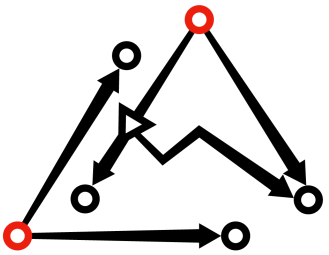

Least Authority

PRIVACY MATTERS

Worldcoin Protocol Cryptography

Security Audit Report

Tools for Humanity Corporation

Final Audit Report: 26 July 2023

Table of Contents

Overview

Background

Tools for Humanity Corporation has requested that Least Authority perform a security audit of the Worldcoin Protocol Cryptographic implementation.

Project Dates

- **April 10 - May 15:** Initial Code Review *(Completed)*
- **May 17:** Delivery of Initial Audit Report *(Completed)*
- **July 17-20:** Verification Review *(Completed)*
- **July 26:** Delivery of Final Audit Report *(Completed)*

Review Team

- Mehmet Gönen, Cryptography Researcher and Engineer
- Anna Kaplan, Cryptography Researcher and Engineer
- Ramakrishnan Muthukrishnan, Security Researcher and Engineer

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the Worldcoin Protocol Cryptographic implementation followed by issue reporting, along with mitigation and remediation instructions as outlined in this report.

The following code repositories are considered in scope for the review:

- semaphore-rs repository: https://github.com/worldcoin/semaphore-rs
- semaphore-mtb repository https://github.com/worldcoin/semaphore-mtb
- developer-portal repository: https://github.com/worldcoin/developer-portal

Specifically, we examined the Git revisions for our initial review:

- semaphore-rs repository: `65043e0c7a2613025db7d9d13182f3696ca763f9`
- semaphore-mtb repository: `b8bd950f138beae8e3147a784bcaffb2d915708e`
- developer-portal repository: `4a19e19f118ddc77e71bd09584285203c65d4a83`

For the verification, we examined the Git revisions:

- semaphore-rs repository: `a405f4c36d102fa64db0b80affbd129188acc99a`
- semaphore-mtb repository: `d878c5a4e226d0437db3e3f80f55a7416e4ea663`
- developer-portal repository: `5b651c0376ee5aeb1711e2c72476dd02eee78280`

For the review, these repositories were cloned for use during the audit and for reference in this report:

- semaphore-rs repository: https://github.com/LeastAuthority/Worldcoin_Semaphore_Rust
- semaphore-mtb repository https://github.com/LeastAuthority/Worldcoin_Semaphore_MTB

- developer-portal repository:
  https://github.com/LeastAuthority/Worldcoin_Developer_Portal

All file references in this document use Unix-style paths relative to the project's root directory.

In addition, any dependency and third-party code, unless specifically mentioned as in scope, were considered out of scope for this review.

Supporting Documentation

The following documentation was available to the review team:

- Website:
  https://worldcoin.org
- Protocol overview:
  https://docs.worldcoin.org

In addition, this audit report references the following documents and links:

- R. Barbulescu and S. Duquesne, "Updating key size estimations for pairings." *IACR Cryptology ePrint Archive*, 2017, [BD17]
- A. Bariant, C. Bouvier, G. Leurent, and L. Perrin, "Algebraic Attacks against Some Arithmetization-Oriented Primitives." *IACR Transactions on Symmetric Cryptology*, 2022, [BBL+22]
- E. Barker, "Recommendation for Key Management: Part 1 – General." *NIST Special Publication*, 2020, [Barker20]
- S. Bowe, A. Gabizon, and I. Miers, "Scalable Multi-party Computation for zk-SNARK Parameters in
- the Random Beacon Model." *IACR Cryptology ePrint Archive*, 2017, [BGM17]
- L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "POSEIDON: A New Hash Function for Zero-Knowledge Proof Systems." *IACR Cryptology ePrint Archive*, 2019, [GKR+19]
- L. Grassi, D. Khovratovich, and M. Schofnegger, "Poseidon2: A Faster Version of the Poseidon Hash Function." *IACR Cryptology ePrint Archive*, 2023, [GKS23]
- J. Groth, "On the Size of Pairing-based Non-interactive Arguments." *IACR Cryptology ePrint Archive*, 2016, [Groth16]
- K. Gurkan, K. W. Jie, and B. Whitehat, "Semaphore: Zero-Knowledge Signaling on Ethereum." *Ethereum Foundation*, 2020, [GJW20]
- T. Kim and R. Barbulescu, "Extended Tower Number Field Sieve: A New Complexity for the Medium Prime Case." *IACR Cryptology ePrint Archive*, 2015, [KB15]
- A. Menezes, P. Sarkar, and S. Singh, "Challenges with Assessing the Impact of NFS Advances on the Security of Pairing-based Cryptography." *IACR Cryptology ePrint Archive*, 2016, [MSS16]
- T. Perrin, "Curves for pairings." 2016, [Perrin16]
- M. Schofnegger, "calc_round_numbers.py." 2021, [S21]
- The Heartbleed Bug:
  https://heartbleed.com
- SetFinalizer mechanism:
  https://pkg.go.dev/runtime#SetFinalizer
- Crate zeroize:
  https://docs.rs/zeroize/latest/zeroize
- Hashicorp Vault:
  https://www.vaultproject.io
- Docker Secrets:
  https://docs.docker.com/engine/swarm/secrets
- BSI TR-02102-1: "Cryptographic Mechanisms: Recommendations and Key Lengths" Version: 2023-1:

