Fine penetration tests for fine websites

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

Review-Report Passbolt Crypto Features 07.2022

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Ing. David Gstir

Index

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

Introduction

"Finally, a password manager built for collaboration. Secure, flexible, and automation ready. Trusted by 10,000 organizations, including Fortune 500 companies, newspapers, governments and defence forces."

From https://www.passbolt.com/

This report describes the results of a security assessment of the Passbolt complex, spanning several of the newer Passbolt features, including the account recovery feature and the ECC key support. Carried out by Cure53 in July 2022, the project included a review of the Passbolt cryptography and a dedicated source code audit.

Registered as *PBL-07,* the project was requested by Passbolt SA in April 2022 and then scheduled for the beginning of the third quarter of 2022 to allow ample time for preparations on both sides. It should be noted that Cure53 has looked at the Passbolt scope before, yet the new features have clearly been tested for the first time during this *PBL-07* project.

As for the precise timeline and specific resources, Cure53 completed the examination in July 2022, specifically in CW28. A total of five days were invested to reach the coverage expected for this assignment, whereas a team of two 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 two separate work packages (WPs):

- **WP1**: Cryptography review and audit of the Passbolt account recovery feature
- **WP2**: Cryptography review and audit of the Passbolt ECC key support

It can be derived from above that white-box methodology was utilized. Cure53 was given access to white-papers, a test server, documentation, as well as all other means of access required to complete the tests. Additionally, source code was shared to make sure the project can be executed in line with the agreed-upon framework.

The project progressed effectively on the whole. All preparations were done in CW27 to foster a smooth transition into the testing phase. Over the course of the engagement, the communications were done using a private, dedicated and shared Slack channel. Involved team members from Cure53 and Passbolt could join the test-related conversations on Slack. The discussions throughout the test were very good and productive and not many questions had to be asked.

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

The scope was well-prepared and clear, greatly contributing to the fact that no noteworthy roadblocks were encountered during the test. Cure53 offered frequent status updates about the test and the emerging findings. Live-reporting was neither specifically requested nor seen as necessary given the manageability of the number and severity of the spotted findings.

The Cure53 team managed to get very good coverage over the WP1-WP2 scope items. Among six security-relevant discoveries, only one was classified as a security vulnerability and five were deemed to be general weaknesses with lower exploitation potential. This outcome clearly demonstrates that the security of the Passbolt test targets is in a very stable state. The small number and minor severities of the findings can be interpreted as a positive sign for the new features offered by Passbolt. None of the findings exceeded the *Medium* score, confirming that no major threats or large attack surface seem to be exposed by Passbolt.

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 team did in terms of attack-attempts, coverage and other test-relevant tasks.

Next, all findings will be discussed in grouped vulnerability and miscellaneous categories, then 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 pertinent to this July 2022 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

- **Crypto reviews & Audits of the new Passbolt features (account recovery, ECC keys)**
  - **WP1**: Cryptography review & Audit of the Passbolt account recovery feature
    - **Functional specs:**
      - https://docs.google.com/document/d/18TDONMdE0iQfB2zDf6MH-WeS4vjdCyUwnzfXz-0SYe4/edit#
    - **Technical specs:**
      - https://docs.google.com/document/d/1_Bksoq1Gnd7sEdTw7L91o6stIb4ou-quaxv1E-LOAZw/edit#
    - **Test server:**
      - https://pro.passbolt.dev
  - **WP2**: Cryptography review & Audit of the Passbolt ECC key support
    - **Relevant sources:**
      - https://github.com/passbolt/passbolt_browser_extension/tree/master/src/all/background_page/utils/openpgp
      - https://bitbucket.org/passbolt_pro/passbolt_pro_api/src/master/src/Utility/OpenPGP/
      - https://bitbucket.org/passbolt_pro/passbolt_pro_api/src/master/src/Service/OpenPGP/
  - **Detailed test-supporting material has been shared with Cure53**
    - https://drive.google.com/drive/folders/179ThgedeZwafq5RxqJsTb19hiPorljFM
  - **All relevant sources were made accessible to Cure53 and/or were available as OSS**

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

Test Methodology

This section documents the testing methodology applied during this *PBL-07* project. It clarifies the coverage achieved during this engagement centered on reviewing the cryptography of the Passbolt account recovery feature. The following notes explain which code parts were inspected and which classes of security bugs were investigated during this Cure53 assessment.

