Quick Security Evaluation

openEngiadina

NGI0 Zero

V 1.0
Amsterdam, January 31st, 2022
Public

1 Introduction

As part of the NLnet projects your project **openEngiadina** has received a basic quick security evaluation from Radically Open Security. The goal of the review is to provide advice and input to consider in the further development of your project. The selected project gets 2 person days plus an extension of 1.5 person days from ROS for a quick security evaluation.

2 Security Evaluation For openEngiadina

Tasks Performed

- Discussion with representatives of the project in the ROS chat, issue tracker and via video conferences.
- Reading of background material and public design discussions on the mailing list.
- Reading of related posts on the activitypub discussion forum and inqlab.net.
- Watching the ERIS: Encoding for Robust Immutable Storage presentation.
- Discussing the scope of the security evaluation, project requested a strong focus on the ERIS specification.
- Reviewing the ERIS specification in version 0.3.0, which from here on will be referred to as *the specification*.
- Interactive discussion in the issue tracker on a number of security and functionality aspects of the specification.

Please note that the OpenEngiadina repositories and code were not analyzed.

Security Considerations

- The specification contains multiple `http://` link references. We recommend switching to `https://` where possible (issue #4).
- The specification makes use of the purl.org link shortener service for multiple document references. This brings with it a number of risks, notably discontinuation of the service at some point in the future and an attacker gaining sufficient access to manipulate the link references. According to the project representatives, these redirections are meant to serve as dynamic web references that can be updated at some point in the future, should the URLs of certain essential resources change. Purl.org is run by the Internet Archive non-profit organization (issue #5).
- Aspects of the specification regarding `CONFIDENTIALITY` vs. `INTERMEDIARY PEER DENIABILITY` security goals were discussed. As a short summary, the current ERIS specification draft does not provide full `CONFIDENTIALITY` in any usage mode due to the custom cryptographic construction for convergent encryption which always generates the same ciphertext for blocks with identical plaintext content. Under some conditions, this allows attacks that break the confidentiality of some of the encoded information (issue #6).

- The specification description of the ChaCha20 usage in §2.1.2. is misleading with regard to the key handling, as it suggests that the encryption key is never reused. For a given convergence secret, all identical data blocks will re-use the same derived ChaCha20 encryption key, well-known IV and well-known counter combination to generate identical encrypted blocks. This construction is intentional as described in §2.3., but should be documented better (issue #7).
- The specification requires compatibility with both the 1KiB and 32KiB ERIS block size for any inputs. However, the inherent storage inefficiencies in using large storage blocks for short inputs (and vice versa) will likely result in a very predictable block size choice based on input size during encoding, as recommended in the specification itself. This presents a clear opportunity for attackers to determine if the original input data was larger than the threshold (16KiB - 1 byte) without access to the read capability of the ERIS-encoded data. This information can be problematic for the stated goal of `INTERMEDIARY PEER DENIABILITY` and `CENSORSHIP RESISTANCE` since intermediary peers are well aware of the block size. As an attack, censors could enforce that all 32KiB ERIS blocks are rejected on the peer or network level, which would prevent the exchange of most existing larger files above the threshold *without* blocking the exchange of most smaller files (issue #9).
- Assuming that a peer-to-peer network is used to fetch blocks of new content, it is to be expected that attackers run intermediary peers or perform surveillance on non-malicious peers. Over time, traffic analysis on block accesses can reveal statistical information about block clusters as well as block to client relationships, and this information can be recorded permanently. The practical attack details depend on the implementation, but we think it is generally plausible that under the right conditions an observer can identify a large percentage of blocks that belong to an ERIS encoded file and reconstruct parts of the ERIS tree structure relationship between blocks based on access patterns. This is simplified if the structure in question is accessed repeatedly or only available from a peer that is under observation. A leak of ERIS block plaintext to ERIS block ciphertext mappings, for example through an ERIS URN that becomes known to an attacker later, may be used for retroactive actions against clients which are known to have requested some of the associated blocks at some point. This scenario is not represented in the official threat model. (issue #11).
- The ERIS design has integrity protections on node blocks and data blocks through the use of cryptographic hashes in a tree structure (Merkle tree). Notably, the URN itself is not protected by any integrity check or signatures and susceptible to getting replaced with a modified URN in transit. A full replacement of the URN can lead to the various impacts, including downloading attacker-controlled files instead of the intended files. The threat model of the specification requires `PEER ENTITY AUTHENTICATION` and `CONFIDENTIALITY` to ensure trust in the messages that are passed between publisher and audience to address this attack potential. However, these properties do not guarantee `DATA INTEGRITY`, so the official threat model of the reviewed specification version does not rule out manipulations of the message exchange that is containing the URN. Note that the specification authors have changed the threat model as a result of this evaluation (issue #16).

- Practical implications of issue #16 were discussed. Partially modifying the URN without knowing the original URN content has different impacts depending on the affected section. We discussed the implications of an attacker with the ability to trigger undetected bitflips in the message between publisher and audience. For example, partial data corruption of the `root reference key pair` section of the URN leads to a denial of service as the ERIS data of the root reference block cannot be found or decrypted correctly.
  We have identified two attacks that involve a modification of the `level of the root reference` section which can be triggered with minor message modifications:
  - In the first attack scenario, the original level is *decreased* by the attacker, which makes the client download a subset of the intended ERIS tree structure with reduced height and misinterpret node blocks as data blocks. In most cases, this will fail at the decoding stage, but there is some probability that the data tail happens to match the expected padding data structure. This results in the client incorrectly using the reference-key pair data as the decoded plaintext file.
  - In the second attack scenario, the original level is *increased* by the attacker, which makes the client download and decode all of the available ERIS structure data, but the client expects there to be one or more additional levels. Unless some specific edge cases are hit which abort the decoding, the client will misinterpret the data blocks as node blocks and convert the correctly decrypted plaintext into a list of reference-key pairs (64 byte per pair). Looking for additional blocks that do not exist, it then asks its block source (likely a peer-to-peer network) for blocks that match these references, *leaking half of the plain-text data to the network peers in the process*.

