HALBORN

Mysten Labs - Sui TypeScript SDK

SDK Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 6th, 2022 - November 1st, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/01/2022	George Skouroupathis
0.2	Draft Review	11/01/2022	Constantin Casmir
0.3	Draft Review	11/01/2022	Gabi Urrutia
1.0	Remediation Plan	05/09/2023	John Saigle
1.1	Remediation Plan Review	05/10/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
George Skouroupathis	Halborn	George.Skouroupathis@halborn.com
John.Saigle	Halborn	John.Saigle@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Mysten Labs engaged Halborn to conduct a security audit on their Sui TypeScript SDK, beginning on October 6th, 2022 and ending on November 1st, 2022 . The security assessment was scoped to the Mysten Labs Sui TypeScript SDK. To begin the test, the Client's team provided the source code for Halborn to conduct security testing using tools to scan, detect, validate possible vulnerabilities found in the SDK and report the findings at the end of the engagement.

The Sui TypeScript SDK is a library that enables clients to connect and work with the Sui platform. It allows users to develop applications by managing secrets and signing, serialization, and by providing a way of communication to the appropriate endpoint.

1.2 AUDIT SUMMARY

The team at Halborn was provided approximately three weeks for the engagement and assigned a full-time security engineer to audit the security of the application. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The goals of our security audits are to improve the quality of the systems reviewed and to target sufficient remediation to help protect users.

During the test, Halborn did not identify any Critical or High-risk issues. However, Halborn detected that the Sui TypeScript SDK makes use of packages with known vulnerabilities.

Moreover, Halborn detected that some dependencies in package.json are not pinned to the exact version and are rather set to a compatible version.

Lastly, it was discovered that the SDK does not perform any data validation on parameters before it performs requests to the back-end. This leaves all the computation needed for the checks to be performed at the back-end,

and also has an impact on the bandwidth available on the server.

To remediate the identified security issues, it is recommended that Mysten Labs implements the following high-level remediation actions:

- Update all packages used in the application.
- Validate each message's parameters before passing it through to the back-end.
- Pin dependencies to exact versions.
- Return custom error from the application's API as to not disclose the technologies on which the application relies.

1.3 SCOPE

The security assessment was scoped to the following components:

Mysten Labs Sui TypeScript SDK:

- MystenLabs / Sui (TypeScript SDK)
  - Git commit ID: 059ede517255f70dd0733d11c36b788ef434a98a

1.4 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of the penetration test. While manual testing is recommended to uncover flaws in logic, process and implementation; automated testing techniques assist enhance coverage of the solution and can quickly identify flaws in it.
The following phases and associated tools were used throughout the term of the audit:

- Storing assets securely
- Exposure of any critical information during user interactions with the blockchain and external libraries
- Application Logic Flaws
- Areas where insufficient validation allows for hostile input
- Application of cryptography to protect secrets
- Input Handling
- Fuzzing of all input parameters
- Technology stack-specific vulnerabilities and Code Audit
- Known vulnerabilities in 3rd party / OSS dependencies

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.

4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	3	2

LIKELIHOOD

IMPACT

	(HAL-01)			
(HAL-04) (HAL-05)	(HAL-02) (HAL-03)			

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
USE OF PACKAGES WITH KNOWN VULNERABILITIES	Low	SOLVED - 10/06/2022
DEPENDENCIES SHOULD BE PINNED TO EXACT VERSIONS	Low	RISK ACCEPTED
LACK OF INPUT VALIDATION	Low	PARTIALLY SOLVED - 10/06/2022
REFLECTED UNSANITIZED INPUT	Informational	RISK ACCEPTED
APPLICATION ERROR MAY DISCLOSE SENSITIVE TECHNOLOGY INFORMATION	Informational	NOT APPLICABLE

FINDINGS & TECH DETAILS

3.1 (HAL-01) USE OF PACKAGES WITH KNOWN VULNERABILITIES - LOW

Description:

The Mysten Sui TypeScript SDK uses some third-party dependencies to work with sandboxes, XLM, string manipulation and globs. However, dependencies create an expected disadvantage in that the actual application's security posture now relies on it.

