Source Code Review of Selected Options Vault Contracts for Thetanuts.Finance

Final Report and Management Summary

2022-02-16

PUBLIC

X41 D-SEC GmbH
Dennewartstr. 25-27
D-52068 Aachen
Amtsgericht Aachen: HRB19989

https://x41-dsec.de/
info@x41-dsec.de

Revision	*Date*	*Change*	*Author(s)*
1	2021-11-30	Draft Report	R. P. Weinmann, M. Vervier
2	2021-11-30	Final Report	R. P. Weinmann, M. Vervier
3	2021-12-14	Public Report Including Client Response	M. Vervier
4	2022-02-16	Update Auditors	M. Vervier

Contents

Dashboard

Target	
Customer	Thetanuts.Finance
Name	CoveredCallVaultV0, SynMiningVaultV1
Type	Source Code Review
Version	c9419488c5508f67832f4efa5556cc1acbb17a6b

Engagement	
Type	Source Code Review
Consultants	Ralf-Philipp Weinmann, Markus Vervier
Engagement Effort	10 person-days, 2021-11-15 to 2021-11-26

Total issues found	5

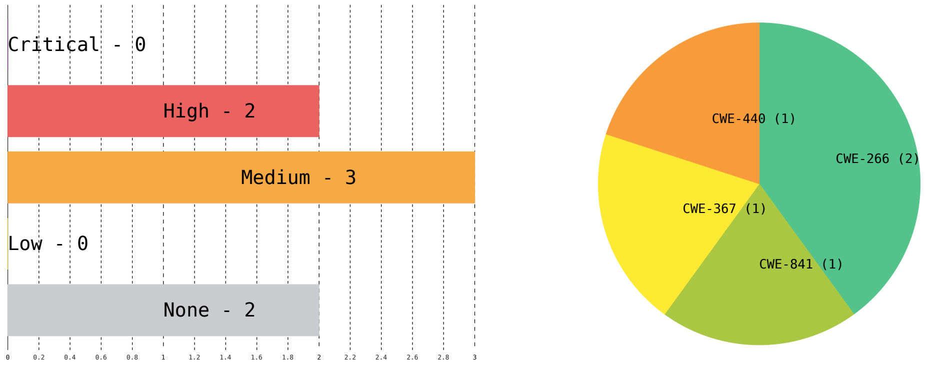

Figure 1: Issue Overview (l: Severity, r: CWE Distribution)

1 Executive Summary

In November 2021, X41 D-Sec GmbH performed a Source Code Review of selected smart contracts by Thetanuts.Finance to identify vulnerabilities and weaknesses in the design and implementation of modeling cash covered and physically settled options.

Thetanuts.Finance plans to provide with users vaults they can deposit cryptocurrency assets into; these vaults then generate yield for the users. The yield is achieved by selling options to market makers based on the assets in the vault. Any serious vulnerabilities in the contracts could result in a loss of funds to customers or the market maker(s) and have negative reputational effects on Thetanuts.Finance.

A total of five vulnerabilities were discovered during the test by X41. None were rated as critical, two were classified as high severity, three as medium, and none as low. Additionally, two issues without a direct security impact were identified.

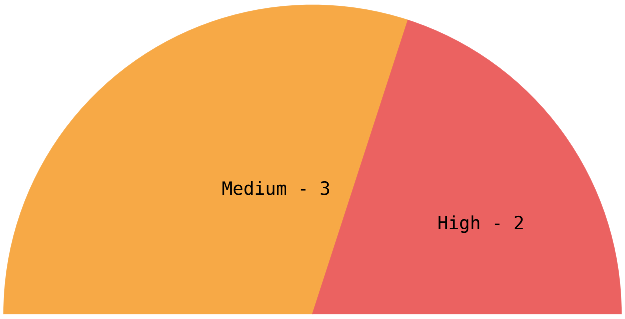

Figure 1.1: Issues and Severity

In a Source Code Review, testers receive all available information about the target. The test was performed by two experienced security experts between 2021-11-15 and 2021-11-26 in collaboration with Ralf-Philipp Weinmann of Comsecuris.

The majority of issues X41 identified were with the Synthetic Mining contract. This contract allowed deposit slippage as well as an economic attack in which an attacker can gain a share of the round premium without being exposed to the risk of the option. These issues have meanwhile been fixed by Thetanuts.Finance. Moreover in the Synthetic Mining contract, asset mispricing can happen if the base currency of the contract is not denoted in USD.

