Fine penetration tests for fine websites

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

Pentest-Report BitNote Web, Infra & Crypto 01.2024

Cure53, Dr.-Ing. M. Heiderich, Dr. M. Conde Pena, Dr. D. Bleichenbacher, MSc. R. Peraglie

Index

Introduction

> *"BitNote allows you to store encrypted text notes directly on a blockchain. Ultimately BitNote allows you to self-custody information, in a similar way that blockchains let you self-custody money."*

From https://bitnote.xyz/

This report, assigned the unique reference ID RVE-01, has been compiled following the completion of a Cure53 penetration test and source code audit against the BitNote web application.

The project originated from initial discussions with Rockwell Ventures management in October 2023. Following confirmation, the review was scheduled for CW03 January 2024 and fulfilled by four vastly experienced technicians from the Cure53 talent pool. For optimal coverage, the client invested twelve working days for the analyses.

Two distinct Work Packages (WPs) were created to separate the two core focus elements. These read as follows:

- **WP1**: White-box pentests & source code audits against BitNote web UI & infra
- **WP2**: White-box pentests & source code audits against BitNote web crypto

A suite of supporting materials were handed over in advance to facilitate the undertakings, such as sources, URLs, documentation, test-user accounts, and other miscellaneous assets. These were also referred to while completing the required preparations, which were finalized in the week before the active review window (CW02) to grant a seamless start. The access to scope-specific sources meant that the assignment adhered to a white-box pentesting methodology.

Communications between the two organizations were facilitated through the establishment of a dedicated and private Slack channel. Hence, all participating team members were able to engage in open conversations regarding progress, findings, issues, etc. This discourse in combination with the ideal scope preparation was conducive to an effective and hindrance-free pentest. Pertinently, live reporting was also conducted for this exercise by using the aforementioned channel.

Cure53 detected nine findings after achieving extensive coverage over the scope elements defined in the two WPs. Of those, five represented security vulnerabilities and four pertained to best practice hardening or minor faults. This total is generally considered modest and commendable for an inaugural audit, reflecting favorably on the scope's security health.

However, the robustness of the two WPs fluctuate substantially. The BitNote web application and its underlying infrastructure (WP1) was affected by two *Critical* persistent XSS vulnerabilities (highlighted in tickets RVE-01-001 and RVE-01-002) that require immediate remediation with utmost priority to ensure a safe user experience. On the other hand, the BitNote web cryptography reviews (WP2) yielded low impact results, with only a minor severity vulnerability and a few fortification opportunities identified in this area.

All in all, one can only conclude that the BitNote web application requires enhancement in order to be sufficiently secured prior to going live. Conversely, the BitNote cryptography has been soundly implemented and requires minimal development initiatives on the whole. Nevertheless, it is recommended that the BitNote application and cryptography are regularly tested in order to ensure that all newly rolled out features and versions are safeguarded to the same degree.

Onward, the *Scope* section next outlines the software components examined and methodologies employed. Next, the report systematically itemizes all *Identified Vulnerabilities* and Miscellaneous Issues; notably, these are presented in order of detection rather than severity rating. Each finding attaches a clear technical explanation, a Proof-of-Concept (PoC) where applicable, and actionable mitigation recommendations, enabling the developer team to understand the risks and address them promptly. To close, the *Conclusions* section offers a critical analysis of BitNote's perceived security posture, highlighting strengths, weaknesses, and final viewpoints concerning the scope’s key areas.

