Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Passbolt Backend & Plugins 06.2021

Cure53, Dr.-Ing. M. Heiderich, BSc. J. Hector, MSc. J. Moritz

Index

Introduction

> *“The password manager your team was waiting for. Free, open source, self-hosted, extensible, OpenPGP based."*
>
> From https://www.passbolt.com/

This report describes the results of a security assessment of the Passbolt complex, spanning Passbolt backend, API and a selection of Passbolt plugins. Carried out by Cure53 in the summer of 2021, the project included a penetration test and a dedicated audit of the source code.

The work was requested by Passbolt SA in mid-April and scheduled for mid-to-late June of the same year. Cure53 has looked at the related components of the Passbolt scope before, as indicated by this project being labelled as *PBL-03.* More specifically, various tests against Passbolt specifications, cryptography and software conducted in the past set the basis for the current project, which is targeting the server-side components.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

In this assessment, a total of nine days were invested to reach the expected levels of coverage, with core investigations taking place in CW25. A team of three senior testers has been composed and tasked with this project's preparation, execution and finalization. For optimal structuring and tracking of tasks, the work was split into three separate work packages (WPs):

- **WP1**: White-Box Tests & Security Assessments against Passbolt Backend & API
- **WP2**: White-Box Tests & Security Assessments against Passbolt Plugins
- **WP3**: White-Box Tests & Security Assessments against Passbolt UI

It can be derived from above that white-box methodology was utilized. Cure53 was given access to all relevant sources and documentation as well as instructions facilitating running the software locally, which was used for the pentesting parts. All preparations were done in mid-June, namely in CW24, so Cure53 could have a smooth start.

The project progressed effectively on the whole. Over the course of the engagement, the communications were done using a private, dedicated and shared Slack channel previously used for *PBL-01* and *PBL-02.* The discussions throughout the test were very productive and not many questions had to be asked. The scope was well-prepared and clear, with no noteworthy roadblocks encountered during the test.

It has to be stated that the Passbolt team, as in other tests before, was very helpful and did whatever was necessary to make it possible for Cure53 to get good coverage over the scope with the chosen approaches. The assistance spanned fast answers to all questions, very quick turnaround times and generally excellent test-support. Ongoing exchanges and interactions positively contributed to the overall outcomes of this project.

Cure53 offered frequent status updates about the test and the emerging findings. Very good coverage over the WP1-WP3 scope items has been reached. Among four only security-relevant discoveries, two were classified to be security vulnerabilities and two to be general weaknesses with lower exploitation potential. Only *Low*-scoring problems and hardening advice can be found on the list of security tickets.

Strengthening the positive impressions Cure53 already gained from examining the client-side Passbolt software, such as the browser extensions, the server-side parts expose a similarly strong security posture. It is clear that the software is being designed and built with security and privacy in mind, which is also mandatory given the purpose of this project. In the following sections, the report will first shed light on the scope and key test parameters, as well as the structure and content of the WPs. A dedicated chapter on test methodology and coverage then clarifies what the Cure53 did in terms of attack-attempts, coverage and other test-relevant tasks.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Next, all four findings will be discussed in grouped vulnerability and miscellaneous categories, while following a chronological order in each group. Alongside technical descriptions, PoC and mitigation advice are supplied when applicable. Finally, the report will close with broader conclusions about this June 2021 project. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team's observations and collected evidence. Tailored hardening recommendations for the Passbolt complex are also incorporated into the final section.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Scope

- **Penetration-Tests & Code Audits against Passbolt Backend, API & Plugins**
  - **WP1**: White-Box Tests & Security Assessments against Passbolt Backend & API
    - https://bitbucket.org/passbolt_pro/passbolt_pro_api/src/master/
  - **WP2**: White-Box Tests & Security Assessments against Passbolt Plugins
    - See plugin folder in sources linked above
  - **WP3**: White-Box Tests & Security Assessments against Passbolt UI
    - https://github.com/passbolt/passbolt_styleguide/tree/master/src/react-extension (all of the *Api** files)
  - **Documentation & setup instructions**
    - https://help.passbolt.com/hosting/install/ce/from-source
    - https://help.passbolt.com/hosting/install/pro/debian/debian.html
  - **Test-supporting material was shared with Cure53**
  - **All relevant sources were made available to Cure53**