The following table shows the vulnerable packages and their respective vulnerabilities, found with pnpm audit:

Title	Package	Severity
Sandbox Escape with RCE	vm2	Critical
Prototype Pollution	@xlmdom/xlmdom	Critical
Regular Expression DoS	trim	High
Regular Expression DoS	glob-parent	High

Code Location:

Listing 1: sui/pnpm-lock.yaml

```
10086    /degenerator/3.0.2:
10087      resolution: {integrity: sha512-c0mef3SNQo56t6urUU6tdQAs+ThoD0o
   ↳ 9B9MJ8HEt7NQcGEILCRFqQb7ZbP9JAv+QF1Ky5plydhMR/IrqWDm+TQ==}
10088      engines: {node: '>= 6'}
10089      dependencies:
10090        ast-types: 0.13.4
10091        escodegen: 1.14.3
10092        esprima: 4.0.1
10093        vm2: 3.9.10
10094      dev: true
```

Listing 2: sui/pnpm-lock.yaml

```
7291    /@xmldom/xmldom/0.7.5:
7292      resolution: {integrity: sha512-V3BIhmY36fXZ1OtVcI9W+
  ↳ FxQqxVLsPKcNjWigIaa81dLC9IolJl5Mt4Cvhmr0flUnjSpTdrbMTSbXqYqV5dT6A
  ↳ ==}
7293      engines: {node: '>=10.0.0'}
7294      dev: true
```

Listing 3: sui/pnpm-lock.yaml

```
18016    /remark-parse/8.0.3:
18017      resolution: {integrity: sha512-E1K9+QLGgggHxCQtLt++
  ↳ uXltxEprmWzNfg+MxpfHsZlrddKzZ/hZyWHDbK3/Ap8HJQqYJRXP+jHczdL6q6i85Q
  ↳ ==}
18018      dependencies:
18019        ccount: 1.1.0
18020        collapse-white-space: 1.0.6
18021        is-alphabetical: 1.0.4
18022        is-decimal: 1.0.4
18023        is-whitespace-character: 1.0.4
18024        is-word-character: 1.0.4
18025        markdown-escapes: 1.0.4
18026        parse-entities: 2.0.0
18027        repeat-string: 1.6.1
18028        state-toggle: 1.0.3
18029        trim: 0.0.1
```

Listing 4: sui/pnpm-lock.yaml

```
8920    /chokidar/2.1.8:
8921      resolution: {integrity: sha512-ZmZUazfOzf0Nve7duiCKD23PFSCs4
  ↳ JPoYyccjUFF3aQkQadqBhfzhjkwBH2mNOG9cTBwhamM37EIsIkZw3nRgg==}
8922      deprecated: Chokidar 2 does not receive security updates since
  ↳  2019. Upgrade to chokidar 3 with 15x fewer dependencies
8923      dependencies:
8924        anymatch: 2.0.0
8925        async-each: 1.0.3
8926        braces: 2.3.2
8927        glob-parent: 3.1.0
```

Risk Level:

Likelihood - 2
Impact - 3

CVSS Vector:

- AV:L/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N

Recommendation:

It is recommended to upgrade the packages to the latest version.

Remediation Plan:

SOLVED: The output of the command `pnpm --prod audit` shows no vulnerable packages when run on commit ID `89992738566dda77895d34252e15760c05b97ce9`.

3.2 (HAL-02) DEPENDENCIES SHOULD BE PINNED TO EXACT VERSIONS - LOW

Description:

The application contains some dependencies that are not pinned to an exact version, but are instead set to compatible version (^x.x.x). This can potentially allow dependency attacks.

Code Location:

Listing 5: sui/sdk/typescript/package.json

```
63  "devDependencies": {
64    "@mysten/sui-open-rpc": "file:../../crates/sui-open-rpc",
65    "@size-limit/preset-small-lib": "^7.0.8",
66    "@types/bn.js": "^5.1.0",
67    "@types/lossless-json": "^1.0.1",
68    "axios": "^0.27.2",
69    "eslint": "^8.23.0",
70    "eslint-config-prettier": "^8.5.0",
71    "eslint-config-react-app": "^7.0.1",
72    "eslint-plugin-prettier": "^4.2.1",
73    "husky": "^7.0.4",
74    "mockttp": "^2.7.0",
75    "prettier": "^2.7.1",
76    "size-limit": "^7.0.8",
77    "ts-auto-guard": "^2.4.1",
78    "ts-node": "^10.9.1",
79    "tslib": "^2.3.1",
80    "tsup": "^6.2.2",
81    "typedoc": "^0.23.11",
82    "typescript": "^4.8.3",
83    "vitest": "^0.22.1",
84    "wait-on": "^6.0.1"
85  },
86  "dependencies": {
87    "@mysten/bcs": "workspace:*",
88    "@noble/hashes": "^1.1.2",
89    "@noble/secp256k1": "^1.6.3",
90    "@scure/bip32": "^1.1.0",
```

```
    "@scure/bip39": "^1.1.0",
    "bn.js": "^5.2.0",
    "buffer": "^6.0.3",
    "cross-fetch": "^3.1.5",
    "jayson": "^3.6.6",
    "js-sha3": "^0.8.0",
    "lossless-json": "^1.0.5",
    "rpc-websockets": "^7.5.0",
    "tweetnacl": "^1.0.3"
  }
```