https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf
- Asymmetric key specs: https://docs.aws.amazon.com/kms/latest/developerguide/asymmetric-key-specs.html
- Semaphore Protocol: https://semaphore.appliedzkp.org/docs/glossary#trusted-setup-files
- Semaphore Trusted Setup MPC Data: https://storage.googleapis.com/trustedsetup-a86f4.appspot.com/semaphore/semaphore_top_index.html
- PSE Snark artifacts: https://www.trusted-setup-pse.org/#Semaphore

Areas of Concern

Our investigation focused on the following areas:

- Correctness of the implementation, including cryptographic constructions and primitives;
- Common and case-specific implementation errors;
- Adversarial actions and other attacks on the code;
- Key management: secure key storage and proper management of encryption and signing keys;
- Exposure of any critical information during user interactions;
- Resistance to Distributed Denial of Service (DDoS) and similar attacks;
- Vulnerabilities in the code leading to adversarial actions and other attacks;
- Protection against malicious attacks and other methods of exploitation;
- Performance problems or other potential impacts on performance;
- Data privacy, data leaking, and information integrity;
- Inappropriate permissions, privilege escalation, and excess authority; and
- Anything else as identified during the initial analysis phase.

Findings

General Comments

Our team performed a security audit of the Worldcoin Protocol cryptographic design and implementation. The scope of the review included Worldcoin's Rust implementation of the `semaphore` protocol and the Go implementation of the Semaphore Merkle Tree Batcher (SMTB). In addition, our team reviewed the Worldcoin `developer-portal`, a set of tools intended to facilitate interaction with the Worldcoin Protocol.

`semaphore-rs` is adapted from `zk-kit` in Rust and uses `ark-circom` for generating Groth16 zk-SNARKs. In our review, we investigated the randomness generation for the nullifier use and could not identify issues. We also specifically checked the implementation of the circuit and protocol in `semaphore-rs`, and could neither find deviations to the implementation of `zk-kit`, nor identify issues within it.

During our review of `semaphore-mtb`, our team analyzed the use of the "lazy Merkle tree construction" and did not identify issues. Our team found that since all of the roots and leaves used in this structure are public values, it is not necessary to take any protection against side-channel attacks, specifically to construct invalid inclusion proofs. The implementation of the SMTB uses `gnark` for generating Groth16 zk-SNARKs to ensure correctness of the batched Merkle tree update. Our team could not identify issues

with the usage of `gnark` besides that the trusted setup is performed by the party proving the correctness of a Merkle tree update. We recommend performing a trusted setup (Issue C).

Our team reviewed the `developer-portal` for potential security issues and identified areas of improvement in the storage of sensitive data and AWS key management service signing mechanism. We recommend the use of secret stores and secure signing algorithms (Suggestion 1, Suggestion 5).

We found that the cryptographic component of the Worldcoin Protocol is generally well-designed and implemented. Given the stated Worldcoin objective to help verify humans with biometric data in a secure and privacy-preserving way, in the tradeoff between security and efficiency, we recommend that security be the priority. Our teams identified areas of improvement to the level of security of the protocol's cryptography.