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Test Methodology

This section aims to make Cure53's pentesting process more transparent and thus highlights the adopted approaches used to find security vulnerabilities across the given scope and work packages. The following subsections divides the pentest into two major categories.

While originally three distinct work packages were assigned, tests against WP1 and WP2 scope items (backend and plugins) were done in parallel. WP3 then covers a general website security audit of the frontend components and UI parts.

Security Assessment of the Passbolt Backend/API and Plugins

This section gives insight into the tests performed regarding the Passbolt backend and API implementation, as well as the plugins shipped with the pro version of Passbolt. Sources were provided for this assessment, so the audit followed a white-box pentesting style.

- Since the language of choice for the implementation of the backend is PHP, it was first checked if dangerous PHP functions are incorrectly used. Although it has been common knowledge that *unserialize*, for example, should not be used with user-input, it is commonly found to be used in such a way. However, checking for the usual pitiful functions in the codebase yielded no results and most of these functions are not even used.
- Once the common pitfalls were eliminated, a closer look at the authentication and login mechanism was taken. For this Passbolt relies on public and private key pairs; the cryptography itself was subject of a previous audit, so the focus was placed on the actual implementation. Here the general code flow during authentication was checked for logical flaws. Further, it was checked if the recovery code can be tricked via Unicode characters in the recovery email. Additionally. it was checked if an account can be recovered using a different private key than the one used during the initial setup. A minor issue in the login mechanism was found when handling the *redirec*t parameter, see PBL-03-001 for more details. No further issues could be found.
- It was checked how the database is queried and whether or not manual query construction is used and susceptible to SQL injection vulnerabilities. However, the developers solely rely on the query builder provided by the CakePHP framework. It is possible to use the framework incorrectly, for example by allowing user-input in the key parameters for the condition array, but these do not occur on the scope.
- Since the API consumes JSON objects, mass assignments frame common issues. These occur when the object in the client request is mapped one to one

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

to the database objects and then saved. Under certain circumstances, this can allow an unauthorized user to escalate their privileges by setting certain parameters, such as the role ID. Thus, it was checked how requests are handled and if such issues can arise. No issues were found due to mitigations in place.

- The API and plugins were checked for proper implementation of access controls that prevents unauthorized access to critical functionality. Access control checks in Passbolt are enforced through database entries in which user IDs are associated with objects they own. This requires that whenever the database is queried, the corresponding user ID is part of the query conditions. Cure53 checked if this is properly implemented and found no issues.
- Furthermore, the codebase was checked for any usage of filesystem IO functions that could lead to arbitrary file-reads or writes. While checking these, it was also investigated how the file upload for avatars is handled and if, for example, issues like path traversals or arbitrary file uploads are possible. Again, the developers demonstrate a good understanding of common security best practices and no issues could be identified.
- Another common issue for web applications concern Cross-Site-Request Forgeries. While it was found that CSRF tokens are in place to prevent these kinds of requests, two endpoints were identified that accept *GET* requests when *POST* should be used. This makes both endpoints susceptible to CSRF, however the impact is rather low given the functionality they provide. More details can be found in PBL-03-002.
- Cure53 checked for the potential of Server-Side-Request Forgeries (SSRF). An interesting area for this was the *DirectorySync* plugin which allows to specify a server. While it was not possible to have a full blown SSRF, the *DirectorySync* plugin could nonetheless be leveraged to check for open ports in the internal network. PBL-03-003 describes this in more detail.
- A lot of things need to be considered and implemented correctly in PHP. For instance, PHP allows weak comparisons which can lead to problems. Another example is deserialization using the *phar://* protocol. Here the team checked for common issues by auditing the provided sources, yet no issues were documented.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Security Assessment of the Passbolt UI and Frontend components

In this section, testing of the frontend components as well as the UI parts of the Passbolt application is described. The frontend communicates with the browser-specific plugin which was the focus of a previous audit, thus explaining why attention was only placed on the UI aspect and the corresponding JavaScript code this time around. Sources were provided and a white-box style of pentesting was followed.

