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 Security White Paper 02.2021

Cure53, Dr.-Ing. M. Heiderich, Dr. N. Kobeissi

Index

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

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 review of a cryptography & security white-paper authored by the Passbolt team (labeled "Security White Paper Passbolt Pro Edition v3.0"), detailing on the security properties and architecture for the Passbolt Pro software.

The work was requested by the Passbolt in December 2020 and carried out by Cure53 in early and mid February 2021. A total of six days were invested to reach the coverage expected for this project and two members of the Cure53 team were involved in preparing, executing and finalizing the review and its resulting report document.

Cure53 was given access to the latest version of the paper available at the time of the review period. In addition, access to a document called "Passbolt Security v3 UX and Security Evolutions" was granted. Communications during the review were done using a dedicated Shared Slack channel in which the work-spaces of Passbolt and Cure53 were glued together and in which all involved personnel were invited into. Communications were very smooth and productive and no questions had to be asked, the scope was well prepared and absolutely clear, no noteworthy roadblocks were encountered during the test. Cure53 gave frequent status updates about the test and the related findings, sent over headlines of issues tracked already and kept the Passbolt team up-to-date.

The Cure53 team managed to get very good coverage over the paper review scope and managed to spot a total of fourteen findings, seven of which were classified to be security vulnerabilities and seven to be general weaknesses with lower exploitation potential. Note that no critical flaws were discovered, only two high priority recommendations were found to be adequate. The remainder of the issues spotted resides in the realm of medium to lower priority. The report will now shed more light on the scope and review target as well as the available material for testing. After that, the report will list all findings and remarks about the reviewed paper in chronological order, first the spotted vulnerabilities and then the general weaknesses discovered in this exercise. Each finding will be accompanied with a technical description, a PoC where possible as well as mitigation or fix advice.

The report will then close with a conclusion in which Cure53 will elaborate on the general impressions gained throughout this test and share some words about the perceived security posture of the scope that is the Security White Paper about Passbolt Pro Edition v3.0.

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

Scope

- **Cryptography Reviews & Audits against Passbolt Protocol Design & Architecture**
  - Cure53 was given access to a white-paper, review-supporting documentation and chat access to the Passbolt team for Q&A during the audits and reviews.
  - The reviews and audits focused on the following areas as requested by Passbolt.
    - Encryption & storage of private key and passphrase in the browser
    - Encryption & storage of private key and passphrase in mobile context (iOS & Android)
    - Private key backups and recovery procedures
    - Private key transfer procedure from browser extension to mobile
    - Encrypted content escrow procedures
  - Cure53 considered the given threat model of Passbolt and as well the general threat models that can be applied for team-oriented password manager software and infrastructure as well as mobile applications.

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

Verifpal Model

In order to inform this audit, a model of the Passbolt protocol was created using Verifpal, and is available on VerifHub.[1]

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-01-001*) for the purpose of facilitating any future follow-up correspondence.

PBL-01-001 Crypto: Secure Channel Enforcement Recommendations *(High)*

As per the *Data in Motion* section found on page 16 of the whitepaper, it was noted that Passbolt's current policy is not enforcing a secure channel for client-server communication. While it may be practically demanding in certain scenarios to enforce such a policy, it is safe to assume that obtaining authentication between client and server would help Passbolt neutralize a critical attack vector. An attacker that successfully mounts a Man in The Middle between a legitimate Passbolt client and server will effortlessly be able to add malicious resources to the server, modify or delete Resources that the client may have write access to, and obtain the encrypted secrets from such Resources. Elevating this scenario further, if the same attacker were to own an account on the server, they could obtain access to the decrypted secrets through using the Resource Sharing feature of Passbolt when applicable.

Moreover, leaving the option of setting up TLS to server administrators could lead to exposing the same attack vector should insecure versions or parameters of TLS be chosen.

