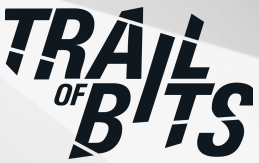

dappOS Virtual Wallet V2

Security Assessment (Summary Report)

August 9, 2023

Prepared for:
dappOS

Prepared by: **Gustavo Grieco and Tarun Bansal**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Executive Summary

Engagement Overview

dappOS engaged Trail of Bits to review the security of version 2 (V2) of its virtual wallet smart contracts.

A team of two consultants conducted the review from July 17 to July 26, 2023, for a total of three engineer-weeks of effort. Our testing efforts focused on finding issues impacting the availability and integrity of the target system. With full access to the source code and documentation, we performed static and dynamic testing of the virtual wallet smart contracts, using automated and manual processes. The off-chain software components used by the system were not included in the scope of this review.

Observations and Impact

The V2 dappOS virtual wallet smart contracts are divided into well-defined components with limited responsibility. The smart contracts are designed to be deployed on multiple blockchains with a focus on minimizing code duplication.

However, some of the issues found during this review are of high severity (TOB-DAOS-1, TOB-DAOS-2) and highlight the need for more thorough testing, including tests to better cover the access control capabilities of every participant in the protocol (users, nodes, and super nodes). The system would also benefit from a comprehensive specification on how rewards and slashes are used to provide incentives for all of the parties involved (TOB-DAOS-6).

The following tables provide the number of findings by severity and category.

EXPOSURE ANALYSIS

Severity	*Count*
High	6
Medium	6
Low	2
Informational	2
Undetermined	1

CATEGORY BREAKDOWN

Category	*Count*
Authentication	1
Data Validation	12
Patching	1
Timing	1
Undefined Behavior	2

Project Summary

Contact Information

The following managers were associated with this project:

Dan Guido, Account Manager
dan@trailofbits.com

Brooke Langhorne, Project Manager
brooke.langhorne@trailofbits.com

The following engineers were associated with this project:

Gustavo Grieco, Consultant
gustavo.grieco@trailofbits.com

Tarun Bansal, Consultant
tarun.bansal@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
July 17, 2023	Pre-project kickoff call
July 21, 2023	Status update meeting #1
July 26, 2023	Delivery of report draft
July 28, 2023	Report readout meeting
August 9, 2023	Delivery of final summary report with fix review (appendix C)

Project Goals

The engagement was scoped to provide a security assessment of the V2 dappOS virtual wallet smart contracts. Specifically, we sought to answer the following non-exhaustive list of questions:

- Can an attacker steal or freeze funds?
- Are there appropriate access control measures in place for users and owners?
- Does the system's behavior match the specification?
- Is it possible to perform system operations without paying the required amount?
- Are there any risks associated with the contracts' use of `delegatecall`?
- Do the contracts avoid or prevent all of the common Solidity flaws?
- Is it possible to destroy a virtual wallet?
- Is it possible to overtake a virtual wallet?

Project Targets

The engagement involved a review and testing of the following target.

dappOS Virtual Wallet V2

Repository	https://github.com/DappOSDao/contracts-core/tree/main
Version	`fbd44704fbab0170fae5b2970b9983ca8d1f0413`
Type	Solidity
Platform	Ethereum

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches included the following:

- **The `VirtualWallet` contract and its dependencies:** These are contract-based accounts to be used as wallets. They support flexible recovery methods to reset their admin controls, and they support gasless transactions. We reviewed how wallets are created and how owners are transferred and reset to look for flaws in the access controls and logic. We checked for replay and malleability issues in the signature handling process. We also checked how gasless transactions are handled by nodes, particularly in corner cases that can produce unexpected reverts (which helped us identify issues such as TOB-DAOS-4).

- **PayDB:** This contract allows users to create cross-chain orders that should be fulfilled by specific participants called nodes. Nodes are required to deposit a certain amount of assets before they start taking orders. We reviewed the complete flow of interactions, from the creation of orders by users to the fulfillment of orders by nodes. We also checked how orders are canceled to make sure the bookkeeping is correct.

