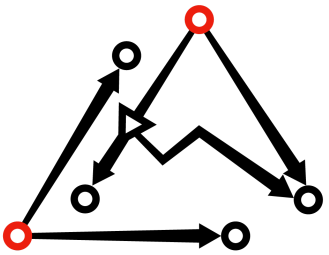

Least Authority

PRIVACY MATTERS

Dogecoin Node Upgrade

Security Audit Report

Final Audit Report: 21 April 2023

Table of Contents

Overview

Background

Least Authority has been requested to perform a security audit of changes to the Dogecoin Core codebase between version 1.14.3 and version 1.14.5. We note that the most recent release at the time of writing, version 1.14.6, is considered out of scope for this report.The goal of the audit is to identify any potential security issues that could impact network operations resulting from the deployment of the upgrade to version 1.14.5.

Project Dates

- **October 3 - November 11**: Code Review *(Completed)*
- **November 16**: Delivery of Initial Audit Report *(Completed)*
- **April 19:** Verification Review *(Completed)*
- **April 20:** Delivery of Final Audit Report *(Completed)*

Review Team

- Jehad Baeth, Security Researcher and Engineer
- Sven M. Hallberg, Security Researcher and Engineer
- Nikos Iliakis, Security Researcher and Engineer
- Giorgi Jvaridze, Security Researcher and Engineer

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the Dogecoin Node upgrade followed by issue reporting, along with mitigation and remediation instructions as outlined in this report.

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

- Dogecoin node v1.14.3: https://github.com/dogecoin/dogecoin/releases/tag/v1.14.3
- Dogecoin node v1.14.4: https://github.com/dogecoin/dogecoin/releases/tag/v1.14.4
- Dogecoin node v1.14.5: https://github.com/dogecoin/dogecoin/releases/tag/v1.14.5

Specifically, we examined the following Git revisions for our initial review. For this review, the scope of the audit was limited to the differences between the below revisions and did not include the entire codebases:

- Dogecoin node v1.14.3: `f80bfe9068ac1a0619d48dad0d268894d926941e`
- Dogecoin node v1.14.4: `4c93783ab64ba96e60b7ea37202c072c66f3f544`
- Dogecoin node v1.14.5 :`31afd133119dd2e15862d46530cb99424cf564b0`

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

- Dogecoin node v1.14.3: https://github.com/LeastAuthority/dogecoin/tree/v1.14.3

- Dogecoin node v1.14.4: https://github.com/LeastAuthority/dogecoin/tree/v1.14.4
- Dogecoin node v1.14.5: https://github.com/LeastAuthority/dogecoin/tree/v1.14.5

For the verification, we examined the Git revision:

- Dogecoin node v1.14.6 : `3a29ba6d497cd1d0a32ecb039da0d35ea43c9c85`

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:

- N/A

Areas of Concern

Our investigation focused on the following areas:

- Adversarial actions and other attacks on the node;
- Attacks that impact funds, such as the draining or manipulation of funds;
- Mismanagement of funds via transactions;
- Denial of Service attacks and security exploits that would impact or disrupt the execution of the upgrade;
- Vulnerabilities within individual components and whether the interactions between the components are secure;
- Exposure of any critical information during interactions with any external libraries;
- Proper management of encryption and signing keys;
- Protection against malicious attacks and other methods of exploitation;
- Data privacy, data leaking, and information integrity;
- Inappropriate permissions and excess authority; and
- Anything else as identified during the initial analysis phase.

Findings

General Comments

The main goal of this security review was to identify any potential security or implementation issues introduced between the 1.14.3 and 1.14.5 revisions of the Dogecoin node codebase. During our investigation, our review team identified several areas of concern, which could result in security issues if not handled correctly during node software updates. While our report is aimed primarily at users and operators of the Dogecoin software, we have identified issues with the codebase that are of concern to Dogecoin developers, and these are highlighted as such. The scope of this review was limited to the changes made in the target revisions. However, it was necessary for our team to build a good understanding of the baseline node implementation in order to reason about the security implications of the changes in scope.

During the investigation of a consensus change in Dogecoin version 1.14.5, our team found that the release of version 1.14.0 had introduced a discrepancy with respect to Dust between the wallet and relay

logic, effectively prohibiting the inclusion of certain transactions in the blockchain, which resulted in them becoming stuck in wallets for an indefinite time. When version 1.14.5 introduced a series of changes that relaxed network requirements, this made the transactions compatible again with network nodes, allowing stuck transactions to be executed. Consequently, we recommend that extra caution be taken during an upgrade, as this issue could result in an unintended transfer or loss of funds if not handled correctly (Issue A).