Risk Level:

Likelihood - 2
Impact - 2

CVSS Vector:

- AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N

Recommendation:

Pinning dependencies to an exact version (=x.x.x) can reduce the chance of inadvertently introducing a malicious version of a dependency in the future.

Remediation Plan:

RISK ACCEPTED: The Mysten Labs team accepted the risk of this finding.

3.3 (HAL-03) LACK OF INPUT VALIDATION - LOW

Description:

During the Sui TypeScript SDK analysis it was observed that the methods of the Provider classes **(sui/sdk/typescript/src/providers/json-rpc-provider.ts and sui/sdk/typescript/src/providers/json-rpc-provider-with-cache.ts)** make direct calls to the Sui back-end application without any kind of input validation.

The lack of input validation on the SDK leaves all kind of validations to the back-end itself, leading to an unnecessary computation waste on that side. A simple validation of input parameters before the request to the server and a proper error handling reduce the computational waste and highly improve the quality and utility of the SDK.

Furthermore, validation of input is an extra security layer which deters attackers from sending arbitrary data to the back-end application, thus preventing injection and fuzzing attacks.

Code Location:

Listing 6: sui/sdk/typescript/src/providers/json-rpc-provider.ts

```
512   async getEventsBySender(
513     sender: SuiAddress,
514     count: number = EVENT_QUERY_MAX_LIMIT,
515     startTime: number = DEFAULT_START_TIME,
516     endTime: number = DEFAULT_END_TIME
517   ): Promise<SuiEvents> {
518     try {
519       return await this.client.requestWithType(
520         'sui_getEventsBySender',
521         [sender, count, startTime, endTime],
522         isSuiEvents,
523         this.skipDataValidation
524       );
```

```
525     } catch (err) {
526       throw new Error(
527         `Error getting events by sender: ${sender}, with error: ${
 ↳ err}`
528       );
529     }
530   }
```

The above example shows the getEventsBySender method, which sends an RPC HTTP request to the back-end gateway to collect the events generated by a specific “sender” address.
This method sends the request directly using the Client.requestWithType method, without firstly checking if the address is valid (has the correct length and format).
This shifts the responsibility of validation to the gateway, whereas a simple check at the SDK could prevent the costly calculation at the back-end.

Screenshots/Videos:

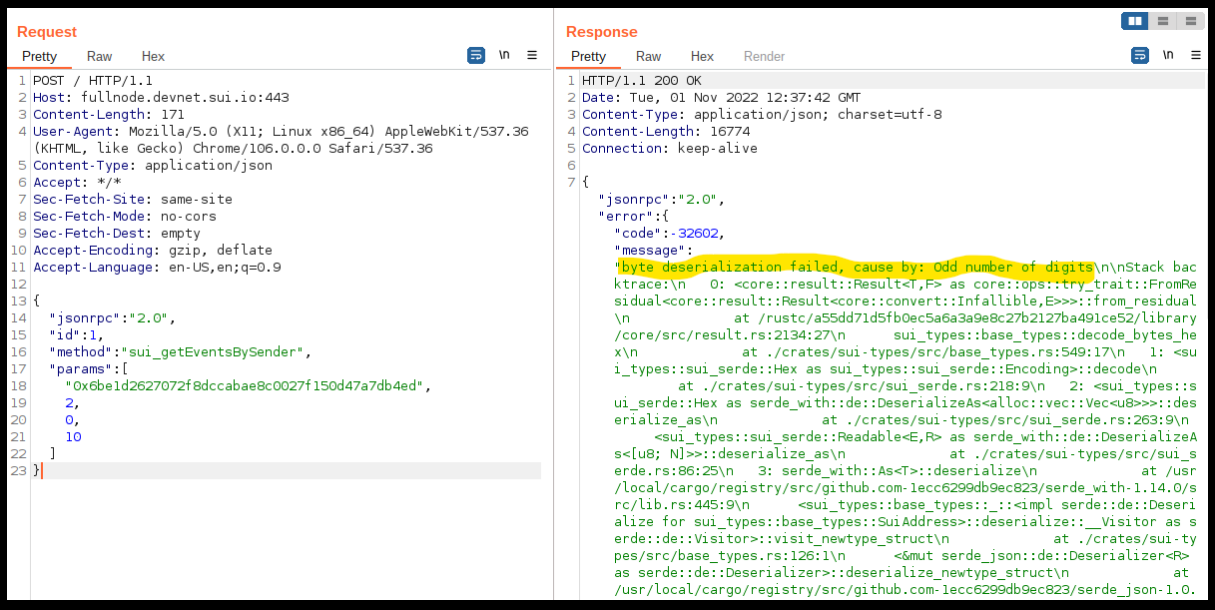


Figure 1: The error returned from the gateway when using a malformed address

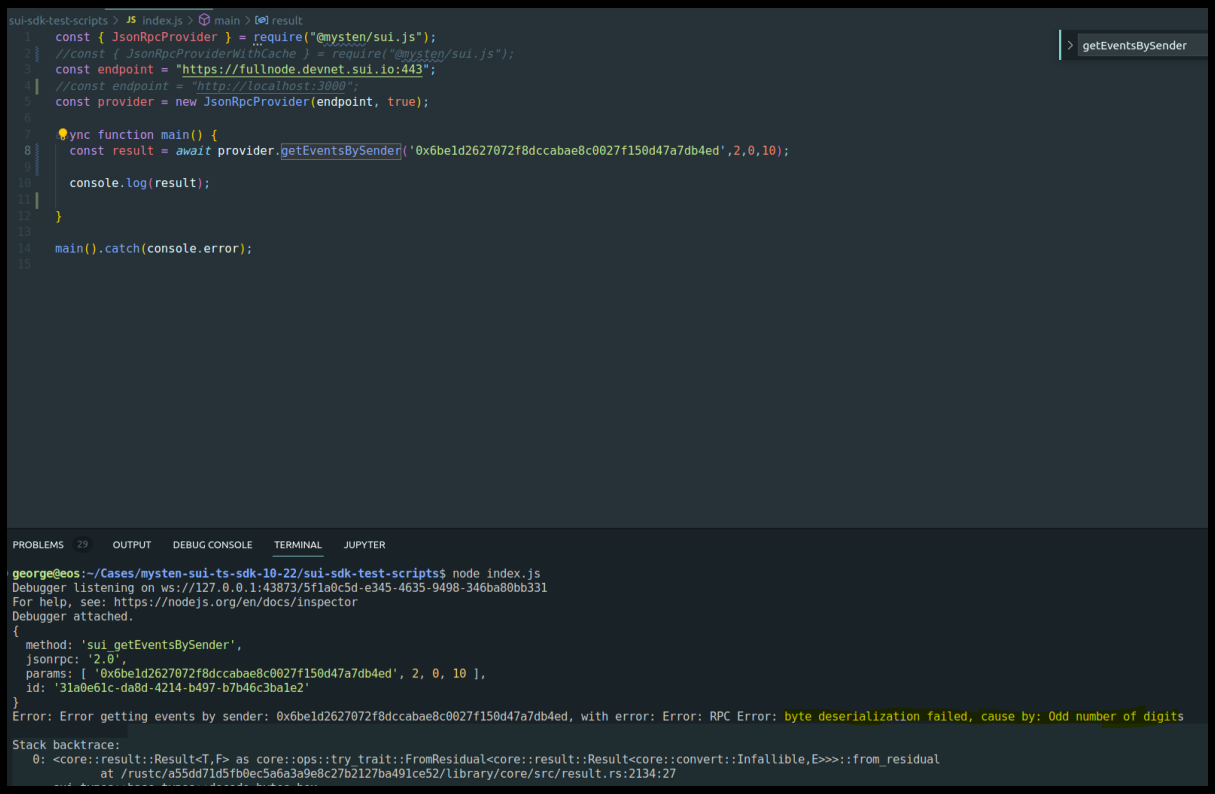


Figure 2: A client application calling getEventsBySender with a malformed address, getting the error response from the gateway

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to implement input validation inside the SDK methods to avoid unnecessary communication with the back-end server. Currently, all kinds of input validation depend on how the user makes use of the SDK, and their responsibility for adding the extra code needed for all the checks.