Coverage Limitations

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. The following list outlines the coverage limitations of the engagement and indicates system elements that may warrant further review:

- We did not look for issues, such as the following, that could enable high-level economic attacks:
    - Misaligned incentives for nodes to participate in the network
    - Insufficient punishment of misbehaving nodes
    - Unreasonable use of fees to incentive users and/or favor the protocol
- We did not look for complex arbitraging opportunities, particularly those requiring front-running of cross-chain orders.
- We did not review the checks and validations performed by the user interface before the user submits a new transaction for the virtual wallet.
- We did not review any specific wallet service.

Codebase Maturity Evaluation

Trail of Bits uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its codebase is mature, immature, or underdeveloped. Deficiencies identified here often stem from root causes within the software development life cycle that should be addressed through standardization measures (e.g., the use of common libraries, functions, or frameworks) or training and awareness programs.

Category	Summary	Result
Arithmetic	Overall, the codebases do not rely on complex arithmetic. Most of the arithmetic-related complexity is located in the code for order handling. While the order handling process has been documented and some tests are applied to check the arithmetic, the use of a fuzzer—or a similar automated testing method—would increase confidence in the arithmetic operations. For example, a fuzzer could stress test the creation and fulfillment of orders and ensure that their output matches their expectations.	**Moderate**
Auditing	While there are events for critical components to monitor the system off-chain, there is a lack of documentation on how to do so. Additionally, an incident response plan describing how the protocol's actors must react in case of failure was not available during the review period.	**Moderate**
Authentication / Access Controls	The codebase relies on tight access controls to protect virtual wallet funds and to guarantee that user operations are executed correctly. However, we found several issues (such as TOB-DAOS-1, TOB-DAOS-14) that can be mitigated by adding access controls to the critical functions. These issues indicate that the access controls of the system need to be reviewed and redesigned to avoid the loss of funds.	**Weak**
Complexity Management	Although most of the functions are short and well defined, the codebases suffer from overall complexity that may be unnecessary. In particular, the use of `delegatecall` should be carefully considered because it can result in code that is more difficult to use and maintain.	**Weak**

	Additionally, there are a number of unused libraries and functions that should be removed.	
Decentralization	While dappOS has made an effort to develop decentralized portfolios, the owner of a deployed contract still retains control of the source chain configuration (TOB-DAOS-13). Also, it is unclear whether super nodes will be decentralized, which are the only ones that can punish nodes if they misbehave.	**Moderate**
Documentation	The quality of the documentation is uneven across the codebases. While most of the functions include NatSpec documentation, some of them, like those in the `PayLoc` contract, are severely undocumented and require more information. Additionally, a specification that details exactly how and when nodes are rewarded and punished is needed in order to verify whether the implementation is correct (TOB-DAOS-12).	**Moderate**
Front-Running Resistance	The system does not offer any protection against front-running, particularly for some important operations such as wallet creation, transaction execution, transaction cancellation, and owner reset. While the lack of front-running protection may support the protocol as an execution market, it can also result in unexpected behavior, such as the reversion of user transactions (TOB-DAOS-1, TOB-DAOS-7).	**Weak**
Low-Level Manipulation	While the use of low-level code is not extensive in the codebases, a few important functions are missing checks that should be added; for example, the ERC-20 token transfer functions in the `TransferHelper` library contract do not implement contract existence checks (TOB-DAOS-2, TOB-DAOS-4).	**Moderate**
Testing and Verification	Although unit tests are used, they currently fail and do not cover some errors or edge cases; as a result, we encountered code that will not handle exceptions correctly, producing situations that can be exploited (TOB-DAOS-10, TOB-DAOS-15). In addition to the lack of unit testing coverage, there are no fuzz tests.	**Weak**