- Passbolt's UI and frontend implementation are based on the ReactJS framework which already provides good built-in mitigation against XSS. As XSS is still possible through the use of the *dangerouslySet** functions, so Cure53 audited the provided sources for any use of these functions.
- The source code audit for potentially dangerous calls was accompanied by dynamic testing, done by providing malicious input in various input fields. No unintended behavior was observed and no issues could be identified.
- The sources were audited more closely in order to find potential logical issues or DOM-based XSS due to untrusted user-input reaching sensitive sinks. The UI code also communicates with the browser-specific plugin (examined in the past). The implementation handling the communication was audited with no flaws to report.
- Parsing of URLs is done in order to extract parameters, so it was checked that no issues in the parsing logic are present and cause unintended behavior. However, untrusted user-input is handled very well and no issues were found.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *PBL-03-001*) for the purpose of facilitating any future follow-up correspondence.

PBL-03-001 WP2: Arbitrary Redirect on login allows Phishing *(Low)*

While analyzing the multi-factor authentication mechanism in search of potential security problems, it has proven possible to append a so-called *redirect* parameter to the URL of the verification page. Using an arbitrary domain inside the above parameter and tricking a victim to use this seemingly trustworthy URL leads to a redirect to an attacker-controlled domain. From that domain a Phishing attempt could be started. The following PoC was created to demonstrate this issue.

URL:
https://localhost/mfa/verify/totp?redirect=http://evil.com

Taking the URL as a starting point to initiate the TOTP verification process results in the *POST* request presented below being triggered.

Resulting *POST* request:

```
POST /mfa/verify/totp?redirect=http://evil.com  HTTP/1.1
Host: localhost
[...]
Referer: [...]
```

If the verification is successful, a response showcased next can be observed.

Response:

```
HTTP/1.1 302 Found
Server: Apache/2.4.41 (Ubuntu)
[...]
Location: http://evil.com
[...]
```

It can be inferred from above that another redirect happens. Specifically, the victim gets transferred to the attacker-controlled domain at *evil.com.* Yet another login form is presented to the victim there. By reaching this stage, the attacker can trick the victim into supplying their credentials again, as they believe the initial login attempt has failed. Of course this time the supplied credentials are sent back to the attacker's domain,

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

eventually resulting in a successful Phishing attack. The issue was attributed to the particular code location furnished below.

Affected file:
plugins/Passbolt/MultiFactorAuthentication/src/Controller/MfaVerifyController.php

Affected code:

```
protected function _handleVerifySuccess()
{
       // Success response depends on request type
       if ($this->request->is('json')) {
       $this->success(__('The multi-factor authentication was a success.'));
       } else {
       $redirectLoop = '/mfa/verify';
       $redirect = $this->request->getQuery('redirect');
       if (is_null($redirect) || substr($redirect, 0, strlen($redirectLoop)) ===
$redirectLoop) {
              $redirect = '/';
           }
       $this->redirect(Router::url($redirect, true));
       }
}
```

In the end, one can see that the *redirect* parameter is being passed down from the *request* parameters without previously being checked against an allow-list of URLs. To mitigate this problem, it is recommended to employ an allow-list-based approach and only permit a specific set of URLs for redirection. The list could include, for example, the main host of the application.