Remediation Plan:

PARTIALLY SOLVED: The target endpoint getEventsBySender no longer exists in the codebase as of commit ID 89992738566dda77895d34252e15760c05b97ce9.

Some methods in the SDK perform validation of parameters before they are passed to the server, but others do not.

3.4 (HAL-04) REFLECTED UNSANITIZED INPUT - INFORMATIONAL

During the assessment, Halborn detected that the https://fullnode.devnet.sui.io back-end gateway returns user input without sanitizing it for display in the web browser.

Various back-end endpoints accept HTTP requests containing user input, and their reply contains the same user input without sanitizing it first, thus introducing the risk of **Cross Site Scripting (XSS)** in clients that choose to display the response without any prior sanitization.

It is worth noting that since the audit's scope did not include a client which was vulnerable to XSS, this finding is for informational purposes only. This, however, does not mean that the response is not displayed, or will not be displayed in the future, on a web page, making it a potential attack vector. In this case, the security posture of the client application rests on that application's developers themselves.

Code Location:

An example of vulnerable code rests on the back-end gateway, in the handler for the sui_getEventsByRecipient type of request.

Screenshots/Videos:

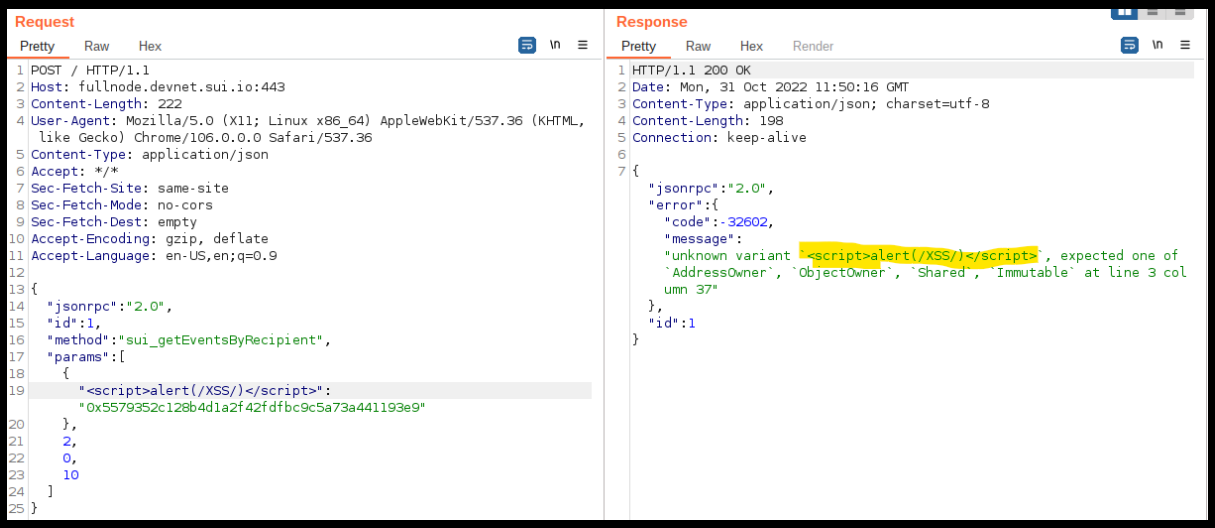


Figure 3: Unsanitized HTTP Response returned from the back-end gateway in an error message

Risk Level:

Likelihood - 1
Impact - 2

CVSS Vector:

- AV:N/AC:H/PR:H/UI:R/S:U/C:L/I:N/A:N

Recommendation:

It is recommended to sanitize all user input that at some point will be reflected onto a page on the web browser, even in generated error messages that are passed down in the HTTP response.
A good way to do this is to ensure that all such input are HTML-entity encoded before being reflected in a server response.

Reference:

Cross Site Scripting Prevention Cheat Sheet

Remediation Plan:

ACKNOWLEDGED: The Mysten Labs team acknowledged this finding.

3.5 (HAL-05) APPLICATION ERROR MAY DISCLOSE SENSITIVE TECHNOLOGY INFORMATION - INFORMATIONAL

Description:

During the audit, Halborn discovered that the application at https://fullnode.devnet.sui.io:443 generates error messages that include sensitive information about its environment, technologies, or associated data.

Some error messages specifically hint to the fact that the back-end server possibly uses TypeScript and the NodeJS engine, and in some cases the full error stack is returned, revealing the use of various Rust libraries in the back-end.

