Coordinated disclosure of vulnerabilities affecting Girault, Bulletproofs, and PlonK

POST APRIL 13, 2022 LEAVE A COMMENT

By Jim Miller

Trail of Bits is publicly disclosing critical vulnerabilities that break the soundness of multiple implementations of zero-knowledge proof systems, including PlonK and Bulletproofs. These vulnerabilities are caused by insecure implementations of the Fiat-Shamir transformation that allow malicious users to forge proofs for random statements.

We've dubbed this class of vulnerabilities Frozen Heart. The word *frozen* is an acronym for FoRging Of ZEro kNowledge proofs, and the Fiat-Shamir transformation is at the *heart* of most proof systems: it's vital for their practical use, and it's generally located centrally in protocols. We hope that a catchy moniker will help raise awareness of these issues in the cryptography and wider technology communities.

This is a coordinated disclosure: we have notified those parties who we know were affected, and they remediated the issues prior to our publication. The following repositories were affected:

- ZenGo's `zk-paillier`
- ING Bank's `zkrp` (deleted)
- SECBIT Labs' `ckb-zkp`
- Adjoint, Inc.'s `bulletproofs`
- Dusk Network's `plonk`
- Iden3's `SnarkJS`
- ConsenSys' `gnark`

The vulnerabilities in one of these proof systems, Bulletproofs, stem from a mistake in the original academic paper, in which the authors recommend an insecure Fiat-Shamir generation. In addition to disclosing these issues to the above repositories, we've also reached out to the authors of Bulletproofs who have now fixed the mistake.

Vulnerabilities stemming from Fiat-Shamir implementation issues are not new (e.g., see here and here), and they certainly were not discovered by us. Unfortunately, in our experience, they are incredibly pervasive throughout the zero-knowledge ecosystem. In fact, we have reported several similar vulnerabilities to some of our previous clients. What's surprising is that despite the pervasiveness of these issues, the cryptography and security communities at large do not appear to be aware of them.

RECENT POSTS

- ABI compatibility in Python: How hard could it be?
- We're streamers now
- Look out! Divergent representations are everywhere!
- We sign code now
- Stranger Strings: An exploitable flaw in SQLite
- We do Windows now
- Porting the Solana eBPF JIT compiler to ARM64
- Working on blockchains as a Trail of Bits intern
- Secure your machine learning with Semgrep
- It pays to be Circomspect
- Magnifier: An Experiment with Interactive Decompilation
- Using mutants to improve Slither
- The road to the apprenticeship
- Shedding smart contract storage with Slither
- libmagic: The Blathering

YEARLY ARCHIVE

- 2020
- 2019
- 2018
- 2017
- 2016
- 2015
- 2014
- 2013

In this blog post, I will first describe the Frozen Heart vulnerability at a high level. I will then discuss the reasons Frozen Heart vulnerabilities occur so commonly in practice (spoiler alert: bad documentation and guidance), steps that the community can take to prevent them, and Trail of Bits' role in leading that charge. Lastly, I will provide the details of our coordinated disclosure.

This is part 1 of a series of posts on the Frozen Heart vulnerability. In parts 2, 3, and 4, I'll describe how these vulnerabilities actually work in practice by covering their impact on Girault's proof of knowledge, Bulletproofs, and PlonK, respectively. Make sure to watch out for these posts in the coming days!

Zero-Knowledge Proofs and the Fiat-Shamir Transformation

This post assumes that you possess some familiarity with zero-knowledge proofs. If you would like to read more about them, there are several helpful blog posts and videos available to you, such as Matt Green's primer. To learn more about the Fiat-Shamir transformation, check out the blog post I wrote explaining it in more detail. You can also check out ZKDocs for more information about both topics.

The Frozen Heart Vulnerability

The Fiat-Shamir transformation is applied to proof systems with the following structure:

1. The prover generates a random value: the commitment.
2. The verifier responds with a random value: the challenge.
3. The prover then uses the commitment, the challenge, and her secret data to generate the zero-knowledge proof.

Each proof system is accompanied by a security proof, which guarantees that it's infeasible for an attacker to forge proofs as long as certain assumptions are met. For proof systems of this structure, one very important assumption is that the challenge value generated by the verifier is completely unpredictable and uncontrollable by the prover. At a high level, the reason is that the zero-knowledge proof generated by the prover is considered valid only if it satisfies a very specific mathematical relationship. The prover may satisfy this relationship in one of two ways:

1. He actually has the necessary secret data to satisfy the relationship, and he generates a zero-knowledge proof the honest way.
2. He does not have the necessary secret data, but he guesses random values and gets lucky.

