Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Pentest-Report IVPN Customer Website & Servers 03.2024

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

Index

Introduction

"What you do online can be tracked by organizations you may not know or trust and become part of a permanent record. A VPN can't solve this on its own, but can prevent your ISP from being able to share or sell your data."

From https://www.ivpn.net/

This report details the findings from a penetration test and source code audit conducted against the IVPN Customer website. This in-depth assessment encompassed the frontend UI, backend components, API endpoints, underlying web servers, and infrastructure.

Commissioned by IVPN Limited in February 2024, Cure53 executed the engagement in March 2024 (calendar week CW11). A total of eight days were dedicated to achieving the coverage expected by the client, in line with the project's overarching requirements.

Three distinct Work Packages (WPs) were created, delineating the various core scope traits. These read as follows:

- **WP1**: White-box pentests & code audits against IVPN Customer website UI
- **WP2**: White-box pentests & code audits against IVPN Customer website API
- **WP3**: Gray-box pentests & scans against related IVPN web-servers & infra

This engagement marks the sixth collaboration between Cure53 and IVPN, demonstrating their ongoing commitment to security. Some aspects of the scope, particularly the IVPN web servers and infrastructure, were previously assessed in November 2019 and February 2023, denoted by the reports referenced as IVP-02 and IVP-05, respectively.

A white-box testing methodology was employed for the website application (WP1 and WP2), leveraging the provided source code, URLs, test user credentials, and other access points. For the underlying web servers and infrastructure (WP3), a gray-box approach was implemented as requested, simulating an external attacker with limited knowledge of the internal configuration. A dedicated team of four senior Cure53 testers was assigned to manage all phases of the project, from preparation and execution to finalization. All necessary preparations were completed in early March 2024 (calendar week CW10) to ensure a seamless testing process.

Effective communication was maintained throughout the engagement via a dedicated Rocket.Chat channel shared by both IVPN and Cure53 teams. This platform facilitated the participation of all relevant personnel from both parties. The well-defined scope and clear communication channels minimized the need for inquiries during the assessment. Cure53 provided numerous status updates on the testing progress and associated findings when required. Live reporting was not specifically requested for this audit.

Cure53 achieved satisfactory coverage across the defined scope areas. The assessment identified a total of four findings, categorized as either security vulnerabilities (two) or general weaknesses with lower exploitability potential (two). The discovery of only two *Low*-severity vulnerabilities and two general weaknesses is a praiseworthy outcome, especially considering that *Critical* and even *High* impact findings were completely avoided.

With this verdict, Cure53 can only commend the IVPN team for their dedication to securing and strengthening the Customer website. However, one can encourage the IVPN developers to maintain their focus on continuous improvement to ensure a consistently high level of security in the face of evolving and sophisticated compromise strategies.

The report will now shed more light on the scope and testing setup, as well as provide a comprehensive breakdown of the available materials. This will be followed by a chapter outlining the *Test Methodology*, which serves to provide greater clarity on the techniques applied and coverage achieved throughout this audit. Subsequently, the report will list all findings identified in chronological order, starting with the *Identified Vulnerabilities* and followed by the *Miscellaneous Issues* unearthed. Each finding will be accompanied by a technical rundown, Proof-of-Concepts (PoCs) where applicable, plus any fix or preventative advice to action.

In summation, the report will finalize with a *Conclusions* chapter in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the IVPN Customer site and server, giving high-level hardening advice where applicable.

Scope

- **Pen.-tests & code audits against IVPN public facing Customer Site & Server**
    - **WP1**: White-box pen.-tests & code audits against IVPN Customer website UI
        - **Primary focus:**
            - IVPN customer journey from initial website visit to service paying user
        - **Out of scope:**
            - Client apps
            - VPN servers
        - **Source:**
            - https://github.com/ivpn/ivpn.net
        - **Commit:**
            - 40c7ad844a3839d9178b71e803d8a507d5e517a3
    - **WP2**: White-box pen.-tests & code audits against IVPN Customer website API
        - **Source**:
            - https://github.com/ivpn/go-services
        - **Commit**:
            - 4eb158e72d0faba3fa4eff0130ac6cad647c5710
    - **WP3**: Gray-box pen.-tests & scans against related IVPN web-servers & infra
        - **Hosts:**
            - Reverse proxy and varnish:
                - *REDACTED*
            - Website and vpnapi:
                - *REDACTED*
            - Accounts:
                - *REDACTED*
            - WireGuard keyserver:
                - *REDACTED*
        - **Host access:**
            - SSH on port 253 with credentials below
        - **Test-user credentials:**
            - SSH:
                - U: strellic
            - SSH 2:
                - U: herrera
    - **Test-supporting material was shared with Cure53**
    - **All relevant sources were shared with Cure53**

Test Methodology

This section documents the testing methodology applied by Cure53 during this project and discusses the resulting coverage, shedding light on how various components were examined. Further clarification concerning areas of investigation subjected to deep-dive assessment is offered, especially in the absence of significant security vulnerabilities detected.

