Source Code Audit on OpenSearch
for Open Source Technology Improvement Fund (OSTIF)

Final Report and Management Summary

2023-09-18

PUBLIC

X41 D-SEC GmbH
Krefelder Str. 123
D-52070 Aachen
Amtsgericht Aachen: HRB19989

https://x41-dsec.de/
info@x41-dsec.de

Organized by the Open Source Technology Improvement Fund

Revision	*Date*	*Change*	*Author(s)*
1	2023-08-10	Final Draft Report	J. M.
2	2023-08-18	Final Report and Management Summary	L. Gommans, L. Rudman
3	2023-09-18	Public Release	L. Gommans

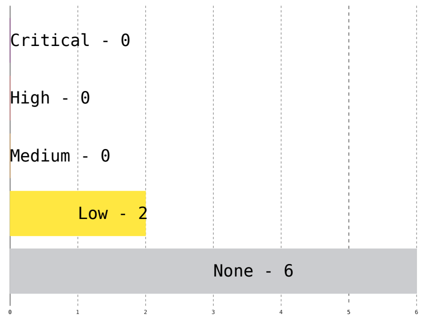
Contents

Dashboard

Target	
Customer	Open Source Technology Improvement Fund (OSTIF)
Name	OpenSearch
Type	Search engine software
Version	2.8.0

Engagement	
Type	Source Code Audit
Consultants	4: L. Rudman, J. M., Luc Gommans, and Niklas Abel
Engagement Effort	42 person-days, 2023-06-27 to 2023-08-10

Total issues found	2



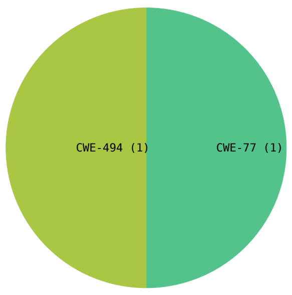


Figure 1: Issue Overview (l: Severity, r: CWE Distribution)

1 Executive Summary

In June 2023, X41 D-Sec GmbH performed a source code audit against OpenSearch to identify vulnerabilities and weaknesses in the source code. The test was organized by the Open Source Technology Improvement Fund (OSTIF)[1].

Two vulnerabilities were discovered during the test by X41, both of which classified as having a low severity. Additionally, six issues without a direct security impact were identified.

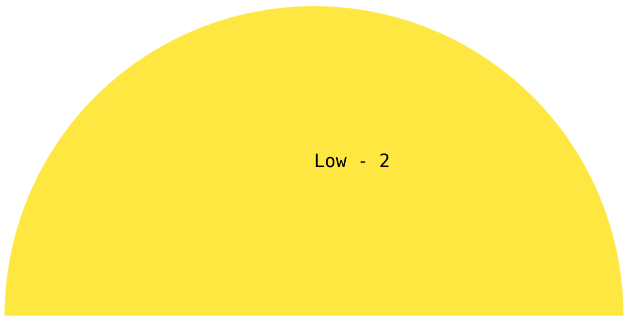


Figure 1.1: Issues and Severity

OpenSearch is an open-source search and analytics suite. Use cases include real-time application monitoring, log analytics, and website search[2]. Vulnerabilities in the software could allow an attacker to obtain sensitive information, or cause unreasonably high resource usage with subsequent costs.

[1] `https://ostif.org`
[2] `https://aws.amazon.com/what-is/opensearch/`

In a source code audit, all information about the system is made available. The test was performed by four experienced security experts between 2023-06-27 and 2023-08-10.

The first discovered issue pertains to including a plugin name in a command to be executed. A malicious plugin name would thus lead to an attacker being able to gain full control of the system. However, the code currently only uses this vulnerable installation method for a fixed list of plugins. The attacker would have to modify the file from which the list is read.

The second issue is that unofficial plugins are installed from an untrusted source without verification. The source, Maven's Central Repository, documents that signature checking needs to be employed to guarantee that users are downloading the original artifact. The integrity checking that is in place is not adequate to satisfy this requirement.

In addition, the testers noted several insecure default values and a mismatch between authenticated and authorized users in cross-cluster operations.

X41 recommends to continue to follow and apply good coding practices throughout the project as well as removing insecure defaults. Due to the widespread use of OpenSearch, it is encouraged to perform recurring security audits, as new vulnerabilities may be introduced when more features are added.

As we conclude the code audit for OpenSearch, it is important to highlight the good condition that the code is in compared to projects of similar complexity. Following the examination of the code by multiple testers, it is remarkable that very few areas of concern were identified.