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	Some important function calls can be front-run using different parameters	Timing	**High**
2	No contract existence check in the TransferHelper library transfer functions	Data Validation	**High**
3	VWManager allows users to call delegatecall on any contract from their virtual wallets	Data Validation	**High**
4	Some important operations can revert, causing gas costs to be impossible to reimburse	Data Validation	**High**
5	Vulnerabilities due to use of outdated Solidity version	Patching	**High**
6	Users can create non-executable and non-cancellable orders to drain server node deposits	Undefined Behavior	**High**
7	Denial-of-service risks on virtual wallet by front-running execute	Data Validation	**Medium**
8	Unsafe ERC-20 transfer for fee payment	Data Validation	**Medium**
9	Lack of two-step process for critical operations	Data Validation	**Medium**
10	Lack of validation of the cross-chain order parameters	Data Validation	**Medium**
11	Anyone can cancel an order to grief users	Data Validation	**Informational**

12	Slashing a lot of orders can be very expensive	Undefined Behavior	**Medium**
13	Owner can delay operations for certain wallets	Data Validation	**Medium**
14	Front-running risks on order execution and wallet creation transactions	Authentication	**Low**
15	The claim function will always set numOnWithdraw to zero	Data Validation	**Low**
16	Both successful and failed virtual wallet transactions can be canceled	Data Validation	**Informational**
17	The use of nonce is error-prone	Data Validation	**Undetermined**

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Code Maturity Categories

The following tables describe the code maturity categories and rating criteria used in this document.

Code Maturity Categories	
Category	**Description**
Arithmetic	The proper use of mathematical operations and semantics
Auditing	The use of event auditing and logging to support monitoring
Authentication / Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system
Complexity Management	The presence of clear structures designed to manage system complexity, including the separation of system logic into clearly defined functions
Cryptography and Key Management	The safe use of cryptographic primitives and functions, along with the presence of robust mechanisms for key generation and distribution
Decentralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Documentation	The presence of comprehensive and readable codebase documentation
Front-Running Resistance	The system's resistance to front-running attacks
Low-Level Manipulation	The justified use of inline assembly and low-level calls
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage

Rating Criteria	
Rating	**Description**
Strong	No issues were found, and the system exceeds industry standards.
Satisfactory	Minor issues were found, but the system is compliant with best practices.
Moderate	Some issues that may affect system safety were found.

Weak	Many issues that affect system safety were found.
Missing	A required component is missing, significantly affecting system safety.
Not Applicable	The category is not applicable to this review.
Not Considered	The category was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

C. Fix Review Results

When undertaking a fix review, Trail of Bits reviews the fixes implemented for issues identified in the original report. This work involves a review of specific areas of the source code and system configuration, not a comprehensive analysis of the system.

On August 3, 2023, Trail of Bits reviewed the fixes and mitigations implemented by the dappOS team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue.

In summary, of the 17 issues disclosed in this report, dappOS has resolved 10 and has partially resolved seven. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Status
1	Some important function calls can be front-run using different parameters	**Partially Resolved** (e629641)
2	No contract existence check in the TransferHelper library transfer functions	**Resolved** (1b2f2de, 598df5c)
3	VWManager allows users to call delegatecall on any contract from their virtual wallets	**Partially Resolved**
4	Some important operations can revert, causing gas costs to be impossible to reimburse	**Resolved** (fddf329, 6b41c47)
5	Vulnerabilities due to use of outdated Solidity version	**Resolved** (1e8c1b9)
6	Users can create non-executable and non-cancellable orders to drain server node deposits	**Resolved**
7	Denial-of-service risks on virtual wallet by front-running execute	**Resolved** (9acd449)

