Fei Protocol

Date	January 2021
Lead Auditor	Valentin Wüstholz
Co-auditors	Alexander Wade, Sergii Kravchenko
Download	PDF

1 Executive Summary

This report presents the results of our engagement with **Fei Protocol** to review some of the smart contracts in their stable coin implementation.

The review has been conducted over two weeks, from **Jan 25, 2021** to **Jan 29, 2021**, by **Valentin Wüstholz**, **Alexander Wade**, and **Sergii Kravchenko**. Additionally, an infrastructure security assessment has been conducted over two weeks from **Feb 8, 2021** to **Feb 19, 2021**, by **Dominik Muhs**. A total of 25 person-days were spent.

2 Scope

Our review focused on the commit hash `ff892c5d`. The list of files in scope and the priorities of the audit are defined by the client and can be found here.

The infrastructure assessment focused on the following assets:

- The `fei.money` domain, specifically `ropsten-app.fei.finance`
- The `icaruscryptolab` AWS organization
- The `fei-protocol/fei-app` at commit hash `eee5d29`

3 Findings

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.

- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

3.1 `GenesisGroup.commit` overwrites previously-committed values Critical

Resolution

This was addressed in fei-protocol/fei-protocol-core#16.

Description

`commit` allows anyone to commit purchased FGEN to a swap that will occur once the genesis group is launched. This commitment may be performed on behalf of other users, as long as the calling account has sufficient allowance:

code/contracts/genesis/GenesisGroup.sol:L87-L94

```
function commit(address from, address to, uint amount) external override onlyGenesisPeriod {
 burnFrom(from, amount);

 committedFGEN[to] = amount;
 totalCommittedFGEN += amount;

 emit Commit(from, to, amount);
}
```

The `amount` stored in the recipient’ s `committedFGEN` balance overwrites any previously-committed value. Additionally, this also allows anyone to commit an amount of “0” to any account, deleting their commitment entirely.

Recommendation

Ensure the committed amount is added to the existing commitment.

3.2 Purchasing and committing still possible after launch Critical

Resolution

This was addressed in fei-protocol/fei-protocol-core#11.

Description

Even after `GenesisGroup.launch` has successfully been executed, it is still possible to invoke `GenesisGroup.purchase` and `GenesisGroup.commit`.

Recommendation

Consider adding validation in `GenesisGroup.purchase` and `GenesisGroup.commit` to make sure that these functions cannot be called after the launch.

3.3 `UniswapIncentive` overflow on pre-transfer hooks Major

Resolution

This was addressed in fei-protocol/fei-protocol-core#15.

Description

Before a token transfer is performed, `Fei` performs some combination of mint/burn operations via `UniswapIncentive.incentivize` :

code/contracts/token/UniswapIncentive.sol:L49-L65

```
function incentivize(
 address sender,
 address receiver,
 address operator,
 uint amountIn
) external override onlyFei {
    updateOracle();

 if (isPair(sender)) {
  incentivizeBuy(receiver, amountIn);
 }

 if (isPair(receiver)) {
        require(isSellAllowlisted(sender) || isSellAllowlisted(operator), "UniswapIncentive: Blocked F
  incentivizeSell(sender, amountIn);
 }
}
```

Both `incentivizeBuy` and `incentivizeSell` calculate buy/sell incentives using overflow-prone math, then mint / burn from the target according to the results. This may have unintended consequences, like allowing a caller to mint tokens before transferring them, or burn tokens from their recipient.

Examples

`incentivizeBuy` calls `getBuyIncentive` to calculate the final minted value:

code/contracts/token/UniswapIncentive.sol:L173-L186

```
function incentivizeBuy(address target, uint amountIn) internal ifMinterSelf {
 if (isExemptAddress(target)) {
  return;
 }

    (uint incentive, uint32 weight,
    Decimal.D256 memory initialDeviation,
    Decimal.D256 memory finalDeviation) = getBuyIncentive(amountIn);

    updateTimeWeight(initialDeviation, finalDeviation, weight);
    if (incentive != 0) {
        fei().mint(target, incentive);
    }
}
```

`getBuyIncentive` calculates price deviations after casting `amount` to an `int256`, which may overflow:

code/contracts/token/UniswapIncentive.sol:L128-L134

```
function getBuyIncentive(uint amount) public view override returns(
    uint incentive,
    uint32 weight,
    Decimal.D256 memory initialDeviation,
    Decimal.D256 memory finalDeviation
) {
    (initialDeviation, finalDeviation) = getPriceDeviations(-1 * int256(amount));
```

Recommendation

Ensure casts in `getBuyIncentive` and `getSellPenalty` do not overflow.