System Design

Our team found that security has been taken into consideration in the design of the Worldcoin cryptographic components. However, we found that there is no trusted setup to compute the Common Reference String (CRS) for the SMTB, and that the CRS from another team is used for the Rust implementation of the `semaphore` protocol, which undermines the security guarantees of the zk-SNARK proof (Issue C). We also found that secret values are not cleared from memory appropriately (Issue A).

In addition, we found that the security level of the cryptographic system is below the recommended level for this type of use case, which can be considered as critical. We recommend that the security level of the system target the recommended higher end of the range. To help achieve this, we recommend using an alternative curve to BN254 to meet minimum security standards (Issue B). We also recommend configuring the Poseidon hash functions to achieve a higher level of 128-bit security (Suggestion 2).

Code Quality

We performed a manual review of the repositories in scope and found the codebases to be generally well-organized. However, we found unresolved TODOs, and unused code and packages, which we recommend be resolved or removed (Suggestion 6, Suggestion 3).

Tests

Sufficient tests are implemented in `semaphore-rs`, `semaphore-mtb`, and the `developer-portal`, which help identify implementation errors that could lead to vulnerabilities and check that the system behaves as intended.

Documentation

Our team found the documentation describing the protocol to be generally sufficient. However, there is no technical specification for the system that outlines the cryptography used and describes in technical detail how `semaphore` is implemented in the Worldcoin Protocol (Suggestion 4).

Code Comments

The three codebases are generally well-commented, with security-critical functions and components being accurately described.

Scope

The scope of this review was sufficient and encompassed the cryptography of the Worldcoin Protocol. Further audits are planned for the Worldcoin Orb biometric tool as well as a Holistic Review of the Worldcoin Protocol.

Dependencies

Our team did not identify issues in the use of the dependencies `zk-kit`, `ark-circom`, and `gnark`.

Specific Issues & Suggestions

We list the issues and suggestions found during the review, in the order we reported them. In most cases, remediation of an issue is preferable, but mitigation is suggested as another option for cases where a trade-off could be required.

ISSUE / SUGGESTION	STATUS
Issue A: Secret Values Are Not Zeroized	Resolved
Issue B: The BN254 Curve Provides Insufficient Security	Unresolved-Planned
Issue C: Current CRS Generation Renders Proofs Insecure	Resolved
Suggestion 1: Do Not Store and Access Secrets in the Environment Variables	Resolved
Suggestion 2: Configure Poseidon Hash Function Parameters for a 128-Bit Security Level	Unresolved-Planned
Suggestion 3: Remove Unused Code From the Codebase	Resolved
Suggestion 4: Create a Semaphore Technical Specification for the Worldcoin Use Case	Resolved
Suggestion 5: Change AWS Key Management Service Signing Mechanism	Unresolved-Planned
Suggestion 6: Resolve TODOs in Codebase	Partially Resolved

Issue A: Secret Values Are Not Zeroized

Location

`semaphore-mtb/prover/proving_system.go`

`semaphore-rs/src/identity.rs`

Synopsis

Our team found instances of secret data not being cleared from memory appropriately. Clearing secrets from memory once they are no longer needed is a known mitigation to memory-dump-based attacks. An attacker that is able to access the memory (thereby exploiting vulnerabilities such as the Heartbleed attack) may be able to retrieve secret values.

Impact

The leakage of cryptographic secret values (seeds or keys) could result in the loss of security properties, such as confidentiality and privacy.

Preconditions

An attacker must be able to read memory regions that contain sensitive data.

Feasibility

Although this type of exploit requires skill, little computational or financial resources are needed.

Mitigation

For Golang codes, we recommend using the `SetFinalizer` mechanism set in the object constructor to clear the memory, despite the fact that it does not guarantee that the secret values will be completely removed from memory.

We also recommend using the `zeroize` Rust crate for zeroization.

Status

The Worldcoin team implemented the `zeroize` Rust crate for zeroization.

Verification

Resolved.

Issue B: The BN254 Curve Provides Insufficient Security

Location

`semaphore-rs/src/circuit.rs`

Synopsis

The Worldcoin Protocol utilizes the BN254 curve in the proving system. In 2016, advances in number theory led to a lower security estimate for this curve. Specifically, its security is now considered to be around 96-bits. This is significantly lower than the 112-bits required by the National Institute of Standards and Technology (NIST) for new products.