  We consider these attacks through the level metadata as problematic and recommend investigating mechanisms and design improvements that prevent them in a more general way (issue #13).
- We discussed the possibility of blocks which are valid for decoding in both the 32KiB and 1KiB block size variants (first 1KiB prefix of the 32KiB file). We think that such a special block can only exist if there is a hash collision in blake2b-256 for the given block data to satisfy the integrity check before the decoding, which is not seen as a practical risk (issue #14).
- The specification is not clear on the exact accepted or required format for nodes if there are one or more `null reference-key pairs`. If there is a null reference-key pair followed by non-null reference key pairs in a node, ERIS implementations may handle this situation differently by aborting, processing the additional pairs or not processing the additional pairs. As a result, the decoded file may be different between two implementations, which is an undesirable outcome (issue #17).
- We discussed the potential use of metadata on the block layers in future specification variants. Changing the design to include a metadata header area in some or all of the blocks would allow the inclusion of more ERIS structure data, a block type identifier or additional integrity components which counteract some of the attacks outlined in other issues. Adding metadata increases complexity and reduces efficiency, so this is offered as part of complex considerations for the specification design and not as a straightforward recommendation (issue #18).

- The specification makes some effort to outline and explain the role of the `convergence secret` and related implications of using either a well-known null convergence secret, a privately shared static convergence secret or a random nonce as convergence secret. However, the involved security considerations are complex and may result in end users over-relying on ERIS security aspects that are not appropriate for a certain usage mode, or in the presence of adversaries that do not fit the official threat model. Therefore we recommend representing the weaker modes with simple names and descriptions that *lower* security expectations, especially for the null convergence secret. Ideally, ERIS-related software should default to the strongest mode unless instructed otherwise by the user.
  The related technical discussion explored potential problems for the low-entropy plain-text content of a block when using a shared convergence secret that it known to the attacker. For example, if the plain-text data content before encoding is slightly longer than a multiple of the block size, a short tail section of the plain text is encoded into a separate block with padding and may be vulnerable to brute-force decryption attempts (issue #19).
- Based on our understanding of the specification, implementations are not strictly required to verify the hash integrity of each block during decoding, which leaves them vulnerable to a number of attacks if they skip verifications. We think this can be clarified in the standard. (issue #23).

Recommendations

- The ocaml-xmppl repository contains domain and path information for a private SSH host (issue #2).
- We highlighted a number of grammar and syntax issues in the specification as well as other minor specification text observations (issue #3).
- The padding scheme in use for the specification can be documented better as an existing standardized scheme. Unfortunately, the relevant ISO standards are not publicly available, although indirect descriptions have been published on Wikipedia and in other places (issue #8).
- The description in `§2.2.1` has an off-by-one error at "`Content larger than 1KiB`" (issue #10).
- The ERIS tree example in `§2.4`, `figure1` represents an edge case where the nodes are completely filled with reference-key pairs. We recommend additional figures with other typical valid tree layouts (issue #12).
- The specification currently has a number of under-specified areas, for example related to the handling of null reference-key pairs that are used as placeholders. We recommend extending the pseudocode or clarifying it via comments to outline the expected implementation behaviour (issue #15).
- The potential use of compression algorithms such as Zstandard was discussed with the project representatives. At the moment, they do not plan on using compression within the standard while evaluating practical compression use cases on the data layer, so this was not evaluated from a security perspective (issue #20).
- We recommend documenting a public security contact for project repositories, for example via a `SECURITY.md` file and a `security@` email contact (issue #22).

- We recommend extending the test vector collection with more accepted and non-accepted test cases that trigger edge case behaviour so that implementations can better test their correctness.

3 Contact details

Your pentester for this project:

- Christian Reitter
  Christian is an IT Security Consultant with experience in the area of software security and security relevant embedded devices. After his M. Sc. in Computer Science, he has worked as a developer and freelance security consultant with a focus on fuzzing research. Notable findings include several firmware vulnerabilities in popular cryptocurrency hardware wallets such as remote code execution, remote theft of secret keys and circumvention of 2FA protection. He has also discovered multiple memory issues in well-known smartcard driver stacks.

If you have any questions about this advice, please contact us at info@radicallyopensecurity.com

For more information about Radically Open Security and its services please visit our website: www.radicallyopensecurity.com.

4 Disclaimer

This evaluation is not to be considered a full audit or pentest. It is important to understand the limits of ROS' services. ROS does not (and cannot) give guarantees that something is secure. Additionally, the above advice is obviously incomparable to a full-blown security audit as performed by ROS or any other professional security company, which takes orders of magnitude more time and effort. Such an audit may be necessary in the future still, and we would be happy to work with you on that.

We recommend for now you treat our feedback as a discreet sanity check by knowledgeable friends; it is not the intention to publicly claim that Radically Open Security audited your code and found it to be safe or 'found no problems'. Rather, the intention is the reverse: we aim to help you capture obvious flaws within the very limited terms of this deal, and to protect the general audience from irresponsible projects putting them at risk. NGI Zero takes the public interest very seriously and wants to have at least a rough understanding of the maturity of your work - hopefully this will guide your own behaviour in terms of any claims you make.

There is no shame in clearly messaging to users that your project is still in an early stage, that they use it at their own risk and that there are no guarantees - being honest with your users can only ever be a good thing.