For both the Synthetic Mining and the Covered Call contract, X41 found that the privileges of the owner of the respective contract can be a security problem and the way the market maker is given access to the contract is too trusting.

In conclusion the audit uncovered relevant flaws that could have negatively impacted the secure operations of Thetanuts.Finance's contracts. Despite these flaws, the attack surface was small and dependencies on third party contracts were limited.

2 Introduction

X41 reviewed the yield-generating vault contracts of Thetanuts Finance which sell options to market makers to generate profit for their users.

These smart contracts are considered sensitive because they have direct control over cryptocurrency assets contributed by third parties.

Attackers could try to attack these contracts to steal all or a part of these cryptocurrency assets, cause assets to be locked in a smart contract or cause other financial or reputational losses to the parties interacting with the smart contracts.

X41 exclusively reviewed the technical implementation of the smart contracts developed by Thetanuts.Finance. The business model, operational security of Thetanuts.Finance, the security of third party services or of related smart-contracts were not in scope of this review.

2.1 Methodology

X41 first tried to gain a higher-level understanding of what the contracts are supposed to do and how those goals are achieved. We matched this understanding with the developers' understanding in two conference calls. Since there was no further written documentation to compare our understanding to, we have reproduced it in Section 4.

Subsequently we performed a manual audit of the source code, evaluating each statement of all function.

X41 adheres to established standards for source code reviewing and penetration testing. These are in particular the *CERT Secure Coding*[1] standards and the *Study - A Penetration Testing Model*[2] of the German Federal Office for Information Security.

[1] https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards

[2] https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/Penetration/penetration_pdf.pdf?__blob=publicationFile&v=1

2.2 Findings Overview

DESCRIPTION	SEVERITY	ID	REF
Contract owner has too much power	HIGH	THETA-PT-21-01	5.1.1
Market maker has too much trust	MEDIUM	THETA-PT-21-02	5.1.2
Economic attack against Synthetic Mining	MEDIUM	THETA-PT-21-03	5.1.3
Deposit slippage possible on Synthetic Mining	MEDIUM	THETA-PT-21-04	5.1.4
Mispricing in Synthetic Mining for certain vaults possible	HIGH	THETA-PT-21-05	5.1.5
Some variables can be made immutable	NONE	THETA-PT-21-100	5.2.1
Common functions can be factored out	NONE	THETA-PT-21-101	5.2.2

Table 2.1: Security-Relevant Findings

2.3 Scope

The audit was performed on the code of commit id c941948[3], contained in the GitHub repository located at:

- `https://github.com/ezoia-com/thetanuts`

The following files were in scope for this audit:

file name	SHA 256 hash
call_v0.sol	262e9a2bd70b1e23ae7eef03e2b25da3d3ab1823621f3399ffbee972a8102321
common_v0.sol	dbcb73f04bfc1a1491ad66a5a10b4bd63ada6eeb5c5da7b334bdf9028a0f92b0
common_v1.sol	3e8e5bc4663f80411553cee454d13f0e539a178663659d83b7444147c4b540e0
link.sol	c10966c18ba5cb1f835bab820aede607dbf39d910b13ff134faa4ac1e76933f9
syn.sol	794db1a2a03dba1c630c1ab7e5f4780017120eece1f09a7f8d023afc8cd189e5

Table 2.2: Files in scope for audit

The contracts containing the main parts of the business logic of this audit were:

- *CoveredCallVaultV0* which is contained in *call_v0.sol* and extends the vault contract *VaultV0* contained in *common_v0.sol*
- *SynMiningVaultV1* which is contained in *syn.sol* and extends the vault contract *VaultV1* contained in *common_v1.sol*

The contract *HistoricalPriceConsumerV3_1* contained in *link.sol* is an auxilliary contract that is used to retrieve pricing information from Chainlink price feeds. In total, the audit scope was 686 lines of code.
The code of third-party dependencies were not in scope for this audit.

[3] the full commit id is c9419488c5508f67832f4efa5556cc1acbb17a6b

2.3.1 Coverage

A security assessment attempts to find the most important or sometimes as many of the existing problems as possible, though it is practically never possible to rule out the possibility of additional weaknesses being found in the future.
In this test, X41 found the time available sufficient to yield a good coverage of the given scope. We took particular care to look out for common Solidity anti-patterns and pitfalls as well as for the following classes of bugs:

- reentrancy issues
- authentication issues
- insufficient/incorrect access control
- defective business logic
- general logic bugs
- economic exploits
- centralization of authority
- trapped funds
- unsafe API usage
- integer overflows/underflows

2.4 Recommended Further Tests

It is recommended to perform dedicated audits for any changes to the reviewed contracts since even minor modifications could potentially have security implications.

