wXTZ

Security Assessment

December 28, 2020

Prepared For:
Christian Arita | *StakerDAO*
christian@stakerdao.com

Prepared By:
Samuel Moelius | *Trail of Bits*
sam.moelius@trailofbits.com

Michael Colburn | *Trail of Bits*
michael.colburn@trailofbits.com

Dominik Teiml | *Trail of Bits*
dominik.teiml@trailofbits.com

Changelog:

November 30, 2020:	Initial report delivered
December 2, 2020:	Engagement Summary corrected
December 23, 2020:	Added Appendix D. Fix Log
December 28, 2020:	Fix Log Findings 3 and 8 corrected

Executive Summary

From November 16 through 25, 2020, StakerDAO engaged Trail of Bits to review the security of Wrapped XTZ (wXTZ). Trail of Bits conducted this assessment over the course of two person-weeks with three engineers working from commits `a0199f15` and `9c39d3c0` of the `wrapped-xtz` repository.

During the first week, we verified that we could build the code and run the unit tests. We also began manual review, focusing on the oven and core components, minus the core's upgradeability mechanism.

During the second week, we continued manual review, focusing on the core's upgradeability mechanism, and the `tzip-7` token implementation.

Our efforts resulted in 13 findings ranging from high to informational severity, as well as some non–security-related findings (Appendix C). One high-severity finding concerns a data validation error in the tzip-7 contract that allows users to mint tokens. A second high-severity finding concerns an edge case where a user can claim tokens owned by the tzip-7 contract itself. The medium-severity finding concerns reliance on dependencies with NPM advisories. One low-severity finding concerns how Ganache CLI is configured. A second low severity finding concerns assumptions made about the behavior of arbitrary value lambdas. The remaining eight findings are of informational severity, meaning they do not pose immediate risks, but potential ones.

We recommend that all of the findings in this report be addressed. Most notably, this includes TOB-WXTZ-013, which could be used to completely undermine the system. It also includes more mundane fixes, such as filling gaps in test coverage and fleshing out the documentation.

A crucial invariant of the system appears to be that the total supply of wXTZ equals the total number of locked XTZ in `mutez`. We recommend that this invariant be incorporated into the tests. This might be accomplished using a Truffle "after" hook, for example.

Finally, we recommend developing a plan for relinquishing control of the core and tzip-7 contracts. It is common within the industry to relinquish control of a contract following a trial period. Doing so demonstrates adherence to the "code is law" principle, giving confidence to the community that uses those contracts. Details for how this might be accomplished are given in TOB-WXTZ-009 and TOB-WXTZ-010.

Update December 23, 2020: StakerDAO provided an updated codebase that addresses issues from this report. See Appendix D for a detailed discussion of the exact status of each issue.

Project Dashboard

Application Summary

Name	wXTZ
Version	a0199f15bd80f55d8a68d92f7bb22e20d2f2a2a09c39d3c087c252632ecde59692fa45e7617bff7c
Type	Reason Ligo
Platforms	Tezos

Engagement Summary

Dates	November 16–23, 2020
Method	Whitebox
Consultants Engaged	3
Level of Effort	4 person-weeks

Vulnerability Summary

Total High-Severity Issues	2	■■
Total Medium-Severity Issues	1	■
Total Low-Severity Issues	2	■■
Total Informational-Severity Issues	8	■■■■■■■■
Total	13	

Category Breakdown

Access Controls	3	■■■
Data Validation	4	■■■■
Documentation	1	■
Error Reporting	1	■
Patching	2	■■
Undefined Behavior	2	■■
Total	13	

Code Maturity Evaluation

Category Name	Description
Access Controls	**Strong.** We observed no problems related to access controls. Core operations requiring administrative permissions appear to check for them wherever necessary. The same is true for tzip-7 operations requiring administrative permissions, and oven operations requiring owner permissions.
Arithmetic	**Strong.** We observed no problems related to arithmetic. Stove Labs said that an earlier bug caused a multiplication overflow. However, we were not able to reproduce it.
Assembly Use	**Satisfactory.** The file `ovenWrapper.religo` contains cut-and-pasted, compiled oven code. A comment notes that this solution is not ideal. While the file appears to be used only in tests, we recommend finding a better long-term solution.
Decentralization	**Weak.** A compromised core administrator could steal wXTZ or deposited XTZ. A compromised tzip-7 administrator could steal wXTZ.
Code Complexity	**Satisfactory.** Aside from whitespace issues, the code is largely easy to read. The fact that many checks are performed in core makes them slightly harder to verify. However, this may be a necessary tradeoff of the design.
Front-Running	**Satisfactory.** An issue was noted where users might “race” to obtain wXTZ owned by the tzip-7 contract itself. No other front-running issues were noted.
Key Management	**Not Applicable.**
Monitoring	**Not Considered.**
Specification	**Moderate.** Documentation does not cover all operations. Also, it is hidden in a “feature” branch.
Testing & Verification	**Moderate.** Many errors are not currently tested for. The core’s upgradeability mechanism is not currently tested. Some existing tests have problems.

Engagement Goals

The engagement was scoped to provide a security assessment of the core, oven, and tzip-7 contracts.

Specifically, we sought to answer the following questions:

- Can the core's upgradability mechanism be abused?
- Are the right permissions checks performed in the core and oven contracts?
- Are there type errors related to the use of `Bytes.pack` and `Bytes.unpack`?
- Does the tzip-7 contract comply with the tzip-7 standard?

Coverage

Core contract. Built. Tests run and reviewed. Code manually reviewed with an emphasis on verifying absence of arithmetic, type, and re-entrancy errors; proper update of storage; and proper application of permissions checks.

Oven contract. Built. Tests run and reviewed. Code manually reviewed with an emphasis on verifying absence of arithmetic, type, and re-entrancy errors; proper update of storage; and proper application of permissions checks.