- Cure53 initiated the analysis by examining the code provided for the frontend UI. This approach complied with a white-box pentesting strategy, as the full UI source code was provided. An initial screening was performed for any high-risk or dangerous functions such as *eval*, assignments to dangerous sinks (such as certain HTML fields), or *location.href* usage. No XSS issues were found, largely due to the strict adherence to Vue.js best practices, which inherently prevents most XSS issues by design.

- The application was scanned for other sinks, including insecure implementation of *postMessage* and message handlers, prototype pollution vulnerabilities, and other DOM XSS sinks, though no associated pitfalls were detected. While investigating the live chat functionality, some *postMessage* handlers were detected that neglected to incorporate origin checks. However, the team was unable to exploit these since their functionality was limited.

- The JavaScript code related to the API calls (*api.js*) performed by the application was also tested for path traversal vulnerabilities and other client-side issues. The attack surface was found to be constrained, since the frontend application handles a negligible amount of user-controlled inputs.

- Next, the assessment team focused on finding vulnerabilities that are commonly encountered and applicable to Vue.js applications specifically. Cure53 could not pinpoint any locations whereby user input was fed directly into the *href* attribute, which could otherwise lead to XSS via a *javascript:* URI. The team also searched for similar areas using the *v-html* directive, as this allows for direct HTML rendering. This directive was employed in numerous locations, though input control was impossible here.

- The team then examined the presence and appropriateness of security headers in server responses. It was found that the application does not include the Content-Security-Policy (CSP) header, which provides beneficial protection against XSS and similar. CSP header implementation advice is proposed in ticket IVP-06-002.

- The team then concentrated on probing the backend UI implementation (WP2). The backend API was written in Go, with primary reviews performed against the code in the *vpnapi* folder based on the scope. The API routes were evaluated in a white-box manner to ensure ideal configuration.

- Firstly, the implementation for authentication and authorization was vetted to determine the presence of any security vulnerabilities. Users can access their account with either their secret account ID or an email/password combination, configurable in the account settings. Both flows were analyzed for potential issues, including the legacy login functionality. Here, Cure53 noticed that an arbitrary string can be encrypted with the *LegacySharedKey* secret, though the team could not detect any tangible security impact emanating from this activity during the evaluation time frame.

- Subsequently, the audit team inspected the scope's Two-Factor Authentication (2FA) using dynamic testing tactics, which can be enabled by users via their account settings. Here, Cure53 noted a lack of rate limiting on the endpoint that accepts 2FA tokens. However, further code reviews verified that rate limiting was only disabled on the staging environment. Additionally, rate limiting was found to be comprehensively applied to all sensitive endpoints that require its usage.

- The team then perused other endpoints that include functionality for the IVPN Customer site and server, such as the account recovery feature. These efforts sought to confirm the viability of injection attacks (e.g., via the Host header) either against the template email or email headers themselves.

- Furthermore, the reset password feature was subjected to in-depth assessments, which confirmed that the tokens exhibit a low TTL and are sufficiently invalidated after (single) use, which conforms with security best practices.

- Elsewhere, testing verified that strict parameter validation is achieved throughout the API, which was deemed a positive attribute since it is responsible for the prevention of an array of issues.

- Additionally, the code was inspected for usage of raw SQL queries, which could lead to SQL injection pitfalls. Most of the queries are created using prepared statements; however, an outlier was observed in the *GiftCardBatchStats* function whereby queries were constructed by joining variables in a raw string. Fortunately, the variables were deemed non-user-controllable after additional investigation.

- The Braintree payment integration and associated GraphQL queries were also checked for potential injection attacks, although no issues in the realm were uncovered.

- Regarding IVPN's web servers and infrastructure, scans were conducted against the in-scope servers to map the internal network’s attack surface and enumerate all internally operating applications.

- Multiple logs were audited to ascertain any erroneous PII storage, though these initiatives were ultimately unfruitful.

- The files and folders on all four hosts were researched in order to find any misconfigured permissions. A single file was verified to be world-readable, which allowed an unprivileged user to read a sensitive private key in the *REDACTED* server, as described in ticket IVP-06-003.

- Lastly, testing to simulate scenarios whereby an attacker could perform internal network requests (such as via an SSRF vulnerability) were performed in an attempt to validate whether the multiple internal APIs were resistant to internally mounted attacks. The internal APIs appropriately check for authorization, thus no correlatory weaknesses were identified.