Impact

Using the BN254 curve undermines the security of the zk-SNARK scheme, such that the feasibility of computing forged proofs that check as valid cannot be ruled out. Such proofs would pass validation, yet violate the constraints imposed by the circuit.

Preconditions

No preconditions are necessary.

Feasibility

The exact feasibility is difficult to estimate. However, the potential gains from a successful attack are high, which suggests that an attacker would have incentive to invest significant resources.

Technical Details

Attacks based on the Tower Number Field Sieve (TNFS) and the derivatives exTNDS and SexTNFS have led to a reduced estimate of the security level of BN254. Scholars and practitioners working on this issue do not entirely agree on the new estimate, but opinions range from 96-bits to 110-bits ([KB15], [MSS16], [BD17], [Perrin16]). Regardless of where on this spectrum the actual value falls, it is still lower than the recommended minimum. Even for applications that only need to remain secure until 2030, NIST requires a security level of at least 112-bits ([Barker20], Table 4), which BN254 does not achieve.

Mitigation

We recommend using an alternative elliptic curve such as BLS12-381, which has approximately a 120-bit security level.

Status

The Worldcoin team acknowledged the Issue and stated that they plan to upgrade the larger proof system, which would resolve this Issue. This is blocked on Ethereum having EVM support for other pairing curves. Hence, at the time of the verification, this Issue remains unresolved.

Verification

Unresolved.

Issue C: Current CRS Generation Renders Proofs Insecure

Location

`semaphore-rs/Cargo.toml#L39`

`semaphore-rs/build.rs#L10`

`semaphore-mtb/prover/proving_system.go#L7-L11`

Synopsis

The Worldcoin Protocol utilizes the Groth16-based zk-SNARK proof-system [Groth16] through the use of `ark-circom` and `gnark`. Groth16 produces preprocessing zk-SNARKs, and their security is based on the existence of a trusted third-party to compute the Common Reference String (CRS) in a preprocessing trusted setup phase.

In `semaphore-rs`, the correct use of the Groth16 zk-SNARK is critical for the unforgeability of the zk-SNARK in the `semaphore` protocol. The Semaphore protocol team mentions that they performed a trusted setup through Powers of Tau on May 29, 2022, for generating their CRS. The Worldcoin team uses the CRS generated by this trusted setup. Since this trusted setup is not performed by the Worldcoin team, and the implementation and execution of the Powers of Tau ceremony is not audited, a potential malicious party could have the ability to forge proofs.

In `semaphore-mtb`, a Groth16 zk-SNARK is being used for verifying the correctness of the Merkle tree update. All inputs to the zk-SNARK are public inputs and, therefore, the Merkle tree can also be constructed locally. In the current design of `semaphore-mtb`, the prover and the executor of the trusted setup are the same entity, which renders all proofs insecure. If the computation of the CRS is performed by a single party, that party has the ability to generate forged proofs about the correctness of the Merkle tree update.

Impact

For `semaphore-rs`: Since the Powers of Tau ceremony can potentially not be trusted, a malicious prover can forge proofs at will, which would be undetectable by any other party. As a result, any proof must be considered potentially forged.

For `semaphore-mtb`: As long as the prover is the only participant in the CRS generation, the prover can forge proofs at will, which would be undetectable by any other party. As a result, any proof must be considered potentially forged.

Preconditions

For `semaphore-rs`: The Powers of Tau ceremony by Semaphore must either be performed at fault or the implementation must be at fault.

For `semaphore-mtb`: The prover must be the only independent participant that contributes randomness to the initial trusted setup phase.

Feasibility

The attacks are straightforward and well-known.

Remediation

We recommend investing in the development of a proper multi-party computation (MPC) to generate the CRS and performing the multi-party computation in an independent manner. Proper and well-tested methods like the Powers of Tau [BGM17] are known and can be executed in a permissionless way.

In addition, the implementation of the Powers of Tau ceremony by Semaphore was out of scope for this audit. Although such an implementation is better than having the prover and executor of the trusted setup be the same entity, our team does not recommend using a Powers of Tau ceremony performed by a different project and instead recommends performing a trusted setup for the Worldcoin project specifically.