2 Introduction

The assessment comprised a security review of OpenSearch[1], utilizing static source code analysis on version 2.8.0. OpenSearch is a search engine which provides full-text search and is considered sensitive as it may process sensitive data depending on the use. From an attacker's perspective, they could attempt to gain access to this data, or the infrastructure systems running OpenSearch, or try to remove log files.

2.1 Methodology

The main objective of this security assessment was the identification of vulnerabilities within the OpenSearch core components by conducting a source code review as well as well as static code analysis.

A manual approach for penetration tests and for code reviews is used by X41. This process is supported by tools such as static code analyzers[2][3][4] and industry standard web application security tools[5].

X41 adheres to established standards for source code reviewing and penetration testing. These are in particular the *CERT Secure Coding*[6] standards and the *Study - A Penetration Testing Model*[7] of the German Federal Office for Information Security.

The workflow of source code reviews is shown in figure 2.1. In an initial, informal workshop regarding the design and architecture of the application a basic threat model is created. This is used to explore the source code for interesting attack surface and code paths. These are then

[1] `https://github.com/opensearch-project/OpenSearch/releases/tag/2.8.0`
[2] `https://github.com/returntocorp/semgrep`
[3] `https://github.com/aquasecurity/trivy`
[4] `https://github.com/ZupIT/horusec`
[5] `https://portswigger.net/burp`
[6] `https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards`
[7] `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1`

audited manually and with the help of tools such as static analyzers and fuzzers. The identified issues are documented and can be used in a GAP analysis to highlight changes to previous audits.

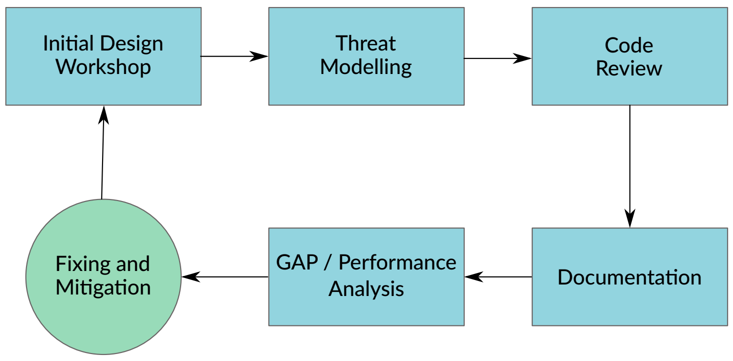


Figure 2.1: Code Review Methodology

2.2 Findings Overview

DESCRIPTION	SEVERITY	ID	REF
Missing Shell Escaping	LOW	OPNSRCH-PT-23-01	4.1.1
Plugins Downloaded from Untrusted Source	LOW	OPNSRCH-PT-23-02	4.1.2
Default Credentials	NONE	OPNSRCH-PT-23-100	4.2.1
SHA-1 Deprecation	NONE	OPNSRCH-PT-23-101	4.2.2
Injectable Coding Patterns in Benchmark Code	NONE	OPNSRCH-PT-23-102	4.2.3
Insecure Defaults	NONE	OPNSRCH-PT-23-103	4.2.4
Password Complexity Requirements	NONE	OPNSRCH-PT-23-104	4.2.5
Mismatch Between Authenticated and Authorized User	NONE	OPNSRCH-PT-23-105	4.2.6

Table 2.1: Security-Relevant Findings

2.3 Scope

During a kickoff call, the testing team, in collaboration with OSTIF and the maintainers, defined and narrowed down the scope of the testing efforts. It was mutually agreed that the primary focus would be on conducting in-depth code analysis of the OpenSearch core components and plugins which are installed by default. The review was pinned to release 2.8.0 of the public GitHub repository `https://github.com/opensearch-project/OpenSearch/releases/tag/2.8.0`.

2.4 Coverage

A security assessment attempts to find the most important or sometimes as many of the existing problems as possible, though it is practically never possible to rule out the possibility of additional weaknesses being found in the future.

The OpenSearchcode was statically analyzed using various tools, including the following:

- semgrep
- trivy
- horusec

In addition, manual code review was conducted where X41 investigated authentication mechanisms including certificate authentication, sources of entropy, data parsing methods, communication between nodes and clusters (e.g., cross-cluster search), the libraries in use, the Docker setup, and setup instructions and recommendations in general. Known vulnerabilities in Elasticsearch were investigated to ensure that any necessary fixes are also applied in OpenSearch.