Scope

- **Pentests & code audits against BitNote web UI, backend & web crypto**
    - **WP1**: White-box pentests & source code audits against BitNote web UI & infra
        - **URL (beta environment):**
            - https://beta.bitnote.xyz/
        - **Sources:**
            - https://beta.bitnote.xyz/pages/js/ww.js
            - https://beta.bitnote.xyz/sw.js
            - new_note.js
            - sign_up.js
    - **WP2**: White-box pentests & source code audits against BitNote web crypto
        - **Encryption:**
            - *See sources of WP1*
        - **Smart contracts:**
            - **Handling of account-creation-related string:**
                - https://beta.bitnote.xyz/pages/contracts/notes_contract.txt
                    - Address on fujinet is:
                        - https://subnets-test.avax.network/c-chain/address/0x18840c3ca5e6d8c9c4fb9379515d19ffcf53aa45
            - **Handling of encrypted-note-related strings:**
                - https://beta.bitnote.xyz/pages/contracts/mod_contract.txt
                    - Address on fujinet:
                        - https://subnets-test.avax.network/c-chain/address/0x59f88524cdddc712d327f60b8b68cb702abe3038
    - **Test-user credentials:**
        - U: audit_ac1
        - U: audit_ac2
        - U: audit_ac3
    - **Documentation:**
        - Product overview:
            - https://bitnote.xyz
        - External JS used:
            - https://github.com/paulmillr/noble-curves/blob/main/src/secp256k1.ts
            - https://beta.bitnote.xyz/pages/js/zxcvbn.js
    - **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., *RVE-01-001*) to facilitate any future follow-up correspondence.

RVE-01-001 WP1: Persistent XSS in blockchain via sharing (*Critical*)

***Fix note**: This issue was mitigated during the testing phase and fix-verified by Cure53.*

Cure53 verified that the title of a blockchain-stored secret note is embedded unsanitized directly into HTML markup before loading the HTML snippet into the DOM. Coupled with the absent Content Security Policy (CSP) reported in ticket RVE-01-003, this behavior allows arbitrary JavaScript execution in a window context with access to the unencrypted BitNote secret notes. Due to the major perceived risk, this ticket received the highest possible severity score, *Critical*.

The following excerpt was obtained from the main page's HTML markup, indicating that the *note_title* is embedded unsanitized directly into HTML markup via a JavaScript template string. This final string is later assigned to an HTML element's *innerHTML* property. As a result, an attacker can inject arbitrary malicious HTML markup into the shared note title in order to leverage JavaScript execution for the purpose of extracting and decrypting alternative secret notes.

Affected main page Javascript code:

```
function newNewNote(params){
    [...]
    var note_title=(checkForProperty(params.note_title)) ?
params.note_title:""
    return `[...]<div class=user_note_titlebox_container>[...]
                 <textarea id=user_note_titlebox [...]>$
{note_title}</textarea>
```

Steps to reproduce:

1. Create a note as the attacker with a title containing the following XSS payload:

   PoC:
   ```
   </textarea>cure53<img src=x onerror=alert(1)><textarea>
   ```

2. Click the three dots and select *Share note*, then forward the share link to the victim.
3. Visit the link as the victim, authenticate, and save the note on the blockchain.

4. The victim will now view the note URL, which should appear akin to the following:

   Note URL:
   https://beta.bitnote.xyz/audit_ac1/?0x018d0d72d027

5. Upon refreshing the web page or visiting the URL from Step 4 directly, one can confirm that the XSS payload triggers and displays an alert box.

To mitigate this issue, Cure53 advises sanitizing all user input prior to embedding it within HTML code by encoding the relevant HTML meta characters. This can be achieved by utilizing a renowned, tried-and-tested library such as React for all operations that construct and manipulate HTML markup. By employing the React library exclusively for these purposes, the Rockwell Ventures team can ensure that all user-input embedded into templates will be optimally sanitized by default. As a result, the developer team will be granted the ability to write small React templates with high audibility and security as standard.

RVE-01-002 WP1: Persistent XSS in note via export (*Critical*)

***Fix note**: This issue was mitigated during the testing phase and fix-verified by Cure53.*

Following the discovery of the vulnerability detailed in ticket RVE-01-001, the test team also confirmed that both the title and content of exported notes were embedded unsanitized in a local HTML file. Similarly, this allows attackers to inject HTML markup in shared notes in order to leverage JavaScript execution through the note content, in the event that the victim exports a shared malicious note.

Web page excerpt:

```
function getExportHTML(json_encrypted_notes, main_assets, keys){
    return `
<!DOCTYPE html><html>
  [...]
        <script>
        [...]
        function unlockManyNotesFromView(cb){ [...]
output_container.firstElementChild.innerHTML+="<tr><td>"+current_note.title
+"</td><td>"+current_note.decrypted+"</td></tr>";
```

To mitigate this issue, Cure53 recommends applying adequate sanitization for the dynamic variables prior to embedding them directly into the HTML markup. This could be implemented with a client-side templating engine such as React in order to render the encrypted notes into an HTML file template within a memory blob, which is subsequently offered as a downloadable file. The downloaded HTML template could again contain a script reference to the React library in order to decrypt the notes and display them to the user.

