Fine penetration tests for fine websites

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

Audit-Report Passbolt DirectoryTree LdapRecord 07.2023

Cure53, Dr.-Ing. M. Heiderich, MSc. N. Krein, BSc. B. Walny

Index

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

Introduction

“Passbolt is the first open source password manager tailored for agile and devops teams first, yet usable by everyone. It is designed to help centralize, organize and share credentials securely. It is built with the vision to re-unite control, productivity and modern security in a single, elegant and collaborative application that is aligned with technical teams preferences and work ethics.”

From https://www.passbolt.com/about

This documentation outlines the scope, test methodology, findings, and final summaries of a penetration test and source code audit against the DirectoryTree LdapRecord PHP library and connected integration.

Passbolt SA initially approached Cure53 with the project proposal in May 2023. Following the successful agreement, the assignment was scheduled for CW29 July 2023 and enacted by a team comprising three senior testers, who conducted the preliminary, execution, and finalization phases of the exercise. Each auditor was selected for their specific skillset and expertise in this particular field. A total of eight working days were invested to ensure maximal breadth of coverage across the scope items.

For ease of test execution, all assessment actions were grouped into two distinct work packages (WPs), which are defined by the following headings:

- **WP1**: Source code audits against DirectoryTree LdapRecord PHP library
- **WP2**: Source code audits against DirectoryTree LdapRecord Passbolt integration

The maintainers from Passbolt SA provided a host of assisting materials to aid Cure53’s efforts against the target entities, including sources, the necessary Docker files to install an on-premise Passbolt setup, as well as detailed scope-relevant information concerning library integration. Both parties agreed that a white-box penetration testing methodology was the most relevant choice for this particular engagement.

The team performed any required preparations in the week prior to active testing (CW28 July 2023) to gain a strong understanding of the scope requirements and resolve any outstanding blockers. Communications between the team members from both organizations were enabled via a dedicated and shared Slack channel. This format proved conducive for an effective and productive collaboration environment, with minimal cross-team queries required. The avoidance of any typical delays or hindrances encountered during procedures of this nature, due to the comprehensive and diligent endeavors of all involved personnel both before and during the assignment, was greatly appreciated by the test team.

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

Cure53 provided frequent status updates regarding the test and corresponding findings. Whilst no specific request for live reporting was relayed, the team nonetheless shared some top-level information concerning the findings as they arose. With respect to the findings, despite ample coverage over the WPs, the audit team was only able to uncover two noteworthy findings, though both were categorized as security vulnerabilities.

The exceptionally small sum of findings, as well as the moderate maximum severity level (Medium), corroborates the opinion that the components in scope already implement performant defensive measures to nullify a plethora of modern attacks. However, it should be noted that both issues were discovered during the examination against the library integration, which underlines some minor oversights in the actual API usage of the provided framework. Nevertheless, the resulting impact was considered to be relatively minimal and does not significantly expand the attack surface of Passbolt itself.

To summarize, Cure53 can only offer its congratulations to the Passbolt SA handlers for their ideal library selection and astute integration. Nonetheless, one can strongly recommend addressing the two findings discussed in this report at the earliest possible convenience, as well as upkeeping the evidently stringent security values. These will undoubtedly ensure that a first-rate security premise can be maintained.

The report will now shed more light on the scope and test setup, as well as the available material for testing. Following this section, there will be a chapter that details the *Test Methodology* used in this exercise. This aims to demonstrate to the client which areas of the software were covered and the tests executed, in lieu of the absence of high impact risks.

Following, the report lists all findings in chronological order. First, the spotted vulnerabilities are discussed, then the general weaknesses discovered in this test (though none from the latter category were identified). Each finding attaches a technical description, a Proof-of-Concept (PoC) or steps to reproduce where applicable, and advice regarding mitigation or fixes.

Finally, the report will conclude with Cure53 elaborating on the general impressions gained throughout the assignment, as well as providing insights into the perceived security posture of the DirectoryTree LdapRecord PHP library and its integration.