2.5 Disclaimer

The information contained in this report does not constitute financial advice.

3 Rating Methodology

Security vulnerabilities are given a purely technical rating by the testers as they are discovered during the test. Business factors and financial risks for Thetanuts.Finance are beyond the scope of a penetration test which focuses entirely on technical factors. Yet technical results from a penetration test may be an integral part of a general risk assessment. A penetration test is based on a limited time frame and only covers vulnerabilities and security issues which have been found in the given time, there is no claim for full coverage.
In total, five different ratings exist, which are as follows:

Severity Rating
NONE
LOW
MEDIUM
HIGH
CRITICAL

3.1 Common Weakness Enumeration

The CWE[1] is a set of software weaknesses that allows vulnerabilities and weaknesses in software to be categorized. If applicable, X41 gives a CWE ID for each vulnerability that is discovered during a test.
CWE is a very powerful method for categorizing a vulnerability. It gives general descriptions and solution advice on recurring vulnerability types. CWE is developed by *MITRE*.[2] More information can be found on the CWE site at `https://cwe.mitre.org/`.

[1] Common Weakness Enumeration

[2] `https://www.mitre.org`

4 High-level description of the smart contracts

This chapter contains a high-level description of the smart contracts as X41 understood their function.

The contracts provide users with a way to generate yield by depositing assets of a fixed type into a vault. This vault then offers a market maker options on these assets in return for depositing a *premium* into the vault for a given *strike price*. The market maker is determined by the owner of the vault – according to discussions with the developers of the contract, the market maker is determined by an auction process.

The contract operates in rounds of fixed duration; the time duration a round is called an *epoch*. At the end of each epoch (a point in time that is called *expiry*), the options are settled. Settlement after the expiry is performed by calling a function of the smart contract. Pricing information for settlement is either retrieved from an on-chain Chainlink oracle contract or can be passed as a signed[1] data structure as a function argument.

Please note that the logic to determine strike prices and premiums for market makers is outside the scope of this audit.

4.1 Covered Calls

The contract `CoveredCallVaultV0` offers a vault that users can deposit ERC20[2] tokens into to generate yield; the tokens are of a given fixed type (for instance WETH[3]). This yield is generated by a covered call strategy: call options for a given expiry and strike price (for instance in USD) are sold to a market maker. In exchange for these call options, the market maker pays a premium into the vault.

The call options sold are European options, henceforth can only be excercised upon expiry and not before. At the time of expiry, one of two situations can happen

[1] the data is signed by a Coinbase public key

[2] Ethereum Request for Comments 20 (token standard)

[3] Wrapped Ethereum

a) In case the option is out of the money, i.e. the current market price is below the option's strike price, the size of the vault for the next round increases by the premium the market maker paid into the vault. The smart contract adjusts internal variables such that the balance of assets for each participating user is increased after the round: each user gains a share of the premium the market maker paid, proportionally to the percentage of assets the user contributed to the round.

b) In case the call option is in the money, i.e. the current market price exceeds the option's strike price, a settlement loss occurs for the users of the vault. This is defined as the difference between the current price of the asset and the strike price, converted back into the underlying asset (cash settlement). The vault is then shrunk by the total settlement loss and the settlement loss is paid out to the market maker. The smart contract then adjusts internal variables leading to the asset balance of each participating user after the round to be decreased by the settlement loss, proportionally to the percentage of assets the user contributed to the round.

The covered call contract requires users to leave their deposits in the vault for at least one epoch unless no round is active. Depositing into the vault signals the users wish to participate not in the current round but in the next round. Withdrawal requests must be pre-announced an epoch before the actual withdrawal can be performed.

4.2 Synthetic Mining

The contract *SynMiningVaultV1* allows users to deposit ERC20 tokens of a given type (e.g. WETH) into a vault that is offered to a market maker for physically settled options. Upon depositing into the contract, ERC20 tokens are minted for the depositor; for instance TN-SMv1-WETHUSDT tokens for a vault in which the so-called *base currency* is WETH and the *quote currency* is USDT. When withdrawing from the vault, these ERC20 tokens are burnt again.

Synthetic mining is a strategy for generating yield that can switch between selling physically settled call and physically settled put options. Similarly to the *CoveredCallV0* contract, internal state variables will be adjusted after each round to reflect the settlement loss or the premium gain for the users.