8	Unsafe ERC-20 transfer for fee payment	**Resolved (b8db602)**
9	Lack of two-step process for critical operations	**Partially Resolved**
10	Lack of validation of the cross-chain order parameters	**Partially Resolved**
11	Anyone can cancel an order to grief users	**Partially Resolved**
12	Slashing a lot of orders can be very expensive	**Resolved**
13	Owner can delay operations for certain wallets	**Resolved (3b9e353)**
14	Front-running risks on order execution and wallet creation transactions	**Partially Resolved**
15	The claim function will always set numOnWithdraw to zero	**Resolved (da0f131)**
16	Both successful and failed virtual wallet transactions can be canceled	**Resolved (d6f7cee)**
17	The use of nonce is error-prone	**Partially Resolved**

Detailed Fix Review Results

TOB-DAOS-1: Some important function calls can be front-run using different parameters
Partially resolved in commit e629641. The `isGateway` parameter is now part of the signature data. However, the `feeReceiver` parameter is still not included in the signature data and can be manipulated by attackers.

The dappOS team provided the following context for this finding's fix status:

> *The isGateway parameter will be used when claiming DappOS exclusive rewards and will not result in any user asset loss. The "feeReceiver" is the address to which gas fees for user transactions are returned, and it should be freely designated by the initiator of the transaction to promote healthy competition and improve user transaction speed experience. Ultimately, we have decided to include the isGateway parameter in the signature, while the "feeReceiver" will still be specified by the caller. Fixed in this commit: e629641*

TOB-DAOS-2: No contract existence check in the TransferHelper library transfer functions
Resolved in commits `1b2f2de` and `598df5c`. The dappOS team added a contract existence check to the `token` argument of all the functions of the `TransferHelper` library contract.

TOB-DAOS-3: VWManager allows users to call delegatecall on any contract from their virtual wallets
Partially resolved. The risk of using a malicious service contract has been documented.

The dappOS team provided the following context for this finding's fix status:

> *dappOS will educate users about the risks associated with service contracts through frontend education. Moreover, security audits will be conducted for service contracts that are officially recognized as safe by dappOS, and the results will be publicly disclosed to users through official dappOS channels. Documentation explaining this has been added: https://github.com/DappOSDao/contracts-core#services*

TOB-DAOS-4: Some important operations can revert, causing gas costs to be impossible to reimburse
Resolved in commits `fddf329` and `6b41c47`. The dappOS team added additional `catch` blocks to capture panic and empty revert messages. The `execute` function of the `VMManager` contract has also been updated to revert and log the return data from the `service` contract.

The dappOS team provided the following context for this finding's fix status:

> *We have added error handling for the remaining two types in PayDB:fddf329, and in VWManager, we return the execution information of the service contract using revert and emit:6b41c47.*

TOB-DAOS-5: Vulnerabilities due to use of outdated Solidity version
Resolved in commit 1e8c1b9. The Solidity version has been upgraded to 0.8.19. However, we advise the dappOS team **to be careful while upgrading the Solidity version to ensure that new opcodes introduced in the updated version are available on all of the target blockchain networks**. If a network does not support some opcodes, then the team can configure the compiler to target a specific EVM version instead of the latest EVM version.

TOB-DAOS-6: Users can create non-executable and non-cancellable orders to drain server node deposits
Resolved. The server node has been given the right to deny refunds for malicious orders.

The dappOS team provided the following context for this finding's fix status:

> *Documentation has been added to explain the matching mechanism between user orders and nodes:*
> *https://github.com/DappOSDao/contracts-core#paydb*
> *https://github.com/DappOSDao/contracts-core#payorder*

TOB-DAOS-7: Denial-of-service risks on virtual wallet by front-running execute
Resolved in commit 9acd449. The dappOS team made `verifyProof` an internal function.

TOB-DAOS-8: Unsafe ERC-20 transfer for fee payment
Resolved in commit b8db602. The dappOS team added a low-level call to the `_feeToken` contract to transfer the tokens and a check on the response and return data. However, **this fix does not include a contract existence check for the `_feeToken` address; if this address does not have a contract deployed, the transaction will still return success even though it did not transfer any tokens**.