TZIP-7 contract. Built. Tests run and reviewed. Code manually reviewed with an emphasis on verifying absence of arithmetic, type, and re-entrancy errors; proper update of storage; and proper application of permissions checks. Checked against the tzip-7 standard. Checked for errors commonly affecting ERC20 tokens. Checked that the pausing mechanism is used consistently.

NPM dependencies. Analyzed using `npm audit`.

Recommendations Summary

This section aggregates all the recommendations made during the engagement. Short-term recommendations address the immediate causes of issues. Long-term recommendations pertain to the development process and long-term design goals.

Short term

❑ **Apply the remediations by updating the dependencies wherever possible. If a vulnerable dependency is relied upon indirectly and no update is available for the corresponding direct dependency, use a patched version of the direct dependency until an update becomes available. For `web3`, document the vulnerability so users know not to rely on its credential storage.** These steps will help protect developers and users from the vulnerabilities in wXTZ's current dependencies. TOB-WXTZ-001

❑ **Adjust `start_ganache-cli.sh` so that `ganache-cli` is bound to `127.0.0.1` (`localhost`) instead of `0.0.0.0`.** This will eliminate unnecessary risk currently posed to wXTZ developers who run the tests. TOB-WXTZ-002

❑ **Add unit tests for the 11 error types not currently checked for in any unit test. Add unit tests for the core's upgradeability mechanism.** Ideally, each function will have at least one test for each of its "happy" (successful) paths, and at least one test for each of its "sad" (failing) paths. A comprehensive set of unit tests will help expose errors, protect against regressions, and provide a sort of documentation to users. TOB-WXTZ-003

❑ **Expand the documentation to cover all of the core, oven, and tzip-7 contract operations. Describe each operation's behavior as well as its inputs and outputs. Move the documentation out of the "feature" branch and into the main branch.** These steps will reduce the likelihood that a wXTZ contract operation's behavior will be misinterpreted. TOB-WXTZ-004

❑ **Have each caller of `runArbitraryValueLambda` incorporate its returned storage into the caller's returned storage.** This will help ensure correct operation if a future version of an arbitrary value lambda modifies storage. TOB-WXTZ-005

❑ **Implement entry points in the core for `setAdministrator`, `setPause`, and `setPauseGuardian`.** This will eliminate the need to add such entry points after the core's deployment. TOB-WXTZ-006

❑ **In the code in Figure 7.1, change `Map.update` to `Big_map.update`.** This will ensure that the code continues to work with future versions of the Reason Ligo compiler. TOB-WXTZ-007

❏ **Disallow the core, the t-zip7 contract, and existing ovens from becoming oven owners by adding code like that in Figure 8.2 to `createOven`.** This will help prevent edge cases that could be discovered from affecting the security of the contracts. TOB-WXTZ-008

❏ **Come up with a long-term plan for disabling the core's admin account.** This could be accomplished by overwriting the core's `isAdmin` lambda with one that always fails. Disabling the account after a fixed period of time will reduce the risk of compromise. TOB-WXTZ-009

❏ **Come up with a long-term plan for disabling the tzip-7 contract admin account.** This could be accomplished by using `setAdministrator` to set the account to a nonexistent address. Disabling the account after a fixed period of time will reduce the risk of compromise. TOB-WXTZ-010

❏ **Make sure the token contract is fully compliant with tzip-7, or explicitly document all the ways in which it deviates.** This will help ensure that contract users do not misinterpret its intended behavior. TOB-WXTZ-011

❏ **Disallow users to claim tokens that the token contract holds.** This will eliminate a vulnerability whereby users can claim the tokens owned by that contract. TOB-WXTZ-012

❏ **Replace `tokenStorage.ledger` on L60 for `newTokens`.** This will eliminate a critical vulnerability that could be used to completely undermine the system. TOB-WXTZ-013

Long term

❏ **Regularly run `npm audit` over the codebase to help reveal vulnerable dependencies.** TOB-WXTZ-001

❏ **Regularly review the logs produced by external software components that you rely on.** We found this bug by reviewing `ganache-cli`'s logs. Regularly reviewing logs can reveal potential attack surface and exploitation attempts. TOB-WXTZ-002

❏ **Ensure unit tests are written for new functions as they are added to the codebase to prevent them from introducing bugs.** TOB-WXTZ-003

❏ **If additional operations are added to the contracts (e.g., using `updateLambdas`) ensure they are properly documented so their behavior is not misinterpreted.** TOB-WXTZ-004

❏ **As new uses of `runArbitraryValueLambda` are added to the codebase, ensure they do not apply this problematic pattern.** This will help prevent future code from breaking. TOB-WXTZ-005

❏ **If additional admin-only operations are added to the tzip-7 contract, ensure that corresponding entry points are added to the core.** This way, such entry points won't need to be added after the core's deployment. TOB-WXTZ-006

❏ **Consider reporting this as a bug to the Reason Ligo compiler maintainers.** Disallowing this code pattern sooner will mean less erroneous code exists in the wild, which will benefit the Tezos community as a whole. TOB-WXTZ-007

❏ **If additional contracts are added to the wXTZ codebase, consider whether they should be disallowed as oven owners.** This will help ensure that edge cases affecting the security of the contracts are not introduced into the codebase. TOB-WXTZ-008

❏ **If additional contracts requiring admin access are added to the codebase, ensure there is a plan to disable such access after a fixed period of time.** Doing so will demonstrate adherence to the "code is law" principle, giving confidence to the community that uses the contracts. TOB-WXTZ-009, TOB-WXTZ-010

❏ **Prioritize complying with industry standards to ease interaction with the contracts.** Deviating from established standards invites misuse. TOB-WXTZ-011

❏ **Avoid introducing special cases like the one described here.** Fewer special cases will result in code that is easier to reason about. TOB-WXTZ-012

❏ **Make sure the testing strategy includes edge cases such as two addresses representing the same account, and make sure to check that all invariants of the system are satisfied after every message.** This will help ensure that similar bugs are not introduced into the system. TOB-WXTZ-013

