Save report v.16 |
| 15/02/2024, 20:16:03 | Save report v.17 |
| 15/02/2024, 20:16:25 | Save report v.18 |
| 15/02/2024, 20:17:21 | Save report v.19 |
| 15/02/2024, 20:18:03 | Save report v.20 |

Generated by VULNREPO