The following TLS Setup is recommended:

- Protocols: TLS 1.3, TLS 1.2
- Cipher suites (TLS 1.3):
  - *TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256*
- Cipher suites (TLS 1.2):
  - *ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-*

[1] https://verifhub.verifpal.com/a833b60af83429dd99b60dc47456f5cb

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

GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384

- TLS curves: X25519, prime256v1, secp384r1
- Certificate type: ECDSA (P-256) (recommended), or RSA (2048 bits)
- DH parameter size: 2048
- HSTS: max-age=63072000 (two years)
- Certificate lifespan: 90 days (recommended) to 366 days
- Cipher preference: client chooses

Should setting up TLS and the management of Server Certificates be too much of a hassle, the use of one the following patterns from the Noise Protocol Framework, which don't rely on digital signatures, is recommended as a drop-in replacement to TLS.

Pattern Name	Pre-flight requirement	Initiator	Number of protocol messages before full security
IK	Pre-authenticated server public key	Client	1: The first Message sent by the server is fully secure
IX	None	Client	2: The second message sent by the client is fully secure

Symbolic Software provides Noise Explorer, an open-source tool using which production-ready cross-platform code for such patterns can be generated and analyzed in depth.

PBL-01-002 Crypto: Server-Side PRNG Recommendations *(Medium)*

Since Passbolt's supports deployment on native as well as containerized environments, the following recommendations are proposed for optimal results when it comes to Pseudo-Random Number Generation.

- Using the only-urandom configuration flag under */dev/gcrypt/random.conf* which disables the use of the blocking /dev/random call, replacing it with a call to /dev/urandom when applicable.
- Enforcing the use of an external entropy source or using a league of entropy provider[2] when possible to ensure that containerized builds of Passbolt don't get initialized with the same random seed which could yield identical server keys and predictable authentication tokens for all deployments.

[2] https://www.cloudflare.com/leagueofentropy/

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

- Ensuring that */dev/urandom* was properly initialized prior to using its output for pseudo-random value generation.

PBL-01-004 Crypto: Nonce Generation Recommendations *(High)*

Passbolt makes use of random nonces in its authentication subprotocol that are generated on both server and client side. In order to make sure that the generated nonce will in fact be used only once, adding to each principal's state a *nonce_key* to be generated (as per PBL-01-002 or PBL-01-003) and a corresponding *nonce_counter* initialized at 0 and incremented by 1 following each derivation is suggested. Derivations would be invoked as follows:

- On the server's side, GnuPG's *gcry_kdf_derive()* could be employed with *GCRY_KDF_SCRYPT* as the algorithm parameter, *nonce_key* as the input key material, nonce_counter as the salt, and the desired output length to derive a fresh nonce.

- The client's side could employ:
  - When using WebExtension: *SubtleCrypto.deriveKey()* with an *HkdfParams* object using SHA-256 as the algorithm parameter, nonce_key as the input key material, and nonce_counter as the salt.
  - When using the PHP: *hash_hkdf()* with sha256 as the algorithm parameter, *nonce_key* as the input key material, *nonce_counter* as the salt and the desired output length.

In any of these cases, the nonce_counter must be incremented after the function call so that the same resulting value after each invocation is never obtained.

Finally, replacing the nonce_key with a fresh value when *nonce_counter* reaches 65,535 then resetting the latter value to 0 is strongly recommended.

PBL-01-005 Crypto: Input Key Validation Recommendations *(Medium)*

In certain scenarios, Passbolt permits importing externally generated keyrings. It is proposed to exclusively support importing private keys while making sure that said key is a legal value which satisfies the minimum security requirements (PBL-01-014).

An invalid key may be the result of a weak prime factor in the case of RSA keys or could simply be of invalid format; either scenario would potentially introduce unwanted behavior that can be avoided at the key material parsing level.

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

PBL-01-006 Crypto: Client Registration Considerations *(Medium)*

Passbolt allows new users to register either through importing their keys as noted in (PBL-01-005) or generating new keys. These requirements add some caveats which must be accounted for in parallel to the recommendations for key generation (PBL-01-003). Then, whenever a private key is either generated or obtained, the corresponding public key should be derived and sent to the server such that the latter checks if any other user happens to be registered using the same public key, thus eliminating any risk of key collision.