In addition, we identified potential security issues that users should be aware of and guard themselves against. These relate to software crashes (Issue B) and the backup and recovery of wallets (Issue C, Issue D). Another group of issues relates to fee adjustments that could negatively impact users who have taken explicit steps to reduce the fees paid on their transactions (Issue E, Issue F, Issue G).Finally, a code quality issue with respect to Dust could conceivably lead to future disruption if the Dogecoin developers are not careful to avoid it (Issue H).

We further note some opportunities for node operators to obtain lower fees on their transactions by switching to Replace-by-Fee (Suggestion 3) or lowering default parameters (Suggestion 1, Suggestion 2). However, we recommend that individual users carefully evaluate these options in light of possible impacts on convenience (Suggestion 1) or network access (Suggestion 2).

Code quality

Our team performed a manual review of the updates to the Dogecoin node codebase and found the code to be decently written and organized. However, the fee system is characterized by a high level of complexity with many interdependent variables, the use, definition, and modification of which can appear in different places. While previous work has gone toward consolidating this, the overall system remains complicated. We recommend that further simplifications of the fee rules and their implementation be explored by Dogecoin developers to reduce the risk of errors and user confusion, especially with respect to future updates.

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: New Dust Rules Could Cause Stuck Transactions to Execute	Unresolved
Issue B: Sensitive Data in Old Core Dump Files	Unresolved
Issue C: Key Derivation Path Has Not Changed	Unresolved
Issue D: Wallet Recovery Not Implemented	Resolved
Issue E: Increased Default Transaction Fee	Resolved
Issue F: Wallet Discard Threshold Replaces Dust Limit	Resolved
Issue G: Increased Fee Requirements From Miners	Resolved
Issue H: Dust Consensus Is Derived from Recommended Fee	Unresolved

Suggestion 1: Consider Lowering the Discard Threshold	Resolved
Suggestion 2: Consider Lowering the Soft Dust Limit	Unresolved
Suggestion 3: Switch to Replace-by-Fee	Unresolved

Issue A: New Dust Rules Could Cause Stuck Transactions to Execute

Location

`src/dogecoin.cpp`

`src/wallet/wallet.cpp`

`src/primitives/transaction.h`

`src/policy/policy.cpp`

Synopsis

Version 1.14.5 changes the rules with respect to Dust outputs in a way that could cause certain transactions that were previously "stuck" to unexpectedly be executed, specifically during a software upgrade.

Impact

This issue could result in the unintended transfers of funds.

Preconditions

The issue is possible if low-output transactions were left pending under the assumption that they had failed permanently.

Technical Details

Version 1.14.5 introduces significant changes to the rules relating to transaction outputs of very low value, also referred to as "Dust." Aside from adjustments to the associated fees, which are effectively advisory, it includes a change to the network *consensus* about what constitutes a legal (so-called "standard") transaction. Any consensus change merits careful investigation, especially with respect to the evolution of the rule in question.

Transaction outputs below a certain value were first prohibited in Dogecoin with version 1.14.0, likely unintentionally as part of the very large code integration leading to that version (cf. pull request 1587). Due to an inconsistency in the code, that version could create transactions that were technically legal, but carried less fees than the network generally required for their inclusion in the blocks. Such transactions could conceivably remain pending in wallets for an indeterminate length of time.

Historically, Dogecoin had allowed Dust outputs but required extra "penalty" fees for their processing. The amount of such fees, as expected by miners and relays to admit a transaction, is computed by the function `GetDogecoinDustFee`. During transaction creation by `CWallet::CreateTransaction`, the function `CTxOut::IsDust` is used to determine if that transaction requires the extra fees.

`IsDust`, however, is also used by `IsStandardTx` as part of determining whether a given transaction is legal with respect to the network consensus. While fee requirements are up to the individual node operator, any transaction not satisfying `IsDust` cannot be accepted. Due to a discrepancy between the

calculation in `IsDust` and that of `GetDogecoinDustFee`, version 1.14.0 could create transactions that did not satisfy `IsDust` and would be created without penalty fees but for which `GetDogecoinDustFee` would report these penalties as required. Consequently, such transactions, if created by the wallet, would never be accepted by relays and miners.

This discrepancy prompted the release of version 1.14.2, which based the calculation of `GetDogecoinDustFee` on the result of `IsDust`. At the same time, the threshold used by `IsDust` was raised to one coin, the value previously used by `GetDogecoinDustFee`. This, again, changed the network consensus, likely unintentionally.

