Quantstamp Security Assessment Certificate

August 31st 2020 — Quantstamp Verified

Polymarket

This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

Type	Gnosis Software Extension / Wallet
Auditors	Jan Gorzny, Blockchain Researcher Ed Zulkoski, Senior Security Engineer Fayçal Lalidji, Security Auditor
Timeline	2020-08-24 through 2020-08-28
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Low
Test Quality	Low

Source Code

Repository	Commit
polymarket-mono	dc3bd26

Goals

- Review specific files for known issues
- Examine the use of dependencies in the project

Total Issues	4 (0 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	1 (0 Resolved)
Low Risk Issues	1 (0 Resolved)
Informational Risk Issues	2 (0 Resolved)
Undetermined Risk Issues	0 (0 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

The code is relatively straightforward despite reliance on many dependencies. The audited files were mostly free of problems, although four issues were found. We found one medium risk issue, one low risk issue, and two informational issues. The code would strongly benefit from additional testing - there appears to be only three test cases, and we were unable to get code overage to run for this project; it's unlikely that it's very high given the low number of tests and large number of files. Quantstamp recommends addressing all issues here and increasing the quality of tests.
Note that only a handful of files (`ProxyWallet.sol`, `ProxyWalletFactory.sol`, `ProxyWalletLib.sol`, `PolymarketWallet.sol`, `GSNModule01.sol`) were in scope for this audit, although there are many more in the repository.

ID	Description	Severity	Status
QSP-1	Incorrect Keccak256 Hashes	Medium	Unresolved
QSP-2	Infinite Approval	Low	Unresolved
QSP-3	Unlocked Pragma	Informational	Unresolved
QSP-4	GSN Inconsistency	Informational	Unresolved

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Findings

QSP-1 Incorrect Keccak256 Hashes

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `ProxyWalletLib.sol`

Description: The `keccak256` constants on L7-8 do not match the expected values mentioned in the comments.

QSP-2 Infinite Approval

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `PolymarketWallet.sol`

Description: The contract uses `approveMaxIfNeeded()` to infinitely approve the `FixedProductMarketMaker`. If a malicious exploit is found in the `FixedProductMarketMaker`, it could potentially drain all tokens from this contract. It may be safer to only approve as needed.

QSP-3 Unlocked Pragma

Severity: *Informational*

Status: Unresolved

File(s) affected: `ProxyWallet.sol`, `ProxyWalletFactory.sol`, `ProxyWalletLib.sol`, `PolymarketWallet.sol`, `GSNModule01.sol`,

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked." For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

QSP-4 GSN Inconsistency

Severity: *Informational*

Status: Unresolved

File(s) affected: `GNSModule01.sol`

Description: The `acceptRelayedCall` member of `GNSModule01` returns `doCall` equal to 0 for accepted calls or 1 for rejected calls. It should be noted that `GSNRecipient` implements internal functions that generate values which can be returned in both accepted or rejected cases:

- `function _approveRelayedCall(bytes memory context) internal pure returns (uint256, bytes memory)`
- `function _rejectRelayedCall(uint256 errorCode) internal pure returns (uint256, bytes memory)`

This will allow the relay hub to gather more specific information about the reason why the call was rejected, in case if the implemented logic get more complex, making a better user experience since different error messages can be displayed.

Recommendation: Even if `GSNModule01` does not inherit from `GSNRecipient` but is used to delegate calls to it in the context of `ProxyWalletFactory`, similar functions can be implemented to return the same values as the implemented functions in `GSNRecipient`.

Adherence to Best Practices

- `ProxyWalletLib.sol`: `import { MemcpyLib } from "./MemcpyLib.sol";` does not appear to be used anywhere.

Test Results

Test Suite Results

Some test output has been removed for this report in order to improve readability.

```
 Contract: PolymarketFixedProductMarketMakerFactory.sol

     ✓ should create a market (892ms)
     ✓ should do a buy (1107ms)
     ✓ can allow a market maker to withdraw his position before oracle responds (3970ms)


  3 passing (7s)

Done in 27.69s.
```

Code Coverage

We were unable to compute the code coverage for this project.