Moreover, we looked for particular vulnerabilities including local privilege escalation, server-side request forgery, object deserialization, command injection, cross-site scripting and cross-site request forgery. X41 also set up a test environment through Docker and tested the APIs[8].

The time allocated to X41 for this assessment was sufficient to yield a reasonable coverage of the given scope.

Suggestions for next steps in securing this scope can be found in section 2.5.

2.5 Recommended Further Tests

X41 recommends to perform code audits of OpenSearch's external dependencies, as they were not in scope of this audit.

[8] Application Programming Interfaces

3 Rating Methodology for Security Vulnerabilities

Security vulnerabilities are given a purely technical rating by the testers as they are discovered during the test. Business factors and financial risks for Open Source Technology Improvement Fund (OSTIF) are beyond the scope of a penetration test which focuses entirely on technical factors. Yet technical results from a penetration test may be an integral part of a general risk assessment. A penetration test is based on a limited time frame and only covers vulnerabilities and security issues which have been found in the given time, there is no claim for full coverage.

In total, five different ratings exist, which are as follows:

Severity Rating
None
Low
Medium
High
Critical

A low rating indicates that the vulnerability is either very hard for an attacker to exploit due to special circumstances, or that the impact of exploitation is limited, whereas findings with a medium rating are more likely to be exploited or have a higher impact. High and critical ratings are assigned when the testers deem the prerequisites realistic or trivial and the impact significant or very significant.

Findings with the rating 'none' are called informational findings and are related to security hardening, affect functionality, or other topics that are not directly related to security. X41 recommends to mitigate these issues as well, because they often become exploitable in the future. Doing so will strengthen the security of the system and is recommended for defense in depth.

3.1 Common Weakness Enumeration

The CWE[1] is a set of software weaknesses that allows the categorization of vulnerabilities and weaknesses in software. If applicable, X41 provides the CWE-ID for each vulnerability that is discovered during a test.

CWE is a very powerful method to categorize a vulnerability and to give general descriptions and solution advice on recurring vulnerability types. CWE is developed by *MITRE*[2]. More information can be found on the CWE website at `https://cwe.mitre.org/`.

[1] Common Weakness Enumeration
[2] `https://www.mitre.org`

4 Results

This chapter describes the results of this test. The security-relevant findings are documented in Section 4.1. Additionally, findings without a direct security impact are documented in Section 4.2.

4.1 Findings

The following subsections describe findings with a direct security impact that were discovered during the test.

4.1.1 OPNSRCH-PT-23-01: Missing Shell Escaping

Severity:	LOW
CWE:	77 – Improper Neutralization of Special Elements used in a Command ('Command Injection')
Affected Component:	InstallPluginsTask.java

4.1.1.1 Description

The function ***getProcessBuilderBasedOnOS()*** executes a command containing the plugin name without any validation or escaping, as shown in listing 4.1. The plugin name cannot already be properly escaped because the target shell, and thus the escaping it would need, is only determined within the function. The exploitability of this issue is limited because the list of plugin names that are allowed to use this method is hardcoded into a file, `plugins.txt`, and distributed with the software.

```java
ProcessBuilder getProcessBuilderBasedOnOS(String plugin, TaskInput taskInput) {
    final String command = taskInput.getOpenSearchBin().resolve("opensearch-plugin") + "
    ↪  install " + plugin;
    final ProcessBuilder processBuilder = new ProcessBuilder();
    if (OS.WINDOWS == OS.current()) {
        processBuilder.command("cmd.exe", "/c", command);
    } else {
        processBuilder.command("sh", "-c", command);
    }
    return processBuilder;
}
```

Listing 4.1: getProcessBuilderBasedOnOS()

The content of the *plugin* variable is passed from *Environment* via *UpgradeCli*, ***accept()***, and ***executeInstallPluginCommand()***. There does not seem to be any validation, such as ensuring that the value only consists of letters, and also no documentation in any of these functions that indicate that the data being passed is expected to already be safe for executing.

4.1.1.2 Solution Advice

X41 recommends to escape parameters for the target shell before executing them as part of a command. If a caller, several class traversals removed, is expected to ensure the safety of the value, this should be part of function documentation. In addition, it is good practice to escape data where is being injected into a target language (in this case: a shell) to prevent (new) callers from mistakenly omitting this.