Findings Summary

#	Title	Type	Severity
1	Code relies on vulnerable NPM packages	Patching	Medium
2	Ganache CLI is configured to listen on all network interfaces	Access Controls	Low
3	Insufficient tests	Error Reporting	Informational
4	Insufficient documentation	Documentation	Informational
5	Calls to `runArbitraryValueLambda` assume storage does not change	Undefined Behavior	Low
6	Core lacks entry points for tzip-7 admin-only operations	Patching	Informational
7	Type confusion in `updateLambdas`	Undefined Behavior	Informational
8	Insufficient validation of newly created oven owners	Data Validation	Informational
9	A compromised core administrator could steal wXTZ or deposited XTZ	Access Controls	Informational
10	A compromised tzip-7 administrator could steal wXTZ	Access Controls	Informational
11	wXTZ deviates from the tzip-7 specification	Data Validation	Informational
12	Possible race condition when wXTZ owns tokens	Data Validation	High
13	Token holders can double their token balances	Data Validation	High

1. Code relies on vulnerable NPM packages

Severity: Medium
Difficulty: Undetermined
Type: Patching
Finding ID: TOB-WXTZ-001
Target: Various

Description
Running `npm audit` over the codebase produces the following warning:

```
    found 1805 vulnerabilities (1000 low, 5 moderate, 800 high) in 1634
scanned packages
```

The 1,805 vulnerabilities involve the 12 advisories in Table 1.1.

NPM Advisory	Description	Dependency
566	Prototype Pollution	`hoek`
598	Memory Exposure	`tunnel-agent`
786	Regular Expression Denial of Service	`braces`
877	Insecure Credential Storage	`web3`
1179	Prototype Pollution	`minimist`
1217	Arbitrary File Write	`decompress`
1464	Insufficient Entropy	`cryptiles`
1500	Prototype Pollution	`yargs-parser`
1523	Prototype Pollution	`lodash`
1547	Signature Malleability	`elliptic`
1555	Remote Memory Exposure	`bl`
1556	Denial of Service	`node-fetch`

Table 1.1: NPM advisories affecting wXTZ's dependencies.

With the sole exception of 877 (`web3`), the remediation is simply to update the dependency. For `web3`, the remediation is to "consider using an alternative module until a fix is made available."

Exploit Scenario
Eve discovers a code path leading to a vulnerable dependency. Eve uses this code path to crash developers' machines, corrupt memory, etc.

Recommendations

Short term, apply the remediations by updating the dependencies wherever possible. If a vulnerable dependency is relied upon indirectly and no update is available for the corresponding direct dependency, use a patched version of the direct dependency until an update becomes available. For `web3`, document the vulnerability so users know not to rely on its credential storage. These steps will help protect developers and users from the vulnerabilities in wXTZ's current dependencies.

Long term, regularly run `npm audit` over the codebase to help reveal vulnerable dependencies.

2. Ganache CLI is configured to listen on all network interfaces

Severity: Low
Difficulty: High
Type: Access Controls
Finding ID: TOB-WXTZ-002
Target: `scripts/sandbox/start_ganache-cli.sh`

Description
Ganache CLI is used as the Tezos blockchain for tests. When it is started, it is told to bind to address `0.0.0.0`, causing it to listen on all interfaces. This poses unnecessary risk to developers who run the tests.

The wXTZ documentation says to run "`npm run sandbox:start`" (Figure 2.1). This command launches the script `start_ganache-cli.sh` (Figure 2.2), and that script launches `ganache-cli` with "`--host 0.0.0.0`" (Figure 2.3).

~~~
## Starting Carthage Sandbox

```
npm run sandbox:start
```
~~~

Figure 2.1: README.md#L9-L13.

```
    "sandbox:start": "./scripts/sandbox/start_ganache-cli.sh",
```

Figure 2.2: package.json#L36.

```
./node_modules/ganache-cli/cli.js --flavor tezos --seed alice --accounts 10 --host 0.0.0.0
```

Figure 2.3: scripts/sandbox/start_ganache-cli.sh#L1.

Using `nmap` from an external machine, we confirmed that the port (8732) was accessible remotely.

Exploit Scenario
Eve discovers a remote code execution vulnerability in `ganache-cli`. Eve uses the bug to get execution on a machine belonging to Alice, a wXTZ developer.

Recommendations
Short term, adjust `start_ganache-cli.sh` so that `ganache-cli` is bound to `127.0.0.1` (`localhost`) instead of `0.0.0.0`. This will eliminate unnecessary risk currently posed to wXTZ developers who run the tests.

Long term, regularly review the logs produced by external software components that you rely on. We found this bug by reviewing `ganache-cli`'s logs. Regularly reviewing logs can reveal potential attack surfaces and exploitation attempts.

3. Insufficient tests

Severity: Informational
Difficulty: Undetermined
Type: Error Reporting
Finding ID: TOB-WXTZ-003
Target: Various in `test` subdirectory

Description

Excluding bridge-related code, there are 23 error types. However, only 12 are checked for in tests (Table 3.1).

Error type	Where tested for
`errorAdminAddressWrongType`	-
`errorAllowanceMismatch`	-
`errorAmountNotZero`	`oven.js`, `onOvenSetDelegate.js`, `default.js`, `onOvenWithdrawalRequested.js`, `withdraw.js`
`errorArbitraryValueKeyNotFound`	-
`errorArbitraryValueWrongType`	-
`errorCoreContractEntrypointTypeMissmatch`	-
`errorLambdaNotAnEntrypoint`	`runEntrypointLambda.js`
`errorLambdaNotArbitrary`	-
`errorLambdaNotFound`	`runEntrypointLambda.js`
`errorLambdaParameterWrongType`	`createOven.js`
`errorNoContract`	-
`errorNoPermission`	`tzip-7.js`
`errorNotAnOvenOwner`	`oven.js`, `onOvenSetDelegate.js`, `onOvenWithdrawalRequested.js`
`errorNotEnoughAllowance`	`tzip-7.js`
`errorNotEnoughBalance`	`tzip-7.js`
`errorOvenMissingDefaultEntrypoint`	`onOvenDepositReceived.js`
`errorOvenNotFound`	-
`errorOvenNotTrusted`	`onOvenDepositReceived.js`
`errorOvenOwnerDoesNotAcceptDeposits`	`withdraw.js`

`errorSenderIsNotAdmin`	-
`errorTokenOperationsArePaused`	-
`errorUnsafeAllowanceChange`	`tzip-7.js`
`errorWXTZTokenContractWrongType`	-

Table 3.1: Error types and the files in which they are tested for.