However, whenever a strike is excercised, a physical settlement is performed: all assets in the vault will be exchanged at the strike price; hence the asset type changes for the next epoch. Simultaneously the option type will be switched from call to put vice versa put to call for the next epoch. For example, if at the end of the first round, the market price of ETH has exceeded the strike price agreed upon with a market maker for a TN-SMv1-WETHUSDT vault, the smart contract will exchange the entire vault for USDT at the strike price with that market maker. In the next round, the vault will then offer physically settled put options on WETH.

5 Results

This chapter describes the results of this test. The security-relevant findings are documented in Section 5.1. Additionally, findings without a direct security impact are documented in Section 5.2.

5.1 Findings

The following subsections describe findings with a direct security impact that were discovered during the test.

5.1.1 THETA-PT-21-01: Contract owner has too much power

Severity:	HIGH
CWE:	266 – Incorrect Privilege Assignment
Affected Component:	`CoveredCallVaultV0`, `SynMiningVaultV1`

5.1.1.1 Description

Both the `CoveredCallVaultV0` and the `SynMiningVaultV1` contract allow the owner of the contract to perform an (emergency) withdrawal of all funds, perform a settlement strike with an arbitrary price without further checks and set an arbitrary expiry. Moreover, the `SyntheticMiningV1` contract allows the owner to set a custom price oracle while the `CoveredCallV0` contract allows the contract owner to change the price feed. All of these possibilities give the owner of the contract the ability to drain the entire vault. Allowing an EOA[1] access to these functions gives it too much power.

5.1.1.2 Solution Advice

X41 recommends mitigating this issue by transferring access control to the above functions to a governance contract or at least a multi-signature contract.

5.1.1.3 Client Response

Right now, the ability to set custom price oracles is needed as Thetanuts.Finance is pricing many new markets, some without existing oracles. Emergency withdrawal will be used in the case potential vulnerabilities are found.
Access control will be via multi-sig when we are officially launched, and will be transferred to a governance contract when the protocol is decentralised.

[1] Externally Owned Account

5.1.2 THETA-PT-21-02: Market maker has too much trust

Severity:	MEDIUM
CWE:	266 – Incorrect Privilege Assignment
Affected Component:	`CoveredCallVaultV0`, `SynMiningVaultV1`

X41 found that decoupling the process of choosing a market maker from the initialization of round parameters allows a market maker to cheat users out of their yield and cause round losses that are higher than intended.

In both the `SynMiningVaultV1` and the `CoveredCallVaultV0` contracts we notice that while the owner chooses a market maker using the ***setMaker()*** contract call, the market maker himself initializes the round and gets to choose the actual round parameters used. Hence, after a market maker has won an auction for a given strike price and premium, he is not bound to these parameters. This allows a market maker to have a risk-free round by using a premium of 0.

Hence, the current design requires the owner of the pool to watch that the market maker initializes the round with the parameters he committed to in the auction.

5.1.2.1 Solution Advice

X41 advises to introduce a *round assigner* role that allows access to a contract function which atomically sets both the market maker and the parameters of the round. This role can then be given to a smart contract for the auction process.

5.1.2.2 Client Response

The vaults are designed for simplicity and safety – the core vaults enforce that market makers cannot buy an immediately profitable position, by checking with the oracles. When the vaults are sufficiently mature, and when strike and premium selection logic has been thoroughly characterised for on-chain behaviour, and when there is sufficient liquidity in the vaults for a price-driven auction, these will be enforced not by modifying the existing code, but by adding a simple, minimal smart contract layer on top of the existing vaults that will check all the parameters before passing on the premium.

As Thetanuts.Finance is working with only known white-listed parties at the beginning, the team does not see this as an immediate threat – the team will also check and ensure that there are no inadvertent mistakes made during the new round, and will swiftly act to correct any issues.