The adjustments in version 1.14.5, finally, relaxed the consensus to an extent that the legacy transactions in question have become both valid and acceptable with respect to their included fees. As a result, they will likely be processed whenever the new version (re-) attempts their submission to the network, typically upon starting the software.

As of version 1.14.5, a "hard Dust limit" of 0.1 cent is imposed for any standard transaction. Below a "soft limit" of one cent, miners and relays will normally require the inclusion of penalty fees. Finally, the wallet software conservatively uses the newly introduced “discard threshold” of one coin by default as its effective Dust limit to remain compatible with network nodes that still run on version 1.14.4 or earlier.

Mitigation

As a matter of course, any pending transactions whose execution is no longer desired must be properly canceled. This can be accomplished, for example, by waiting for any desired transactions to complete, shutting down the node, and restarting it with the `-zapwallettxes` and `-rescan` command-line interface (CLI) arguments. The `-zapwallettxes` argument causes all transactions in the wallet to be deleted and `-rescan` searches the blockchain for any missing transaction history. See this related issue for further details.

When transactions become “stuck” in the wallet and need to be replaced, we recommend ensuring that the replacement transaction consumes the same transaction outputs as the original, so that the original becomes invalid by virtue of the replacement being accepted.

We also further recommend that due diligence be undertaken during version upgrades.

Status

The Dogecoin development team has acknowledged this issue and noted that, while the recommended mitigation will not be implemented at this time, it will be taken into consideration for future releases.

Verification

Unresolved.

Issue B: Sensitive Data in Old Core Dump Files

Location

`src/support/lockedpool.cpp:226`

Synopsis

Up to version 1.14.4, the software could write sensitive data to disk when crashing. Version 1.14.5 mitigates the issue on common operating systems, but remnants of previous crashes could remain.

Impact

This issue could result in the compromise of sensitive data.

Preconditions

For this issue to occur, an attacker must obtain file system access with the permissions of the user account running the Dogecoin software.

Feasibility

Straightforward.

Technical Details

Up to version 1.14.4, the cryptographic keys protecting accounts could be included in "core dump" files that are generated under certain conditions during a crash of the software. Version 1.14.5 adds a call to `madvise` that excludes sensitive data from core dumps on Linux and FreeBSD.

Mitigation

We recommend disabling the creation of core files in the operating system, securely deleting any that had been previously generated, and transfering funds to newly created wallets.

Status

The Dogecoin development team has acknowledged this issue and noted that, while the recommended mitigation will not be implemented at this time, it will be taken into consideration for future releases.

Verification

Unresolved.

Issue C: Key Derivation Path Has Not Changed

Location

`src/wallet/wallet.cpp:129`

Synopsis

Version 1.14.5 erroneously claims to change the way keys are generated in Hierarchical Deterministic (HD) wallets. This could mislead the user and cause recovery attempts to fail in the case of a lost wallet.

Impact

This issue could result in the temporary loss of access to funds.

Preconditions

For this issue to occur, HD key derivation must be enabled on the wallet upon creation, which is the default configuration.

Technical Details

Version 1.14.5 purports to change the way keys are generated in HD wallets. However, the key derivation process as implemented by the function `DeriveNewChildKey` remains, in fact, unchanged. The difference lies only in the display of the key derivation path, shown for example in the output of the `dumpwallet` command.

The change could mislead the user and cause the temporary loss of access to funds if third-party software were employed to recover keys using the wrong path.

The key derivation path used by the Dogecoin software version 1.14.5 begins with `m/0'/0'`, which is the same as what it had been in previous versions.

Mitigation

We suggest alerting users to take note of the correct derivation path and make sure it is used in any recovery attempt.

Remediation

We recommend checking that the software reports the actual path used, either by reverting the change to the display, or by adjusting the key derivation code to match it.

Status

The Dogecoin development team stated that the issue is still being investigated. As such, this issue remains unresolved at the time of verification.

Verification

Unresolved.

Issue D: Wallet Recovery Not Implemented

Synopsis

The Dogecoin software does not offer the function needed to recover HD wallets from a saved seed.

Impact

This issue could result in the temporary loss of access to funds.

Preconditions

This issue is possible if the `wallet.dat` file and an up-to-date backup are not available.

Technical Details

The main benefit of deterministically generated HD wallets is that all keys therein can be regenerated from a single secret seed. The seed might be derived from an externally managed passphrase or, as is the case in Dogecoin, it might be randomly generated and saved to the wallet alongside the account keys that are created from it. In this scenario, the benefit remains in the fact that any backup of the wallet can be used to regenerate all account keys, even those that are not contained in that backup.