TOB-DAOS-9: Lack of two-step process for critical operations
Partially resolved. The dappOS team added front-end checks and documentation to educate users about the risks associated with this operation. However, these measures do not fully mitigate the risk that ownership of a virtual wallet could be transferred to an erroneous address.

The dappOS team provided the following context for this finding's fix status:

> *Regarding information confirmation, it aligns with the level of risk associated with transfer operations and does not directly result in user asset loss due to attacks. To maintain a good user experience, dappOS still allows one-step completion of virtual wallet transfers. However, it will provide users with prompt messages and a two-step*

confirmation process on the frontend operation page. Documentation explaining this has been added:
https://github.com/DappOSDao/contracts-core#changeowner

TOB-DAOS-10: Lack of validation of the cross-chain order parameters
Partially resolved. Documentation has been added to specify that the super node verifies the correct execution of the order with the correct `tokenOut` and `amountOut` values. However, the `tokenOut` and `amountOut` values are not included in the `payOrderId`, and anyone can execute orders.

The dappOS team provided the following context for this finding's fix status:

> *Documentation has been added to explain the mechanism of creating orders in the dappOS system and the superNode's maintenance of consensus for penalties:*
> *https://github.com/DappOSDao/contracts-core#paydb*
> *https://github.com/DappOSDao/contracts-core#payorder*
> *https://github.com/DappOSDao/contracts-core#punishment*

TOB-DAOS-11: Anyone can cancel an order to grief users
Partially resolved. The platform now requires server nodes to execute canceled orders, reducing the risk that anyone could cancel orders to grief users. Documentation has been added to specify that an order is considered canceled only when the assigned server node or the super node cancels it.

The dappOS team provided the following context for this finding's fix status:

> *Documentation has been added to explain the cancellation mechanism:*
> *https://github.com/DappOSDao/contracts-core#cancel*

TOB-DAOS-12: Slashing a lot of orders can be very expensive
Resolved. The dappOS team added a minimum refund threshold for server node punishment.

The dappOS team provided the following context for this finding's fix status:

> *Documentation has been added to provide an explanation of the cancellation of orders and the associated penalty mechanism:*
> *https://github.com/DappOSDao/contracts-core#cancel*
> *https://github.com/DappOSDao/contracts-core#punishment*

TOB-DAOS-13: Owner can delay operations for certain wallets
Resolved in commit 3b9e353. The dappOS team implemented a two-step process to update the source chain configuration with a 24-hour delay.

TOB-DAOS-14: Front-running risks on order execution and wallet creation transactions
Partially resolved. The front-running risks of order execution are mitigated by the `createWalletIfNotExists` function, which fetches the existing wallet of the provided owner. However, the `createWallet` function can still be front-run to make calls to it revert.

The dappOS team provided the following context for this finding's fix status:

> *If a transaction is front-run during the execution of the contract, the createWalletIfNotExists function of the PayDB contract can read the correct wallet of the owner and invoke the transaction execution function with it as a parameter. Documentation has been added:*
> *https://github.com/DappOSDao/contracts-core#virtual-wallet*

TOB-DAOS-15: The claim function will always set numOnWithdraw to zero
Resolved in commit `da0f131`. The claim function now sets the correct value of `nomOnWithdraw`. However, we still advise adding a specific unit test to make sure that this feature works as expected.

TOB-DAOS-16: Both successful and failed virtual wallet transactions can be canceled
Resolved in commit `d6f7cee`. The dappOS team added a check to ensure that only valid orders can be canceled.

TOB-DAOS-17: The use of nonce is error-prone
Partially resolved. The correct use of the nonce has been documented, with advice to developers not to use the same nonce again.

The dappOS team provided the following context for this finding's fix status:

> *Documentation has been added to provide an explanation of the parameters composing the code and payOrderId:*
> *https://github.com/DappOSDao/contracts-core#code*
> *https://github.com/DappOSDao/contracts-core#payorderid*
> *https://github.com/DappOSDao/contracts-core#nonce-1*