Screenshots/Videos:

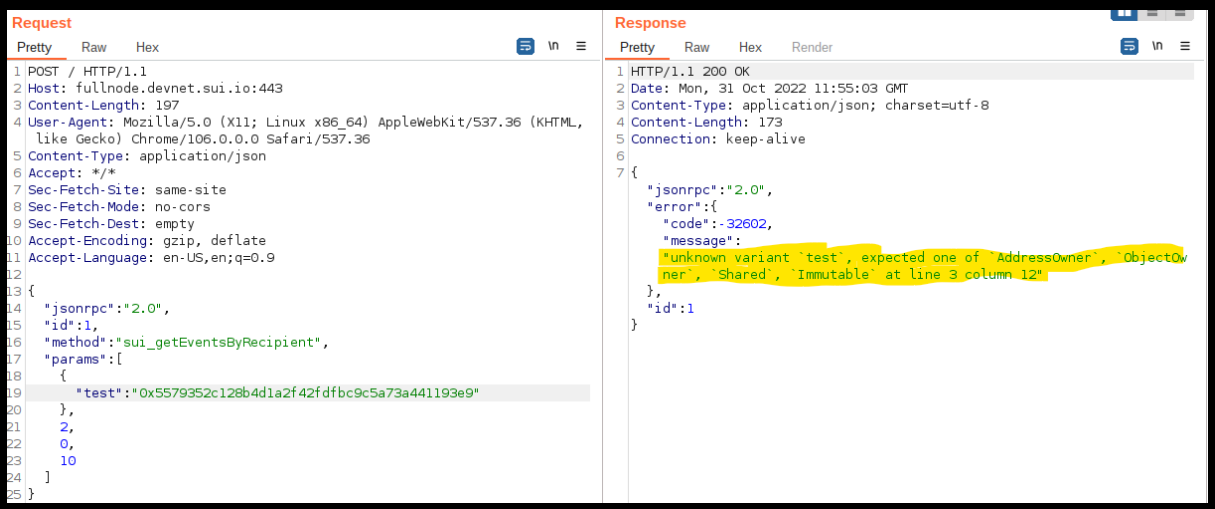
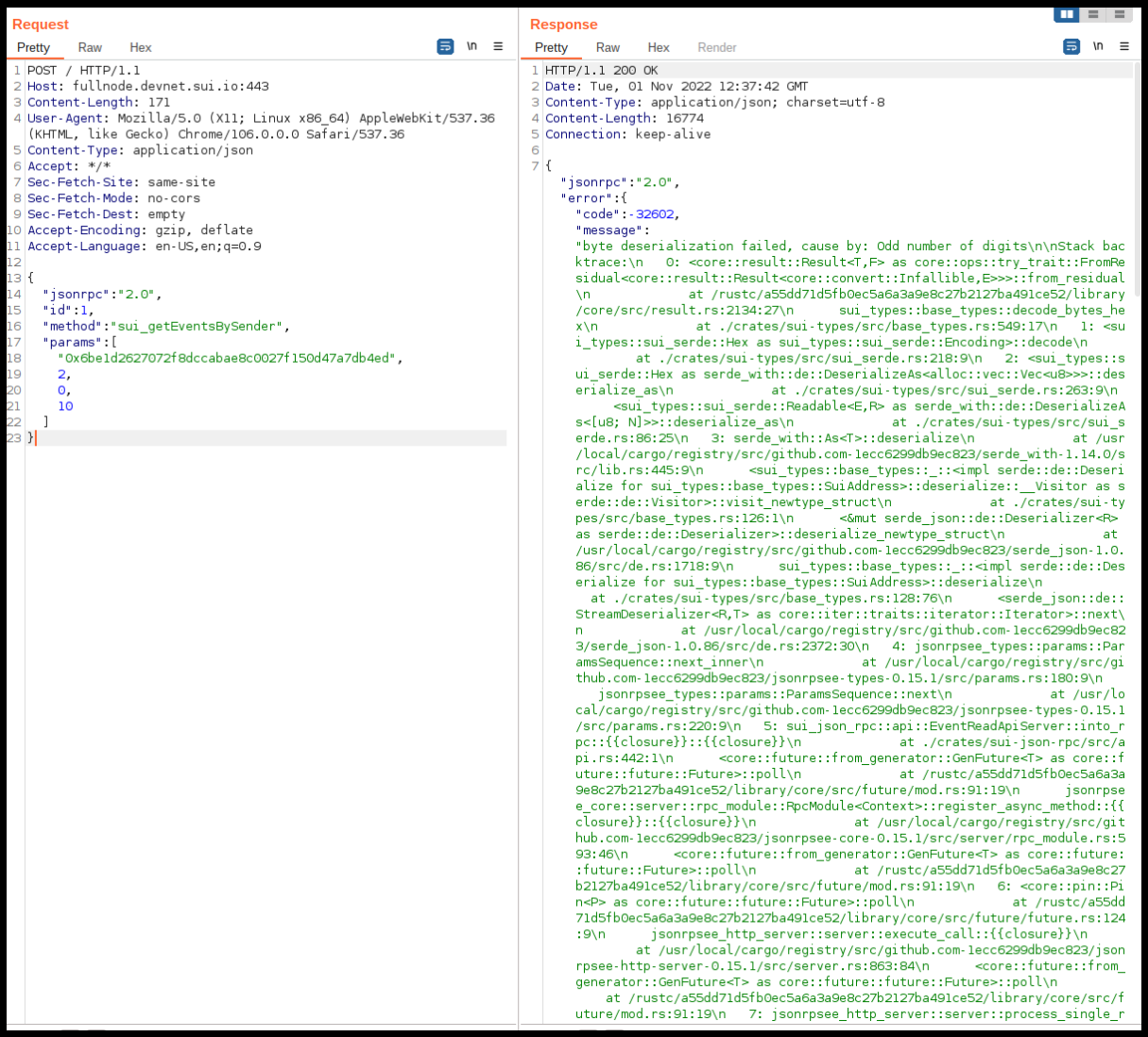


