ExVul

Oct 2024

EXVUL WEB3 SECURITY AUDIT FOR OKX

WEB3 SECURITY

https://exvul.com

Table of Contents

1.EXECUTIVE SUMMARY 3
1.1 Methodology 3

2. FINDINGS OVERVIEW 5
2.1 Project Info And Contract Address 5
2.2 Summary 5
2.3 Key Findings 6

3. DETAILED DESCRIPTION OF FINDINGS 7
3.1 Inadequate CoinMintPayload error handling 7
3.2 Inadequate regex error handling 8
3.3 No check if builder.InscriptionTxCtxDataList is 0 9
3.4 Should clear the privatekey , when return to prevent the privatekey in the memroy long time 10
3.5 ingeger overflow risk 11
3.6 Unused parameters 12

4.CONCLUSION 13

5. APPENDIX 14
5.1 Basic Coding Assessment 14
5.1.1 Apply Verification Control 14
5.1.2 Authorization Access Control 14
5.1.3 Forged Transfer Vulnerability 14
5.1.4 Transaction Rollback Attack 14
5.1.5 Transaction Block Stuffing Attack 14
5.1.6 Soft Fail Attack Assessment 14
5.1.7 Hard Fail Attack Assessment 14
5.1.8 Abnormal Memo Assessment 14
5.1.9 Abnormal Resource Consumption 15
5.1.10 Random Number Security 15
5.2 Advanced Code Scrutiny 15
5.2.1 Cryptography Security 15
5.2.2 Account Permission Control 15
5.2.3 Malicious Code Behavior 15
5.2.4 Sensitive Information Disclosure 15
5.2.5 System API 15

6. DISCLAIMER 16

7. REFERENCES 17

1.EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by go-wallet-sdk to review Wallet SDK implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

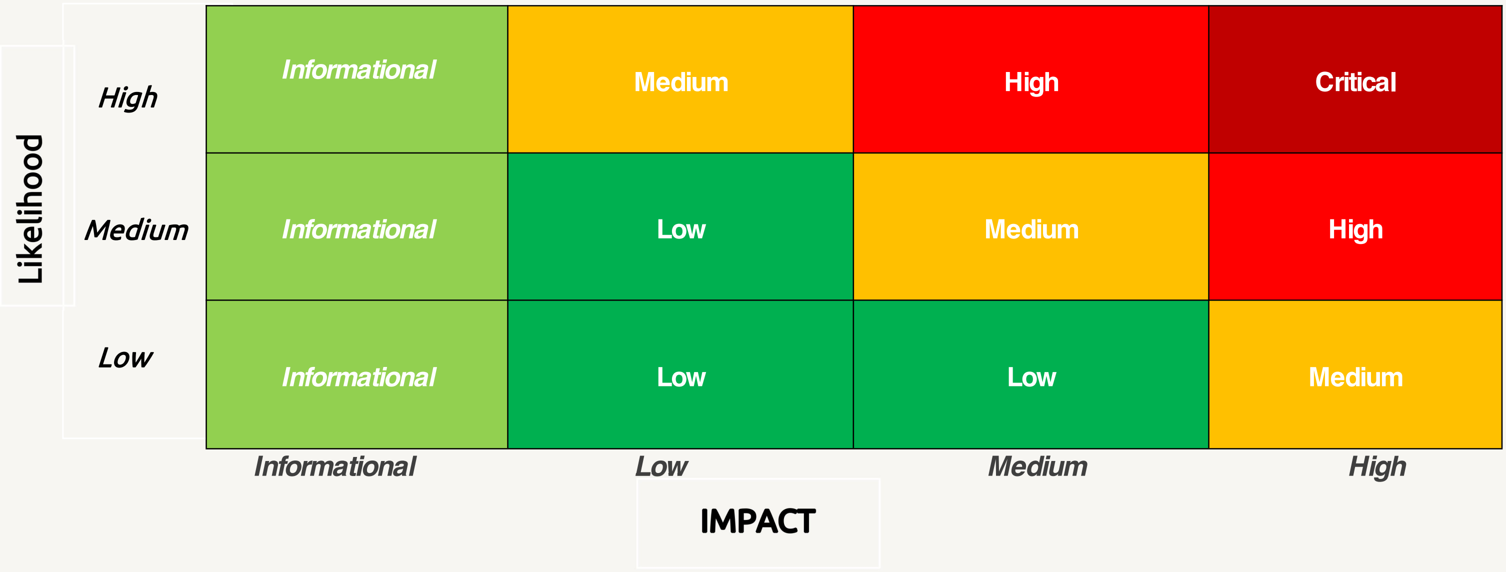
The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into for: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly, Critical, High, Medium, Low, Informational shown in table 1.1.