Cryptography and code review

This audit of the Passbolt account recovery feature consisted of a manual inspection of the relevant cryptographic code segments and an analysis of the general design. This included the API endpoints provided by the backend and the graphical user-interface code consisting of the browser extension and web application code.

Cure53 started with an analysis of the overall recovery flow design, checking for logical flaws which typically mean that malicious users or external attackers are capable of gaining insights into sensitive data or, alternatively, there is a way for them to hijack the recovery flow. The goal here was to recover the user's passwords stored in Passbolt or potentially control the administrator-account.

The initial steps were followed by a thorough analysis of the PHP backend code. Cure53 first and foremost checked for flaws in the code dealing with cryptography. Specifically, the choices of cryptographic algorithms for the individual PGP operations and secure random number generation were researched. Since the underlying PGP operations are implemented in third-party libraries, the usage of these APIs was inspected for errors and flaws. The underlying libraries themselves were assumed to be secure and were not investigated in depth due to the limited budget.

Further, all input validation and parsing code of PGP messages in the backend and frontend was reviewed. The focus was on potential issues where adversaries could abuse mistakes to cause harm to the security of the stored user-passwords. The code was also checked for any key reuse issues. Specifically, this pertains to the fact that a user-key must not be used as an organization-key or, similarly, a revoked key must not be available for reconfiguring.

The next test-target was the backend API, which was reviewed for correct authorization checks. Flaws in the checks could lead to attacks ranging from simple information leaks to full administrator-account-takeovers. Another aspect of the backend is secure storage of the organization key. Since this key is used to recover private keys of other users, its storage is highly critical. It turned out that the private key is never stored within Passbolt, but only the public key is persisted on the backend.

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

As the frontend code is responsible for generating PGP keys, importing them and performing various PGP cryptographic operations during the account recovery flow, the relevant code parts were also checked for flaws and erroneous use of the *OpenPGP.js* API. Finally, the newly added foundation for the ECC key support was reviewed for proper validation of the PGP key properties and usage of the PGP library APIs. The manual code review was combined with testing against a local test instance to verify various leads and check their exploitability.

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-07-001*) for the purpose of facilitating any future follow-up correspondence.

PBL-07-003 WP1: Unauthenticated API endpoints reveal users *(Low)*

The account recovery flow has a number of REST API endpoints which can be accessed without authentication. One of these is the URI */account-recovery/requests/{requestId}/{userId}/{tokenId}.* The endpoint is called after the user initially confirms the account's recovery email and it will check for the existence of a user account with the identifier specified in the placeholder *{userId}.* This happens before the endpoint validates whether the request identifier or token are valid.

In case the user does not exist, the request is immediately aborted and a response containing the error message "*The user does not exist or is not active.*" is returned. However, this can be abused by malicious actors who wish to test whether a user identifier exists, which could provide valuable information for further attacks.

Proof-of-Concept:

The following cURL command demonstrates this problem by using an all zero UUIDs for *requestId, userId* and *tokenId,* which do not exist in the database:

```
curl -H 'Accept: application/json'
'http://localhost:8080/account-recovery/requests/00000000-0000-0000-0000-
000000000000/00000000-0000-0000-0000-000000000000/00000000-0000-0000-0000-
000000000000.json?api-version=v2'
{"header":{"id":"826f7259-5bab-466e-a39e-
df8fecc1327f","status":"error","servertime":1657894845,"action":"a4029634-71ea-
5e22-bdfb-c05d694a4af4","message":"The user does not exist.","url":"\/account-
recovery\/requests\/00000000-0000-0000-0000-000000000000\/00000000-0000-0000-
0000-000000000000\/00000000-0000-0000-0000-000000000000.json?api-
version=v2","code":404},"body":""}
```