Also, the core's upgradeability mechanism is largely untested.

Unit tests help expose errors and provide a sort of documentation of the code. Moreover, unit tests exercise code in a more systematic way than any human can, and thus help protect against regressions.

Exploit Scenario
Eve exploits a flaw in wXTZ that would likely have been revealed through unit tests.

Recommendations
Short term, add unit tests for the 11 error types not currently checked for in any unit test. Add unit tests for the core's upgradeability mechanism. Ideally, each function will have at least one test for each of its "happy" (successful) paths, and at least one test for each of its "sad" (failing) paths. A comprehensive set of unit tests will help expose errors, protect against regressions, and provide a sort of documentation to users.

Long term, ensure unit tests are written for new functions as they are added to the codebase to prevent them from introducing bugs.

4. Insufficient documentation

Severity: Informational	Difficulty: Not applicable
Type: Documentation	Finding ID: TOB-WXTZ-004
Target: `feature/docs` branch	

Description

The wXTZ contracts would benefit from more thorough documentation. The documentation should also be placed front and center, not hidden away in a "feature" branch.

Currently, the documentation covers only these core lambdas:

- `createOven`
- `onOvenDepositReceived`
- `onOvenWithdrawalRequested`

The documentation does not cover these core lambdas:

- `composeBurnOperation`
- `composeMintOperation`
- `isAdmin`
- `isOvenOwner`
- `isTrustedOven`
- `updateLambdas`
- `onOvenSetDelegate`

For the core lambdas that are covered, only `createOven`'s behavior is described. For `onOvenDepositReceived` and `onOvenWithdrawalRequested`, only their inputs and outputs are given.

Finally, the documentation describes the wXTZ token only as a tzip-7 smart contract. The documentation should mention all of the ways the contract differs from a standard tzip-7 smart contract, e.g., the pausing mechanism. (See also TOB-WXTZ-011.)

Exploit Scenario

Alice, a Tezos developer, writes a contract that interacts with the wXTZ contracts. Alice misinterprets how a wXTZ contract operation works and loses funds as a result.

Recommendations

Short term, expand the documentation to cover all of the core, oven, and tzip-7 contract operations. Describe each operation's behavior as well as its inputs and outputs. Move the documentation out of the "feature" branch and into the main branch. These steps will reduce the likelihood that a wXTZ contract operation's behavior will be misinterpreted.

Long term, if additional operations are added to the contracts (e.g., using `updateLambdas`) ensure they are properly documented so their behavior is not misinterpreted.

5. Calls to `runArbitraryValueLambda` assume storage does not change

Severity: Low
Type: Undefined Behavior
Target: various

Difficulty: High
Finding ID: TOB-WXTZ-005

Description

All calls to `runArbitraryValueLambda` discard the returned storage, assuming it does not change. While this is true of all current arbitrary value lambdas, a future update could break this assumption.

For example, `composeMintOperation` *does* return the storage it is passed, leaving it unchanged (Figure 5.1). However, this assumption is implicit where `composeMintOperation` is invoked in `createOven` (Figure 5.2).

```
((arbitraryValueLambdaParameter, storage): (arbitraryValueLambdaParameter, storage)):
arbitraryValueLambdaReturnValue => {
    ...
    (operations, storage, Bytes.pack(()));
}
```

Figure 5.1:
contracts/partials/wxtz/core/lambdas/arbitrary/composeMintOperation/composeMintOperation.religo#L1-L34.

```
    let (mintWXTZOperationList, _, _) = runArbitraryValueLambda((
        {
            lambdaName: "arbitrary/composeMintOperation",
            lambdaParameter: composeMintOperationParameter,
        },
        storage
    ));
```

Figure 5.2:
contracts/partials/wxtz/core/lambdas/createOven/createOven.religo#L56-L62.

Exploit Scenario

An arbitrary value lambda is updated using `updateLambdas`. The new lambda modifies the storage, causing existing code to break.

Recommendations

Short term, have each caller of `runArbitraryValueLambda` incorporate its returned storage into the caller's returned storage. This will help ensure correct operation if a future version of an arbitrary value lambda modifies storage.

Long term, as new uses of `runArbitraryValueLambda` are added to the codebase, ensure they do not apply this problematic pattern. This will help prevent future code from breaking.

6. Core lacks entry points for tzip-7 admin-only operations

Severity: Informational
Difficulty: High
Type: Patching
Finding ID: TOB-WXTZ-006
Target: core contract

Description

The core needs to be able to mint and burn wXTZ tokens. This requires the core to hold the tzip-7 contract "admin" address. It also makes the core the only sender capable of invoking certain admin-only operations on the tzip-7 contract. However, the core lacks entry points for many of those operations.

As shown in Figure 6.1, the core mints wXTZ tokens in `onOvenDepositReceived`. In Figure 6.2, this requires the core to hold the tzip-7 contract "admin" address.

```
let composeMintOperationParameter: composeMintOperationParameter = {
    to_: ovenOwner,
    value: Tezos.amount / 1mutez // TODO: extract as tezToNat(tez)
};
```

Figure 6.1: contracts/partials/wxtz/core/lambdas/onOvenDepositReceived/onOvenDepositReceived.religo#L42-L45.

```
// only the admin is allowed to mint tokens
switch(Tezos.sender == tokenStorage.admin) {
    | true => unit
    | false => (failwith(errorNoPermission): unit)
};
```

Figure 6.2: contracts/partials/wxtz/tzip7/mint/mint.religo#L8-L12.