Figure 4: A custom error from NodeJS



Request

```
POST / HTTP/1.1
Host: fullnode.devnet.sui.io:443
Content-Length: 171
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36
Content-Type: application/json
Accept: */*
Sec-Fetch-Site: same-site
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: empty
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

{
  "jsonrpc":"2.0",
  "id":1,
  "method":"sui_getEventsBySender",
  "params":[
    "0x6be1d2627072f8dccabae8c0027f150d47a7db4ed",
    2,
    0,
    10
  ]
}
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 01 Nov 2022 12:37:42 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 16774
Connection: keep-alive

{
  "jsonrpc":"2.0",
  "error":{
    "code":-32602,
    "message":
    "byte deserialization failed, cause by: Odd number of digits\n\nStack bac
    ktrace:\n   0: <core::result::Result<T,F> as core::ops::try_trait::FromRe
    sidual<core::result::Result<core::convert::Infallible,E>>>::from_residual
    \n             at /rustc/a55dd71d5fb0ec5a6a3a9e8c27b2127ba491ce52/library
    /core/src/result.rs:2134:27\n      sui_types::base_types::decode_bytes_he
    x\n             at ./crates/sui-types/src/base_types.rs:549:17\n   1: <su
    i_types::sui_serde::Hex as sui_types::sui_serde::Encoding>::decode\n
           at ./crates/sui-types/src/sui_serde.rs:218:9\n   2: <sui_types::s
    ui_serde::Hex as serde_with::de::DeserializeAs<alloc::vec::Vec<u8>>>::des
    erialize_as\n             at ./crates/sui-types/src/sui_serde.rs:263:9\n
          <sui_types::sui_serde::Readable<E,R> as serde_with::de::DeserializeA
    s<[u8; N]>>::deserialize_as\n             at ./crates/sui-types/src/sui_s
    erde.rs:86:25\n   3: serde_with::As<T>::deserialize\n             at /usr
    /local/cargo/registry/src/github.com-1ecc6299db9ec823/serde_with-1.14.0/s
    rc/lib.rs:445:9\n      <sui_types::base_types::_::<impl serde::de::Deseri
    alize for sui_types::base_types::SuiAddress>::deserialize::__Visitor as s
    erde::de::Visitor>::visit_newtype_struct\n             at ./crates/sui-ty
    pes/src/base_types.rs:126:1\n      <&mut serde_json::de::Deserializer<R>
    as serde::de::Deserializer>::deserialize_newtype_struct\n             at
    /usr/local/cargo/registry/src/github.com-1ecc6299db9ec823/serde_json-1.0.
    86/src/de.rs:1718:9\n      sui_types::base_types::_::<impl serde::de::Des
    erialize for sui_types::base_types::SuiAddress>::deserialize\n
      at ./crates/sui-types/src/base_types.rs:128:76\n      <serde_json::de::
    StreamDeserializer<R,T> as core::iter::traits::iterator::Iterator>::next\
    n             at /usr/local/cargo/registry/src/github.com-1ecc6299db9ec82
    3/serde_json-1.0.86/src/de.rs:2372:30\n   4: jsonrpsee_types::params::Par
    amsSequence::next_inner\n             at /usr/local/cargo/registry/src/gi
    thub.com-1ecc6299db9ec823/jsonrpsee-types-0.15.1/src/params.rs:180:9\n
       jsonrpsee_types::params::ParamsSequence::next\n             at /usr/lo
    cal/cargo/registry/src/github.com-1ecc6299db9ec823/jsonrpsee-types-0.15.1
    /src/params.rs:220:9\n   5: sui_json_rpc::api::EventReadApiServer::into_r
    pc::{{closure}}::{{closure}}\n             at ./crates/sui-json-rpc/src/a
    pi.rs:442:1\n      <core::future::from_generator::GenFuture<T> as core::f
    uture::future::Future>::poll\n             at /rustc/a55dd71d5fb0ec5a6a3a
    9e8c27b2127ba491ce52/library/core/src/future/mod.rs:91:19\n      jsonrpse
    e_core::server::rpc_module::RpcModule<Context>::register_async_method::{{
    closure}}::{{closure}}\n             at /usr/local/cargo/registry/src/git
    hub.com-1ecc6299db9ec823/jsonrpsee-core-0.15.1/src/server/rpc_module.rs:5
    93:46\n      <core::future::from_generator::GenFuture<T> as core::future:
    :future::Future>::poll\n             at /rustc/a55dd71d5fb0ec5a6a3a9e8c27
    b2127ba491ce52/library/core/src/future/mod.rs:91:19\n   6: <core::pin::Pi
    n<P> as core::future::future::Future>::poll\n             at /rustc/a55dd
    71d5fb0ec5a6a3a9e8c27b2127ba491ce52/library/core/src/future/future.rs:124
    :9\n      jsonrpsee_http_server::server::execute_call::{{closure}}\n
           at /usr/local/cargo/registry/src/github.com-1ecc6299db9ec823/json
    rpsee-http-server-0.15.1/src/server.rs:863:84\n      <core::future::from_
    generator::GenFuture<T> as core::future::future::Future>::poll\n
        at /rustc/a55dd71d5fb0ec5a6a3a9e8c27b2127ba491ce52/library/core/src/f
    uture/mod.rs:91:19\n   7: jsonrpsee_http_server::server::process_single_r
```