Likelihood	Informational	Low	Medium	High
High	Informational	Medium	High	Critical
Medium	Informational	Low	Medium	High
Low	Informational	Low	Low	Medium

IMPACT

Table 0.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impact security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy code on our private test environment and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given code with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- Code and business security testing: We further review the business logic and examine the system operation to identify possible pitfalls and/or errors.
- Additional Recommendations: We also provide additional advice on coding and development from the perspective of proven programming practices.

Category	Assessment Item
Basic Coding Assessment	Apply Verification Control
	Authorization Access Control
	Forged Transfer Vulnerability
	Forged Transfer Notification
	Numeric Overflow
	Transaction Rollback Attack
	Transaction Block Stuffing Attack
	Soft Fail Attack
	Hard Fail Attack
	Abnormal Memo
	Abnormal Resource Consumption
	Secure Random Number
Advanced Source Code Scrutiny	Asset Security
	Cryptography Security
	Business Logic Review
	Source Code Functional Verification
	Account Authorization Control
	Sensitive Information Disclosure
	Circuit Breaker
	Blacklist Control
	System API Call Analysis
	Contract Deployment Consistency Check
Additional Recommendations	Semantic Consistency Checks
	Following Other Best Practices

Table 0.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

2. FINDINGS OVERVIEW

2.1 Project Info And Contract Address

Project Name: go-wallet-sdk

Audit Time: October 15, 2024 - October 28, 2024

Language: Go

File Name	Link
go-wallet-sdk	https://github.com/okx/go-wallet-sdk/commit/1810380535560a104190b35bca080e7141bf5c45

2.2 Summary

Severity	Found	
Critical	0	
High	0	
Medium	0	
Low	5	🟦🟦🟦🟦🟦
Informational	1	🟩

2.3 Key Findings

ID	Severity	Findings Title	Status	Confirm
NVE- 001	**Low**	Inadequate CoinMintPayload error handling	Ignore	Confirmed
NVE- 002	**Low**	Inadequate regex error handling	Ignore	Confirmed
NVE- 003	**Low**	No check if builder.InscriptionTxCtxDataList is 0	Ignore	Confirmed
NVE- 004	**Low**	Should clear the privatekey , when return to prevent the privatekey in the memroy long time	Ignore	Confirmed
NVE- 005	**Low**	ingeger overflow risk	Ignore	Confirmed
NVE- 006	**Informational**	Unused parameters	Ignore	Confirmed

Table 2.3: Key Audit Findings

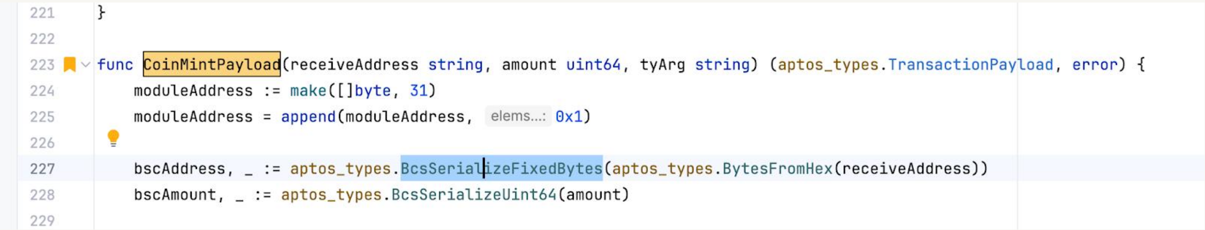
3. DETAILED DESCRIPTION OF FINDINGS

3.1 Inadequate CoinMintPayload error handling

ID:	NVE-001	Location:	aptos/aptos.go
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