Status

The Worldcoin team stated that two CRSs are implemented in the system: `Semaphore` and `MerkleTreeBatcher`. The Semaphore CRSs utilized are those published by PSE, while the `MerkleTreeBatcher` CRS was generated using a hybrid trusted setup ceremony.

Verification

Resolved.

Suggestions

Suggestion 1: Do Not Store and Access Secrets in the Environment Variables

Location

`Worldcoin_Developer_Portal/web/src/backend`

Synopsis

A number of secrets such as `GENERAL_SECRET_KEY` and `HASURA_GRAPHQL_ADMIN_SECRET` are defined as environment variables. These secrets can get accidentally leaked into another system (such as a logging service) when a developer decides to log all the process environment variables, if a developer is not cautious. Environment variables are also passed into child processes, and that too can have unintended consequences that might lead to an exploit.

Mitigation

We recommend using a secret store like Hashicorp Vault or Docker Secrets, instead of environment variables, to store sensitive data.

Status

The Worldcoin team acknowledged that secrets in the `developer-portal`, if not carefully managed, could potentially be leaked into other systems (such as logging services) or passed into child processes. The team added that they are aware of the potential risks and are actively exploring secret hardening measures that would also apply to other systems in the organization.

Verification

Resolved.

Suggestion 2: Configure Poseidon Hash Function Parameters for a 128-bit Security Level

Location

`semaphore-rs/src/poseidon`

Synopsis

In the Worldcoin Protocol, the Poseidon hash function is implemented with 65 rounds, and the degree of S-Boxes is five. According to best practices and guidelines, as noted in [S21], this number of rounds for Poseidon is sufficient for 128-bit security. However, it is possible to reduce this security level by applying an interpolation attack, which is defined in [BBL+22]. In section 4.3 of this research paper, the authors conduct a complexity analysis for this attack by assuming `r` is the total number of rounds for Poseidon, and `t` is the degree of S-Boxes.

In this case, the security level for the Poseidon hash function is equal to log(d*log(d)*(log(d) + log(p))*log(log(d))) where `d=t^(r-2)`, which is the degree of the univariate polynomial, and `p` is the field size. In Worldcoin, `t=5` and `r=65`. Hence, the complexity is approximately equal to 114. Therefore, the attack detailed in this research would reduce the expected security of the Poseidon hash function and, by extension, the overall security of the system.

In addition, `Poseidon2` – a new version of the Poseidon hash function – has been designed, as explained in [GKS23], in order to mitigate the attack described above.

Mitigation

To achieve a higher security level, we recommend that the Worldcoin team either increase the total number of rounds for Poseidon, or the degree of S-Boxes, with respect to the complexity formula noted above. According to the calculations in [BBL+22], Poseidon should have at least 74 rounds for the 128-bit security level.

Status

The Worldcoin team stated that they will likely develop their own protocol to resolve this Issue and other concerns. Additionally, since this would require re-keying the users, the team noted that it would be implemented as part of a planned major upgrade. This would require upstreaming the change to Semaphore, which would require updated circuits and a new setup ceremony performed by the Semaphore team.

Verification

Unresolved - Planned.

Suggestion 3: Remove Unused Code From the Codebase

Location

`src/mimc_hash.rs`

`src/mimc_tree.rs`

Synopsis

There are some instances of unused code in the codebase. This reduces readability and can confuse reviewers.

Mitigation

We recommend removing all unused code and packages.

Status

The Worldcoin team has implemented the mitigation as recommended.

Verification

Resolved.

Suggestion 4: Create a Semaphore Technical Specification for the Worldcoin Use Case

Synopsis

The Worldcoin World ID Protocol project has sufficient general documentation that describes each of the components and how they interact with each other, in addition to sufficient user documentation needed to learn how to interact with the system. However, our team found that the Worldcoin cryptographic component documentation can be improved to include a technical specification of the implementation of `Semaphore` in the Worldcoin Protocol.

Mitigation

We recommend improving the technical documentation of the implementation.

Status

The Worldcoin team documented their use of `Semaphore` in a published Protocol whitepaper.

Verification

Resolved.