4.1.2 OPNSRCH-PT-23-02: Plugins Downloaded from Untrusted Source

Severity:	LOW
CWE:	494 – Download of Code Without Integrity Check
Affected Component:	InstallPluginCommand.java

4.1.2.1 Description

Unofficial plugins are downloaded from Maven's Central Repository as shown in listing 4.2. The documentation[1] mentions that, without signatures, data integrity is not guaranteed by the Central Repository:

> When people download artifacts from the Central Repository, they might want to verify these artifacts' PGP signatures against a public key server. If there are no signatures, then users have no guarantee that they are downloading the original artifact.

```java
/** Downloads the plugin and returns the file it was downloaded to. */
private Path download(Terminal terminal, String pluginId, Path tmpDir, boolean isBatch) throws
↪ Exception {

    if (OFFICIAL_PLUGINS.contains(pluginId)) {
        [...]
        return downloadAndValidate(terminal, url, tmpDir, true, isBatch);
    }

    // now try as maven coordinates, a valid URL would only have a colon and slash
    String[] coordinates = pluginId.split(":");
    if (coordinates.length == 3 && pluginId.contains("/") == false &&
    ↪ pluginId.startsWith("file:") == false) {
        String mavenUrl = getMavenUrl(terminal, coordinates, Platforms.PLATFORM_NAME);
        terminal.println("-> Downloading " + pluginId + " from maven central");
        return downloadAndValidate(terminal, mavenUrl, tmpDir, false, isBatch);
    }
```

Listing 4.2: Downloading from Maven Central Repository

[1] https://maven.apache.org/repository/guide-central-repository-upload.html

Downloads are verified by checking a hash sum as detailed in the informational note in section 4.2.2. Only for official plugins is PGP[2] verification performed. If an untrusted third party is able to overwrite an installed plugin in the repository, or can get an administrator to install a malicious plugin, they would gain full access to the system.

Although executable code is downloaded from a source that does explicitly not guarantee its contents without signature, the circumstances for exploitation involve either abusing an unknown vulnerability in the Central Repository or social engineering, leading to a low rating for this finding.

4.1.2.2 Solution Advice

X41 recommends to perform signature validation per Maven's recommendation. Users should supply the expected PGP key (fingerprint) as obtained from a trusted source such as the developer's official website, since Maven offers no mechanism for secure key distribution.

[2] Pretty Good Privacy

4.2 Informational Notes

The following observations do not have a direct security impact, but are related to security hardening, affect functionality, or other topics that are not directly related to security. X41 recommends to mitigate these issues as well, because they often become exploitable in the future. Doing so will strengthen the security of the system and is recommended for defense in depth.

4.2.1 OPNSRCH-PT-23-100: Default Credentials

Affected Component:	OpenSearchRealm.java, internal_users.yml

4.2.1.1 Description

OpenSearch uses the default credentials `admin`:`admin`, among others, and does not require nor ask for changing the password when logging in. Users may forget to change the password, which could allow an attacker to gain access to the system.

4.2.1.2 Solution Advice

X41 recommends to require an initial password to be provided via environment variables, or fall back to generating and printing a secure password the first time OpenSearch runs. It should be noted that printing the password may result in it being logged and stored persistently. Alternatively, or in addition, a password change could be required before the account can be used. Otherwise, the authentication process can be changed from logging in with username and password, replacing it with the FIDO2 login standard.

4.2.2 OPNSRCH-PT-23-101: SHA-1 Deprecation

Affected Component:	InstallPluginCommand.java

4.2.2.1 Description

Plugins' SHA-1[3] hashes are verified as a fallback to SHA-512 as shown in listing 4.3. This hash function is known to have practical weaknesses[4]. However, the expected hash is downloaded from the same location as the data to be checked, making this not a security feature but rather one that protects from data corruption.

```java
private Path downloadAndValidate(
    final Terminal terminal,
    final String urlString,
    final Path tmpDir,
    final boolean officialPlugin,
    boolean isBatch
) throws IOException, PGPException, UserException {
    Path zip = downloadZip(terminal, urlString, tmpDir, isBatch);
    pathsToDeleteOnShutdown.add(zip);
    String checksumUrlString = urlString + ".sha512";
    URL checksumUrl = openUrl(checksumUrlString);
    String digestAlgo = "SHA-512";
    if (checksumUrl == null && officialPlugin == false) {
        // fallback to sha1, until 7.0, but with warning
        terminal.println(
            "Warning: sha512 not found, falling back to sha1. This behavior is deprecated and
            ↪ will be removed in a "
                + "future release. Please update the plugin to use a sha512 checksum."
        );
        checksumUrlString = urlString + ".sha1";
        checksumUrl = openUrl(checksumUrlString);
        digestAlgo = "SHA-1";
    }
```

Listing 4.3: downloadAndValidate()

[3] Secure Hashing Algorithm 1
[4] https://shattered.io/

The fallback was marked[5] as deprecated in 2017 with the plan to remove it in version 7.0 as per a source code comment, or in "a future release" per the information that is shown to the user. The original project has since released version 8.8.2, indicating that the fallback should have been removed, but the comment is still present[6].