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
a2efcc9b2f9da7f76ad25bab1f8d1f2f4f5cad2c79e1055948e3a87bdef1cb8a ./contracts/DAI.sol
e0c83f660406434973f07026d34c9bf11929a59e4e2602bfc21626dbfd34c4ad ./contracts/EtherForwarder.sol
b63bb3aa7b2c07dd9d5bd50004f23f5323113dc68a7d19475c16df7f2767e386 ./contracts/Exports.sol
bc28026c9face315241e12fabe181f0c401093edddbfdb91fa3de91249def7b4 ./contracts/FactoryLib.sol
d1512312fc9c48a9fb49f8e6a773b27bc37ce8d42dfb29ea3174872d1284d978 ./contracts/GSNLib.sol
b87db14200c4e877b84ba6474ee190f24355e48917703be1fbbc02f968bf5df5 ./contracts/GSNModule01.sol
90eb41e1a9471e7c703ae240739580a93ef416a9d98f571b2688386c066e30b3 ./contracts/IGSNModule.sol
1e2a23227fea430f3b82a6ce5447f3af6b020bb387f24ea84e292d53ead81ec4 ./contracts/IProxyWalletFactory.sol
c5fa25ce39ddf78535b377690c832e67d02997442ca4aa61213f57ca277046ab ./contracts/MemcpyLib.sol
d131685e5b568f36eb69896f806dd3db51bb7ed94f753e8e445f814359138108 ./contracts/Migrations.sol
58879890eeac9a41d90d005814380ca1e4339f209a45ffdaa27669579bbfe668 ./contracts/PolymarketFixedProductMarketMakerFactory.sol
05b302f8a1c06e10568c96aeb5675948a2cc61f8b714b6cd1aa522d033cc6ea0 ./contracts/PolymarketFPMMDeployerLib.sol
071cb1d30d6d5551f02f95dd568164dce62e80fae0a66e4d9e5f6b1a0c10b3a8 ./contracts/PolymarketLMSRMarketMakerFactory.sol
c2b25c28e6720b2b5f3e822a787fef562c35ecf4ecf6b26c2e5849a3545b1f0e ./contracts/PolymarketMarketMakerFactoryLib.sol
f06fe0720f58b48d4d46da8311192a7daa11771f0e881ecde2e5691666c504b6 ./contracts/PolymarketWallet.sol
9820a2ab1e5bfdf2e5ce99de57cc3776359b2ca27efe0c7e794b36b7be482b8d ./contracts/PolymarketWalletData.sol
a59235f067aa852673a40f49f2a4416738f636f01045c7471e9a94edfa7f834c ./contracts/PolymarketWalletLib.sol
82e3518dd1fc4656b7ec4f9a1e236f918df96aec5d81403a6efd7ce09e929b4c ./contracts/ProxyWallet.sol
b6fe11600c09b83f71197d7ef947fb26353e245fcf4144e806d54443c7b1f2c1 ./contracts/ProxyWalletFactory.sol
9f6968d2ea90568c010f624691d047c6f0057244e2c0a7b54b593ef5c8095717 ./contracts/ProxyWalletLib.sol
c2f91fb014edb4b2e2f16775d45790d9106e4f9c8b632f5f93c30c8d34222f3c ./contracts/RevertCaptureLib.sol
0dbbf46f399c0fcecdc036fff945551427ce4538541c6cb2fbcc44d936ab3ba6 ./contracts/RStoreLib.sol
88866305fe9c3557f215dd1d41b5f25ac0e0155d3b8fe7ea851f4b458d49ac73 ./contracts/SliceLib.sol
b4d9447e29ef14bc6c6294a47b06fd79e5dfafecdb4c1f7fe612a3f511bca927 ./contracts/StringLib.sol
```

Tests

```
ae17a8fd38c3db3105a08f04f6d2b935a861367b97457dd29e1bcf59c760b198 ./test/test.js
```

Changelog

- 2020-08-28 - Initial report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.