3.4 `BondingCurve` allows users to acquire FEI before launch

Medium

Resolution

This was addressed in fei-protocol/fei-protocol-core#59

Description

`BondingCurve.allocate` allocates the protocol’s held PCV, then calls `_incentivize`, which rewards the caller with FEI if a certain amount of time has passed:

code-update/contracts/bondingcurve/BondingCurve.sol:L180-L186

```
/// @notice if window has passed, reward caller and reset window
function _incentivize() internal virtual {
    if (isTimeEnded()) {
        _initTimed(); // reset window
        fei().mint(msg.sender, incentiveAmount);
    }
}
```

`allocate` can be called before genesis launch, as long as the contract holds some nonzero PCV. By force-sending the contract 1 wei, anyone can bypass the majority of checks and actions in `allocate`, and mint themselves FEI each time the timer expires.

Recommendation

Prevent `allocate` from being called before genesis launch.

3.5 `Timed.isTimeEnded` returns `true` if the timer has not been initialized Medium

Resolution
This was addressed in fei-protocol/fei-protocol-core#62

Description

`Timed` initialization is a 2-step process:

- `Timed.duration` is set in the constructor: https://github.com/ConsenSys/fei-protocol-audit-2021-01/blob/d31114d834e62b4f3d4fa7b1c0b0c70fbff623a4/code-update/contracts/utils/Timed.sol#L15-L20
- `Timed.startTime` is set when the method `_initTimed` is called: https://github.com/ConsenSys/fei-protocol-audit-2021-01/blob/d31114d834e62b4f3d4fa7b1c0b0c70fbff623a4/code-update/contracts/utils/Timed.sol#L43-L46

Before this second method is called, `isTimeEnded()` calculates remaining time using a `startTime` of 0, resulting in the method returning `true` for most values, even though the timer has not technically been started.

Recommendation

If `Timed` has not been initialized, `isTimeEnded()` should return `false`, or `revert`

3.6 Overflow/underflow protection Medium

Resolution

This was partially addressed in fei-protocol/fei-protocol-core#17 by using `SafeMath` for the specific example given in the description.

Description

Having overflow/underflow vulnerabilities is very common for smart contracts. It is usually mitigated by using `SafeMath` or using solidity version ^0.8 (after solidity 0.8 arithmetical operations already have default overflow/underflow protection).

In this code, many arithmetical operations are used without the ‘safe’ version. The reasoning behind it is that all the values are derived from the actual ETH values, so they can’t overflow.

On the other hand, some operations can’t be checked for overflow/underflow without going much deeper into the codebase that is out of scope:

code/contracts/genesis/GenesisGroup.sol:L131

```
uint totalGenesisTribe = tribeBalance() - totalCommittedTribe;
```

Recommendation

In our opinion, it is still safer to have these operations in a safe mode. So we recommend using `SafeMath` or solidity version ^0.8 compiler.

3.7 Unchecked return value for IWETH.transfer call Medium

Resolution

This was addressed in fei-protocol/fei-protocol-core#12.

Description

In `EthUniswapPCVController`, there is a call to `IWETH.transfer` that does not check the return value:

code/contracts/pcv/EthUniswapPCVController.sol:L122

```
weth.transfer(address(pair), amount);
```

It is usually good to add a require-statement that checks the return value or to use something like `safeTransfer`; unless one is sure the given token reverts in case of a failure.

Recommendation

Consider adding a require-statement or using `safeTransfer`.

3.8 `GenesisGroup.emergencyExit` remains functional after launch Medium

Resolution

This was partially addressed in fei-protocol/fei-protocol-core#14 and fei-protocol/fei-protocol-core#13 by addressing the last two recommendations.