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

Scope

- **Source code audits & penetration tests against DirectoryTree LdapRecord PHP library**
  - **WP1**: Source code audits against DirectoryTree LdapRecord PHP library
    - **URL:**
      - https://ldaprecord.com/
    - **Source:**
      - https://github.com/DirectoryTree/LdapRecord
  - **WP2**: Source code audits against DirectoryTree LdapRecord Passbolt integration
    - **API URL:**
      - https://bitbucket.org/passbolt_pro/passbolt_pro_api/src/master/plugins/PassboltEe/DirectorySync/
    - **Styleguide (frontend):**
      - https://github.com/passbolt/passbolt_styleguide/tree/master/src/shared/models/userDirectory
      - https://github.com/passbolt/passbolt_styleguide/tree/master/src/shared/services/api/userDirectory
      - https://github.com/passbolt/passbolt_styleguide/tree/master/src/shared/services/forms/userDirectory
      - https://github.com/passbolt/passbolt_styleguide/tree/master/src/react-extension/components/Administration/DisplaySimulateSynchronizeUserDirectoryAdministration
      - https://github.com/passbolt/passbolt_styleguide/tree/master/src/react-extension/components/Administration/DisplaySynchronizeUserDirectoryAdministration
      - https://github.com/passbolt/passbolt_styleguide/tree/master/src/react-extension/components/Administration/DisplayTestUserDirectoryAdministration
      - https://github.com/passbolt/passbolt_styleguide/tree/master/src/react-extension/components/Administration/DisplayUserDirectoryAdministration
    - **Installation materials and supplementary scope information:**
      - **Comprehensive scope information:**
        - **Docker compose file:**
          - https://gist.github.com/nourcy/12ed6f8d8ac10805bad553393d4a456a
        - **OpenLDAP LDIF file:**
          - https://gist.github.com/nourcy/7fd7d89ca9284fc83cb8dbf4d674ae6c
  - **Test-supporting material was shared with Cure53**
  - **All relevant sources were shared with Cure53**

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

Test Methodology

The *Test Methodology* section provides a definitive overview of the coverage achieved by Cure53 during this pentest iteration against the designated scope. Since two WPs were prescribed for auditing purposes, this section is divided into two subsections for ease of reference, each of which clarify the multitude of pentesting methods instigated. As a consequence of the minimal yield of security-relevant findings, the following passages serve to elucidate the entire procedural analyses and draw attention to any initially promising attack vectors that were either unsuccessful or did not evoke any noteworthy outcomes.

WP1: Source code audits against DirectoryTree LdapRecord PHP library

The *LdapRecord* PHP library provides a framework to facilitate easier operational integration with LDAP directories. This represents the core library utilized in Passbolt's user directory feature, thus providing administrators the ability to synchronize a list of groups and users. Cure53's primary objectives for WP1 were to estimate the library's security premise in general and ascertain whether any additional attack surface is exposed via the Passbolt web application, as summarized below:

- Cure53 initiated the *LdapRecord* library assessment by checking the official GitHub repository[1] for previously reported weaknesses or glaring security faults.
- Next, the issues page, commit log, and public CVE directory were subjected to stringent examination. Despite exhaustive efforts, the Cure53 consultants were unable to detect any previously reported limitations, except for minor and common bugs that incur negligible security impact.
- The evidently abundant development-specific and generic updates attest to active library maintenance. The ensuing avoidance of deficiencies in this area was noted with commendation.
- Nonetheless, the library is written in PHP, which oftentimes facilitates an extensive volume of potential pitfalls that may evoke security compromise. As a result, the auditors exhaustively scrutinized the source code for risks that are likely to emanate from libraries of this ilk.
- Cure53 placed particular emphasis on reviewing framework functionalities that may, in turn, induce developers to write insecure code. To provide one example, this can occur when seemingly secure API calls to construct LDAP queries do not possess the necessary escaping functionalities.
- During this process, the query's API documentation was extensively studied in code to ensure each call operates as originally intended. Inherently risk-laden functionalities - such as the *rawFilter()* API - are marked with explicit warnings to enforce user input omission.