In the CoinMintPayload function, the aptos_types.BcsSerializeFixedBytes and aptos_types.BcsSerializeUint64 functions are called to process the receiveAddress and amount parameters, but the return error values of these two functions are not handled. Unhandled errors may cause the program to continue executing and generate unexpected errors when these functions fail to execute.



```go
}

func CoinMintPayload(receiveAddress string, amount uint64, tyArg string) (aptos_types.TransactionPayload, error) {
    moduleAddress := make([]byte, 31)
    moduleAddress = append(moduleAddress, elems...: 0x1)

    bscAddress, _ := aptos_types.BcsSerializeFixedBytes(aptos_types.BytesFromHex(receiveAddress))
    bscAmount, _ := aptos_types.BcsSerializeUint64(amount)

```

Recommend:

Added error checking to check for errors after calling BcsSerializeFixedBytes and BcsSerializeUint64, and return errors appropriately to ensure interrupts and feedback when errors occur.

Status: Ignore

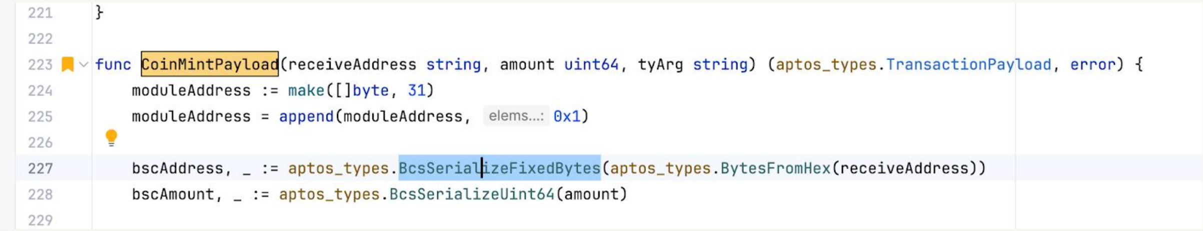
Customer response: In this scenario, other factors will cause errors and interceptions, which will be included in the overall optimization of the code style.

3.2 Inadequate regex error handling

ID:	NVE-002	Location:	aptos/aptos.go
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

In the ShortenAddress function, regexp.Compile("^0x0*") is used to compile the regular expression, but the returned error value is ignored. This incorrect omission may cause the program to continue executing when the regular expression compilation fails, which may cause re.ReplaceAllString to behave abnormally.



```
}

func CoinMintPayload(receiveAddress string, amount uint64, tyArg string) (aptos_types.TransactionPayload, error) {
    moduleAddress := make([]byte, 31)
    moduleAddress = append(moduleAddress, elems...: 0x1)

    bscAddress, _ := aptos_types.BcsSerializeFixedBytes(aptos_types.BytesFromHex(receiveAddress))
    bscAmount, _ := aptos_types.BcsSerializeUint64(amount)

```

Recommend:

Check for errors after compiling the regular expression and handle them appropriately to ensure that execution does not continue if regular expression compilation fails.