Description

`emergencyExit` is intended as an escape mechanism for users in the event the genesis `launch` method fails or is frozen. `emergencyExit` becomes callable 3 days after `launch` is callable. These two methods are intended to be mutually-exclusive, but are not: either method remains callable after a successful call to the other.

This may result in accounting edge cases. In particular, `emergencyExit` fails to decrease `totalCommittedFGEN` by the exiting user’s commitment:

code/contracts/genesis/GenesisGroup.sol:L185-L188

```
burnFrom(from, amountFGEN);
committedFGEN[from] = 0;

payable(to).transfer(total);
```

As a result, calling launch after a user performs an exit will incorrectly calculate the amount of FEI to swap:

code/contracts/genesis/GenesisGroup.sol:L165-L168

```
uint amountFei = feiBalance() * totalCommittedFGEN / (totalSupply() + totalCommittedFGEN);
if (amountFei != 0) {
 totalCommittedTribe = ido.swapFei(amountFei);
}
```

Recommendation

- Ensure `launch` cannot be called if `emergencyExit` has been called
- Ensure `emergencyExit` cannot be called if `launch` has been called
- In `emergencyExit`, reduce `totalCommittedFGEN` by the exiting user’s committed amount

3.9 Unchecked return value for transferFrom calls Medium

Resolution

This was addressed in fei-protocol/fei-protocol-core#12.

Description

There are two `transferFrom` calls that do not check the return value (some tokens signal failure by returning false):

code/contracts/pool/Pool.sol:L121

```
stakedToken.transferFrom(from, address(this), amount);
```

code/contracts/genesis/IDO.sol:L58

```
fei().transferFrom(msg.sender, address(pair), amountFei);
```

It is usually good to add a require-statement that checks the return value or to use something like `safeTransferFrom`; unless one is sure the given token reverts in case of a failure.

Recommendation

Consider adding a require-statement or using `safeTransferFrom`.

3.10 `GovernorAlpha` proposals may be canceled by the proposer, even after they have been accepted and queued Minor

Resolution

This was addressed in fei-protocol/fei-protocol-core#61

Description

`GovernorAlpha` allows proposals to be canceled via `cancel`. To cancel a proposal, two conditions must be met by the proposer:

- The proposal should not already have been executed: https://github.com/ConsenSys/fei-protocol-audit-2021-01/blob/d31114d834e62b4f3d4fa7b1c0b0c70fbff623a4/code-update/contracts/dao/GovernorAlpha.sol#L206-L208
- The proposer must have under `proposalThreshold()` TRIBE balance: https://github.com/ConsenSys/fei-protocol-audit-2021-01/blob/d31114d834e62b4f3d4fa7b1c0b0c70fbff623a4/code-update/contracts/dao/GovernorAlpha.sol#L210-L211

The latter condition is completely under the control of the proposer, meaning that a proposer may cancel proposals in any of these states: `Pending`, `Active`, `Canceled`, `Defeated`, `Succeeded`, `Queued`, `Expired`.

Recommendation

Prevent proposals from being canceled unless they are in the `Pending` or `Active` states.

3.11 `Pool`: claiming to the pool itself causes accounting issues Minor

Resolution

This was addressed in fei-protocol/fei-protocol-core#57

Description

In `Pool.sol`, `claim(address from, address to)` is used to claim staking rewards and send them to a destination address `to`:

code-update/contracts/pool/Pool.sol:L229-L238

```
function _claim(address from, address to) internal returns (uint256) {
    (uint256 amountReward, uint256 amountPool) = redeemableReward(from);
    require(amountPool != 0, "Pool: User has no redeemable pool tokens");

    _burnFrom(from, amountPool);
    _incrementClaimed(amountReward);

    rewardToken.transfer(to, amountReward);
    return amountReward;
}
```

If the destination address `to` is the pool itself, the pool will burn tokens and increment the amount of tokens claimed, then transfer the reward tokens to itself.

Recommendation

Prevent claims from specifying the pool as a destination.