[1] https://github.com/DirectoryTree/LdapRecord

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

- A supplementary endeavor here pertained to determining whether any of these (and indeed WP2's) functionalities are used within Passbolt itself, and if so by what means. Supporting guidance in this respect can be found in PBL-09-001.
- Complementing the evaluation for insecure API calls was an appraisal of the alternatives and essential escaping functionalities to guarantee the purported security assurances had been met. Testing verified that the *EscapedValue* class enforces compliance with and correct utilization of the necessary flags.
- The audit team also honed in on potential PHP pitfalls that might be obfuscated and embedded in unassuming mechanisms, such as session handling, caching, or serialization.
- *LdapRecord* supports a number of serialization mechanisms for model instances. Thus, Cure53 deemed it apt to evaluate any security exposure via dangerous methods, such as PHP's unserialize. However, no associated faults were observed since only recommended serialization methods - primarily secure *json* array handling - were leveraged. Notably, some degree of reflection was considered possible.
- Internally, *LdapRecord* adapts the *Simple Cache* interface, which is implemented according to specification and does not imbue any substantial attack surface.
- Lastly, Cure53 confirmed that correlating and potentially sensitive aspects (such as the Connection manager) function as expected and implement various measures for authentication handling. Albeit, the method by which they are integrated is entirely dependent on the project that adopts the *LdapRecord* framework, which is only relevant for WP2.

WP2: Source code audits against DirectoryTree LdapRecord Passbolt integration

This section stipulates the advanced approaches applied against Passbolt's Users Directory feature, as defined in WP2. The core goal here was to ensure the *LdapRecord* library is correctly integrated into Passbolt's codebase and does not expose any auxiliary attack surface.

- Passbolt enables an additional frontend for the Users Directory feature, which was henceforth subjected to deep dive examination for any typical client-side security shortcomings.
- The implemented React components and templates were validated to successfully repel commonly encountered faults, such as XSS, due to optimal usage of the framework's APIs. These efforts also verified that no potentially vulnerable sinks had been incorporated, including setting elements' *innerHTML* attribute or working with reference escape hatches.
- Alternative actions that may have facilitated security compromise also reviewed favorably, such as the absence of arbitrary user input during the dynamic construction of the *href* attribute.

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

- Other positive attributes noted by Cure53 here were correct CSRF token handling, as well as ideal propagation and fault-free functionality of respective header fields from the main app.
- Elsewhere, the testers sought to assess the attack surface incurred by the Passbolt Pro API, which was completed by examining the handful of defined endpoints via the newly introduced routes. No pertinent issues were acknowledged from an authorization standpoint, since all routes were only available via an authenticated and admin-authorized user, thus sufficient protection had been established.
- However, in the handful of exposed controllers, three alternate dialects for verifying the user permissions that access these routes were observed; specifically, the first by checking *$this->User->role()*; the second by checking *$this->User->isAdmin()*; and the third by implementing a global *beforeFilter()* that verifies the role. This process is on occasion conducted by either using weakly typed comparisons via "==" and or strongly typed comparisons via "*!*==".
- This particular review raised some concern and exhibited opportunities for hardening. Cure53 recommends always utilizing a *beforeFilter()* for each controller that confidently verifies the role via *isAdmin()* once for all exposed administrative endpoints. This will guarantee that newly inserted endpoints are inherently protected and expose scant leeway for potential oversight errors.
- Despite this requirement, no seemingly forgotten or unprotected endpoint was detected throughout the engagement.
- In addition, the *LdapRecord* integration was inspected according to the API documentation and in adherence with the insight obtained during WP1 evaluations.
- Two distinct LDAP injections were identified and documented in tickets PBL-09-001 and PBL-09-002 respectively. Both issues allow malicious admins to extract information from the connected LDAP server, despite the inability to access it otherwise.
- The root cause of the first issue emanates from lax use of *rawFilter()*, which should be prevented. The second manifests via direct use of *field_mappings*, which may be leveraged in tandem with verbose exception logging to extract sensitive information from the LDAP server. Here, one must underline that the severity risk associated with these fault areas depends on the individual LDAP usage for each client.
- Lastly, user input handling in general was considered sufficiently protected. Other types of injection vectors are prevented due to correct input and output validation of data originating from the LDAP instance. Positively, no SQL injections into the Passbolt storage drivers were located, which can be attributed to Passbolt's correct adoption of parameterized queries.

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

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, all tickets are given a unique identifier (e.g., *PBL-09-001*) to facilitate any future follow-up correspondence.