Other tzip-7 operations also require the sender to be the "admin." One example is `setPause`, which requires the sender to be the admin when unpausing (Figure 6.3). However, the core lacks an entry point for `setPause`.

```
switch (setPauseParameter) {
    | true => {
        switch (Tezos.sender == tokenStorage.pauseGuardian) {
            | false => (failwith(errorNoPermission): unit)
            | true => unit
        }
    }
    | false => {
        switch (Tezos.sender == tokenStorage.admin) {
            | false => (failwith(errorNoPermission): unit)
            | true => unit
        };
    }
};
```

Figure 6.3: contracts/partials/wxtz/tzip7/setPause/setPause.religo#L6-L19.

More generally, the core lacks entry points for the following operations, which could require admin privileges under certain circumstances:

- `setAdministrator`
- `setPause`
- `setPauseGuardian`

Such entry points could be added after the core's deployment using `updateLambdas`. However, deploying the core with such entry points already implemented would be less error-prone.

Exploit Scenario
StakerDAO tries to add entry points for the above operations after the core's deployment. A mistake in a script causes the wrong lambdas to be overwritten. The wXTZ contracts become inoperable.

Recommendations
Short term, implement entry points in the core for `setAdministrator`, `setPause`, and `setPauseGuardian`. This will eliminate the need to add such entry points after the core's deployment.

Long term, if additional admin-only operations are added to the tzip-7 contract, ensure that corresponding entry points are added to the core. This way, such entry points won't need to be added after the core's deployment.

7. Type confusion in `updateLambdas`

Severity: Informational
Type: Undefined Behavior
Target: `updateLambdas.religo`

Difficulty: High
Finding ID: TOB-WXTZ-007

Description

The function `updateLambdas` uses `Map.update` to update `storage.lambdas` (Figure 7.1). Since `storage.lambdas` is a `big_map`, this should be `Big_map.update`.

```
    let updateLambdasAccumulator: updateLambdasAccumulator = storage.lambdas;
    let updateLambdasIterator: updateLambdasIterator =
        ((updateLambdasAccumulator, lambdaUpdate): updateLambdasIteratorParameter):
updateLambdasAccumulator => {
            let (lambdaName, optionalPackedLambda) = lambdaUpdate;
            // optionalPackedLambda can be Some/None to upsert/remove the entry
            Map.update(lambdaName, optionalPackedLambda, updateLambdasAccumulator)
        };
```

Figure 7.1: [contracts/partials/wxtz/core/lambdas/updateLambdas/updateLambdas.religo#L26-L32]().

Note that, in its current form, the code compiles and exhibits correct behavior. However, the code should be fixed in the event that future compilers are not so lenient.

Exploit Scenario

A future version of the Reason Ligo compiler does not allow `big_maps` to be updated with `Map.update`. The function `updateLambdas` no longer compiles. Time and effort is wasted trying to determine the source of the error.

Recommendations

Short term, in the code in Figure 7.1, change `Map.update` to `Big_map.update`. This will ensure that the code continues to work with future versions of the Reason Ligo compiler.

Long term, consider reporting this as a bug to the Reason Ligo compiler maintainers. Disallowing this code pattern sooner will mean less erroneous code exists in the wild, which will benefit the Tezos community as a whole.

8. Insufficient validation of newly created oven owners

Severity: Informational
Type: Data Validation
Target: `createOven.religo`

Difficulty: High
Finding ID: TOB-WXTZ-008

Description
In some parts of the code, the addresses of the known wXTZ contracts are treated as special cases. An example appears in Figure 8.1. (See also TOB-WXTZ-012.) To avoid obscure edge cases, the core should disallow itself, the t-zip7 contract, and existing ovens from becoming oven owners.

```
    if (Tezos.sender == coreContractAddress) {
         /**
          * If the deposit comes from the wXTZ Core, then do nothing.
          * This prevents an endless core-hook transaction loop
          */
         ([]: list(operation), storage)
     } else {
```

Figure 8.1: contracts/partials/wxtz/core/lambdas/createOven/oven/default/default.religo#L14-L20.

Since there is no apparent reason to allow the wXTZ contracts to become oven owners, code similar to Figure 8.2 should be added to `createOven`.

```
     switch (ovenOwner == lambdaExtras.selfAddress) {
          | false => ()
          | true => failwith("core cannot be oven owner"): unit
     };

     let wXTZTokenContractAddress: address = getWXTZTokenContractAddress(storage);
     switch (ovenOwner == wXTZTokenContractAddress) {
          | false => ()
          | true => failwith("wXTZ contract cannot be oven owner"): unit
     };

     let ovenOwnerOwner: option(address) = Big_map.find_opt(ovenOwner, storage.ovens);
     switch (ovenOwnerOwner) {
          | None => ()
          | Some(ovenOwnerOwner) => failwith("existing oven cannot be oven owner"): unit
     };
```

Figure 8.2: Code to be added to `createOven`.

Exploit Scenario
An edge case is discovered where having an oven owned by the core allows a user to mint wXTZ tokens. Eve exploits the bug knowing that ovens owned by core are permitted.

Recommendations

Short term, disallow the core, the t-zip7 contract, and existing ovens from becoming oven owners by adding code like that in Figure 8.2 to `createOven`. This will help prevent edge cases that could be discovered from affecting the security of the contracts.

Long term, if additional contracts are added to the wXTZ codebase, consider whether they should be disallowed as oven owners. This will help ensure that edge cases affecting the security of the contracts are not introduced into the codebase.

9. A compromised core administrator could steal wXTZ or deposited XTZ

Severity: Informational
Type: Access Controls
Target: `createOven.religo`

Difficulty: High
Finding ID: TOB-WXTZ-009

Description
If the core administrator account were compromised, it could be used to steal all deposited XTZ.