RVE-01-004 WP1: Insufficient master password policy *(Medium)*

Generally speaking, users are required to select a username and master password in order to create a BitNote account. Once this has been completed, a long-term P-521 EC key pair is generated and used to derive the note encryption keys. In addition, the private key component of the user's P-521 EC key pair is encrypted with a key derived from the master password, while the encrypted private key is stored on-chain.

Since the encrypted private key is publicly accessible, one can argue that the strength of the master password is more crucial to security than in other similar centralized setups, whereby a server breach would be required for a malicious party to retrieve access to contained information. However, extremely weak passwords (including those comprising a single character or simple dictionary word) are still accepted as master passwords despite an explicit user warning. Although the UI includes a utility to generate strong passwords, as well as an estimation of the selected master password's strength via *zxcvbn*, the current password policy is considered insufficient given the fatal effect that a compromised master password would incur over user accounts. Moreover, derivatives of the master password are easily available on-chain, as mentioned previously.

To mitigate this issue, Cure53 discourages permitting users to select weak master passwords[1]. Given that *zxcvbn* is already in use, passwords with an estimated strength lower than at 80 bits should be disallowed. In addition, it is also advisable to establish a minimum length for the password of at least 8 characters (ideally 12) to facilitate strong master password configurations, which will neutralize the probability of password leakages[2].

RVE-01-005 WP2: Transaction origin phishing attack on referral address *(Low)*

***Fix note:** This issue was mitigated after the testing phase and fix-verified by Cure53.*

Cure53 found that the notes smart contract uses the transaction's origin address by employing Solidity's *tx.origin* expression in order to specify a new referral for the address in question. Accordingly, adversaries can establish a malicious contract that overwrites the caller's referral address upon invocation.

This could be implemented in a payable fallback function that invokes the mod contract's *initAccount* function, which in turn forwards the *ref* argument unaltered to the notes contract's *setReferrer* method. The *tx.origin* address is related to the victim invoking the malicious contract, while the *ref* argument is attacker-controlled.

However, this attack vector can only target specific victims and requires them to invoke an attacker-controlled contract using the blockchain key pair dedicated to and generated by BitNote. As such, this ticket has been assigned a *Low* severity rating only.

[1] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf
[2] https://www.passcape.com/text/articles/rockyou_leaked_passwords.pdf

Affected contract:
notes-contract.sol

Affected code:
```
function setReferrer(address ref) external {
	require(msg.sender == mod_contract);
	refferal_addr[tx.origin] = ref;
}
```

To mitigate this issue, Cure53 suggests passing the caller address of the mod contract as a second argument to the *setReferrer* method and leveraging it in favor of the *tx.origin* address. By doing so, it is assured that only the referrer of the mod contract's immediate caller address can be overwritten. Henceforth, any phishing attack attempts would require the attacker to lure the victim into submitting a signed transaction to the mod contract directly with the dedicated BitNote blockchain account.

RVE-01-007 WP1: Full password decryption for biometric authentication *(Medium)*

***Note:** The biometric authentication feature was removed by BitNote until the proposed fix can be integrated, so the issue as reported by Cure53 no longer exists.*

Cure53 detected that the cryptography introduced within the biometric authentication feature utilizes predictable secrets in order to encrypt the master password stored within the web page's client-side storage. As a result, adversaries with physical access to the unlocked device can fully decrypt the master password without entering any biometric credentials.