PBL-09-001 WP2: LDAP injection via custom group/user filters *(Low)*

Whilst auditing the LDAP filter functionalities within the Passbolt Pro API's Directory Sync plugin, Cure53 noted that both the user and group filtering for LDAP queries are vulnerable to LDAP injections. Here, user input is directly passed into the *LdapRecord* query builder's *rawFilter* function, which incurs the potential for arbitrary user-selected LDAP queries. With this primitive, all contents of the LDAP database can be retrieved via a standard blind search that requests a character at a time, then subsequently observing the response. Nonetheless, this ticket's severity marker was reduced to *Low*, due to the requirement for an authenticated admin and the separation between Passbolt data storage and the LDAP. However, one must stipulate that the risk impact may be higher depending on the data saved in the LDAP instance.

Affected file:
passbolt_pro_api/plugins/PassboltEe/DirectorySync/src/Utility/LdapDirectory.php

Affected code:

```
private function _customizeUsersQuery(Builder $query): Builder
{
	$userCustomFilter = $this->directorySettings->getUserCustomFilters();
	if (is_callable($userCustomFilter)) {
[...]
	} elseif (is_string($userCustomFilter)) {
		try {
		$filter = Parser::parse($userCustomFilter);
		$query->rawFilter(Parser::assemble($filter));
[...]
private function _customizeGroupsQuery(Builder $query): Builder
{
[...]
}
```

The following HTTP request additionally underlines this flaw by querying the sensitive *userPassword* attribute present in the majority of LDAP installations. Since the *rawFilter()* directly accepts filter queries, a rogue admin user can thus extract sensitive fields that they should typically not be permitted to access:

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

Sample request:

```
POST /directorysync/settings/test.json?api-version=v2 HTTP/2
Host: docker.passbolt.local
[...]
Content-Type: application/json

{
  "enabled": true,
  "group_path": "",
  "user_path": "",
  "group_custom_filters": "(&(cn=meow))",
  "user_custom_filters": "(|(userPassword={CLEARTEXT}password))"
[...]
}
```

Sample response:

```
{
  "header": {
       "id": "8bc514f3-4cad-4493-a127-f8f690b97b2e",
       "status": "success",
       "servertime": 1689843618,
       "action": "eefa8673-805e-5ce9-be3a-4062ab608d76",
       "message": "The operation was successful.",
       "url": "/directorysync/settings/test.json?api-version=v2",
       "code": 200
  },
  "body": {
       "users": [{
              "type": "user",
              "id": "4aa844ea-b8ce-103d-9b6d-8b235a4af9c7",
              "directory_name": "uid=user1,ou=users,dc=passbolt,dc=local",
              "directory_created": "2023-07-17T09:16:00+00:00",
              "directory_modified": "2023-07-17T09:16:00+00:00",
              "user": {
                     "username": "uwilliams@example.org",
                     "profile": {
                            "first_name": "Kenneth  # Assuming given name is the
first part of the name",
                     "last_name": "Kenneth Johnson"
                     }
              },
[...]
       }]
  }
}
```

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

Whilst one could argue that this behavior is an intentional design decision and feature, a number of risk considerations are evoked that should be heeded. The capability to extract data from the LDAP database might not be immediately obvious to clients, particularly considering that the functionality is hidden behind a seemingly mundane mechanism. However, the *LdapRecord* documentation[2] clearly states "*Raw filters are not escaped. Do not accept user input into the raw filter method.*" Hence, this feature should be redesigned to neutralize the aforementioned security impact.

To mitigate this issue, Cure53 recommends reevaluating this design selection and warning clients that all contents of the LDAP entries may potentially be retrieved. To completely eliminate any injection potential here, the developer team could leverage the existing LdapRecord {or,*and}Filter* to prevent arbitrary filter clauses and automatic user input escaping. Albeit, one specific drawback will emerge following this implementation; namely, the loss of the wildcard search feature, which may represent a required use case. To reiterate, if the developer team opts to retain this design trait, clients should be informed regarding the consequential implications at the very least.

PBL-09-002 WP2: Arbitrary LDAP data exfiltration via *fields_mapping* (Medium)

Whilst assessing the Directory Sync APIs, the test team confirmed a method by which to arbitrarily extract data from the configured LDAP instance via the custom *fields_mapping* attribute. This field mapping feature intends to map variables from LDAP to Passbolt for display following successful retrievals. This mapping process was verified dynamic to each request and alterable ad hoc, for reasons unknown. This allows for arbitrary retrieval of LDAP fields, such as potentially sensitive *userPasswords*, by simply requesting them in a custom mapping.