3.12 Assertions that can fail Minor

Description

In `UniswapSingleEthRouter` there are two assert-statements that may fail:

code/contracts/router/UniswapSingleEthRouter.sol:L21

```
assert(msg.sender == address(WETH)); // only accept ETH via fallback from the WETH contract
```

code/contracts/router/UniswapSingleEthRouter.sol:L48

```
assert(IWETH(WETH).transfer(address(PAIR), amountIn));
```

Since they do some sort of input validation it might be good to replace them with require-statements. I would only use asserts for checks that should never fail and failure would constitute a bug in the code.

Recommendation

Consider replacing the assert-statements with require-statements. An additional benefit is that this will not result in consuming all the gas in case of a violation.

3.13 Simplify API of GenesisGroup.purchase Minor

Description

The API of `GenesisGroup.purchase` could be simplified by not including the `value` parameter that is required to be equivalent to `msg.value`:

code/contracts/genesis/GenesisGroup.sol:L79

```
require(msg.value == value, "GenesisGroup: value mismatch");
```

Using `msg.value` might make the API more explicit and avoid requiring `msg.value == value`. It can also save some gas due to fewer inputs and fewer checks.

Recommendation

Consider dropping the `value` parameter and changing the code to use `msg.value` instead.

4 Infrastructure Security Assessment

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

4.1 Clickjacking and Missing Content Security Policy Major Fixed

Resolution

After multiple iterations, the following Content Security Policy has been put into effect:

```
default-src 'self';
script-src 'self' 'unsafe-inline' https://www.googletagmanager.com;
object-src 'none';
style-src 'self' 'unsafe-inline';
img-src 'self';
media-src 'none';
frame-src 'none';
font-src 'self';
connect-src 'self'
  https://api.amplitude.com
  https://eth-ropsten.alchemyapi.io
  https://eth-mainnet.alchemyapi.io
  https://api.thegraph.com;
frame-ancestors 'none'
```

The CSP is transmitted through the following headers:

- Content-Security-Policy
- X-Content-Security-Policy
- X-WebKit-CSP

as well as through corresponding `meta` HTML tags. Additionally, the following frame-busting JavaScript code has been added to prevent Clickjacking attacks in the unlikely event that existing CSP measures fail or are bypassed:

```
<script>
 if (self == top) {
   document.documentElement.style.display = 'block ';
 } else {
   top.location = self.location;
 }
</script>
```

Description

A content security policy (CSP) provides an added layer of protection against cross-site scripting (XSS), clickjacking, and other client-side attacks that rely on executing malicious content in the context of the website.

Specifically, the lack of a content security policy allows an adversary to perform a clickjacking attack by including the target URL (such as `app.fei.money`) in an `iframe` element on their site. The attacker then uses one or more transparent layers on top of the embedded site to trick a user into performing a click action on a different element.

This technique can be used to spawn malicious Metamask dialogues, tricking users into thinking that they are signing a legitimate transaction.

Affected Assets

All S3-hosted web sites.

Recommendation

It is recommended to add content security policy headers to the served responses to prevent browsers from embedding Fei-owned sites into malicious parent sites. Furthermore, CSP can be used to limit the permissions of JavaScript and CSS on the page, which can be used to further harden the deployment against a potential compromise of script dependencies.

It should be noted that security headers should not only be served from Cloudfront but any public-facing endpoint. Otherwise, it will be trivial for an attacker to circumvent the security headers added by Cloudfront, e.g. by embedding the `index.html` file directly from the public-facing S3 bucket URL.

Besides CSP headers, clickjacking can also be mitigated by directly including frame-busting JavaScript code into the served page.

4.2 S3 Buckets Cleartext Communication Medium Fixed

Resolution

Direct access to S3 buckets through `s3.amazonaws.com` is now rejected, while unencrypted HTTP traffic to the previously affected assets now consistently redirects to the HTTPS equivalents.