Status: Ignore

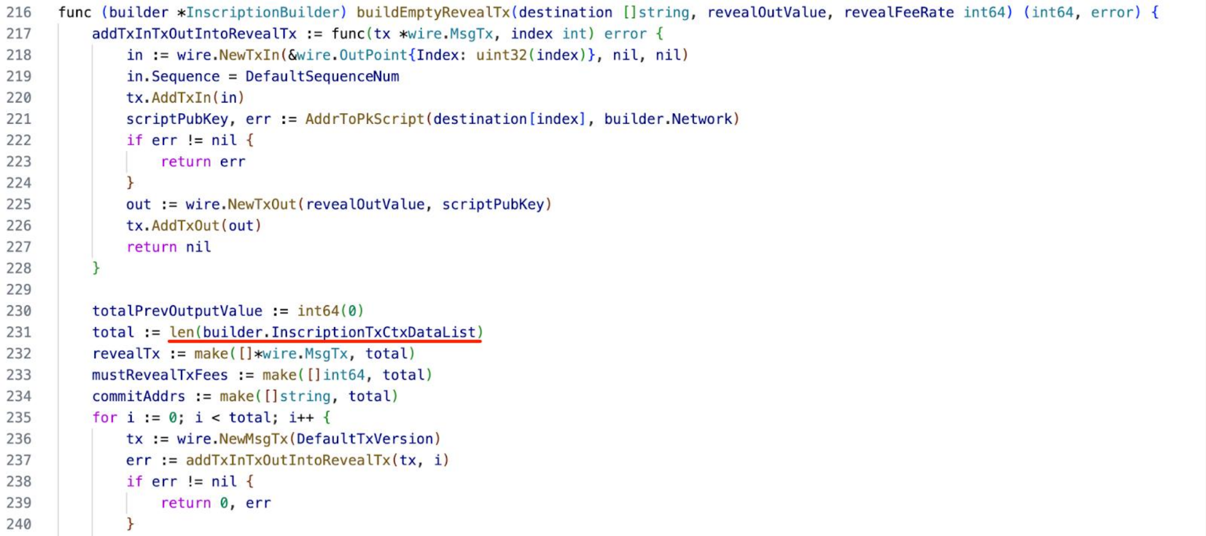
Customer response: In this scenario, other factors will cause error interception, which will be included in the overall optimization of the code style and will be combined with the data on the end for judgment and processing.

3.3 No check if builder.InscriptionTxCtxDataList is 0

ID:	NVE-003	Location:	bitcoin/inscribe.go
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

In the buildEmptyRevealTx function, it is recommended to check the length of builder.InscriptionTxCtxDataList to prevent unexpected behavior when its length is 0. If the list is empty, then continuing the execution may cause index error or null reference exception. You can check it at the beginning and return an error immediately if the length is 0.



```go
func (builder *InscriptionBuilder) buildEmptyRevealTx(destination []string, revealOutValue, revealFeeRate int64) (int64, error) {
    addTxInTxOutIntoRevealTx := func(tx *wire.MsgTx, index int) error {
        in := wire.NewTxIn(&wire.OutPoint{Index: uint32(index)}, nil, nil)
        in.Sequence = DefaultSequenceNum
        tx.AddTxIn(in)
        scriptPubKey, err := AddrToPkScript(destination[index], builder.Network)
        if err != nil {
            return err
        }
        out := wire.NewTxOut(revealOutValue, scriptPubKey)
        tx.AddTxOut(out)
        return nil
    }

    totalPrevOutputValue := int64(0)
    total := len(builder.InscriptionTxCtxDataList)
    revealTx := make([]*wire.MsgTx, total)
    mustRevealTxFees := make([]int64, total)
    commitAddrs := make([]string, total)
    for i := 0; i < total; i++ {
        tx := wire.NewMsgTx(DefaultTxVersion)
        err := addTxInTxOutIntoRevealTx(tx, i)
        if err != nil {
            return 0, err
        }
```

Recommend:

Added length check for builder.InscriptionTxCtxDataList. If the length is 0, an error is returned immediately and execution stops.