While Dogecoin allows extracting all keys, including the HD seed, from a wallet via the `dumpwallet` command, its converse `importwallet` does not recognize the contained seed value and there is, as of version 1.14.5, no provision to specify it. Therefore, it might be hard in practice to fully recover a wallet from an outdated backup, despite being possible in principle.

Mitigation

We recommend regularly backing up the full `wallet.dat` file by using the `backupwallet` command, rather than relying on `dumpwallet`.

Remediation

We recommend that the software add functionality to recover a wallet and its associated funds from a given HD seed as included in the output of `dumpwallet`.

Status

The Dogecoin development team acknowledged the finding but stated that this is intended behavior. Therefore, we consider this issue resolved.

Verification

Resolved.

Issue E: Increased Default Transaction Fee

Location

`src/wallet/wallet.h`

Synopsis

An internal variable used in the determination of transaction fees has been increased from zero to the recommended minimum. This value could then be used in place of a lower value if the user had explicitly reduced the minimum.

Impact

This issue could lead to an increase in the actual fees paid.

Preconditions

This issue is possible if the user reduces the parameters `-minrelaytxfee` and `-mintxfee` without specifying an explicit `-paytxfee`.

Technical Details

The constant `DEFAULT_TRANSACTION_FEE` is used to initialize the variable `payTxFee`, which corresponds to the `-paytxfee` CLI argument. Prior to version 1.14.4, this value had been set to zero. Version 1.14.4 changed it to one coin, which equaled the default minimum fee at the time. Version 1.14.5 adjusted both (`RECOMMENDED_MIN_TX_FEE` and `DEFAULT_TRANSACTION_FEE`) to one cent.

If the user explicitly reduced the minimum values via `-minrelaytxfee` and `-mintxfee` to obtain a lower rate, they would have previously paid those values. This becomes ineffective with version 1.14.4 unless the user also specifies `-paytxfee` with their desired value.

NB: Prior to version 1.14.4, `payTxFee` had been inadvertently ignored altogether.

Mitigation

We recommend using the `-paytxfee` parameter to set the desired transaction fee.

Status

The Dogecoin development team acknowledged the finding but stated that this is intended behavior. Therefore, we consider this issue resolved.

Verification

Resolved.

Issue F: Wallet Discard Threshold Replaces Dust Limit

Location

`src/wallet/wallet.h`

Synopsis

A new internal variable called `discardThreshold` is introduced that the wallet uses as its effective Dust limit in place of former uses of `nDustThreshold`. If a user lowers the latter via the CLI argument `nDustLimit`, this change could cause the new, higher, value to silently take effect instead.

Impact

This issue could lead to an increase in the actual fees paid.

Preconditions

This issue is possible if the user reduces the wallet parameter `-dustlimit`.

Technical Details

As of version 1.14.5, the variable `discardThreshold`, which corresponds to the `-discardthreshold` CLI argument, is split out to serve the role that `-dustlimit` previously held with respect to transactions created by the wallet.

The developers note in the relevant commit (a4d96554) that at the time, 97% of the network enforced a Dust limit of one coin. However, if the user had relied on one of the remaining nodes and explicitly reduced the `-dustlimit` parameter to retain more of their "change" in transactions, this would silently become ineffective with version 1.14.5.

Mitigation

We recommend using the `-discardthreshold` parameter instead of `-dustlimit` to control the desired value in the wallet.

Status

The Dogecoin development team implemented the mitigation by aligning the two default parameters, `-dustlimit` and `-discardthreshold`.

Verification

Resolved.

Issue G: Increased Fee Requirements From Miners

Location

`src/wallet/wallet.h`

Synopsis

Two internal variables that control transaction fee requirements have effectively switched places in terms of their default value. According to the new values, getting transactions mined is generally becoming more expensive even though getting transactions relayed through the network (to a miner) is becoming less expensive.

Impact

This issue could negate savings imparted by a direct or cheap network access to a miner.

Preconditions

This issue is possible if a user relies on direct or cheap access to a miner for lower fees.

Technical Details

As of version 1.14.4, the default value for parameter `-blockmintxfee` is increased by a factor of 1000 to one cent. This controls the minimum amount of fees that a miner expects to gain (on average) from a given block of transactions. At the same time, the default for parameter `-minrelaytxfee` is *reduced* by a factor of 1000 to a tenth of a cent. Previously, these values were set to 0.001 cent and 0.1 cent, respectively. Therefore, there was the potential to reduce fees 100-fold if one could connect to a miner directly. With the reversal of the relationship between these default values, it becomes much less profitable.