When querying for a different user identifier, but still specifying invalid *tokenId* and *requestId* values, the backend responds with a different message. The latter message indicates that the user exists, as the following snippet shows:

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

```
curl -H 'Accept: application/json'
'http://localhost:8080/account-recovery/requests/00000000-0000-0000-0000-
000000000000/1e94a0fb-f6b4-4198-bc4a-e646cec9d342/00000000-0000-0000-0000-
000000000000.json?api-version=v2'-version=v2'
{"header":{"id":"2f55082f-ec11-4f33-a643-
0056cc6c23ea","status":"error","servertime":1657895037,"action":"a4029634-71ea-
5e22-bdfb-c05d694a4af4","message":"The authentication token could not be
found.","url":"\/account-recovery\/requests\/00000000-0000-0000-0000-
000000000000\/1e94a0fb-f6b4-4198-bc4a-e646cec9d342\/00000000-0000-0000-0000-
000000000000.json?api-version=v2","code":404},"body":""}
```

Affected file:
passbolt_pro-passbolt_pro_api-bf1c3e91031c/plugins/Passbolt/AccountRecovery/src/Service/AccountRecoveryRequests/AccountRecoveryRequestGetService.php

Affected code:

```
public function getNotCompletedOrFail(
        string $requestId,
        string $userId,
        string $token,
        ?string $clientIp = null
    ): AccountRecoveryRequest {
        // Assert policy is not set to disabled
        (new AccountRecoveryOrganizationPolicyGetService())->getOrFail();

        // Assert user exist, is active and not deleted
        $userEntity = (new UserGetService())-
        >getActiveNotDeletedOrFail($userId);

        // Assert token exist and is valid and belong to the user and is of the
        right type
        $tokenService = new AuthenticationTokenGetService();
       $tokenEntity = $tokenService->getActiveOrFail($token, $userId,
       AuthenticationToken::TYPE_RECOVER);
[...]
```

It is recommended to always first check the *authentication* token: only after this is ensured to be valid, the application should provide more details in the error messages.

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-07-001 WP2: PGP key validation bypass using invalid Base64 *(Medium)*

While inspecting the PGP message parsing code in *passbolt_pro-passbolt_pro_api-bf1c3e91031c,* Cure53 discovered that the internal method *OpenPGPBackendArmored-ParseTrait::unarmor()* performs insufficient validation of the input. PGP messages can be “armored”, that is ASCII-encoded, and this is typically done for easier transmission within regular emails. This ASCII-encoded format consists of a Base64-encoded binary blob with a special prefix and suffix that indicate the type of the PGP message (generic message, public key etc.). Passbolt uses armored PGP messages throughout its API to transmit keys and other PGP-signed messages.

The *unarmor()* method is used to convert an ASCII-encoded PGP message back to its binary format. As shown below, the implementation performs rudimentary streamlining of the input and then extracts the Base64-encoded segment after the ASCII header. Afterwards, the result is fed to the PHP’s *base64_decode()*[1] function. If not explicitly specified, as is the case here, this function will default to a non-strict mode. As a result, it will ignore invalid Base64 characters, then silently drop them during decoding. This ends in the function almost always accepting the input and never returning *false*. As a consequence, *unarmor()* will never return *false* when a PGP message contains invalid Base64 characters.

Malicious clients might abuse this to store invalid PGP messages. When parsed by any other PGP implementation, these could trigger errors. This can occur in the PHP backend as it uses two PGP implementations: the *PHP GnuPG* extension for encryption and decryption and *Openpgp-php* for most other operations. In this case, *PHP GnuPG* will trigger an error since it receives the armored messages as input.