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., *IVP-06-001*) to facilitate any future follow-up correspondence.

IVP-06-001 WP2: VPN config generator query params unsanitized *(Low)*

***Fix Note:** This issue was fixed and the fix was verified by Cure53 in early April 2024. The documented problem no longer exists.*

IVPN offers users the ability to connect to the VPN via OpenVPN or WireGuard. To leverage this feature, the user can download the OpenVPN or WireGuard configuration files from the API server via the IVPN Account dashboard.

The IVPN Account dashboard also allows users to configure various options like the IP address or port before they download their VPN configuration file. It was discovered that the API serving these configuration files takes these options via query parameters, which are not sanitized. As such, the parameters can contain newline characters and inject additional fields into the configuration.

An attacker could create a URL with malicious query parameters to inject custom fields into the VPN configuration files. If a user were to download and use these configuration files to connect to the VPN, their traffic may be intercepted by an adversary. The user could trust configuration file downloads from this URL, since it matches the IVPN domain. However, given that this user interaction is fairly unlikely, the ticket received a reduced severity score of *Low*.

PoC URL:
https://REDACTED/v5/config/ivpn-openvpn-config.zip?verify_x509_name=x%20name-prefix%0ainjected-key%20x%0a%3b

Steps to reproduce:
1. Download the OpenVPN configuration from the URL above.
2. Extract the *.zip* and open any *.ovpn* file in a text editor.
3. Note that the *injected-key x* field is injected into the config.

To mitigate this issue, Cure53 recommends sanitizing the values of the query parameters for the configuration generators. Moreover, the IVPN team should ensure that newlines are also removed to prevent new field injection.

IVP-06-003 WP3: World-readable config file reveals private key *(Low)*

Fix Note: *This issue was fixed and the fix was verified by Cure53 in early April 2024. The documented problem no longer exists.*

While investigating the file permissions of sensitive files on the *REDACTED* server, Cure53 found that a specific config file is improperly secured against access from unprivileged users. Any local user on the system can access the contents of this file, including a secret private key that is assumedly related to IVPN's Apple Store key.

However, an attacker would already need to have a foothold into the system and all other sensitive configuration files were found to be properly secured. Thus, the overall impact of this shortcoming was deemed *Low*.

Privatekey.p8 **access:**
```
[herrera@xor etc]$ cat /opt/ivpn/vpnapi/etc/privatekey.p8
-----BEGIN PRIVATE KEY-----

MIGTAgEAMBMGByqGSM49AgEGCCqGSM49A[redacted]7T0q
-----END PRIVATE KEY-----
```

To mitigate this issue, Cure53 recommends applying the correct file flags on this file, as well as preventing access from unauthorized users.

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.

IVP-06-002 WP1: Lack of Content-Security-Policy header *(Info)*

Fix Note: *This issue was partly fixed and the fix was verified by Cure53 in early April 2024. The documented problem no longer exists.*

Testing verified that the application does not implement a Content-Security-Policy (CSP) header. This security header serves as an additional layer of defense, allowing one to define policies for certain HTML tags such as script elements, which includes the origin a resource can be loaded from and other beneficial aspects. In general, the primary objective of this header is to ensure that abusive HTML injection is either completely deterred or rendered highly difficult to achieve.

Notably, this omission does not directly facilitate security flaws, though it could provide adversaries with unnecessary advantages to exploit other brittle areas.

To mitigate this issue, Cure53 advises integrating a CSP header that is as strict as possible into every server response, including error responses such as *4xx* items. The Customer site maintainers should set all HTTP headers at a specific, shared, and central location by adopting a load-balancing server or similar infrastructure (rather than configuring them at random). If the developers deem this approach impractical, remediation can alternatively be achieved by using the web-server configuration and a matching module.

IVP-06-004 WP2: Lack of email confirmation for user accounts *(Low)*

Fix Note: *The documented behavior is as expected and is not considered a problem; no fix is provided here.*

The observation was made that users are not required to confirm their emails when opting to use the alternative login process via email and password. This enables attackers to pre-register accounts using other users' emails and subsequently activate 2FA for these accounts.

This would effectively prevent the victim user from utilizing the IVPN application with their email, since they would be prevented from logging in due to the OTP token required for 2FA.