For example, if an attacker uses `updateLambdas` to overwrite the core's `onWithdrawalRequested` lambda with one that always succeeds, they can then withdraw the funds deposited in all ovens.

Since the core necessarily holds the tzip-7 contract admin address (see TOB-WXTZ-006), a compromise of the core administrator account could also be used to steal wXTZ (see TOB-TOB-010).

Exploit Scenario
Eve is a wXTZ developer with access to the core administrator's credentials. Eve steals all deposited XTZ and disappears.

Recommendation
Short term, come up with a long-term plan for disabling the core's admin account. This could be accomplished by overwriting the core's `isAdmin` lambda with one that always fails. Disabling the account after a fixed period of time will reduce the risk of compromise.

Long term, if additional contracts requiring admin access are added to the codebase, ensure there is a plan to disable such access after a fixed period of time. Doing so will demonstrate adherence to the "code is law" principle, giving confidence to the community that uses the contracts.

10. A compromised tzip-7 administrator could steal wXTZ

Severity: Informational
Difficulty: High
Type: Access Controls
Finding ID: TOB-WXTZ-010
Target: tzip-7 contract

Description
If the tzip-7 contract administrator account were compromised, it could be used to steal wXTZ.

An attacker could accomplish this by burning tokens in one account and minting them in another. Note that a burn followed by a mint is effectively the same as a transfer. Thus, an attacker with control of the tzip-7 administrator account has effectively unlimited transfer power.

Exploit Scenario
Eve is a wXTZ developer with access to the tzip-7 contract's administrator credentials. Eve steals all the wXTZ and disappears.

Recommendation
Short term, come up with a long-term plan for disabling the tzip-7 contract admin account. This could be accomplished by using `setAdministrator` to set the account to a nonexistent address. Disabling the account after a fixed period of time will reduce the risk of compromise.

Long term, if additional contracts requiring admin access are added to the codebase, ensure there is a plan to disable such access after a fixed period of time. Doing so will demonstrate adherence to the "code is law" principle, giving confidence to the community that uses the contracts.

11. wXTZ deviates from the tzip-7 specification

Severity: Informational
Type: Data Validation
Target: `partials/wxtz/tzip7/transfer/transfer.religo`

Difficulty: Not applicable
Finding ID: TOB-WXTZ-011

Description
The tzip-7 specification says that unless the transaction sender is the `from` account, the transfer function must check if sufficient approval has been granted, and decrease that approval accordingly. However, when the `from` account is the token contract itself, this is not done:

```
        let thisContractIsTokenOwner = Tezos.self_address == transferParameter.from_;
        let newAllowances = switch(senderIsTokenOwner || thisContractIsTokenOwner) {
           | true => tokenStorage.approvals
```

Figure 11.1: `contracts/partials/wxtz/tzip7/transfer/transfer.religo#L16-L18`.

In this way, the token contract deviates from the standard.

Recommendation
Short term, make sure the token contract is fully compliant with tzip-7, or explicitly document all the ways in which it deviates. This will help ensure that contract users do not misinterpret its intended behavior.

Long term, prioritize complying with industry standards to ease interaction with the contracts. Deviating from established standards invites misuse.

12. Possible race condition when wXTZ owns tokens

Severity: High
Difficulty: Low
Type: Data Validation
Finding ID: TOB-WXTZ-012
Target: `partials/wxtz/tzip7/transfer/transfer.religo`

Description
As described in TOB-WXTZ-011, the token contract does not check approvals when the `from` address is the contract itself. As a result, anyone is allowed to claim those tokens.

Exploit Scenario
A feature of the system relies on the token contract holding its own tokens. A user is able to claim all of those tokens, completely undermining that functionality.

Recommendation
Short term, disallow users to claim tokens that the token contract holds. This will eliminate a vulnerability whereby users can claim the tokens owned by that contract.

Long term, avoid introducing special cases like the one described here. Fewer special cases will result in code that is easier to reason about.

13. Token holders can double their token balances

Severity: High
Difficulty: Low
Type: Data Validation
Finding ID: TOB-WXTZ-013
Target: `partials/wxtz/tzip7/transfer/transfer.religo`

Description

A user with a non-zero token balance can call `Transfer` with their own address as the first two parameters, and their balance as the third. Their `newTokens` BigMap will have their balance set to 0 before L60. However, since L60 references the old ledger, it will be read as their initial balance, and doubled on L65.

```
        let receiverBalance = Big_map.find_opt(transferParameter.to_, tokenStorage.ledger);
        let receiverBalance = switch (receiverBalance) {
        | Some(value) => value
        | None => defaultBalance
        };
        let newReceiverBalance = receiverBalance + transferParameter.value;
        let newTokens = Big_map.update(
               transferParameter.to_,
               Some(newReceiverBalance),
               newTokens
        );
        // save new balances and allowances in token ledger and approvals
        let newStorage = {
                      ...tokenStorage,
                      ledger: newTokens,
                      approvals: newAllowances
        };
        // no operations are returned, only the updated token storage
        (emptyListOfOperations, newStorage);
```

Figure 13.1: contracts/partials/wxtz/tzip7/transfer/transfer.religo#L60-L78.

Note: This vulnerability is analogous to the hack on the defi platform bZx in September, 2020.

Exploit Scenario

A user is able to double their token balances. This violates major invariants of the system.

Recommendation

Short term, replace `tokenStorage.ledger` on L60 for `newTokens`. This will eliminate a critical vulnerability that could be used to completely undermine the system.

Long term, make sure the testing strategy includes edge cases such as two addresses representing the same account, and make sure to check that all invariants of the system are satisfied after every message. This will help ensure that similar bugs are not introduced into the system.