Mitigation

This issue can only be mitigated by finding access (directly or indirectly) to a miner that does not follow the new defaults.

Status

The Dogecoin development team acknowledged the finding but stated that this is intended behavior. Therefore, we consider this issue resolved.

Verification

Resolved.

Issue H: Dust Consensus Is Derived from Recommended Fee

Location

`src/policy/policy.h`

Synopsis

The internal parameter that configures the network consensus on the "hard" Dust limit is derived indirectly from the recommended transaction fee. A change of the fee recommendation could thus inadvertently cause a consensus change.

Impact

This issue could cause the network to fork and user transactions to become stuck.

Preconditions

This issue is possible if the constant `RECOMMENDED_MIN_TX_FEE` is changed in the code while the constant `DEFAULT_HARD_DUST_LIMIT` is derived from it.

Technical Details

As discussed in Issue A, the "hard" Dust limit is a network consensus parameter. It is set in the code from the constant `DEFAULT_HARD_DUST_LIMIT`. This is set to one tenth of `DEFAULT_DUST_LIMIT`, which, in turn, is defined as equal to `RECOMMENDED_MIN_TX_FEE`. The latter was introduced as a general base constant from which other fee values are derived in order to clarify their relationship. As we have outlined, however, fee policies are fundamentally different from network consensus. A change to the latter would

force out any node not compliant with it, effectively fragmenting the network. At the least, users running outdated software versions would find their transactions unable to complete.

As Issue A has demonstrated, consensus changes are not to be made lightly. The naming ("recommended") and placement (with other fee policy parameters) of `RECOMMENDED_MIN_TX_FEE` make it harder than necessary to realize that a consensus change would result from its modification.

Mitigation

Users can change their node's hard Dust limit via the `-harddustlimit` CLI parameter. We recommend that node operators closely monitor the network so they can react to unexpected parameter changes.

Remediation

We recommend that Dogecoin developers separate the definition of `DEFAULT_HARD_DUST_LIMIT` from `RECOMMENDED_MIN_TX_FEE`.

Status

The Dogecoin development team has acknowledged this issue and noted that, while the recommended mitigation will not be implemented at this time, it will be taken into consideration for future releases.

Verification

Unresolved.

Suggestions

Suggestion 1: Consider Lowering the Discard Threshold

Location

`src/wallet/wallet.h`

Synopsis

As mentioned in Issue F, the newly introduced wallet Discard Threshold is conservatively set to one coin. After a sufficient proportion of the network switches to version 1.14.5, this will be 100 times higher than the (soft) Dust limit actually in effect.

Mitigation

We recommend lowering the discard threshold to one cent once network conditions allow it.

Status

The Dogecoin development team has implemented the mitigation as suggested.

Verification

Resolved.

Suggestion 2: Consider Lowering the Soft Dust Limit

Location

`src/wallet/wallet.h`

`src/wallet/wallet.cpp`

Synopsis

As described in Issue A, the soft Dust limit is an "advisory" software control, which transactions relays and miners are intended to accept. It is set 10 times higher than the hard limit imposed for transaction validity. An opportunity exists for the user, if a miner can be identified (and reliably accessed), that is configured with a lower soft Dust limit.

If access to a miner with a lowered Dust limit (down to the network-wide hard Dust limit of 0.1 cent) can be obtained, the user may consider lowering their wallet's `-dustlimit` and `-discardthreshold` parameters accordingly.

Mitigation

We recommend seeking out good access to miners and surveying their apparent configuration. Care must be taken to monitor network conditions with respect to the effective soft Dust limit as well as overall network consensus for changes to the hard Dust limit. Otherwise, the user's transactions could become stuck when conditions change.

Status

The Dogecoin development team stated that they are investigating an alternative solution. As such, this suggestion remains unresolved at the time of verification.

Verification

Unresolved.

Suggestion 3: Switch to Replace-by-Fee

Location

`src/wallet/wallet.h`

Synopsis

A wallet-side parameter has been reduced in version 1.14.5 that lowers the cost of the Replace-by-fee (RBF) strategy for manually increasing the fees paid on a pending transaction as implemented by the `bumpfee` command. This makes RBF favorable to the alternative of "child pays for parent" (CPFP) where a subsequent transaction is made with fees to cover itself and its predecessor.

Mitigation

We recommend users who have been using CPFP to change to RBF after upgrading to version 1.14.5.

Status

The Dogecoin development team stated that the issue is still being investigated. As such, this suggestion remains unresolved at the time of verification.

Verification

Unresolved.

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, and zero-knowledge 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.