Description

The system’s S3 buckets are configured to allow unencrypted traffic:

```
$ curl -v http://fei.money.s3.amazonaws.com/index.html
*   Trying 52.219.112.162:80...
* TCP_NODELAY set
* Connected to fei.money.s3.amazonaws.com (52.219.112.162) port 80 (https://github.com/ConsenSys/fei-protocol-audit-20
> GET /index.html HTTP/1.1
> Host: fei.money.s3.amazonaws.com
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< x-amz-id-2: 0QtzqEhGn7gHUjjiAxpniOMXKQ1O1ouT6Tp8iQG2EfvlKbg0ZgEbDdkQrJrJL2OyJF1VyZkPjjU=
< x-amz-request-id: D6250FE8F76E84F0
< Date: Tue, 09 Feb 2021 13:07:54 GMT
< Last-Modified: Mon, 11 Jan 2021 20:38:09 GMT
< ETag: "ec826fa83693f3db3a989fcbeb5adef1"
< Accept-Ranges: bytes
< Content-Type: text/html
< Content-Length: 3675
< Server: AmazonS3
<
< ...
```

Affected Assets

- `arn:aws:s3:::ropsten-app.fei.money/*`
- `arn:aws:s3:::www.fei.money/*`
- `arn:aws:s3:::feiprotocol.com/*`
- `arn:aws:s3:::www.app.fei.money/*`
- `arn:aws:s3:::www.ropsten-app.fei.money/*`
- `arn:aws:s3:::app.fei.money/*`
- `arn:aws:s3:::fei.money/*`

Recommendation

It is recommended to enforce encryption of data in transit using TLS certificates. To accomplish this, the `aws:SecureTransport` can be set in the S3 bucket’ s policies.

4.3 Missing Log Aggregation Medium Fixed

Resolution

CloudFrond and CloudTrail have been enabled. These components send endpoint-related and organizational log messages into S3 buckets where they can be queried using AWS Athena. The security review process section of this report contains sample queries for Athena.

Description

There is no centralized system that gathers operational events of AWS stack components. This includes S3 server access logs, configuration changes, as well as Cloudfront-related logging.

Recommendation

It is recommended to enable CloudTrail for internal log aggregation as it integrates seamlessly with S3, Cloudfront, and IAM. Furthermore, regular reviews should be set up where system activity is checked to detect suspicious activity as soon as possible.

4.4 Enforce Strict Transport Security Medium Fixed

Resolution

All domains in scope now ship with the following header:

`strict-transport-security: max-age=63072000; includeSubdomains`

Description

The HTTP `Strict-Transport-Security` response header (often abbreviated as HSTS) lets a web site tell browsers that it should only be accessed using HTTPS, instead of using HTTP. This prevents attackers from stripping TLS certificates from connections and removing encryption.

Recommendation

It is recommended to deliver all responses with the `Strict-Transport-Security` header. In an S3-Cloudfront setup, this can be achieved using Lambda@Edge lambda functions.

4.5 Server Information Leak Minor

Description

Responses from the `fei.money` domain and related assets leak server information in their response headers. This information can be used by an adversary to prepare more sophisticated attacks tailored to the deployed infrastructure.

Note: At the time of reporting, this issue was deemed not possible to fix due to technical limitations on AWS-hosted static sites using S3 and CloudFront.

Examples

```
$ curl -I https://fei.money/static/media/
HTTP/2 404
x-amz-error-code: NoSuchKey
x-amz-error-message: The specified key does not exist.
x-amz-error-detail-key: static/media/index.html
date: Tue, 09 Feb 2021 13:49:34 GMT
server: AmazonS3
x-cache: Error from cloudfront
via: 1.1 fa133af2508a341e1ff6bfff526ba095.cloudfront.net (CloudFront)
x-amz-cf-pop: TXL52-C1
x-amz-cf-id: x0eNuDCrilaFgOT3fz4g1CpdRIfFCxBta7Pif4wexsXpN3weVLv7uw==
```

Recommendation

It is recommended to remove any headers that hint at server technologies and are not directly required by the frontend.

4.6 Missing Route53 Domain Lock Minor Fixed

Resolution

A transfer lock on both the `feiprotocol.com` and `fei.money` domains has been requested.

Description

Domain registrars often give customers the option to lock a domain. This prevents unauthorized parties from transferring it to another registrar, either through malicious interaction with the registrar itself, or compromised domain owner credentials. No domain currently has a lock enabled.

Affected Assets

- fei.money
- feiprotocol.com

Recommendation

It is recommended to set a lock for the affected domains, assuming that the registrar allows domain locks:

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the navigation pane, choose Registered Domains.

3. Choose the name of the domain that you want to update.
4. Choose Enable (to lock the domain) or Disable (to unlock the domain).
5. Choose Save.

