Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Pentest-Report Tuum MetaMask Identify 07.2023

Cure53, Dr.-Ing. M. Heiderich, M. Pedhapati, L. Herrera

Index

Introduction

> *"Identify Snap is a sophisticated plugin that developers can incorporate into a variety of applications to enhance MetaMask's functionality beyond its native capabilities. Specifically, Identify Snap extends MetaMask's features by adding support for Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs)."*

From https://docs.tuum.tech/identify/basics/introduction

This report describes the results of a security assessment of the Tuum MetaMask Identify complex and its underlying codebase. The project, which included a dedicated source code audit and a review of the feature-set, was carried out by Cure53 in July 2023.

Registered as *TUU-01*, the examination was requested by Tuum Technologies, Inc., in June 2023 and then scheduled to start almost right away in the following month Although this was the first collaboration between Cure53 and Tuum, sufficient time was available for preparations on both sides.

In terms of the exact timeline and specific resources allocated to *TUU-01*, Cure53 completed the research in July 2023, more precisely in CW27 and CW28 of 2023. In order to achieve the expected coverage for this task, a total of eight days were invested. In addition, it should be noted that a team of three senior testers was formed and assigned to preparation, execution, and delivery of this project.

For optimal structuring and tracking of tasks, the examination was split into two separate work packages (WPs):

- **WP1**: Source code audits against Tuum MetaMask Identify & codebase
- **WP2**: Code audits & feature reviews of Tuum MetaMask Identify & codebase

As the titles of the WPs indicate, white-box methodology was utilized. Cure53 was provided with documentation, as well as all further means of access required to complete the tests. Additionally, all sources corresponding to the test-targets were shared to make sure the project can be executed in line with the agreed-upon framework.

Overall, the project progressed effectively. To facilitate a smooth transition into the testing phase, all preparations were completed in late June and early July 2023, specifically in the week preceding the testing period.

Throughout the engagement, communications were conducted via a private, dedicated and shared Slack channel. Stakeholders - including the Cure53 testers and the internal staff from Tuum - could participate in discussions in this space.

The quality of the interactions throughout the test was excellent, with no outstanding queries. These steady exchanges contributed positively to the overall outcomes of this project. The scope was well prepared and clear, which played a major role in avoiding significant roadblocks during the test. Cure53 offered frequent status updates about the test and the emerging findings, but no live reporting was requested for this assignment.

The Cure53 team succeeded in achieving very good coverage of the WP1-WP2 scope items. Only four findings were documented as part of *TUU-01.* Among them, three were classified as security vulnerabilities and one was categorized as a general weakness with lower exploitation potential. Both the limited number of security-relevant flaws, and the fact that none of the findings exceeded *Medium* scores in terms of the assigned risk-factors, contribute to the positive outcome of this project.

The following sections first describe the scope and key test parameters, as well as how the WPs were structured and organized. Next, all findings are discussed in grouped vulnerability and miscellaneous categories. Flaws assigned to each group are then discussed chronologically. In addition to technical descriptions, PoC and mitigation advice will be provided where applicable.

The report closes with drawing broader conclusions relevant to this July 2023 project. Based on the test team's observations and collected evidence, Cure53 elaborates on the general impressions and reiterates the verdict. The final section also includes tailored hardening recommendations for the Tuum MetaMask Identify complex.

Scope

- **Code audits & security reviews of Tuum MetaMask Identify & codebase**
    - **WP1**: Source code audits against Tuum MetaMask Identify & codebase
        - **Primary focus:**
            - General tests & attacks against browser add-ons, extension Snap-Ins, independent of the specific use-case of the complex as a crypto wallet snap.
        - **Source:**
            - https://github.com/tuum-tech/identity-snap/tree/v1.3.2/packages/snap
        - **Documentation:**
            - https://docs.tuum.tech/identify/basics/introduction
    - **WP2**: Code audits & feature reviews of Tuum MetaMask Identify & codebase
        - **Primary focus:**
            - Identity and credential management features
            - DID management features
            - VC and VP handling
            - Possible information leaks via Snap features & vulnerabilities
            - Authn & login
        - **Sources and documentation:**
            - *See WP1*
    - **Test-supporting material was shared with Cure53**
    - **All relevant sources were shared with Cure53**

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, all tickets are given a unique identifier (e.g., *TUU-01-001*) to facilitate any future follow-up correspondence.