Steps to reproduce:

1. As the victim, visit the BitNote login page using the Chrome browser, then register biometrics by entering the credentials and clicking *Use Biometric ID*.
2. Log out upon successful biometric registration.
3. As the attacker, visit the BitNote main page within the same browser session and execute the following JavaScript on the BitNote domain's login page (*F12->Developer Console->Copy & Paste JavaScript*):

   PoC:
   ```
   rp_id =
   [...atob("fjyO2XdGulgSum/oPJ5YqZlg16J51gVQrWxjve9d")].map(c=>c.charCo
   deAt(0))
   authData = new Uint8Array([...rp_id,0,0,0,0,0])
   creds = { response: {authenticatorData: authData, clientDataJSON:
   authData}}
   navigator.credentials.get = async _ => creds
   window.crypto.subtle.verify = async () => true
   pw_box = document.getElementsByClassName("signin_mp_c_input")[0]
   pw_box.setAttribute('type', 'text')
   Object.defineProperty(pw_box, "value", {
   ```

```
        get() { return this.getAttribute('value'); },
        set(value) { alert("The password is " + value);
        return this.setAttribute('value', value);
        },
});
```

4. Proceed by entering the username of the victim from Step 2 and click *Use Biometrics ID*.
5. Verify that the user's password is now displayed in an alert box and within the password input field.

To mitigate this issue, Cure53 advises implementing a two-pronged remediation approach. Firstly, the BitNote team should avoid storing the master password directly and alternatively adopt the PBKDF2 function output, which represents the derived encryption key required to decrypt the blockchain and ECDH private keys. This revised approach will mean that any attackers successful in breaking the encryption will no longer be able to retrieve the master password, directly neutralizing cross-service attacks.

In addition, one can recommend cryptographically protecting the confidentiality of the encryption key by encrypting it with key material derived from secrets stored on the biometric authenticator. This could be implemented by deriving the key material from the output of a Pseudo-Random Function (PRF) offered by the authenticator, which is accessible via the *prf* extension of the WebAuthn specification[3]. By doing so, the key material can only decrypt the encryption key if the biometric authenticator grants access to the key material.

[3] https://w3c.github.io/webauthn/#prf-extension

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, while a vulnerability is present, an exploit may not always be possible.

RVE-01-003 WP1: Absent Content Security Policy *(Medium)*

Fix note: *This issue was mitigated after the testing phase and fix-verified by Cure53.*

Testing verified that the application’s HTML markup is served without a Content Security Policy (CSP), which facilitates exploiting the vulnerabilities outlined in RVE-01-001 and RVE-01-002, since inline JavaScript is permitted.

To mitigate this issue, Cure53 strongly recommends integrating a sufficient CSP as an HTML tag to the served HTML markup in order to only allow scripts with a white-listed checksum of a cryptographically secure hash function[4], as indicated in the below example. By doing so, only white-listed JavaScript code can be executed, preventing trivial secret extraction in the event of an HTML injection. In addition, one can advise imposing enhanced restrictions on the other integrated CSP implementations in order to nullify advanced extraction techniques. Notably, a CSP within an HTML *<meta>* tag can also be applied to the downloaded HTML file containing the extracted secret notes.

Example CSP:

```
default-src 'self'; script-src
'sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC'
```

RVE-01-006 WP2: Security non-reinstatable post-MP compromise *(Info)*

By design, the private key component of a user’s EC P-521 keypair is stored on-chain, encrypted with a key that is derived from the user’s master password. In the event that a user’s master password is compromised, the private key can be recovered and all user notes created to date should be considered compromised.

In this regard, if a user’s master password is compromised (which would be fatal and necessitates strict password policy enforcement, as proposed in ticket RVE-01-004), the security of the user’s BitNote account cannot be re-established. Specifically, newly-created notes, as well as notes that the user chooses to migrate by editing and saving them, will remain insecure. This owes to the absence of a BitNote mechanism serving to amend the account’s master password and re-generate a new EC P-521 key pair, which would effectively reinstate the security of the aforementioned note types. Following cross-team conversations, the confirmation was made that the client is proposing to address this concern in the future by offering an option to import notes into a new account.

[4] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cont[...]icy/script-src#white[...]_hashes

Albeit, the user would be required to issue an additional payment in order to register a new account.

To mitigate this issue, Cure53 recommends implementing a mechanism that allows the user to re-establish the security of a compromised account and continue using the service, assuming that the content of past notes is compromised and requires migration. This can be accomplished by either allowing the user to modify the master password and refresh the EC P-521 key or by integrating an import functionality that permits importing notes into other accounts.