Status: Ignore

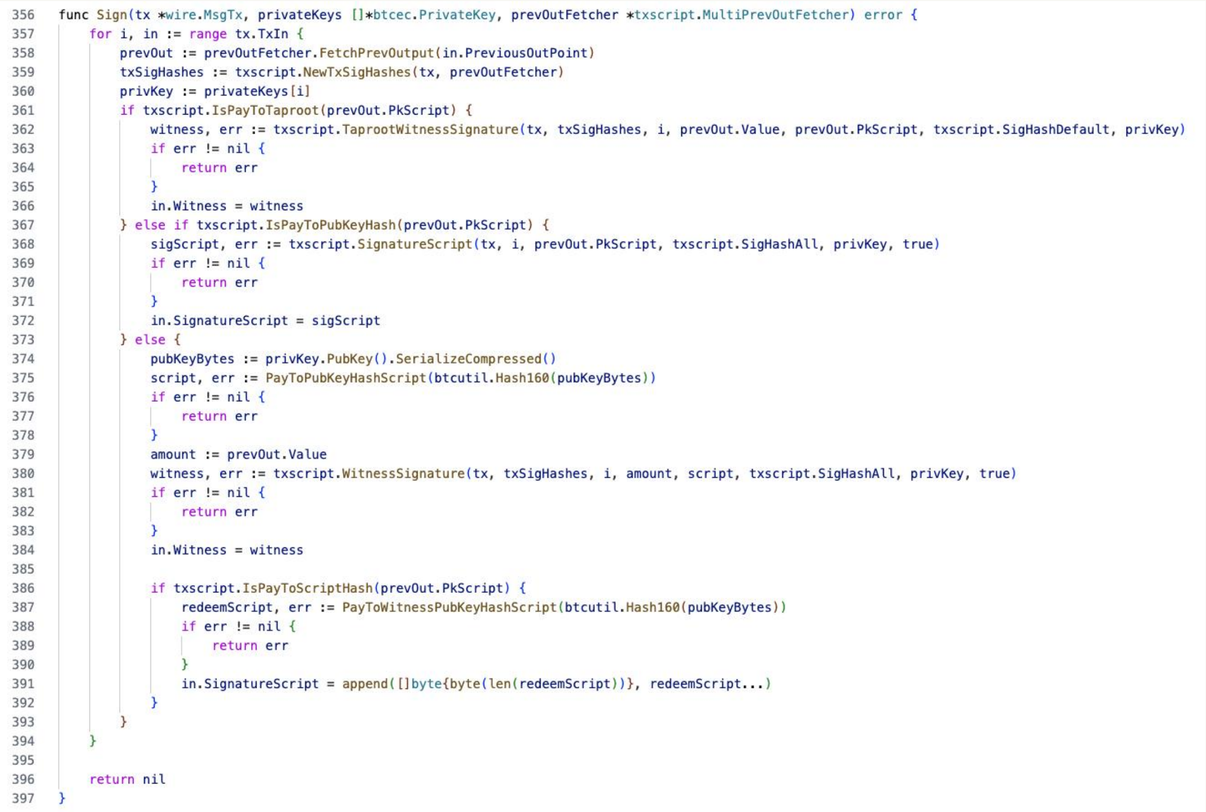
Customer response: If it is not returned in time, it just means that the reveal transaction cannot be constructed, and it will not cause other problems.

3.4 Should clear the privatekey , when return to prevent the privatekey in the memroy long time

ID:	NVE-004	Location:	bitcoin/inscribe.go
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

In the Sign function, privateKeys is not cleared before returning, which may cause the private key to reside in the memory for too long, causing potential security risks. You can clear the content of privateKeys before returning to ensure that the private key is not kept in the memory for a long time.



```go
func Sign(tx *wire.MsgTx, privateKeys []*btcec.PrivateKey, prevOutFetcher *txscript.MultiPrevOutFetcher) error {
    for i, in := range tx.TxIn {
        prevOut := prevOutFetcher.FetchPrevOutput(in.PreviousOutPoint)
        txSigHashes := txscript.NewTxSigHashes(tx, prevOutFetcher)
        privKey := privateKeys[i]
        if txscript.IsPayToTaproot(prevOut.PkScript) {
            witness, err := txscript.TaprootWitnessSignature(tx, txSigHashes, i, prevOut.Value, prevOut.PkScript, txscript.SigHashDefault, privKey)
            if err != nil {
                return err
            }
            in.Witness = witness
        } else if txscript.IsPayToPubKeyHash(prevOut.PkScript) {
            sigScript, err := txscript.SignatureScript(tx, i, prevOut.PkScript, txscript.SigHashAll, privKey, true)
            if err != nil {
                return err
            }
            in.SignatureScript = sigScript
        } else {
            pubKeyBytes := privKey.PubKey().SerializeCompressed()
            script, err := PayToPubKeyHashScript(btcutil.Hash160(pubKeyBytes))
            if err != nil {
                return err
            }
            amount := prevOut.Value
            witness, err := txscript.WitnessSignature(tx, txSigHashes, i, amount, script, txscript.SigHashAll, privKey, true)
            if err != nil {
                return err
            }
            in.Witness = witness

            if txscript.IsPayToScriptHash(prevOut.PkScript) {
                redeemScript, err := PayToWitnessPubKeyHashScript(btcutil.Hash160(pubKeyBytes))
                if err != nil {
                    return err
                }
                in.SignatureScript = append([]byte{byte(len(redeemScript))}, redeemScript...)
            }
        }
    }

    return nil
}
```