For a secure proof system, it's effectively impossible for malicious provers to achieve option 2 (i.e., they have only a one in 2^{128} probability of guessing right) as long as the random challenge is completely unpredictable. But if a malicious prover can predict this value in a certain way, it's actually easy for him to commit a proof forgery by finding random

- 2012

CATEGORIES

- Apple (13)
- Attacks (10)
- Audits (2)
- Authentication (5)
- Binary Ninja (13)
- Blockchain (50)
- Capture the Flag (11)
- Careers (2)
- CodeQL (3)
- Compilers (23)
- Conferences (28)
- Containers (2)
- Cryptography (40)
- Crytic (3)
- Cyber Grand Challenge (7)
- DARPA (21)
- Dynamic Analysis (12)
- Education (15)
- Empire Hacking (7)
- Engineering Practice (12)
- Events (6)
- Exploits (27)
- Fuzzing (29)
- Go (4)
- Guides (10)
- Internship Projects (31)
- iVerify (4)
- Kubernetes (2)
- Linux (1)
- Machine Learning (6)
- Malware (7)
- Manticore (15)
- McSema (11)
- Meta (12)
- Mitigations (9)
- osquery (22)
- Paper Review (11)
- People (1)
- Podcast (1)
- Press Release (27)
- Privacy (9)
- Products (6)
- Program Analysis (16)
- Recruitment (1)
- Remote Work (1)
- Research Practice (19)
- Reversing (14)
- Rust (4)
- SafeDocs (1)
- Sinter (1)
- Slither (3)
- Sponsorships (12)
- Static Analysis (28)
- Symbolic Execution (17)
- Training (2)

values that will satisfy the necessary mathematical relationship. This is exactly how the Frozen Heart vulnerability works.

This vulnerability is generic; it can be applied to any proof system that insecurely implements the Fiat-Shamir transformation. However, the exact details of the vulnerability depend on the proof system in question, and the impact of the vulnerability depends on how the surrounding application uses the proof system. For examples on how the vulnerability impacts different systems, be sure to read my upcoming posts describing Frozen Heart vulnerabilities in Girault's proof of knowledge, Bulletproofs, and PlonK. You can also check out my previous Fiat-Shamir blog post, in which I show how this works for the Schnorr proof system.

Preventing Frozen Heart Vulnerabilities

The Frozen Heart vulnerability can affect any proof system using the Fiat-Shamir transformation. To protect against these vulnerabilities, you need to follow this rule of thumb for computing Fiat-Shamir transformations: **the Fiat-Shamir hash computation must include all public values from the zero-knowledge proof statement and all public values computed in the proof (i.e., all random "commitment" values)**.

Here, the zero-knowledge proof statement is a formal description of what the proof system is proving. As an example, let's look at the Schnorr proof system. The (informal) proof statement for the Schnorr proof system is the following: the prover proves that she knows a secret value, `x`, such that `h = g`^{x} `mod q`, where `q` is a prime number and `g` is a generator of a finite group. Here, `h`, `g`, and `q` are all public values, and `x` is a private value. Therefore, we need to include `h`, `g`, and `q` in our Fiat-Shamir calculation.

But we're not done. In step 1 of the protocol, the prover generates a random value, `r`, and computes `u = g`^{r} (here, `u` is "the commitment"). This `u` value is then sent to the verifier as part of the proof, so it is considered public as well. Therefore, `u` needs to be included in the Fiat-Shamir calculation. You can see that all of these values are included in the hash computation in ZKDocs.

A Frozen Heart vulnerability is introduced if some of these values (specifically, `h` or `u`) are missing. If these values are missing, a malicious prover can forge proofs for random `h` values for which she does not know the discrete log. Again, to see how this works, read my previous Fiat-Shamir blog post.

The details are crucial here. Even for the Schnorr proof system, which is arguably the most simple zero-knowledge proof protocol used in practice, it can be easy to make mistakes. Just imagine how easy it is to introduce mistakes in complex proof systems that use multiple Fiat-Shamir transformations, such as Bulletproofs and PlonK.

The Problem

- Uncategorized (23)
- Working at Trail of Bits (2)
- Year in Review (4)
- Zero Knowledge (9)

Tweets from @trailofbits

Trail of Bits
... · Nov 23

Echidna v2.0.4 will auto-generate a coverage report in HTML, following the same approach as the text file report. It will show colors to signal which lines are covered either without errors (green), with a revert (yellow) or not covered at all (red).
github.com/crytic/echidna
...