Affected file:
passbolt_pro-passbolt_pro_api-bf1c3e91031c/src/Utility/OpenPGP/Traits/OpenPGPBackendArmoredParseTrait.php

[1] https://www.php.net/manual/en/function.base64-decode.php

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

Affected code:

```
private function unarmor(string $text, string $header = 'PGP PUBLIC KEY BLOCK')
    {
        // @codingStandardsIgnoreStart
        $header = \OpenPGP::header($header);
        $text = str_replace(["\r\n", "\r"], ["\n", ''], $text);
        if (
            ($pos1 = strpos($text, $header)) !== false &&
            ($pos1 = strpos($text, "\n\n", $pos1 += strlen($header))) !== false
        ) {
            $pos2 = strpos($text, "\n=", $pos1 += 2);
            if ($pos2 === false) {
                // no CRC, consider the key invalid
                return false;
            }

            return base64_decode($text = substr($text, $pos1, $pos2 - $pos1));
        }

        return false;
        // @codingStandardsIgnoreEnd
    }
```

It is recommended to always enable the strict mode of *base64_decode()* by setting the second argument to *true.*

PBL-07-002 WP1: Weak crypto permitted in organization key validation *(Medium)*

A review of the backend logic employed to configure the account recovery feature (endpoint */account-recovery/organization-policies*) showed that the administrator has to decide on the procedures around organization keys. Specifically, they can either let the client generate the organization PGP key or generate one with a tool of their choice and import it. In both cases, the backend verifies that the key meets certain criteria before accepting it.

These checks currently do not ensure that weak ciphers like DSA or ElGamal with small, insecure key sizes are rejected. The existing code shown below would actually be capable of rejecting DSA and ElGamal keys, but this is currently not enabled.

It should also be noted that the frontend is not capable of generating keys with these algorithms, since the *OpenPGP.js* in the recent version has these disabled by default. Hence, an administrator can still import a weak key generated by some external tool. This can lead to situations where such a weak key is imported and could, consequently, be used to secure user private keys.

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

The scenario presented above would indicate that a compromise of the Passbolt database would be fatal. In other words, decrypting all private keys of the users and, thus, reaching all their stored passwords, would become much easier due to the weak cryptographic premise.

Affected file:
passbolt_pro-passbolt_pro_api-bf1c3e91031c/src/Service/OpenPGP/PublicKeyValidationService.php

Affected code:

```
public static function parseAndValidatePublicKey(string $armoredKey, ?array
$rules = null): array
    {
        [...]
        foreach ($rules as $ruleName) {
            switch ($ruleName) {
                case self::IS_VALID_ALGORITHM_RULE:
                    if (!self::isValidAlgorithm($keyInfo['type'])) {
                        $validationErrors[$ruleName] = __('The algorithm is
                        invalid.');
                    }
                    break;
[...]

public static function isValidAlgorithm(?string $algorithm = null, $strict =
false): bool
    {
        if (!isset($algorithm)) {
            return false;
        }
        $supported = \OpenPGP_PublicKeyPacket::$algorithms;
        if ($strict) {
            // Minus legacy items such as DSA, ELGAMAL
            // Default in openpgp.js v5
            unset($supported[16]);
            unset($supported[17]);
        }
        foreach ($supported as $i => $a) {
            if ($algorithm === $a) {
                return true;
            }
        }
        return false;
    }
```

It is recommended to enable the strict key algorithm validation mode for the organization key and only approve or accept RSA and ECC keys with sufficient key size.

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

PBL-07-004 WP1: Finished account recovery aids future key compromise *(Low)*

During a code review of the final step in the account recovery process, an observation about possible facilitation of future compromisation of the keys was made. Specifically, when the user successfully finishes the account recovery, the PGP messages created during the process are left in the database. This could aid attackers who get a hold of the Passbolt database.

Each user enrolled in the account recovery feature creates an encrypted copy of their private key. This private key is symmetrically encrypted using a randomly generated key. In turn, the randomly generated key is encrypted using the organization key and placed alongside the encrypted private key into the Passbolt database.

Whenever the administrator confirms a user's account recovery request, the password in question is decrypted and re-encrypted with the user's temporary private key, specifically created when issuing the request. The result is called the *account recovery response* and it is stored in the Passbolt database. To complete the account recovery, the user fetches this response and can use it to recover their private key and re-encrypt it with a new passphrase.