4.7 Weak IAM Password Policy Minor Fixed

Resolution

This has been fixed by the client with the following notes:

- Enforced 14 character password length
- Enabled 90 day password expiration
- Prevent password reuse
- Require one uppercase, one lowercase, one number, one non-alphanumeric character
- Require 2FA on all users via this doc and this post (Create new Force_MFA policy, attach it to the new Engineers group, and then assign all users (including Dominik) to this group
- Also requiring 2FA on command line access. Using `src/infra/aws-token.sh` for generating the credentials and putting them in `~/.aws/config`

Description

The password policy for IAM users currently does not enforce the use of strong passwords, multi-factor authentication, and regular password rotation.

Currently, only a minimum password length of 8 is enforced.

Recommendation

- Require a minimum password length of 14
- Set a password expiration policy of at most 90 days
- Disallow the reuse of passwords
- Enable mandatory multi-factor authentication with a virtual app

4.8 Review Access Key Expiration Minor Fixed

Resolution

This issue is considered resolved with the implementation of regular security review meetings.

Description

It is recommended to only create access keys when absolutely necessary. There should be no access keys given out to root users. Instead, temporary security credentials (IAM Roles) should be created.

Recommendation

It is recommended to read the Best practices for managing AWS access keys and incorporate the security practices where reasonable.

4.9 Dependency Security Minor Fixed

Resolution

This issue has been resolved by implementing Snyk for continuous dependency security scanning. This allows the developers to review potential risks of included packages and receiving automated pull requests with fixes if necessary.

Furthermore, a manual review of select dependencies has been conducted by the penetration tester without significant, actionable results. The following dependencies have been checked:

- bignumber
- numeral
- validator
- web-vitals

Description

The Yarn audit feature currently finds two low-severity dependency issues:

- Prototype pollution in `ini` - a dependency of `react-scripts`
- Insecure Credential Storage in `web3`

```
yarn audit v1.22.4
┌───────────────┬──────────────────────────────────────────────────────────────
│ low           │ Prototype Pollution                                          │
├───────────────┼──────────────────────────────────────────────────────────────
│ Package       │ ini                                                          │
├───────────────┼──────────────────────────────────────────────────────────────
│ Patched in    │ >1.3.6                                                       │
├───────────────┼──────────────────────────────────────────────────────────────
│ Dependency of │ react-scripts                                                │
├───────────────┼──────────────────────────────────────────────────────────────
│ Path          │ react-scripts > react-dev-utils > global-modules >           │
│               │ global-prefix > ini                                          │
├───────────────┼──────────────────────────────────────────────────────────────
│ More info     │ https://www.npmjs.com/advisories/1589                        │
└───────────────┴──────────────────────────────────────────────────────────────
┌───────────────┬──────────────────────────────────────────────────────────────
│ low           │ Insecure Credential Storage                                  │
├───────────────┼──────────────────────────────────────────────────────────────
│ Package       │ web3                                                         │
├───────────────┼──────────────────────────────────────────────────────────────
│ Patched in    │ No patch available                                           │
├───────────────┼──────────────────────────────────────────────────────────────
│ Dependency of │ web3                                                         │
├───────────────┼──────────────────────────────────────────────────────────────
│ Path          │ web3                                                         │
├───────────────┼──────────────────────────────────────────────────────────────
│ More info     │ https://www.npmjs.com/advisories/877                         │
└───────────────┴──────────────────────────────────────────────────────────────
2 vulnerabilities found - Packages audited: 1963
Severity: 2 Low
Done in 11.90s.
```

Recommendations

It is recommended to apply the `ini` patch, which is already available. For `web3`, it is recommended to monitor the repository's Github issue https://github.com/ConsenSys/fei-protocol-audit-2021-01/issues/2739 and upgrade as soon as a fix is available.

For additional dependency security, it is recommended to integrate a security monitoring service. Snyk has a free plan which allows unlimited tests on public repositories, and 200 tests per month for private ones. A bot will automatically add a pull request to bump vulnerable dependency versions.

It should be noted that the quality and reliability of such automated contributions are highly dependent on the quality of the test suite. It is recommended to build strict tests around core functionality and expected dependency behaviour to detect breaking changes as soon as possible.