PBL-03-002 WP2: Inconsistent CSRF protections *(Low)*

It was found that two endpoints of the *DirectorySync* plugin accept *GET r*equests for state changing operations. Since the application only verifies CSRF tokens for *POST, PUT, PATCH* and *DELETE* requests, these endpoints are not protected against CSRF attacks.

An attacker can exploit this by luring an admin to click a link on an attacker-prepared website. By this action, the admin can unintentionally trigger the synchronization with an external *LDAP* or *ActiveDirectory* server. Furthermore, an attacker can trick an admin into removing a user or a group from the synchronization ignore-list, which might allow the attacker to synchronize user accounts that should not be available in the application. For the latter the *id* of the user or group must be known to the attacker.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

To reproduce this issue an authenticated administrator has to click a link on the following attacker-prepared website which is hosted on e.g. *http://evil.com/crsf.html.*

PoC HTML:

```
<a href="http://localhost/directorysync/synchronize.json">Click me</a>
<a href="http://localhost/directorysync/ignore/toggle/Users/<ID>">Click me</a>
```

After clicking a link, a *GET* request with the administrator's session cookie is submitted to the corresponding endpoint and, thus, the action is being triggered.

For state changing operations, it is recommended to make use of the request method *POST* instead of *GET.* When the request method is changed to *POST*, the CSRF protection mechanisms of the underlying framework take effect.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

PBL-03-003 WP2: Admin SSRF allows port scanning in the internal network *(Info)*

It was found that an administrator can specify arbitrary hosts and ports to connect to an external *LDAP* or *ActiveDirectory* server using the *DirectorySync* plugin. If the system is configured with multiple administrators and one of them does not have command line access, this user can scan the internal network for running services. However, successful exploitation of this vulnerability depends on the user model in the particular setup.

The following two requests show how a malicious admin can determine whether a port on the specified server is open or closed. If a port is closed, the message *"The settings provided are incorrect. No LDAP server is available."* is returned to the user. Otherwise, the message *"The settings provided are incorrect. Unable to bind to LDAP: Can't contact LDAP server"* is shown.

PoC request for closed port:

```
POST /directorysync/settings/test.json?api-version=v2 HTTP/1.1
Host: passbolt.local
Cookie: passbolt_session=<ADMIN SESSION>;
Content-Type: application/json
X-Csrf-Token: <TOKEN>
Content-Length: 692

{
"directory_type":"ad",
"domain_name":"example.local",
"connection_type":"plain",
"server":"127.0.0.1",
"port":1337,
[...]
}
```

PoC response closed port:

```
[...]
  "message": "The settings provided are incorrect. No LDAP server is
available.",
    "url": "\/directorysync\/settings\/test.json?api-version=v2",
[...]
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

PoC request open port:

```
POST /directorysync/settings/test.json?api-version=v2 HTTP/1.1
Host: passbolt.local
Cookie: passbolt_session=<ADMIN SESSION>;
Content-Type: application/json
X-Csrf-Token: <TOKEN>
Content-Length: 692

{
"directory_type":"ad",
"domain_name":"example.local",
"connection_type":"plain",
"server":"127.0.0.1",
"port":80,
[...]
}
```

PoC response open port:

```
[...]
  "message": "The settings provided are incorrect. Unable to bind to LDAP: Can\
u0027t contact LDAP server",
    "url": "\/directorysync\/settings\/test.json?api-version=v2",
[...]
```

It is recommended to add a configuration entry that allows configuring a set of allowed hosts and ports. With such an entry, an administrator without command line access would not be able to scan arbitrary hosts in the internal network.

PBL-03-004 WP3: Cross-Origin-related HTTP security headers missing *(Info)*

It was found that the Passbolt platform is missing several of the newer[1] Cross-Origin-infoleak-related HTTP security headers in its responses. This does not directly lead to a security issue, yet it might aid attackers in their efforts to exploit other problems, such as for example issues relating to the Spectre attack[2]. The following list enumerates the headers that need to be reviewed to prevent flaws linked to headers.

- **Cross-Origin Resource Policy (CORP)** and *Fetch Metadata Request* headers allow developers to control which sites can embed their resources, such as images or scripts. They prevent data from being delivered to an attacker-controlled browser-renderer process, as seen in *resourcepolicy.fyi* and *web.dev/fetch-metadata*.
- **Cross-Origin Opener Policy (COOP)** lets developers ensure that their application window will not receive unexpected interactions from other websites,

[1] https://security.googleblog.com/2020/07/towards-native-security-defenses-for.html
[2] https://meltdownattack.com/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

allowing the browser to isolate it in its own process. This adds an important process-level protection, particularly in browsers which do not enable full Site Isolation; see *web.dev/coop-coep.*

- **Cross-Origin Embedder Policy (COEP)** ensures that any authenticated resources requested by the application have explicitly opted-in to being loaded. Today, to guarantee process-level isolation for highly sensitive applications in Chrome or Firefox, applications must enable both COEP and COOP; see *web.dev/coop-coep.*

Overall, missing Cross-Origin security headers can be considered a bad practice that should be avoided in times where attacks such as Spectre are known to be well-exploitable and exploit code is publicly available. It is recommended to add the aforementioned headers to every relevant server response. Resources explaining those headers are available online, explaining both the proper header setup[3] as well as the possible consequences of not setting them after all[4].

[3] https://scotthelme.co.uk/coop-and-coep/
[4] https://web.dev/coop-coep/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Conclusions

As noted in the *Introduction,* Cure53 is impressed with the general outcomes of this and previous tests of the Passbolt complex. Tested in June 2021 by a four-member team, the Passbolt backend, API and various Passbolt plugins make a great impression from a security standpoint. This is also reflected by only a few findings outlined in this report. It should also be noted that two flaws that were classified as vulnerabilities were both rated as having *Low* impact.

The main focus of this audit was the backend implementation which is done using PHP. Choosing PHP can introduce a lot of problems when the developers are unaware of typical pitfalls. However, the Passbolt developers demonstrate a strong understanding of the language, as well as exceptional familiarity with common security best practices. This is supported by the lack of findings which stem from unsanitized user-input, like XSS, SQLi or Remote Code Execution. A minor exception to this is the arbitrary redirect discussed in PBL-03-001. Reviewing the code revealed that untrusted input is strictly validated and sanitized. which contributes to the overall great security posture of the application.

Choosing to build the application on the CakePHP and the ReactJS frameworks, along with proper implementation of both, has been a good decision by the developers. It helps in mitigating security risks and, in the end, contributes to a good security posture. Besides good handling of user-input, the implemented access control mechanism is marked by robustness, with solid checks in place when necessary and keeping the complexity rather minimal.

PBL-03-002 discusses a CSRF issue which is the result of selecting to implement the given features using the *GET* method rather than the *POST*. This leads to the CSRF token check being omitted which would prevent this kind of issue. However, the impact here is rather low and a fix is fairly straightforward. In regard to file upload handling, the developers implemented an allow-list of extensions as well as accepted mime-types. In combination with strict input checks, the approach mitigates security issues like arbitrary file uploads and path traversals.

Full blown SSRF was also not possible, however PBL-03-003 outlines a scenario where *DirectorySync* could be used to discover open internal ports. It should be noted though, that this is rather far-fetched in terms of actual feasibility. Nevertheless, it still serves as a pointer to a potential issue in the future.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

All in all, the Passbolt application is in a very good state and capitalizes on a number of security strengths, especially gained by extensive knowledge of the developers who implement comprehensive mitigations and anticipate attacks quite well. Fixing the issues outlined as the result of this Cure53 June 2021 should further improve the already good security standing of the Passbolt complex.

Cure53 would like to thank Remy Bertot, Cedric Alfonsi and Max Zanardo from the Passbolt SA team for their excellent project coordination, support and assistance, both before and during this assignment.