Once this process is completed, the recovery response is no longer needed and could be removed. This is, however, not performed, as seen from the *RecoverCompleteService::complete()* below. For Passbolt, this means that any attacker who manages to reach the Passbolt database can also retrieve these responses. While they are useless without the user's private key, they might become decryptable in the future if the key algorithm of the user's temporary key is no longer secure.

Affected file:
passbolt_pro-passbolt_pro_api-bf1c3e91031c/src/Service/Setup/RecoverCompleteService.php

Affected code:

```
public function complete(string $userId): void
    {
        $user = $this->validateData($userId);
        $token = $this->buildAuthenticationTokenEntity($userId);

        if (!$this->AuthenticationTokens->save($token)) {
            throw new ValidationException(
                __('Could not update the authentication token data.'),
                $token,
                $this->AuthenticationTokens
            );
```

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

```
        }

        $this→dispatchEvent(RecoverCompleteServiceInterface
              ::COMPLETE_SUCCESS_EVENT_NAME, [
            'user' => $user,
            'data' => $this->request->getData(),
        ]);
    }
```

It is recommended to remove the PGP message containing the account recovery response once it was used. What is more, it might be advisable to re-encrypt the user's private key with a new random passphrase if it has been deployed already.

Keeping the same passphrase indefinitely means that once the passphrase is known to an attacker, they can use it to decrypt any future passwords the user places into Passbolt, whenever the adversary gains access to the database again.

PBL-07-005 WP1: Unusable organization key not rejected *(Low)*

Testing the account recovery setup showed that it is possible to upload an unusable PGP key which will make the account recovery feature unusable. When configuring the account recovery policy, the administrator has to upload the organization key. This key can either be generated from within Passbolt, or can be generated using an external tool and then imported into Passbolt. While generating the key within Passbolt is appropriate, any external key might not meet the requirements of the key consistent with what Passbolt would deem secure.

Testing showed that it is possible to import an RSA key which has only the capability to sign data. Specifically, such a key can be imported during the initial setup without error, but will consequently break the account recovery process. The following PGP private key can be used to test the issue. The password for the PGP private key is "*test*".

Proof-of-Concept:

```
-----BEGIN PGP PRIVATE KEY BLOCK-----

lQdGBGLRnF0BEAC6DZxA0ROTnN2K/t8p2581hRYUVZcL7QV765g4U/qG1zsRGtWv
DWRnJXxysbnxUODEXTJwwDr9u0j5wmvzFNgZUBu2b4ba6f3RNX41s0VMVls4t2U3
dPtfl95KqTq4ZLsXn5Z7jiStc8zXgsr7SBDVVay/UFs5cs8cJ00nsa4tiHg/KWo3
imvesa81x1cSQsD9f0ao7lGRzCwoyHcFpgrlWCKr2EpfkuPuqTL/+sy8jZfQWisH
H1z46oaSpevRYnElNV/GAE+ocQrMvKf2ninwe4BbTHcYdQtyCbIq+8K/9n5ChaIO
PuFRpAG9Z7GUCeTRS5deuoWvCBqla/DkJw2mYbzc/usV5TNd79tKRCnWEbUmQ2J7
yV75vhoAboF+XLu8an7cIS4pcuYU4QCis+kM9frZatts5lFpbiUua+iahS4DBhAJ
NSzRv/J/QXe2fgaueD8pglMMTZMvklIiYn0m8j69CqT/voYHGE/BHPH9t0GrQuIk
sH+dMQ2PQrtG/M+5QfqQCYtvUafG9rXe5VJNAa13jskVdh32s1XN5+xnPVxE6fo0
```