Recommend:

It is recommended to clear privateKeys before returning to prevent the private key from staying in memory for too long, causing potential security risks.

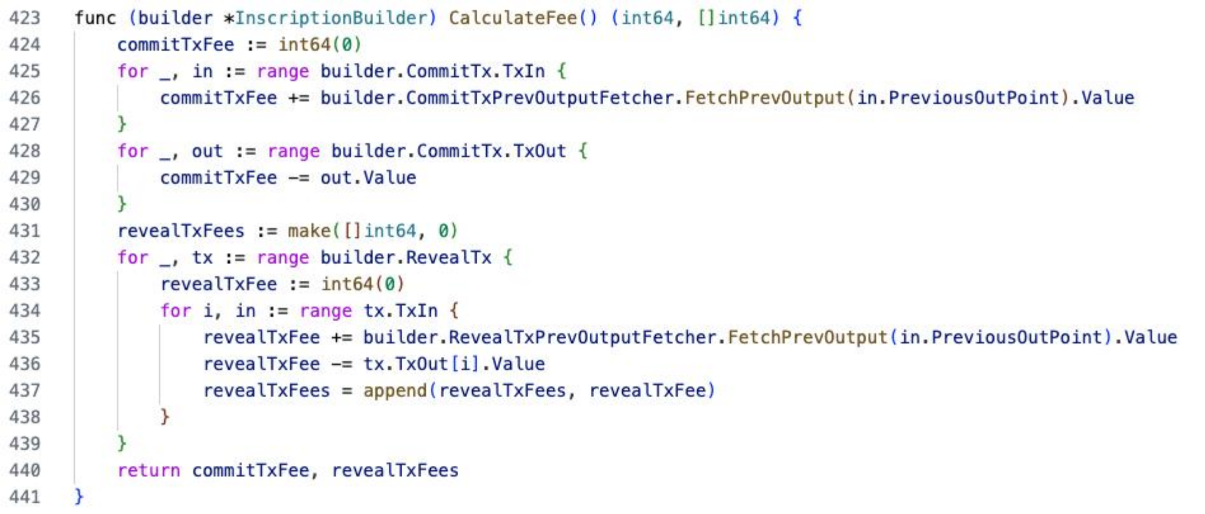
Status: Ignore

3.5 ingeger overflow risk

ID:	NVE-005	Location:	bitcoin/inscribe.go
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Low

Description:

In the CalculateFee function, due to multiple accumulation operations (such as commitTxFee += ... and revealTxFee += ...), there may be a risk of integer overflow, especially in the case of large values. To prevent overflow, you can check the result of each accumulation before accumulation, or use a safer method to handle the accumulation operation.



```
func (builder *InscriptionBuilder) CalculateFee() (int64, []int64) {
    commitTxFee := int64(0)
    for _, in := range builder.CommitTx.TxIn {
        commitTxFee += builder.CommitTxPrevOutputFetcher.FetchPrevOutput(in.PreviousOutPoint).Value
    }
    for _, out := range builder.CommitTx.TxOut {
        commitTxFee -= out.Value
    }
    revealTxFees := make([]int64, 0)
    for _, tx := range builder.RevealTx {
        revealTxFee := int64(0)
        for i, in := range tx.TxIn {
            revealTxFee += builder.RevealTxPrevOutputFetcher.FetchPrevOutput(in.PreviousOutPoint).Value
            revealTxFee -= tx.TxOut[i].Value
            revealTxFees = append(revealTxFees, revealTxFee)
        }
    }
    return commitTxFee, revealTxFees
}
```