In order to encrypt a client's keyring, it is advised to replace the use of a password and a call to haveibeenpwned with an offline alternative: a randomly generated passphrase. A five-word passphrase generated using one of the EFF's wordlists[3] and 5 random dice (or nonces as per (PBL-01-004) has been proven to be more secure (~2^{44} bits of security) than an 11-character combination of mixed-case alphanumerics and symbols (~2^{28} bits of security).

PBL-01-009 Crypto: Undefined Scenario for Removing User from Group *(Medium)*

Assuming a user was once part of a group with access to a certain Resource. The scenario of Removing the user from that would provide forward secrecy, as the user would no longer be able to access any newly added that the group was granted privilege to access. However, the user might have kept copies of the plaintext secrets before access was revoked. If this scenario falls under the threat model of Passbolt, it would be advised to enforce a credential reset, if applicable, whenever a user's access to a shared Resource is revoked.

PBL-01-010 Crypto: Deprecated HTTP Header *(Low)*

The *X-XSS-Protection 1; mode=block* HTTP header is currently deprecated and is only supported on legacy browsers. To provide modern browsers with comparable XSS protection, it is highly advised to add a Content-Security-Policy (CSP) header that disables inline JavaScript.

Configuring CSP as *default-src https:* could be a good starting-point. While this policy may be customized with additional parameters to suit Passbolt's requirements, it would be ideal to transfer all of the resources needed to load the client through a secure channel while segregating HTML and script code into separate files.

[3] https://www.eff.org/deeplinks/2016/07/new-wordlists-random-passphrases

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-01-003 Crypto: Client-Side Key Generation Recommendations *(Info)*

Passbolt makes use of a web client to target users across various platforms. Using *SubtleCrypto.generateKey()* rather than *Crypto.getRandomValues()* is advised for OpenPGP key generation operations on the WebExtension Client. Noting that this would not be as much of an issue on a client's side as it on the server's, ensuring that */dev/urandom* was properly initialized before calling *SubtleCrypto.generateKey()* on the WebExtension or *random_bytes()* on PHP would be beneficial.

PBL-01-007 Crypto: Server Setup Extensions for HSM Support *(Info)*

Passbolt's server setup could be extended to add support for managing the server's private key and escrow key(s) when applicable using hardware security modules (HSMs) which would minimize the threat in case of compromise and aid in recovery. Ideally, the server's keypair as well as any escrow keys would be generated using a Threshold Cryptosystem such as Shamir Secret Sharing.

This would accompany a hierarchical security policy where for each operation to be performed on the server, depending on severity, a specific number of shares would be to be present physically at the time of execution.

It is currently unclear whether server keys are strictly generated, or can be imported during the initial server setup. Should that be one of Passbolt's business requirements, it would be recommended that the instructions in (PBL-01-005) and (PBL-01-006) for key setup be followed.

PBL-01-008 Crypto: Authentication Protocol Considerations *(Info)*

Conflicting information was found when comparing the "Authentication" section from the API reference and the "Login Steps" section in the Whitepaper. The protocol described in the API reference was taken into consideration as it seemed to be the more accurate version. Assuming the execution of the authentication protocol under a secure channel as expressed in (PBL-01-001), there were no major security findings.

A Verifpal model of the Authentication protocol and the Resource Sharing protocol can be found attached to this report.

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

Furthermore, there exists a certain amount of redundancy in sending the client's public key fingerprint to the server multiple times during a single protocol execution. Sending the client's public key fingerprint once after the establishment of a handshake should be enough for the protocol to be executed correctly.

PBL-01-011 Crypto: Commit Signatures Considerations *(Info)*

Requiring commit signatures by Passbolt developers to complement what is listed under the "Other Best Practices" section in the Whitepaper is advised. A malicious actor may forge a legitimate Passbolt contributor's identity by attaching the latter's name and email to commits that induce vulnerabilities on a forked Passbolt repository. Digitally signed commits would thus enable the open source community to fork and modify Passbolt while being assured that commits authored by genuine Passbolt contributors reaffirm Passbolt's guarantees of quality and security.