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

```
/mIJnAXw9D6wxksJGBOYLggeAlhymFWkYQiFBwyf3Z1gSBN5Hie9hIAvwWyGHQz/
WFqRqDOsiLzxn9JAhyX7IJ7lE2CkXrsqzvI5YVtIHNRovz+UVH0UQCjziQARAQAB
/gcDAtvoWYxRX8VG5cn0TfkQXIqK+b6G7Wg9iajTv7ZmzzHdsWKy01VqId4GyxN2
b/BWPILkhqQGZNxcyoJXqgrRFG6FoSRBG3IzlfchOFOLvj6ckb8TPhGY+06z/U6y
upZntHFocWaQTOKgfYrS+p2cl41+yzk+Tmt8SEa1k9BbWNAhpYdJWpzrNYKL4RF9
rirKMeBrRfT5PIHmryyTI/MXHbSf+iqoGFXDJnl3jPt4kyohAuiIIBLR0wmeV9AN
/e6/Ls3jUsf7zNJYEg/4g4zxPfia21K6gaHnsgHA4UrS7vb+dw7xAZBq/RGga6C3
GAeHFyvHYstEtyEn4/o3+e7dCqX5StYtmksvMeYms+yHY0fM89rGoPlP9qXCaKhX
kDS/YIK58tyWV5uTibYLKTJmvp42cL+/0p2koEli/BYmvcTF3LJxRnZjaoLa+mVc
0LQ8tg0X4I1Ic9m1ajFIE1iGMIOXYDCauM+P+wJS09H1ppiyGwlMxVeLifZsrGDZ
kfU7YhrnBStD8eQraccE9KEi+HsD/MWsuSvlJXpdyqhlKaTJXZVt0Bhsmcdupdgo
WcL7c5+/yLG/7kGbtNvuv5WTZXyqKFo6W4rxU6I0KsBG6Q6FV4VXrm9gV38/VxTC
uTykjJo8sLpNwyRBnHaSncQ9u1Y00XoLqmujOq2mFmViewKk+lTK6r2I5Hh8SW1t
cbaSZXiwdcjA4qRmVroZABEOY7yJEcNhEwGUvztLo2ajDVuQJMB/NSMGkne5D482
1Jv9naSSMce8PqGtk/SDBqeLBAK04T6bYclfQmqzOM8H9WQ8E0Qdl6bIKCsawUrh
DjaB+riMd8x8xaKYMCCt7h/UhQD5/Dm32cP5nILYwHX97iWbRu68Ja4RGUqzcTTQ
4oJ0ggbB0sDKbF4Og8hNdouOHbrp4Vb6JLZmTtao8hrMUeEr4jm3EwMXkYWO+bmH
ueHy5BxcvAo7w2huecuSZx52Y9kUceSIgLWx/RWqyEK3iGq0OpdadCX1iS+8wV83
u1u7hPTPyMEMVAVzN6NGbr7MbGd4p8tB/8V/MqtDV2xXzH1kQNjYf3zXY3z0MM7d
jwflGD6U7eqVkCjNR0IeJt2XlJU/xvOWOAHtVjEeDtCnxjCIOBEGk/cVVnLud03E
saN0ckhqLfnVLxCtXD3XEv2Nn5FYngNBfN9VSrUJJKFo1+qYnKsRmMW4m2Bcxuhi
kKSbPm3PRYKSxOwo/D33RjDSEHriQqJPuh51Xbdn7m7U2Jf7MUdwrynlRhWifM1z
KCVhWyyxhKmYu7n2H4meM3m1dLxifqTh8A8pYTyWh78Rj15OZPpiNFGKfLVLq4wY
nRWvZGA77k6O6e2FH6dPaNE7vuskSrT1rw+QprHvAQHlPAnLSFUGsdESGPWfj+yS
x9IVB13J39jBD6Qq3HGoq2msYCbkY9UB383hDnW17HHCg0VL/Un44/T4J3CG4WZk
X+j20q+bkolx9G6hojBLKdRlFs42qREbl7jtylqKk9z89agzXSlia3EQ6cYNiu5k
MX1DxKy3i83LGgTSKFZG1Afbt0Ntl0pr39cdZbmVh56/KrARz/3ThhkpF0n8LalA
jlsVvZ6jw6TaWy6snYsCFpzz2B3qqMpjSCdBg3Tn/RAKguTLqMRMjrzFiUJ/MIIc
Wxsiq/T6tdqAhA7lps4GuBgOn0VCUHVeIOcY47yakrC2hqTTt5+zFzmN/rmljWAP
tymiAJXFr655ezQBLig+m4W+6y+GaVPZDL/k+ha3DRhbS1p8EhlvnJi0L09yZ2Fu
aXphdGlvbiBLZXkgc2lnbm9ubHkgPGRhdmlkQHNpZ21hLXN0YXIuYXQ+iQJMBBMB
CgA2FiEE+spGsHOLhCb0jsbAy2mTW0kTYIYFAmLRnF0CGwMECwkIBwQVCgkIBRYC
AwEAAh4BAheAAAoJEMtpk1tJE2CGMm4QALS7vEZRla6nPH15IHq+XqpTnaCLM2ap
tOEtEq+xbN7YwfVE1R+rX+hR2N67ZeSfwzojvQyKM1OgJ5+39n0LcBIMkhduvdio
OyGky+PN2nws1echKTTG64fdPLY1E8BBkvBx36b29sKYKDVnArIegvtDQnU/ULK2
yQPKsn21t6ja+e07tQKHlqo5Sd7trtuisF/zYWTTrByAIllMb3y/zJq/oFWJJv1X
dZPfkMONR9FhrjQEHWm30D5TY2Xi2KkPkiqd+KDAd/e1LWMSAxe+uRhmRumJbt7M
S84Gb7YuKl6zv7kkMA8RsAWhAkRnGmLfEf29bMMIprlLB6Y1zLkP1Ia+AN18zTJ5
/ZRUlF3SuKvsmc5oF7pVi21pXKVJw8Y3F6EGhbT8JZYx4J3l++H2Vyjzl6zZyTru
5lBft63pSoQ6/ovacgP/u4VmLLLzhFFYL6WpRVuS0V3e0+viMuISm4Wx61UtmgrJ
segBtSFIfkxE1R/wMQDIYcsSbZSRXCPssHXqXi+Nm8pd0TrtQfVL8C2u/TN4Lb/C
fhQGRMvYZ/glnCUqgW6ECRBjZWJWd/5vphoYUaVxM2tDOwN6sqiNUHnnrf9+hBVh
11EYovNhwfQcPNhItkOkJBOHzapjk6jpavwJOTfD4Nhb9RoEgOI+MCse2rr7HDC1
eVYqFRwUm/Kn
=JKMZ
-----END PGP PRIVATE KEY BLOCK-----
```