5 Security Review Process

In an additional effort to achieve security-in-depth, it is recommended to implement a schedule or recurring security review meetings. The goal of these meetings is to complete a checklist to enforce security best-practices, as well as find anomalies in the system as soon as possible to commence mitigation and investigations.

This section outlines recommendations for the contents of such a checklist. It should be noted that security requirements are likely to change, and thus, this list should be treated as a working document as the project's infrastructure and attack surface change.

5.1 CloudTrail Anomalies

Event filter query template:

```
SELECT useridentity.username,
         sourceipaddress,
         eventtime,
         additionaleventdata
FROM cloudtrail_logs
WHERE {{ event filter }}
        AND eventtime >= '<yyyy-mm-dd>'
        AND eventtime < '<yyyy-mm-dd>';
```

Event Filter	Description
`eventname = 'ConsoleLogin'`	Sign-in activity
`eventname = 'AddUserToGroup'`	User added to group
`eventname = 'ChangePassword'`	User password change
`eventname LIKE '%AccessKey%'`	Key management events
`eventname LIKE '%MFADevice'`	MFA deactivation/deletion/resync
`eventname = 'StopLogging'`	Logging stopped
`eventname LIKE '%BucketPolicy%'`	Bucket policy activity
`eventname LIKE '%GroupPolicy%'`	Group policy activity
`eventname LIKE '%UserPolicy%'`	User policy activity
`eventname LIKE '%RolePolicy%'`	Role policy activity

Aggregate statistics about failed authentication and user authorization attempts can be gathered with the following query:

```
SELECT count (*) AS totalEvents,
         useridentity.arn,
         eventsource,
         eventname,
         errorCode,
         errorMessage
FROM cloudtrail_logs
WHERE (errorcode LIKE '%Denied%'
        OR errorcode LIKE '%Unauthorized%')
        AND eventtime >= '2021-02-17'
        AND eventtime < '2021-02-17'
GROUP BY  eventsource, eventname, errorCode, errorMessage, useridentity.arn
ORDER BY  eventsource, eventname
```

For investigative purposes or the goal of covering new infrastructure components, it might be necessary to add more event names to the review process. AWS does not provide a comprehensive list of event names per stack component. An external list of CloudTrail event names is available on the GorillaStack blog.

Note: In case of issues with Athena or ingestion into the database, CloudTrail allows users to view the unfiltered event history for a user-specified time range as well. Particularly notable is the ability to filter by resource types, of which the following are relevant to the Fei AWS infrastructure:

- AWS::S3::Bucket
- AWS::CloudTrail::Trail
- AWS::IAM::AccessKey
- AWS::IAM::MfaDevice
- AWS::IAM::Group
- AWS::IAM::Policy
- AWS::IAM::Role

5.2 Cloudfront Endpoint Anomalies

Top 10 endpoints hit in a given time frame:

```sql
SELECT uri,
         status,
         count(*) AS ct
FROM cloudfront_logs_fei_landing
WHERE date >= DATE('2021-02-01')
        AND date <= DATE('2021-02-28')
GROUP BY  uri, status
ORDER BY  ct DESC limit 10
```

This query can be filtered further by adding `AND status = 500` or a similar condition to find suspicious response codes.

5.3 Route53 CNAME Review

Subdomain takeover vulnerabilities occur when a subdomain is pointing to a service, e.g. a previously deleted CloudFront endpoint or S3 bucket. This allows an attacker to set up a page on the service that was being used and point their page to that subdomain. Especially with wildcard certificates on the system, e.g. `*.fei.money`, this can lead to an exploitation of user trust and enables attacks that can result in reputational and financial loss.

It is recommended that DNS records in Route53 are reviewed regularly and removed as soon as the underlying resource is decommissioned.

5.4 External Monitoring and Notifications

Beyond manual checks, it is recommended that a service such as Assertible is used. This will allow the development team to detect unavailable endpoints and enforce regularly-checked assertions, such as proper return codes or page content. Furthermore, such a service should integrate other means of communication such as Slack notifications, SMS messages, or arbitrary webhook calls to notify an on-duty developer as quickly as possible.

Appendix 1 - Disclosure