PBL-01-012 Crypto: Cross-Device Secret Key Transfer Protocol Revision *(Info)*

While the protocol proposed under the "Enable cross-device secret key transfer using QR Codes" section appears to be functional, an alternative, simpler protocol which relies on the light cross-platform TXQR[4] library is a solution which does not require managing server pagination. The solution proposed would not dismiss the role of the server nonetheless. A number security caveats must be addressed in order for the transfer protocol to be truly complete whether Passbolt opts to transition to TXQR or not:

- An additional data validation layer could be added for extra assurance while serializing/deserializing QR data.
- The QR data could be generated using a Transient-Key Cryptographic Scheme which allows its data to be decrypted during a specific temporal interval with the aim of disallowing the use of QR codes as long-term backup recovery tokens.
- A revocation scheme could the above and when used if a transfer is in any of the defined "Cancelled", "Error", or "Complete" states from the "Fig. QR Code Exchange State Diagram".

PBL-01-013 Crypto: Post-Quantum Public-Key Cryptography Adoption *(Info)*

While no quantum computer that threatens the security of public-key cryptographic primitives such as RSA, ECDSA, etc… exists as of the date of the formulation of this report, the emergence of such technology has been imminent and anticipated by the Cryptographic community. In anticipation of this event the National Institute of Standards and Technology (NIST) initiated the process of Post-Quantum Cryptography (PQC) standardization in 2016. The NIST PQC competition has recently announced the final

[4] https://github.com/divan/txqr

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

candidates of its third round and is aiming to publish standardized drafts before the year 2024.

Given that the cornerstone of Passbolt is a public-key primitive which can be targeted by a powerful-enough quantum computer in the future, all recorded secrets and client-server communications would be rendered retroactively vulnerable and reducible to plaintext. In order to mitigate this risk and protect data from future attacks, it is advisable to transition to post-quantum primitives when deemed possible.

PBL-01-014 Crypto: Keyring Minimum Security Requirement Definitions *(Info)*

NIST's Special Publication 800-57 Part 3 Rev. 5[5] recommends using combinations of primitives and keys that provide a minimum of 112 bits of security. For purposes of compliance and better security overall, Passbolt would be therefore required to withdraw support for primitives that do not fulfill the aforementioned requirement.

Applying said advisory can be reflected through not accepting primitives weaker than RSA-2048 or its equivalent. The additional precautionary step of supporting primitives with no less than 128 bits of security provides better assurance with a negligible performance drawback (i.e RSA-3072, x25519, and equivalents, or better).

[5] https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-5/final

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

Conclusions

This project centered around a critical review of the latest cryptographic design specification drafts for the Passbolt software. The cryptography of the project was examined by two members of Cure53 over the course of six days, with the results amounting to seven findings of varying severity and seven informational findings.

The Passbolt team provided complete, clear cryptographic specifications on time and this was central to the production of a complete audit with valuable results.

Despite the relatively large number of findings, all findings were written from a critical perspective of the specification, and therefore do not necessarily have a practical impact on the software implementation for Passbolt. PBL-01-001 and PBL-01-004, which deal with transport layer encryption as well as nonce generation and management for authentication operations, were marked as high severity because they outline omitted information in the provided specifications that could cause implementation decisions to be made with high potential security impact. PBL-01-002, PBL-01-005, and PBL-01-009 also deal with undefined behavior in the specifications, but which less severe potential consequences on software implementations.

The remainder of the issues, most of which are informational-severity, deal with recommendations towards hardening the specification without touching upon any details that could have a current tangible security impact on the security of the Passbolt cryptographic design.

In conclusion, while Passbolt's existing cryptographic specification design could benefit from clearer and more complete definitions in certain areas (as outlined in this report), the specification itself illustrates a relatively well-designed protocol with no immediate outstanding security issues.

Cure53 would like to thank Remy Bertot, Thomas Oberndörfer and the rest of the Passbolt team for their excellent project coordination, support and assistance, both before and during this assignment.