5.1.3 THETA-PT-21-03: Economic attack against Synthetic Mining

Severity:	MEDIUM
CWE:	841 – Improper Enforcement of Behavioral Workflow
Affected Component:	*SynMiningVaultV1*

X41 found a way for an attacker to gain a percentage of the premium in some rounds that is almost risk-free for him.
In the synthetic mining scenario, a user depositing into the vault immediately takes part in the current round, regardless of the point in time he paid into the round. The size of his share of the round premium does not depend on how late he entered the round. An attacker can exploit this fact using in the following scenario:

1. Epoch is very close to coming to an end (e.g. seconds before options expiry)
2. Current market price makes option very unlikely to be exercised
3. Attacker hence decides to deposit *amt* into vault
4. Attacker waits until expiry has been reached
5. Attacker calls ***settleStrike_coinbaseOracle()*** or ***settleStrike_chainlink()***. This sets *expiry=0* in the internal function ***_settleStrike***.
6. Attacker calls ***initWithdraw(amt)***, leading to a direct withdrawal since *expiry* equals 0 (line 109 of *syn.sol*)

In the last step of the above scenario, the attacker is being paid out a higher amount than he paid into the vault as the premium paid by the market maker at the beginning of the round was distributed to all participants of the round.

5.1.3.1 Solution Advice

Three changes are necessary:

- The ***deposit()*** function must always reference the current *epoch* of *valuePerLPX1e18* when minting tokens, not *epoch - 1* when a round already is active.
- The function ***initNewRound()*** has to recalculate the current *epoch*'s *valuePerLPX1e18* (to include the premium)
- The internal function ***_settleStrike*** has to recalculate *valuePerLPX1e18* based on current *valuePerLPX1e18* of the current *epoch* and not the previous one.

This effectively causes the premium to be redistributed among all holders of Synthetic Mining tokens at the beginning of a round and not at the end.
The issue was addressed by customer in revision 4cce81a before X41 proposed any solution advice. X41 has reviewed that the changes address the issue.

5.1.3.2 Client Response

Thetanuts.Finance acknowledges the issue, and has made changes that Auddefi has since reviewed.

5.1.4 THETA-PT-21-04: Deposit slippage possible on Synthetic Mining

Severity:	MEDIUM
CWE:	367 – Time-of-check Time-of-use (TOCTOU) Race Condition
Affected Component:	*SynMiningVaultV1*

The *SynMiningVaultV1* contract does not check for a minimum size of assets in the pool before starting a round. X41 found that this can lead to deposit slippage for the market maker.
Consider the following scenario:

1. Attacker deposits *amt* to the vault
2. Attacker front-runs market maker's transaction with the contract call to ***initNewRound*** with a transaction containing a contract call to one of the ***settleStrike*** functions[2] followed by a call to ***initWithdraw(amt)***.
3. The internal function ***_settleStrike()*** causes *expiry* to be set to *0*.
4. The ***initWithdraw(amt)*** call leads to a direct withdrawal as *expiry* is 0 (line 109 of *syn.sol*).
5. The ***initNewRound*** call of the market maker will cause the round to be entered with deposit slippage.

Due to the front-run withdrawal, the market maker potentially pays a round premium that is higher than he intended.

5.1.4.1 Solution Advice

Introduce a *minSizeVault* parameter for the **initNewRound()** function of *SynMiningVaultV1* and check it against the variable *amtIntVault* before starting the round.
The customer has addressed this issue in revision c58e65d6. X41 has reviewed that the changes fix the issue.

5.1.4.2 Client Response

Thetanuts.Finance acknowledges the issue, and has made changes that Auddefi has since reviewed.

[2] ***settleStrike_coinbaseOracle()*** or ***settleStrike_chainlink()***

5.1.5 THETA-PT-21-05: Mispricing in Synthetic Mining for certain vaults possible

Severity:	HIGH
CWE:	440 – Expected Behaviour Violation
Affected Component:	*SynMiningVaultV1*

X41 found that for the Synthetic Mining vault, the ***settleStrike_coinbase()*** function may lead to mispricing of assets. Affected are vaults which are set up with a *QUOTE* currency that is not denoted in USD.

The price that is being fed to the internal ***_settleStrike()*** function by externally callable functions like ***settleStrike_coinbaseOracle()*** is pricing information for *BASE* tokens relative to *QUOTE* tokens. However, the Coinbase price oracle API only gives signed price data denoted in USD.

As an example: for a Synthetic Mining Pool with WBTC as quote currency and WETH as base currency, an attacker issuing a call to ***settleStrike_coinbaseOracle()*** will cause the ***_settleStrike()*** function to try to withdraw an amount of WETH that is several thousand times higher[3] than the correct amount. If the allowance of the market maker is configured correctly, this transaction will revert. However, if this is not the case and the transaction succeeds, an attacker can use this bug to steal from the market maker.