Recommend:

Check the accumulated value and after each accumulation operation, check whether overflow occurred. If overflow occurred, return an error.

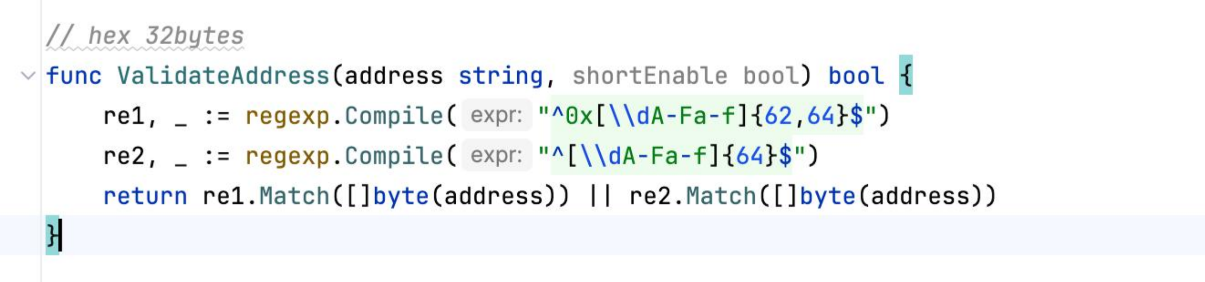
Status: Ignore

3.6 Unused parameters

ID:	NVE-006	Location:	aptos/aptos.go
Severity:	Informational	Category:	Business Issues
Likelihood:	Low	Impact:	Informational

Description:

In the ValidateAddress function, a shortEnable parameter is passed in, but the parameter is not used in the function. This may confuse the caller and make them think that the shortEnable parameter has the function of affecting address validation.



```
// hex 32bytes
func ValidateAddress(address string, shortEnable bool) bool {
    re1, _ := regexp.Compile( expr: "^0x[\\dA-Fa-f]{62,64}$")
    re2, _ := regexp.Compile( expr: "^[\\dA-Fa-f]{64}$")
    return re1.Match([]byte(address)) || re2.Match([]byte(address))
}
```

Recommend:

If the shortEnable parameter does not affect the function logic and is not needed in the design, you can directly remove it from the parameter list. If the shortEnable parameter is to support a specific address format, you can add the corresponding logic to the function.

Status: Ignore

Customer response: As a follow-up optimization of the code optimization project.

4.CONCLUSION

In this audit, we thoroughly analyzed **go-wallet-sdk** Wallet-Sdk implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be PASSED. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. APPENDIX

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: Critical

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the code
- Result: Not found
- Severity: Critical

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the code.
- Result: Not found
- Severity: Critical

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.6 Soft Fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.7 Hard Fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

5.1.8 Abnormal Memo Assessment

- Description: Assess whether there is abnormal memo vulnerability in the code.
- Result: Not found
- Severity: Critical

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in code processing.
- Result: Not found
- Severity: Critical

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

5.2.2 Account Permission Control

- Description: Examine permission control issue in the code
- Results: Not Found
- Severity: Medium

5.2.3 Malicious Code Behavior

- Description: Examine whether sensitive behavior present in the code
- Results: Not found
- Severity: Medium

5.2.4 Sensitive Information Disclosure

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

6. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

7. REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error.

https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation.

https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure.

https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors.

https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

ExVul

EXVUL
WEB3 SECURITY

www.exvul.com

contact@exvul.com

www.x.com/EXVULSEC

github.com/EXVUL-Sec