TUU-01-002 WP2: DApp origin not displayed in the Snap UI (*Medium*)

***Fix note**: This issue has been mitigated by the Tuum Technologies team. Cure53 has positively verified the deployed fix.*

No origin is displayed in any of the Snap's dialogs to indicate which dApp is requesting permission to perform a given action. This is problematic because it could facilitate a situation of a malicious dApp taking over and performing actions from other, trusted dApps. It was also discovered that RPC calls can be initiated from within third-party iframes embedded inside the victim's page. These include - but are not limited to - *createVP* requests, which are used during the login flow.

Furthermore, this issue is considered more severe since many websites serve ads from third-party iframes despite the fact that they do not control their contents. If adversaries were able to serve an advert on a domain that is making use of the Identify, they would be able to trigger RPC calls. From there, it could be imagined that tricking users could be successful, given that no origin is displayed in the Snap's UI. Thus, it would be credible to assume that the request originated from a trusted website.

To mitigate this issue, Cure53 recommends creating a default template which should be included in all dialogs utilized by the Snap component. This needs to clearly display the origin that performed the RPC call. Be advised that the origin is available in the *onRpcRequest* handler.

TUU-01-003 WP2: No user confirmation dialog in GDrive configuration (*Low*)

***Fix note**: This issue has been mitigated by the Tuum Technologies team. Cure53 has positively verified the deployed fix.*

The *configureGoogleAccount* RPC method is utilized to change the *googleAccessToken* from the currently connected account. However, no user confirmation is prompted to the requesting user before a successful operation.

The token in question is then used by the *googleDriveVCStore* to import and export the user's VCs data to their Google Drive, specifically for the purpose of using an application from another browser or device.

By exploiting the existing weakness, a malicious dApp that was already granted the required permissions, would be able to change the user's *access* token and replace it with the attacker's one. Next, when the user attempts to export their VCs, they will end up uploading their data to the attacker's Google Drive.

Affected file:
/identity-snap-1.3.2/packages/snap/src/rpc/gdrive/configureGoogleAccount.ts

Affected code:

```
export const configureGoogleAccount = async (
  identitySnapParams: IdentitySnapParams, { accessToken }: GoogleToken,
) => {
  const { snap, state, account } = identitySnapParams;
  try {
    await verifyToken(accessToken);
    const coinType = await getCurrentCoinType();
    state.accountState[coinType][
      account.evmAddress
    ].accountConfig.identity.googleAccessToken = accessToken;
    await updateSnapState(snap, state);
    return true;
  } catch (error) {
    throw error;
  }
};
```

Steps to reproduce:

1. Make sure the MetaMask extension and the Identify are installed.
2. Start the *Example* site from Tuum Tech's *identity-snap* repository and connect the MetaMask account *Step 1* to this item.
3. Access the MetaMask extension and extract wallet address.
4. Open the DevTools and execute the JavaScript below. Remember to replace the *{{metamask_address}}* variable with the value extracted in *Step 3*.

   Command:

   ```
   postMessage({"target":"metamask-contentscript","data":{"name":"metamask-
   provider","data":{"method":"wallet_invokeSnap","params":
   {"snapId":"local:http://localhost:8080","request":
   {"method":"configureGoogleAccount","params":
   {"metamaskAddress":"{{metamask_address}}",
   "accessToken":"1337"}}},"jsonrpc":"2.0","id":1}}}, "*");
   ```

To mitigate this issue, Cure53 recommends requesting the permission of the user via the *snapDialog* function. Furthermore, similarly to TUU-01-004, the dialog should display the user's email address associated with the *access* token. The user must be able to make an informed decision on whether they want to configure their account as connected to the specified Google Drive.

TUU-01-004 WP2: No user-email displayed in *sync* dialog for GDrive (*Medium*)

***Fix note**: This issue has been mitigated by the Tuum Technologies team. Cure53 has positively verified the deployed fix.*

The *syncGoogleVCs* RPC method, which is responsible for syncing the users' VCs stored locally with their Google Drives, does not display the email associated with the user's *access* token in the *confirmation* dialog UI.