To highlight this particular limitation, the following example requests have been supplied. Notably, the required administrative access does not imply access to the underlying LDAP instance. The Passbolt API provides established obfuscation features to hide the LDAP connection settings, whilst the LDAP server may only be reachable within the deployed Passbolt API network. Similarly to the previous ticket, the overall severity depends on the use and type of information stored within the LDAP records.

Sample request:

```
POST /directorysync/settings/test.json?api-version=v2 HTTP/2
Host: docker.passbolt.local
[...]
Te: trailers

{
   "enabled":true,
   "group_path":"",
```

[2] https://ldaprecord.com/docs/core/v2/searching#raw-filters

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

```
   "user_path":"",
   "group_custom_filters":"(&(cn=meow))",
   "user_custom_filters":"(&(cn=*Exf*))",
[...]
   "fields_mapping":{
[...]
      },
      "openldap":{
         "user":{
            "id":"entryUuid",
            "firstname":"givenname",
            "lastname":"userPassword",
            "username":"mail",
            "created":"createtimestamp",
            "modified":"modifytimestamp"
         },
[...]
}
```

Returned response:

```
{
  "header": {
       "message": "The operation was successful.",
       "code": 200
       [...]
  },
  "body": {
       "users": [{
             "type": "user",
             "id": "8411a828-b9b1-103d-814b-8b235a4af9c7",
             "directory_name": "uid=user31339,ou=users,dc=passbolt,dc=local",
             "directory_created": "2023-07-18T12:22:33+00:00",
             "directory_modified": "2023-07-18T12:22:33+00:00",
             "user": {
                   "username": "ben+ldap@cure53.de",
                   "profile": {
                         "first_name": "Exfiltration Test"
                         "last_name":"{CLEARTEXT}SuperSecretCure53Password"
                   }
             }
       }]
  [...]
}}}
```

To mitigate this issue, Cure53 discourages sending a custom field mapping on each request. The field mapping cannot be configured via the UI directly, hence sending it on each request is surplus to requirement. The default mapping is configured within a configuration file via *DirectorySync/config/config.php*. Subsequently, one can recommend statically following the mappings of this configuration file instead.

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

Conclusions

The concise, eight day pentest against the LdapRecord PHP library and connected Passbolt integration concluded without any substantially risk-inducing findings, which is an undeniably praiseworthy indication of a robust security foundation that repels a swathe of common bug classes.

In terms of the coverage achieved by the testers, the audit commenced with a review of the LdapRecord library. Soon after this had been initiated, Cure53 was able to confirm that the underlying codebase had been written from the ground up and in compliance with security best practices. This viewpoint is corroborated by the complete lack of tangible, high-risk deficiencies, which is even more impressive considering that the framework is subjected to frequent and recurring amendments. Despite exhaustive compromise attempts, the consultant team was unable to pinpoint any emergent weaknesses that may be susceptible to exploitation.

From the ensuing outcomes, one can confidently verify that the integration of the LdapRecord library does not widen the attack surface of Passbolt itself. For further clarification on this viewpoint, please refer to the *Test Methodology* section, which outlines the various techniques applied against the focus characteristics. Nonetheless, one aspect that may benefit from improvement concerns the utilization of the library's query builder itself. Here, the use of raw filters - which are inherently insecure and should not be used on user input - is applied to construct dynamic LDAP queries. Via custom group and user filters, unintentional access to the entire LDAP database may be facilitated. Albeit, this fault area is far more severe in theory than in practice, since this functionality is only exposed to administrative Passbolt users.

The second finding may be of greater pertinence, since the code to handle custom field mappings does not prevent the insertion of arbitrary fields. Thus, verbose exception logging can disclose sensitive fields from the LDAP database itself. Whilst this incurs slightly higher impact than the previous finding, both offer nominal impact upon the Users Directory feature’s security model in general.

In conclusion, Cure53 finalizes this procedure with a unanimously positive verdict. Considering the low volume of issues detected, one hopes that the *Test Methodology* section elaborates the overall testing process and instills confidence in the audited codebases’ degree of defensive resilience. The underlying code quality is excellent and written in conformance with strong industry operations, though the documented minor findings confirm that some hardening measures can be incorporated for airtight defense-in-depth. As mentioned previously, LdapRecord is a sound choice for handling LDAP functionalities and its integration into Passbolt can only be deemed a resounding success.

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

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