RVE-01-008 WP2: Side-channel attack hardening guidance *(Low)*

Cure53 analyzed the resilience of the implemented cryptographic primitives with regards to side-channel attacks, particularly focusing on compromises based on timing differences. Here, the team can confirm that substantial timing differences are avoided; in particular, scalar multiplication uses a constant number of point additions. However, some smaller timing leaks may still be possible, as discussed next.

The implementation of ECDSA uses BigInt for arithmetic operations over the underlying finite field, while the implementation of the modular inversion employs a non-constant-time algorithm. The author of the underlying library is generally aware of timing attacks and claims that measurements demonstrate insignificant timing leaks. With this, the neutralization of only substantial timing leaks is essentially a reactive approach; Cure53 recommends enforcing a more proactive strategy in this situation.

The library’s author argues that countermeasures against side-channel attacks could introduce vulnerabilities, which is valid to a certain extent. However, a number of simple countermeasures can be incorporated, such as randomizing the coordinates of projective and Jacobian points[5] as proposed by Coron, for example. This does not remove all timing differences, but rather ensures timing differences remain independent of secret information. Coron’s proposal has been criticized due to the fact that some potential cases are not taken into consideration[6]; however, the proposed attack is only significant for ECDH implementations.

The current implementation leverages non-deterministic ECDSA signatures via additional entropy. It should be noted that this randomization renders timing attacks increasingly challenging to perform, since it prevents the retrieval of precise timing information by repeating the same measurement multiple times. Hence, this ticket’s impact score has been downgraded to *Low*. To mitigate this issue, Cure53 advises incorporating appropriate countermeasures through library updates rather than a quick patch.

[5] https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4d5d6dfdb582c[...]9039#page=10
[6] http://download.mmag.hrz.tu-darmstadt.de/media/FB20/Dekanat/Publikationen/CDC/TI-03-01.zvp.pdf

RVE-01-009 WP2: Note sender/receiver unbound to share links *(Low)*

***Fix note:** This issue was mitigated after the testing phase and fix-verified by Cure53.*

While analyzing the feature that permits inter-user note sharing, Cure53 noted that the unique user identifiers acting as the sender and receiver of the shared notes are not authenticated. This enables an attack strain whereby an adversary intercepting a share link from user *audit_ac1* to user *audit_ac2* can appropriately craft a valid share link that they can send to *audit_ac1*, essentially impersonating *audit_ac2* in this particular action.

In particular, when user *audit_ac1* shares a note with user *audit_ac2*, a share link of the following form is generated:

Share link:
*beta.bitnote.xyz/audit_ac2/?
su=audit_ac1&sm=_4fBhON8UGmSQcHZ2IkiwUf3MINXt2FROCcmkFIyL8M6Bu0uo3lTCss
F-4dc_M5IzQsGgJ3qNrc5-1sFqD_GE0Pihy6wi-EqAwoEoNxzxbdl1G2dxKPtq9pIoj5T-
PCpHenEm3J5hIFz8IBOAB77k1qBoLo5aMddmLIiRFcd4PrheG5lXg&st=dOShp1XNtWa_h
H3Xw9pO3S9-XuXjycQUnGhXIznASU4PuR3gLZTUxtBg*

Given that the shared note is encrypted with AES-256-GCM using a key derived from an ECDH based on the users' P-521 key pairs, a third user that intercepts the share link (deviating from *audit_ac1* and *audit_ac2*) can craft the following share link:

Attacker-created share link:
*beta.bitnote.xyz/audit_ac1/?
su=audit_ac2&sm=_4fBhON8UGmSQcHZ2IkiwUf3MINXt2FROCcmkFIyL8M6Bu0uo3lTCss
F-4dc_M5IzQsGgJ3qNrc5-1sFqD_GE0Pihy6wi-EqAwoEoNxzxbdl1G2dxKPtq9pIoj5T-
PCpHenEm3J5hIFz8IBOAB77k1qBoLo5aMddmLIiRFcd4PrheG5lXg&st=dOShp1XNtWa_h
H3Xw9pO3S9-XuXjycQUnGhXIznASU4PuR3gLZTUxtBg*

Notably, the above link is equivalent to the link that *audit_ac2* would obtain legitimately if they opted to share the note with *audit_ac1*. However, the share link can be sent from the third user to *audit_ac1* and will be accepted.