A. Vulnerability Classifications

Vulnerability Classes	
Class	**Description**
Access Controls	Related to authorization of users and assessment of rights
Auditing and Logging	Related to auditing of actions or logging of problems
Authentication	Related to the identification of users
Configuration	Related to security configurations of servers, devices, or software
Cryptography	Related to protecting the privacy or integrity of data
Data Exposure	Related to unintended exposure of sensitive information
Data Validation	Related to improper reliance on the structure or values of data
Denial of Service	Related to causing system failure
Documentation	Related to documenting or recording use scenarios
Error Reporting	Related to the reporting of error conditions in a secure fashion
Patching	Related to keeping software up to date
Session Management	Related to the identification of authenticated users
Timing	Related to race conditions, locking, or order of operations
Undefined Behavior	Related to undefined behavior triggered by the program
Standards	Related to complying with industry standards and best practices

Severity Categories	
Severity	**Description**
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth
Undetermined	The extent of the risk was not determined during this engagement

Low	The risk is relatively small or is not a risk the customer has indicated is important
Medium	Individual user information is at risk, exploitation would be bad for client’s reputation, moderate financial impact, possible legal implications for client
High	Large numbers of users, very bad for client’s reputation, or serious legal or financial implications

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploit was not determined during this engagement
Low	Commonly exploited, public tools exist or can be scripted that exploit this flaw
Medium	Attackers must write an exploit, or need an in-depth knowledge of a complex system
High	The attacker must have privileged insider access to the system, may need to know extremely complex technical details, or must discover other weaknesses in order to exploit this issue

B. Code Maturity Classifications

Code Maturity Classes	
Category Name	**Description**
Access Controls	Related to the authentication and authorization of components.
Arithmetic	Related to the proper use of mathematical operations and semantics.
Assembly Use	Related to the use of inline assembly.
Centralization	Related to the existence of a single point of failure.
Upgradeability	Related to contract upgradeability.
Function Composition	Related to separation of the logic into functions with clear purpose.
Front-Running	Related to resilience against front-running.
Key Management	Related to the existence of proper procedures for key generation, distribution, and access.
Monitoring	Related to use of events and monitoring procedures.
Specification	Related to the expected codebase documentation.
Testing & Verification	Related to the use of testing techniques (unit tests, fuzzing, symbolic execution, etc.).

Rating Criteria	
Rating	**Description**
Strong	The component was reviewed and no concerns were found.
Satisfactory	The component had only minor issues.
Moderate	The component had some issues.
Weak	The component led to multiple issues; more issues might be present.
Missing	The component was missing.

Not Applicable	The component is not applicable.
Not Considered	The component was not reviewed.
Further Investigation Required	The component requires further investigation.

C. Non–Security-Related Findings

This appendix contains findings that do not have immediate or obvious security implications.

- **Whitespace problems:**
    - **The code is indented with both spaces and tabs.** Depending upon an editor's tab-width setting, the code could display incorrectly, making it difficult to read. We recommend choosing either spaces or tabs, and indenting all files consistently.
    - **Many files have lines that end in whitespace.** Some editors try to remove such whitespace. This causes `git` to believe the files should be committed. We recommend removing such whitespace.
    - **Many files do not end in a newline.** This can make them difficult to parse, e.g., with shell scripts. We recommend that every file end in a newline.

- **In `runArbitraryValueLambda.religo`, the variable `lambdaExtras` is unused:**

```
// TODO: extract lambdaExtras for
let lambdaExtras: lambdaExtras = {
    selfAddress: Tezos.self_address
};
```

  Consider either using the variable in the subsequent call to `arbitraryValueLambda`, or adjusting the comment to indicate that the variable is intentionally unused.

- **The `transfer.religo` entry point has confusing terminology.** `sender` is initially used to refer to the transaction sender, and later in the code, to the `from` account. We recommend calling the transaction sender "`sender`," and the `from` account "benefactor," "token owner," or "`from` account."

- **`lambdaNames` should be constant variables.** There are multiple places (e.g., here) where magic strings are used for `lambdaNames`. Extracting these to a constant would make the code more robust.

- **The core's `default` lambda has a misleading comment.** The comment is at odds with the implementation of the function, hence remedying or removing it would improve readability.

```
/**
 * Lambda to handle the Default entrypoint call, used to send XTZ / delegation rewards
 */
```

- **In two locations, the term "vault" should be "oven."** The two locations are in `test/unit/core/lambdas/{createOven.js, onOvenDepositReceived.js}`, and are depicted below.

```
it('should be possible to delegate to the vault owner him/herself', async () => {

            /**
             * Manager will act as a mock vault, without %default
             */
```

- **Unnecessary `include` in `onOvenSetDelegateInit.religo`.** The following `include` is not needed:

```
#include "../arbitrary/composeMintOperation/composeMintOperationInit.religo"
```

- **Problems with `test/integration/core.js`.** The following "`require`" statement is needed near the top of the file:

```
const { readFileSync } = require('fs');
```

  Also, the declaration of helpers should include `coreAddress`:

```
           helpers = { tzip7Helpers, coreHelpers, coreAddress };
```

D. Fix Log

From December 22 to 23, 2020, Trail of Bits reviewed StakerDAO's fixes for the issues identified in this report. The fixes were spread across the following seven pull requests. At the time of this writing, all except the last one (44) have been merged.

- `https://github.com/StakerDAO/wrapped-xtz/pull/15`
- `https://github.com/StakerDAO/wrapped-xtz/pull/16`
- `https://github.com/StakerDAO/wrapped-xtz/pull/17`
- `https://github.com/StakerDAO/wrapped-xtz/pull/19`
- `https://github.com/StakerDAO/wrapped-xtz/pull/20`
- `https://github.com/StakerDAO/wrapped-xtz/pull/21`
- `https://github.com/StakerDAO/wrapped-xtz/pull/44`

StakerDAO fixed or partially fixed 9 of the 13 findings identified in this report. We reviewed the fixes to ensure they would be effective. StakerDAO chose to not fix, or has not yet fixed, the remaining four findings.