5.1.5.1 Solution Advice

Remove the ***settleStrike_coinbaseOracle*** function from *SynMiningVaultV1*. Alternatively, provide a ***settleStrike_coinbaseOracle*** that takes two signed data structures, one for the price of the quote currency in USD and another for the base currency in USD. This function then has to check that the timestamps for both the signed quote currency price and the signed base currency price are reasonably close [4] to each other.

5.1.5.2 Client Response

Thetanuts.Finance acknowledges the issue. The protocol is currently unaffected by the issue, as all pairings are in the base currency of USD. The team will not launch non-USD denominated vaults in the meantime, and will consider the provided advice for implementation in the future.

[3] because 1 ETH is worth > 4000 USD at the time of writing this report

[4] within 60 seconds of each other, the Coinbase price oracle API updates every minute, see `https://blog.coinbase.com/introducing-the-coinbase-price-oracle-6d1ee22c7068`

5.2 Side Findings

The following observations do not have a direct security impact, but are related to security hardening, affect functionality, or other topics that are not directly related to security. X41 recommends to mitigate these issues as well, because they often become exploitable in the future. Doing so will strengthen the security of the system and is recommended for defense in depth.

5.2.1 THETA-PT-21-100: Some variables can be made immutable

Affected Component: `SynMiningVaultV1`, `VaultV0`

5.2.1.1 Description

We noticed that some variables are only initialized during contract creation and can not be modified by any of the functions in the contracts. Unless setters are added for these variables, it makes sense to declare these variables immutable to save storage costs.

5.2.1.2 Solution Advice

Make the following variables immutable:

in `SynMiningVaultV1` : `QUOTE`, `BASE`, `BASE_DECIMALS` and `QUOTE_DECIMALS`

in `VaultV0` : `START_TIME`, `PERIOD`, and `DUST_FLOOR`.

5.2.1.3 Client Response

Thetanuts.Finance accepts the recommendations and will declare variables only initialised by constructor immutable.

5.2.2 THETA-PT-21-101: Common functions can be factored out

Affected Component: `VaultV0`, `VaultV1`

5.2.2.1 Description

X41 found that cases of code duplication exist across the vault implementations.

5.2.2.2 Solution Advice

The following functions can be factored out:

- ***settleStrike_chainlink()***
- ***settleStrike_chainlink_fallback()***
- ***settleStrike_coinbaseOracle()***
- ***settleStrike_MM()***
- ***setExpiry()***
- ***setMaker()***
- ***setMaxCap()***

5.2.2.3 Client Response

Thetanuts.Finance acknowledges the issue. Code duplication exists as all VaultV0 are meant to be phased out in favour of VaultV1's structure.

6 About X41 D-Sec GmbH

X41 D-Sec GmbH is an expert provider for application security and penetration testing services. Having extensive industry experience and expertise in the area of information security, a strong core security team of world-class security experts enables X41 D-Sec GmbH to perform premium security services.

X41 has the following references that show their experience in the field:

- Review of the Mozilla Firefox updater[1]
- X41 Browser Security White Paper[2]
- Review of Cryptographic Protocols (Wire)[3]
- Identification of flaws in Fax Machines[4,5]
- Smartcard Stack Fuzzing[6]

The testers at X41 have extensive experience with penetration testing and red teaming exercises in complex environments. This includes enterprise environments with thousands of users and vendor infrastructures such as the Mozilla Firefox Updater (Balrog).

Fields of expertise in the area of application security encompass security-centered code reviews, binary reverse-engineering and vulnerability-discovery. Custom research and IT security consulting, as well as support services, are the core competencies of X41. The team has a strong technical background and performs security reviews of complex and high-profile applications such as Google Chrome and Microsoft Edge web browsers.

X41 D-Sec GmbH can be reached via `https://x41-dsec.de` or `mailto:info@x41-dsec.de`.

[1] https://blog.mozilla.org/security/2018/10/09/trusting-the-delivery-of-firefox-updates/

[2] https://browser-security.x41-dsec.de/X41-Browser-Security-White-Paper.pdf

[3] https://www.x41-dsec.de/reports/Kudelski-X41-Wire-Report-phase1-20170208.pdf

[4] https://www.x41-dsec.de/lab/blog/fax/

[5] https://2018.zeronights.ru/en/reports/zero-fax-given/

[6] https://www.x41-dsec.de/lab/blog/smartcards/

Acronyms

CWE	Common Weakness Enumeration	9
EOA	Externally Owned Account	13
ERC20	Ethereum Request for Comments 20 (token standard)	10
WETH	Wrapped Ethereum	10