To mitigate this issue, Cure53 recommends including unique identifiers (e.g., usernames) of both the note-sharing and note-receiving users as Additional Authenticated Aata (AAD) in AES-GCM. Another alternative approach would be to involve this context information in the key derivation step[7]. In the latter case, however, one would need to amend the key derivation function, though it would prevent the reuse of shared links in the same direction (i.e. from *audit_ac1* to *audit_ac2* but at a later point in time).

[7] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/nist.sp.800-56Ar3.pdf#5.8%20Key-[...]Schemes

Conclusions

Cure53 would now like to take the opportunity to comment on the project from a wider perspective post-finalization, offering insight into the positive and negative aspects observed throughout the vetting process. In sum, the two WPs exhibited differing defensive capabilities based on the evidence collected, as discussed below.

The source code corresponding to the BitNote core was provided by the client, who provided swift assistance for all queries via the established Slack channel in spite of the time zone variance.

BitNote offers a decentralized service whereby users can register an account, then subsequently create (and share) end-to-end encrypted notes, which are stored in the Avalanche blockchain. Despite the innovative paradigms underpinning BitNote, the design and decentralized model in tandem incur certain inherent limitations that must be taken into account to maximize user security.

Perhaps the most noteworthy point of contention is that blockchain's inherent nature coupled with BitNote's composition facilitate public availability of a user's master password derivatives, which can be employed to mount brute-force attacks against passwords of this ilk. As such, master password strength must be considered fundamental compared with correlating centralized environments. With this in mind, Cure53 believes that the current master password assurances are non-comprehensive, as documented in ticket RVE-01-004.

The frontend was stringently inspected. Although the source code is implemented in vanilla JavaScript and avoids the majority of dependencies, the custom logic responsible for constructing the HTML markup suffers from multiple DOM-based XSS vulnerabilities (see tickets RVE-01-001 and RVE-01-002) due to absent sanitization and CSP implementations (see ticket RVE-01-003).

Rather than hot-patch the aforementioned issues individually, one can advise refactoring the user interface with a templating system that thoroughly sanitizes all user-input variables by default. This will help to prevent the exploitation of vulnerabilities similar to RVE-01-001 and RVE-01-002, which currently remain obfuscated within the frontend source code.

The cryptography involved in BitNote's design was also subjected to deep dive examination. In general, the testing team acknowledged sound algorithm usage in the Web Crypto API regarding end-to-end note encryption. Moreover, the use of authenticated encryption was noted with distinction.

To caveat the aforementioned strengths, the protection installed to avoid side-channel attacks would benefit from improvement, as indicated in ticket RVE-01-008. In addition, the cryptography underlying the biometric login feature was deemed subpar and served for obfuscation rather than confidentiality shielding purposes. Consequently, a malicious actor can fully decrypt the plain master password, as described in ticket RVE-01-007.

Concerning the cryptography's design, Cure53 noted that the sender and receiver of a shared note are not bound to the generated share link, which permits an attacker that has intercepted this link to impersonate the receiver to a certain extent, as reported in ticket RVE-01-009. Additionally, in the event of a master password compromise, the user will not be able to reinstate the security of their current account even if they migrate the note content. As such, the user is forced to create a new account, requiring additional payment as well as re-encryption (see ticket RVE-01-006).

The two smart contracts involved in the account creation and note encryption protocols were systematically assessed by the testing team. In this area, a plausible phishing attack opporutnity on the referral address was detected and detailed in ticket RVE-01-005, though the exploitation likelihood is negligible.

In summary, BitNote's cryptographical elements appeared astute from a security perspective, with the pertinent exception of the biometric logic weakness described previously. Some leeway for general augmentation and refinement was acknowledged and reflected in the various *Miscellaneous Issues*, which should be reviewed at the earliest possible convenience. The frontend garnered a concerning impression in comparison and should be refactored entirely rather than patching the vulnerabilities in isolation.

Considering that BitNote is still in beta phase at present and will expand its feature set moving forward, Cure53 strongly advises performing follow-up security audits to ensure that the attack surface remains as constrained and future-proofed as possible.

Cure53 would like to thank Rockwell and Michael from the Rockwell Ventures team for their excellent project coordination, support, and assistance, both before and during this assignment.