4.2.2.2 Solution Advice

X41 advises to follow through with the deprecation because, also for OpenSearch, it appears that enough time has passed for developers to upgrade their plugins.

[5] `https://github.com/opensearch-project/OpenSearch/commit/5b711c283dbc233f80de5e0adb26723c22d678c7#diff-4aa75ce0cf2748f4941183e993ae79bbc0e3926e23c445dc95930d845bc6e105R376-R378`

[6] `https://github.com/elastic/elasticsearch/blob/v8.8.2/distribution/tools/plugin-cli/src/main/java/org/elasticsearch/plugins/cli/InstallPluginAction.java#L555`

4.2.3 OPNSRCH-PT-23-102: Injectable Coding Patterns in Benchmark Code

Affected Component:	client/benchmark/

4.2.3.1 Description

The code in the `client/benchmark` directory does not follow best coding practices, however we where unable to identify a location where it is currently in use. The last time a commit occurred in that directory (excluding repository-wide changes) appears to be in 2018, before the rename from Elasticsearch to OpenSearch.

In listing 4.4, an instance is shown where a URI[7] is built without correctly encoding special characters that are valid in a *String* but are not supposed to occur in a path name.

```java
private RestSearchRequestExecutor(RestClient client, String indexName) {
    this.client = client;
    this.endpoint = "/" + indexName + "/_noop_search";
}
```

Listing 4.4: Unescaped URI Concatenation in RestClientBenchmark.java

Listing 4.5 shows an excerpt of the README file, advising users to download data from the Elasticsearch S3[8] tenant via plain text HTTP[9]. An attacker could supply a large file that extracts to something well beyond the size of any storage medium, leading to a DoS[10] situation if the process is an automated deployment or test environment. However, as it appears the code is not commonly used, this is not considered a realistic risk. Bzip2 does not appear to have regular vulnerabilities: supplying malicious data also appears to be unlikely to be able to exploit, beyond DoS, a system running this code. Nevertheless, these risks are not necessary to accept by upgrading the README to use HTTPS[11] instead.

```
#### Bulk indexing

Download benchmark data and decompress them.

Example invocation:
```

[7] Uniform Resource Identifier
[8] Simple Storage Service
[9] HyperText Transfer Protocol
[10] Denial of Service
[11] HyperText Transfer Protocol Secure

```

```
wget http://benchmarks.elasticsearch.org.s3.amazonaws.com/corpora/geonames/documents-2.json.bz2
bzip2 -d documents-2.json.bz2
mv documents-2.json client/benchmark/build
gradlew -p client/benchmark run --args ' rest bulk localhost build/documents-2.json geonames
↪ 8647880 5000'
```
```

Listing 4.5: Plain HTTP Download in client/benchmark/README.md

A JSON[12] object is built by using the ***String.format()*** method, inserting a string without escaping, as shown in listing 4.6.

```java
RestBulkRequestExecutor(RestClient client, String index) {
    this.client = client;
    this.actionMetadata = String.format(Locale.ROOT, "{ \"index\" : { \"_index\" : \"%s\" } }%n",
    ↪  index);
}
```

Listing 4.6: JSON Crafting through String Concatenation in RestClientBenchmark.java

4.2.3.2 Solution Advice

X41 recommends to remove the code if no longer in use or to follow general security practices also in code only meant to be run on development systems.

[12] JavaScript Object Notation

4.2.4 OPNSRCH-PT-23-103: Insecure Defaults

Affected Component:	Docker container, Documentation

4.2.4.1 Description

The official Docker container *`opensearchproject/opensearch`*, as used by the `docker-compose.yml` from `https://opensearch.org/downloads.html` is distributed with demo certificates, using publicly available private keys. While this is noted in the "Security configuration" documentation, neither the "Download & Get Started" page, nor the "Installing OpenSearch" documentation point out this fact. This may easily be overlooked by administrators.