Upon further analysis, it was discovered that both the */accounts/change-email* and */accounts/set-auth* endpoints are susceptible to this attack, given that both permit altering the user's email without any corresponding actions needed.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Affected file:
/go-services-production/services/accounts/managers/accounts_mgr.go

Affected code #1:
```
func (mgr AccountsMgr) ChangeEmail(account *accounts.Account, newEmail 
string) error {
	[...]
	if err := mgr.data.SetAccountEmail(account.ID, newEmail); err != nil 
{
		return fmt.Errorf("cannot change email: %s", err)
	}
	account.Email = newEmail
	return nil
}
```

Affected code #2:
```
func (mgr AccountsMgr) SetAuth(account *accounts.Account, authType 
accounts.AuthType, email string, passwordHash string) error {
	if authType == accounts.AuthTypeEmail {
		isUsed, err := mgr.data.IsEmailInUse(email)
		[...]
		if err := mgr.data.SetAccountEmail(account.ID, email); err != 
nil {
			return err
		}
		account.Email = email
		account.PasswordHash = passwordHash
	} else {
		[...]
	}
	return nil
}
```

To mitigate this issue, Cure53 recommends inserting an additional procedure for both the */accounts/change-email* and */accounts/set-auth* endpoints, whereby a verification link is sent to the email address of the affected user that must be clicked to complete the email modification.

Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Conclusions

This closing chapter of the report serves to underscore Cure53's varying impressions of the IVPN Customer website's security health, as extrapolated following a eight day analysis against the scope in Q1 2024. In short, the testing team detected minimal security concerns during this engagement, attesting to a framework that is effectively capable of resisting the vast majority of breach and threat circumstances.

The IVPN Customer website and underlying servers presented a substantially secure posture during the assessment. This is reflected in the limited findings identified within this report, consisting solely of two *Low*-severity vulnerabilities and two general weaknesses. The successful mitigation of a wide range of common web application risks is a testament to the effectiveness of the security measures implemented by the project overseers.

A white-box testing methodology, leveraging access to source code and other resources, enabled a comprehensive evaluation of the application's security controls. Additionally, the established Rocket.Chat communication channel ensured smooth information exchange and minimized potential roadblocks. The well-defined scope and shared understanding of target areas further streamlined the testing process. Finally, the provision of a deployed staging environment by IVPN facilitated efficient testing of both the website and API functionalities. Albeit, it is important to acknowledge that the limited functionality within the defined scope and the inherently minimal attack surface of the application also played a role in the relatively low number of identified findings.

Concerning the coverage, the frontend for the primary IVPN Customer site was reviewed extensively for any client-side vulnerabilities, including DOM XSS and prototype pollution. Cure53 noted the absence of a Content-Security-Policy (CSP) header on the site (IVP-06-002); as a consequence, the premise fails to safeguard against XSS issues. The team recommends implementing these defense-in-depth headers to repel connected security threats moving forward. Fortunately, Cure53 could not detect any client-side vulnerabilities in the interface despite extensive efforts, partly owing to sound usage of the Vue.js frontend framework.

A deep-dive code review of the API associated with IVPN's customer journey was extensively performed, focusing on all exposed API endpoints. This led to the discovery that the alternative login feature via email and password lacks an email confirmation requirement, as indicated in ticket ticket IVP-06-004.

Another detriment affecting the API entails OpenVPN and WireGuard configuration files, whereby the endpoint in question takes certain query parameters to assign fields in the config files. However, these parameters are unsanitized and can be utilized to inject potentially malicious fields into the VPN configuration, as highlighted in ticket IVP-06-001.

The audit team also validated that strict enforcement of parameter validation is performed, including type checks and other actions, prior to internal API calls. The dev team's astute coding pattern here preemptively avoids a swathe of potential errors and inaccuracies.

The assessment of the backend codebase yielded similarly positive results. The Cure53 testers thoroughly investigated functionalities to identify any potentially risky calls, such as those enabling code execution, SQL injection, or other malicious behavior. The team's strenuous initiatives proved that untrusted user input is appropriately handled on the server-side, nullifying the potential for exploitation via common attack and injection vectors.

The examination of IVPN's web servers and infrastructure identified a single finding, documented in ticket IVP-06-003, which describes the ability for an unprivileged user to gain unauthorized read access to a sensitive private key on the *REDACTED* server. It is important to note that no other vulnerabilities were discovered within the web server or infrastructure environment.

To finalize, Cure53 is undeniably impressed with the overall security posture of the IVPN Customer website and its underlying infrastructure. The codebase exhibits assured standards of quality, while the implemented architecture and frameworks demonstrate a strong foundation in secure design principles. This viewpoint is further supported by the limited number of findings identified: only two *Low*-severity vulnerabilities were discovered during this engagement. These results are a clear indication of the proactive measures taken by the IVPN team to continually strengthen the security of their Customer website.

Cure53 would like to thank Nick Pestell, Iain, and Juraj Hilje from the IVPN Limited team for their excellent project coordination, support, and assistance, both before and during this assignment.