Figure 5: A full-stack trace showing a deserialization error and Rust libraries being used

Risk Level:

Likelihood - 1
Impact - 2

CVSS Vector:

- AV:N/AC:H/PR:H/UI:N/S:U/C:N/I:L/A:L

Recommendation:

It is recommended to always return a generic error message to end users, to avoid any possibility of disclosing potentially critical information about the technologies or environment related to the application.

Reference:

CWE-209: Generation of Error Message Containing Sensitive Information

Remediation Plan:

NOT APPLICABLE: While relevant in a closed-source context, there is no risk for the TypeScript SDK. As the project is open source, there is no risk in revealing that JavaScript and Rust are in use on the server.

AUTOMATED TESTING

Description:

Part of the assessment was Static Code Analysis, which Halborn performed using the NodeJSScan tool. NodeJSScan is a Static security code scanner (SAST) specially built for Node.js applications.

Results:

NodeJSScan only discovered informational issues which do not pose any risk for the Sui TypeScript SDK, or they do not apply for an SDK.

Figure 6: Statistics of issues discovered by NodeJSScan

The following table shows the vulnerabilities discovered by NodeJSScan:

Title	Severity
ANTI CSRF CONTROL	INFO
HELMET HEADER CHECK CROSSDOMAIN	INFO
HELMET HEADER XSS FILTER	INFO
HELMET HEADER NOSNIFF	INFO
HELMET HEADER HSTS	INFO
HELMET HEADER REFERRER POLICY	INFO
HELMET HEADER CHECK CSP	INFO
HELMET HEADER FEATURE POLICY	INFO
HELMET HEADER FRAME GUARD	INFO
HELMET HEADER CHECK EXPECT CT	INFO
HELMET HEADER IENOOPEN	INFO
HELMET HEADER X POWERED BY	INFO
HELMET HEADER DNS PREFETCH	INFO
RATE LIMIT CONTROL	INFO

THANK YOU FOR CHOOSING

// HALBORN