4.2.4.2 Solution Advice

X41 recommends to use secure defaults, such as not including the demo certificates, requiring the user to provide this. Alternatively, unique certificates may be generated when none are provided. Users should actively decide to use demo certificates, and they should be made clearly aware that this is an insecure configuration.

4.2.5 OPNSRCH-PT-23-104: Password Complexity Requirements

Affected Component:	Default configuration, Documentation

4.2.5.1 Description

The default configuration and example given in the documentation requires a password to "contain at least one uppercase letter, one lowercase letter, one digit, and one special character". This policy is unsuitable to determine password strength, poses an annoyance for users and may result in users following predictable patterns, such as using short words with a single digit and an exclamation mark appended, or using leetspeak.

4.2.5.2 Solution Advice

X41 recommends to remove the password complexity requirement. zxcvbn[13], which is already in use, is better suited to estimate password strength.

[13] `https://github.com/zxcvbn-ts/zxcvbn`

4.2.6 OPNSRCH-PT-23-105: Mismatch Between Authenticated and Authorized User

Affected Component:	Cross-Cluster Search

4.2.6.1 Description

When using cross-cluster search, the user-facing cluster handles authentication, whereas the search cluster handles authorization. The search cluster does not authenticate the user itself and trusts the user-facing cluster to have authenticated the user.

The authenticated user and the authorized user are only matched via their username, and they may be in completely different user databases, using different credentials.

It is possible that a user `Alice` one one cluster refers to a different person `Alice` on the other cluster.

This also allows the requesting cluster to pose as any given user on the search cluster without authenticating as that user. This is rated an informational note because the clusters perform TLS[14] authentication with a shared CA[15] and in general seem to be designed to have full trust between each other. If a cluster was not trusted for full access, the informational note would become a finding as the impact would change to be more severe.

4.2.6.2 Solution Advice

X41 recommends to introduce a user realm in the cluster-to-cluster communication, ensuring that both clusters authenticate and authorize against the same user database.

[14] Transport Layer Security
[15] Certificate Authority

5 About X41 D-Sec GmbH

X41 D-Sec GmbH is an expert provider for application security and penetration testing services. Having extensive industry experience and expertise in the area of information security, a strong core security team of world-class security experts enables X41 D-Sec GmbH to perform premium security services.

X41 has the following references that show their experience in the field:

- Source code audit of the Git source code version control system[1]
- Review of the Mozilla Firefox updater[2]
- X41 Browser Security White Paper[3]
- Review of Cryptographic Protocols (Wire)[4]
- Identification of flaws in Fax Machines[5,6]
- Smartcard Stack Fuzzing[7]

The testers at X41 have extensive experience with penetration testing and red teaming exercises in complex environments. This includes enterprise environments with thousands of users and vendor infrastructures such as the Mozilla Firefox Updater (Balrog).

Fields of expertise in the area of application security encompass security-centered code reviews, binary reverse-engineering and vulnerability-discovery. Custom research and IT security consulting, as well as support services, are the core competencies of X41. The team has a strong technical background and performs security reviews of complex and high-profile applications such as Google Chrome and Microsoft Edge web browsers.

X41 D-Sec GmbH can be reached via `https://x41-dsec.de` or `mailto:info@x41-dsec.de`.

[1] https://x41-dsec.de/security/research/news/2023/01/17/git-security-audit-ostif/
[2] https://blog.mozilla.org/security/2018/10/09/trusting-the-delivery-of-firefox-updates/
[3] https://browser-security.x41-dsec.de/X41-Browser-Security-White-Paper.pdf
[4] https://www.x41-dsec.de/reports/Kudelski-X41-Wire-Report-phase1-20170208.pdf
[5] https://www.x41-dsec.de/lab/blog/fax/
[6] https://2018.zeronights.ru/en/reports/zero-fax-given/
[7] https://www.x41-dsec.de/lab/blog/smartcards/

Acronyms

API	Application Programming Interface	9
CA	Certificate Authority	23
CWE	Common Weakness Enumeration	11
DoS	Denial of Service	19
HTTP	HyperText Transfer Protocol	19
HTTPS	HyperText Transfer Protocol Secure	19
JSON	JavaScript Object Notation	20
PGP	Pretty Good Privacy	15
S3	Simple Storage Service	19
SHA-1	Secure Hashing Algorithm 1	17
TLS	Transport Layer Security	23
URI	Uniform Resource Identifier	19