ID	Title	Severity	Status
01	Code relies on vulnerable NPM packages	Medium	Partially Fixed
02	Ganache CLI is configured to listen on all network interfaces	Low	Partially Fixed
03	Insufficient tests	Informational	Fixed
04	Insufficient documentation	Informational	Not Fixed
05	Calls to `runArbitraryValueLambda` assume storage does not change	Low	Not Fixed
06	Core lacks entry points for tzip-7 admin-only operations	Informational	Fixed
07	Type confusion in `updateLambdas`	Informational	Fixed
08	Insufficient validation of newly created oven owners	Informational	Fixed
09	A compromised core administrator could steal wXTZ or deposited XTZ	Informational	Not Fixed
10	A compromised tzip-7 administrator could steal wXTZ	Informational	Not Fixed

11	wXTZ deviates from the tzip-7 specification	Informational	Fixed
12	Possible race condition when wXTZ owns tokens	High	Fixed
13	Token holders can double their token balances	High	Fixed

Detailed Fix Log

Finding 1: Code relies on vulnerable NPM packages
Partially Fixed. At the time of this writing, all dependencies have been updated to their latest versions. An update to Table 1.1 appears below. While the overall number of advisories affecting the codebase decreased (from 12 to 9), one new advisory was added (1589).

NPM Advisory	Description	Dependency
566	Prototype Pollution	`hoek`
598	Memory Exposure	`tunnel-agent`
877	Insecure Credential Storage	`web3`
1179	Prototype Pollution	`minimist`
1464	Insufficient Entropy	`cryptiles`
1523	Prototype Pollution	`lodash`
1547	Signature Malleability	`elliptic`
1556	Denial of Service	`node-fetch`
1589	Prototype Pollution	`ini`

NPM advisories affecting wXTZ's dependencies following PR #17.

Note that the advisories affect more than just the immediate dependency `ganache-cli`. They also also affect `commitizen` (1589), `cz-conventional-changelog` (1589), and `truffle` (877, 1179, 1556).

Finding 2: Ganache CLI is configured to listen on all network interfaces
Partially Fixed. The `start_ganache-cli.sh` script no longer passes the "`--host 0.0.0.0`" option by default. Now, the `package.json` script that calls `start_ganache-cli.sh` determines whether this option is passed. In particular, `env:start` passes the option, but `sandbox:start` does not.

Finding 3: Insufficient tests
Fixed. As of PR #15, there are 29 error types. All are tested for except `errorNoContract`, which is returned only by a contract used for testing (`getViews`).

Finding 4: Insufficient documentation
Not Fixed.

Finding 5: Calls to `runArbitraryValueLambda` assume storage does not change

Not Fixed.

Finding 6: Core lacks entry points for tzip-7 admin-only operations
Fixed. The core now has entry points for the tzip-7 contract's `setAdministrator`, `setPause`, and `setPauseGuardian` operations. Each entry point checks that the sender is the core administrator, and has a test to verify that the check is effective. We manually reviewed the tests and verified that they pass.

Finding 7: Type confusion in `updateLambdas`
Fixed. `Map.update` was changed to `Big_map.update` as we recommended.

Finding 8: Insufficient validation of newly created oven owners
Fixed. The recommended checks were implemented. Tests were added to verify that the checks are effective. We manually reviewed the tests and verified that they pass.

Finding 9: A compromised core administrator could steal wXTZ or deposited XTZ
Not Fixed.

Finding 10: A compromised tzip-7 administrator could steal wXTZ
Not Fixed.

Finding 11: wXTZ deviates from the tzip-7 specification
Fixed. Approval checks are now performed within a `canTransfer` function, which does not have this edge case.

Finding 12: Possible race condition when wXTZ owns tokens
Fixed. The removal of the just mentioned edge case eliminated this vulnerability.

Finding 13: Token holders can double their token balances
Fixed. The code now features an `updateLedgerByTransfer` function that decreases the sender's balance and increases the receiver's balance. The latter operation uses the ledger produced by the former operation. Thus, the vulnerability is no longer present.

Detailed Issue Discussion

Responses from StakerDAO for each issue are included as quotes below.

Finding 1: Code relies on vulnerable NPM packages

> *Partially fixed in PR #17. Reason for only a partial fix for this finding is the dependency on `ganache-cli`, which even after updating still carries over ~500 vulnerabilities.*

Finding 2: Ganache CLI is configured to listen on all network interfaces

> *Partially fixed in PR #44. This finding won't be addressed further due to intricacies surrounding docker networking on development machines.*

Finding 3: Insufficient tests

> *Fixed in PR #15. Every error of wXTZ Core is now covered by tests at least once + other new tests.*

Finding 4: Insufficient documentation

> *In progress.*

Finding 5: Calls to `runArbitraryValueLambda` assume storage does not change

> *Won't fix. We've decided not to apply suggestions from this finding since they do not pose immediate concern. One extra issue discovered was that the exact same assumption made about storage, applies to the operations returned from arbitrary lambdas as well.*

Finding 6: Core lacks entry points for tzip-7 admin-only operations

> *Fixed in PR #21. The following entry points (lambdas) have been added to wXTZ Core:*
>
> - *tzip-7/setAdministrator*
> - *tzip-7/setPause*
> - *tzip-7/setPauseGuardian*
>
> *Additionally, we've added an extra lambda that the finding did not point out was missing, but was related:*
>
> - *setArbitraryValue*
>
> *This entry point is useful for updating the wXTZ Token contract address or the wXTZ Core admin itself.*

Finding 7: Type confusion in `updateLambdas`

> *Fixed in PR #19.*

Finding 8: Insufficient validation of newly created oven owners

Fixed in PR #20. New error code `errorInvalidOvenOwner = "16"` *was introduced to cover all invalid wXTZ Oven owner cases.*

Finding 9: A compromised core administrator could steal wXTZ or deposited XTZ

Won't fix.

Finding 10: A compromised tzip-7 administrator could steal wXTZ

Won't fix.

Finding 11: wXTZ deviates from the tzip-7 specification
Finding 12: Possible race condition when wXTZ owns tokens
Finding 13: Token holders can double their token balances

Fixed in PR #16. Thorough refactor was applied to address the findings above.