It is recommended to inspect the public key properties for the capability to encrypt data before storing it as the organization key.

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

PBL-07-006 WP2: Missing consistent ruleset for PGP cipher requirements *(Low)*

While reviewing the PGP key validation in the backend and comparing it with the key generation logic from the frontend, it was noticed that there is no consistent ruleset for PGP key ciphers.

PGP keys include a multitude of properties regarding preferred cryptographic algorithm use. These include what symmetric ciphers and hash algorithms should be used when a sender encrypts data to this key. Since PGP has been around for a long time, the list of options includes weak ciphers which should not be used anymore. In other words, this pertains to ciphers which no longer meet current state-of-the-art cyber security standards.

Passbolt does not place any requirements on the keys but relies on the underlying PGP library to have proper defaults. While GnuPG and similar libraries strive to have good defaults and disable weak algorithms, this is not always the case for all libraries. As Passbolt uses multiple different PGP libraries throughout the codebase, this can become a problem when one library has different defaults for algorithms than the others do.

It must be noted that Passbolt already has the capability to reject DSA and ElGamal keys, as well as RSA keys with too small key sizes. This is good from a security perspective, but it is recommended to extend these checks to also validate the hash algorithms used for signatures, ensuring that only hash algorithms which are secure can be accepted.

That determines that *SHA-256*, *SHA-348* and *SHA-512* should be used. As for symmetric ciphers, only all AES variants should be allowed. For ECC keys, it is suggested to find the subset of commonly supported curves and verify that keys do not use any other curve. It was, for example, found that keys based on the curve *secp256k1* can be imported but will cause an error in the backend since they are not supported there.