Suggestion 5: Change AWS Key Management Service Signing Mechanism

Location

`web/src/backend/kms.ts`

Synopsis

In the Worldcoin Protocol `developer-portal`, the RSA signing mechanism uses the `RSASSA_PKCS1_V1_5_SHA_256` cipher suite. Due to known vulnerabilities, `RSA PKCS1.5` padding has been deprecated or discouraged by various standards and organizations, including NIST and IETF

(Internet Engineering Task Force). More secure alternatives are recommended instead, such as `RSA-PSS` or `RSA-OAEP`.

Moreover, although `RSA-2048` provides a 112-bit security and is recommended by NIST, a higher security level is recommended for critical systems. Bundesamt für Sicherheit in der Informationstechnik (BSI) – the Federal Office for Information Security – recommends using a 120-bit security level instead of a 112-bit security by the end of 2023.

Mitigation

We recommend replacing `RSA-2048` with a more secure signing algorithm, such as `RSA-PSS`, `RSA-OAEP`, or ECDSA – all of which are available in the AWS KMS asymmetric key library.

Status

The Worldcoin team stated that the current algorithm must be kept for compatibility reasons. Additionally, the team noted that they are actively working with key third-parties to extend support for either `ECC_NIST_P256` and/or `RSASSA_PSS_SHA_256` and therefore deprecate the current mechanism.

Verification

Unresolved.

Suggestion 6: Resolve TODOs in Codebase

Location

Examples (non-exhaustive):

`src/field.rs`

`src/lazy_merkle_tree.rs`

`src/merkle_tree.rs`

`src/util.rs`

`pages/api/_reset-client-secret.ts#LL79C70-L80C1`

Synopsis

There are many unresolved TODO items in the code comments of the in-scope repositories, which may lead to a lack of clarity and cause confusion about the completion of the implementation. Resolving TODOs prior to a comprehensive security audit of the code allows security researchers to better understand the full intended functionality of the code, indicates completion, and increases readability and comprehension.

Mitigation

We recommend that TODOs be resolved or removed from the codebases.

Status

The Worldcoin team has addressed critical TODOs & FIXMEs in the `developer-portal` and stated that they also plan to resolve and/or remove the TODOs in the `semaphore-rs` repository in the future.

Verification

Partially resolved.

About Least Authority

We believe that people have a fundamental right to privacy and that the use of secure solutions enables people to more freely use the Internet and other connected technologies. We provide security consulting services to help others make their solutions more resistant to unauthorized access to data and unintended manipulation of the system. We support teams from the design phase through the production launch and after.

The Least Authority team has skills for reviewing code in multiple Languages, such as C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, JavaScript, ZoKrates, and circom, for common security vulnerabilities and specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture in cryptocurrency, blockchains, payments, smart contracts, zero-knowledge protocols, and consensus protocols. Additionally, the team can utilize various tools to scan code and networks and build custom tools as necessary.

Least Authority was formed in 2011 to create and further empower freedom-compatible technologies. We moved the company to Berlin in 2016 and continue to expand our efforts. We are an international team that believes we can have a significant impact on the world by being transparent and open about the work we do.

For more information about our security consulting, please visit https://leastauthority.com/security-consulting/.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis

Our audit techniques include manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's website to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. As we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation. We hypothesize what vulnerabilities may be present and possibly resulting in Issue entries, then for each, we follow the following Issue Investigation and Remediation process.

Documenting Results

We follow a conservative and transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even before having verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyze the feasibility of an attack in a live system.

Suggested Solutions

We search for immediate and comprehensive mitigations that live deployments can take, and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our Initial Audit Report, and before we perform a verification review.

Before our report, including any details about our findings and the solutions are shared, we like to work with your team to find reasonable outcomes that can be addressed as soon as possible without an overly negative impact on pre-existing plans. Although the handling of issues must be done on a case-by-case basis, we always like to agree on a timeline for a resolution that balances the impact on the users and the needs of your project team.

Resolutions & Publishing

Once the findings are comprehensively addressed, we complete a verification review to assess that the issues and suggestions are sufficiently addressed. When this analysis is completed, we update the report and provide a Final Audit Report that can be published in whole. If there are critical unaddressed issues, we suggest the report not be published and the users and other stakeholders be alerted of the impact. We encourage that all findings be dealt with and the Final Audit Report be shared publicly for the transparency of efforts and the advancement of security learnings within the industry.