Why is this type of vulnerability so widespread? It really comes down to a combination of ambiguous descriptions in academic papers and a general lack of guidance around these protocols.

Let's look at PlonK as an example. If you inspect the details of the protocol, you will see that the authors do not explicitly explain how to compute the Fiat-Shamir transformation (i.e., hash these values). Instead, the authors instruct users to compute the value by hashing the "transcript." They do describe what they mean by "transcript" in another location, but never explicitly. A few implementations of the proof system got this computation wrong simply because of the ambiguity around the term "transcript."

Believe it or not, PlonK's description is actually better than important descriptions in most papers. Sometimes the authors of a paper will actually specify an insecure implementation, as was the case for Bulletproofs (which we'll see in an upcoming follow-up post). Furthermore, most academic papers present only the interactive version of the protocol. Then, after they've described the protocol, they mention in passing that it can be made non-interactive using the Fiat-Shamir transformation. They provide you with the original Fiat-Shamir publication from the 1980s, but they rarely say how this technique should actually be used!

Can you imagine if all cryptographic primitives had such documentation issues? We have an entire RFC dedicated to generating nonces deterministically for ECDSA, but it is still implemented incorrectly. Imagine if there were no standards or guidance for ECDSA, and developers could implement the algorithm only by reading its original paper. Imagine if this paper didn't explicitly explain how to generate these nonces and instead pointed the reader to a technique from a paper written in the 1980s. That's essentially the current state of most of these zero-knowledge proof systems.

To be clear, I'm not trying to condemn the authors of these papers. These protocols are incredibly impressive and already take a lot of work to build, and it's not the job of these authors to write comprehensive implementation details for their protocols. But the problem is that there really is no strong guidance for these protocols, and so developers have to rely largely on academic papers. So, my point is not that we should blame the authors, but rather that we shouldn't be surprised at the consequences.

The Solutions

The best way to address these issues is to produce better implementation guidance. Academic papers are not designed to be comprehensive guides for implementation. A developer, particularly a non-cryptographer, using only guidance from these papers to implement a complex protocol is likely to make an error and introduce these critical vulnerabilities. This is the exact reason we created ZKDocs. With ZKDocs, we aim to provide clearer implementation guidance, focusing particularly on areas of protocols that are commonly messed up, such as Fiat-Shamir transformations. If you're creating your own zero-knowledge proof implementation, check out our Fiat-Shamir section of ZKDocs!

It's also worth mentioning that these issues could be more or less eradicated if test vectors for these protocols were widely available. Implementations with incorrect Fiat-Shamir transformations would fail tests using test vectors from correct implementations. However, given the limited guidance for most of these proof systems, producing test vectors for all of them seems unlikely.

Lastly, investigating these Frozen Heart vulnerabilities was a good reminder of the value of code audits. My team and I reviewed a lot of public repositories, and we found a good number of implementations that performed these transformations correctly. Most of these implementations were built by groups of people who performed internal and, typically, external code audits.

Coordinated Disclosure

Prior to our disclosure, my teammates and I spent the last few months researching and reviewing implementations for as many proof systems as possible. Once our research was complete, we disclosed the vulnerabilities to ZenGo, ING Bank, SECBIT Labs, Adjoint Inc., Dusk Network, Iden3, ConsenSys, and all of their active forks on March 18, 2022. We also contacted the authors of the Bulletproofs paper on this date.

As of April 12, 2022, ZenGo, Dusk Network, Iden3, and ConsenSys have patched their implementations with the required fixes. ING Bank has deleted its vulnerable repository. The authors of the Bulletproofs paper have updated their section on the Fiat-Shamir transformation. We were not able to get in contact with SECBIT Labs or Adjoint Inc.

- Zengo submitted this patch.
- Dusk Network submitted this patch.
- Iden3 submitted this patch.
- ConsenSys submitted this patch.

We would like to thank ZenGo, ING Bank, Dusk Network, Iden3, ConsenSys, and the Bulletproofs authors for working swiftly with us to address these issues.

Acknowledgments

I would like to thank each of my teammates for assisting me in reviewing public implementations and David Bernhard, Olivier Pereira, and Bogdan Warinschi for their work on this vulnerability. Lastly, a huge thanks goes to Dr. Sarang Noether for helping me understand these issues more deeply.

By James Miller Posted in Cryptography, Zero Knowledge

← Towards Practical Security Optimizations for Binaries

The Frozen Heart vulnerability in Girault's proof of knowledge →