This is problematic because a malicious dApp can change the Google Drive account in use via the issue described in ticket TUU-01-003. This would be done without any user interaction. During subsequent *sync* attempts, the user might be tricked into importing or exporting their VCs data to an attacker-controlled Google Drive account.

Steps to reproduce:

1. Ensure that the MetaMask extension and the Identify are installed.
2. Start the *Example* site from Tuum Tech's *identity-snap* repository and connect it to the MetaMask account from *Step 1*.
3. Access the MetaMask extension and extract wallet address.
4. Configure the connection between a Google Drive account and the Identify.
5. Open the DevTools and execute the JavaScript below. Make sure to replace the *{{metamask_address}}* variable with the value extracted in *Step 3*.

Command:

```
postMessage({"target":"metamask-contentscript","data":{"name":"metamask-provider","data":{"method":"wallet_invokeSnap","params":{"snapId":"local:http://localhost:8080","request":{"method":"syncGoogleVCs","params":{"metamaskAddress":"{{metamask_address}}"}}},"jsonrpc":"2.0","id":1}}},"*");
```

To mitigate this issue, Cure53 recommends requesting the additional *https://www.googleapis.com/auth/userinfo.email* permission during Google's OAuth flow when retrieving a valid *access* token with permission to write and read the user's Google Drive account. Afterward, a request to *https://www.googleapis.com/oauth2/v1/userinfo?access_token=* should be made to retrieve the user's email address. This email address should be displayed in the *confirmation* dialog upon externally syncing the user's VCs.

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, whilst a vulnerability is present, an exploit may not always be possible.

TUU-01-001 WP1: Newlines handled incorrectly in Snap UI *(Low)*

***Fix note**: This issue has been mitigated by the Tuum Technologies team. Cure53 has positively verified the deployed fix.*

The Identify lacks optimal handling for newlines that appear in some of the Snap dialogs within the MetaMask extension. This facilitates spoofing of messages for certain dialogs, which could be used to trick users into performing certain actions on behalf of the Snap component.

The impact of this problem was appropriately downgraded to *Low* as attacks are only limited to phishing attempts.

Affected file:
/identity-snap-1.3.2/packages/snap/src/snap/account.ts

Affected code:

```
const dialogParamsForPrivateKey: SnapDialogParams = {
  type: 'prompt',
  content: panel([
    heading('Connect to EVM Account'),
    text('Enter your ECDSA private key for the following Account'),
    divider(),
    text(`EVM Address: ${evmAddress}`),
  ]),
  placeholder: '2386d1d21644dc65d...',
};
privateKey = PrivateKey.fromString(
  (await snapDialog(snap, dialogParamsForPrivateKey)) as string,
).toStringRaw();
```

Steps to reproduce:

1. Have the MetaMask extension and the Identify installed.
2. Start the *Example* site from Tuum Tech's *identity-snap* repository and connect with the MetaMask account from *Step 1.*
3. Open the DevTools and execute the JavaScript below.

Command:

```
postMessage({"target":"metamask-contentscript","data":{"name":"metamask-
provider","data":{"method":"wallet_invokeSnap","params":
{"snapId":"local:http://localhost:8080","request":
{"method":"getAccountInfo","params":
{"metamaskAddress":"0x1","externalAccount":
{"blockchainType":"evm","data":{"address":"1337\r\n\r\nMessage from 
Identity Snap:\r\n\r\n\r\n\r\nBefore proceeding, you have to access 
https://attacker.com and confirm your private key due to 
2FA."}}}}},"jsonrpc":"2.0","id":1}}})
```

To mitigate this issue, Cure53 recommends stripping newlines from user-input before creating dialogs using the *snapDialog* function.

Conclusions

Note: All of the aforementioned unveiled issues and vulnerabilities have been correctly mitigated and fixed by the Tuum Technologies team within in v1.4.2 of the MetaMask Identify[1]. The Cure53 team was able to review and fully verify the applied fixes.