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

Conclusions

Cure53 would like to congratulate the Passbolt team on achieving a solid security premise for their new features. After spending five days on examining the Passbolt test-targets, two members of the Cure53 team conclude the project on the positive note. The account recovery feature and the ECC key support, as well as the cryptographic premise, only require some minor work to be further strengthened. Cure53 was in constant communication with the customer through a dedicated Slack channel. The communication was excellent and help was provided whenever requested. For the sake of context, it should be clarified that this security assessment featured three repositories:

- *passbolt_browser_extension-master* containing the browser extension source code,
- *passbolt_pro-passbolt_pro_api-bf1c3e91031c* containing the PHP backend code
- *passbolt_styleguide-master* containing the web application code.
- Together, the three above repositories form the Passbolt password manager software.

The focus of this assessment was on whether the cryptographic operations were consistent and correct within the user-account recovery process. As Passbolt uses PGP, special attention was paid to the correct usage of the PGP library APIs. The backend of the application is written in PHP with the CakePHP framework. The frontend is written in JavaScript using React for user interface rendering. The frontent consists of a browser extension the user has to manually install and a web application served by the web server hosting the backend. While these two parts are tightly coupled, the clear code structure made it convenient for the testing team to find the interesting parts and audit them for correctness. Similarly, the backend code is well-structured and uses common design patterns, which renders understanding of the code quite easy.

Overall, one vulnerability and five miscellaneous issues were identified in the account recovery logic. None of these findings could be seen as fatal for the security of the feature and they underline that the developers have done a great job with this implementation. The concept of an account recovery feature for a hosted password manager with multiple users is generally a dangerous one from a security perspective. Subtle flaws in its design or implementation might lead to a full compromise, potentially affecting all user passwords. The architecture of this account recovery feature shows that the Passbolt team understood these problems and carefully designed their feature to hold up against malicious users and external attackers.

The recovery flow is well-designed and ensures that common flaws do not affect the implementation. One example of this is the initiation of the account recovery process. As

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

any user must be able to trigger this, this would normally be a great entry point for attackers. Passbolt ensures that while everyone can trigger this, the user has to manually confirm the recovery process via a dedicated email. Only then the administrator will be notified. Here the item again must go through a manual confirmation step and the request can be reviewed before it is accepted. The implementation also shows care for detail in a sense that all input provided to the backend by the client has to go through strict validation rules before it is accepted and processed. One place where this audit showed a bit of space for improvement is the parsing and validation of the PGP messages and keys. While not directly exploitable, the current code could be more strict to prevent error conditions and potential abuse.

The core of the Passbolt account recovery is a single PGP key, namely the organization key. The public key part is stored in the backend and used to secure backups of each user's PGP key, which secures the passwords of that user. This key is - as such - critical to the security of every user's passwords. As shown by the findings of this assessment, the current requirements placed on the key are sufficient, but could be improved to strengthen the overall security posture. Passbolt will never store the private key part of the organization key, but will place this responsibility on the administrator-user. On the one hand, this makes Passbolt more secure, since it is not possible to steal this key when gaining access to the backend. On the other hand, it will place the responsibility of securely storing this key on the administrator, which can be problematic when they have too little experience with this topic or process.

Currently, a malicious administrator would be fatal to the security of the Passbolt complex. This was discussed with the developers and they confirmed that it is planned to mitigate this by implementing a shared secret where multiple administrator-users have to collaborate to accept or reject every account recovery. This could certainly help reduce risks even further.

To conclude, this summer 2022 security review achieved very good coverage of all working packages / test targets. Cure53 can confirm that the test targets demonstrate security soundness in relation to the attempted attacks and areas covered. Moving forward, the Passbolt project could continue to benefit from recurrent security assessments of this feature in order to ensure the identified issues have been addressed accordingly. It is important to remember that changes within one part of the application may have an unintentional security impact on other parts. Thus, Cure53 advises for the future security reviews to be scheduled and performed, both in-house and externally.

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