Cure53 can conclude that the Tuum MetaMask Identify boasts a very solid security posture. In light of only four findings stemming from this July 2023 project, the Tuum team can be congratulated on their successful efforts towards providing robust protection mechanisms for their components. Nevertheless, it is still recommended to swiftly and properly resolve all of the unveiled security-relevant issues in order to further strengthen the Tuum MetaMask Identify and its underlying codebase.

In the frames of *TUU-01,* the available codebase was subjected to rigorous review. The Cure53 testing team leveraged multiple compromise techniques, which verified the strong efficacy of Tuum in minimizing the attack surface. The limited number of the identified issues clearly indicates the development team's successful integration of abundant precautionary measures into the Identify project. Ample evidence - including the fact that none of the issues exceeded *Medium* risk scores - suggests that the Tuum Technologies team is not only aware of common security errors, but has also taken proactive steps to mitigate and prevent them. The testing team started by auditing the Snap *Manifest* file for insecure configuration patterns, including overly lax permissions. It was corroborated that Snap only requests the required permissions, which is considered best practice and limits the attack surface.

Despite the highly constrained attack surface, the Cure53 team evaluated every exposed RPC method accessible. This was done in an attempt to uncover risk-inducing behaviors, resulting in one minor issue identified in the *getAccountInfo* method. Due to a lack of proper sanitization, newlines could be utilized to spoof the message displayed in the Snap dialog, as clarified in TUU-01-001. Additionally, strict parameter validation is being enforced before all the RPC calls. Among other aspects, this includes type-checks. As such, secure coding pattern preemptively prevents a range of common mistakes from occurring.

The testing team placed particular emphasis on issues that would cause private key leakage or other information leaks. Attention was thus given to the *googleDriveVCStore* and *snapDataStore* functionalities, since they are employed for storing user-data. This led to the discovery of two issues related to the Google Drive integration.

[1] https://github.com/tuum-tech/identify/tree/v1.4.2

Firstly, the Google Drive configuration step was found to lack the necessary user confirmations (TUU-01-003). As such, it was not leveraged to prevent malicious dApps from switching the user's Google Drive account to a malicious one.
Secondly, during the syncing process, it was noted that the *confirmation* dialog lacks the indication of which account the user-data is being pulled from or sent to (see TUU-01-004 for more details). Under specific circumstances, this could lead to information leakage.

The *jsonpath* library, which is used to perform complex queries within the data store, was also extensively analyzed for issues connected to its direct manipulation of user-data. Fortunately, it was not possible to detect any issues. Several tests that were performed by Cure53, including attempts for information leakage via DoS, failed. In regard to storage, the utilization of *snap_manageState* to store the user-data was deemed to be a good solution. This is because the stored contents are automatically encrypted using a Snap-specific key, alongside being automatically decrypted when retrieved.

Additional checks for issues associated with the interaction between authorized but malicious dApps and the Snap components were completed as well. All attempts failed and were prevented by MetaMask's security checks. Next, checks for path traversal and open redirect issues in functions performing external requests were reviewed, as the ability to perform arbitrary or partial requests on behalf of Snap would be problematic. Once again, proper steps were taken by Tuum to prevent issues of this nature.

The testing team also looked for issues in the *login* implementation provided by the Tuum Technologies team (DID Auth) using the Snap. No issues specific to this feature were uncovered, although an attack scenario was found to also affect the *login* flow in the context of Cure53 moving on to the Snap UI components. Due to the lack of the dApp origin being displayed in the Snap dialogs, a malicious dApp could trick users into performing actions on behalf of a trusted dApp (see TUU-01-002).

Following the completion of this security audit, Cure53 garnered a positive impression regarding the security offering established by the Identify. This viewpoint is further corroborated by the low number of vulnerabilities noted by three members of the Cure53 testing team, as well as the complete absence of any *Critical*- or *High*-severity problems on the scope. This evidently indicates adequate nullification of highly damaging attack vectors. An even better security standard can be attained by effectively addressing and mitigating the findings presented in this *TUU-01* report. The Tuum Technologies team has already laid a commendable groundwork to incorporate future development enhancements and booster security standing of MetaMask Identify solutions.

Cure53 would like to thank Kiran Pachhai, and Donald Bullers from the Tuum Technologies, Inc. team for their excellent project coordination, support and assistance, both before and during this assignment.