Fine penetration tests for fine websites

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

# Pentest-Report IVPN Apps & Daemon 03.2021

Cure53, Dr.-Ing. M. Heiderich, BSc. C. Kean, BSc. B. Walny, MSc. R. Peraglie, MSc. F. Fäßler

## Index

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

## 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 describes the results of a security assessment targeting the IVPN complex. Carried out by Cure53 in the frames of a broader security-centered cooperation with IVPN, the project included a penetration test and a source code audit, with the main targets being the mobile and desktop applications that IVPN offers for various platforms, as well as the IVPN daemon software.

As for timeline and resources, the work was requested by IVPN in late 2020 and then scheduled for late January and early February 2021. The testing team consisted of five senior testers from the Cure53 team. The budget allocated to the project stood at eighteen person-days.

To best respond to the goals communicated by IVPN, four distinct work packages (WPs) were formulated. All WPs consistently utilized penetration testing and source code auditing as key approaches, yet the targets were shifting. In WP1 Cure53 focused on the IVPN iOS application, while the Android branch took center stage in WP2. Next, the IVPN Electron application was assessed in WP3. Finally, the IVPN demon was subjected to extensive examinations in WP4.

As all code pertinent to the IVPN complex is available on GitHub as OSS, the Cure53 testers used white-box methodology during this project. Alongside source code access, the team members were supplied with a variety of builds for various platforms. Builds linked to both staging and production servers, with various debugging means enabled, were furnished to Cure53. The ultimate goal was to enable acquisition of good coverage and expected research depth.

All preparations were completed in January 2020, namely CW03 and early CW04, so that the Cure53 team could have a smooth start. The project moved forward without any interruptions. Communications during the test were done using a Rocketchat instance into which IVPN invited the involved Cure53 team members. Since the scope was clear and well-prepared, not many discussions were needed. The IVPN team was very helpful and assisted Cure53 with various aspects of this work.

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

Nevertheless, Cure53 gave frequent status updates about the test and the related findings. Live-reporting was used for all spotted findings, with the IVPN bug tracker running on Bitbucket used for this purpose.

The Cure53 team managed to reach a good coverage over the WP1-4 scope items. Fourteen security-relevant discoveries were made on the scope. Four items were classified to be security vulnerabilities and ten need to be seen as general weaknesses with lower exploitation potential. Two findings were considered *Critical* in terms of severity. The first shows how a local attacker can abuse the tested software to achieve a *root* privilege escalation via a race condition (see IVP-03-007). The second has similar consequences but leverages weaknesses in WireGuard, as demonstrated in IVP-03-013. The remaining issues were mostly classified as *Medium* and might be seen as less concerning threats.

Note that at the time of authoring of this final report document, all but three issues were addressed by the IVPN team and the fixes were verified successfully by Cure53. One of the three remaining issues was flagged as a false alert and two to be of acceptable risk for now and a possible work-in progress for later releases. Each issue description in this final report was amended with a fix note describing it's current state. Proof-of-Concept material was removed from this report upon IVPN's request.

In the following sections, the report will first shed light on the scope and key test parameters, as well as the structure and content of the WPs. Next, all findings will be discussed in grouped vulnerability and miscellaneous categories, then following chronological order in each array. Alongside technical descriptions, PoC and mitigation advice are supplied when applicable. Finally, the report will close with broader conclusions about this January-February 2021 project. Cure53 elaborates on the general impressions and reiterates the verdict based on the testing team's observations and collected evidence. Tailored hardening recommendations for the IVPN complex are also incorporated into the final section.

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

## Scope

- **Penetration-Tests & Code Audits against IVPN Apps & IVPN Daemon Software**
  - **WP1**: Penetration-Tests & Source Code Audits against IVPN iOS App
    - https://github.com/ivpn/ios-app
    - Testflight invite received.
  - **WP2**: Penetration-Tests & Source Code Audits against IVPN Android App
    - https://github.com/ivpn/android-app
  - **WP3**: Penetration-Tests & Source Code Audits against IVPN Electron App
    - **CLI:**
      - https://github.com/ivpn/desktop-app-cli
    - **UI:**
      - https://github.com/ivpn/desktop-app-ui2
  - **WP4**: Penetration-Tests & Source Code Audits against IVPN Daemon
    - https://github.com/ivpn/desktop-app-daemon
  - **Test-user accounts were provided for Cure53**
  - **Debugging-friendly binaries were shared with Cure53**
  - **Test-supporting material was shared with Cure53**
  - **Sources were shared and available as OSS**

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

## Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *IVP-03-001*) for the purpose of facilitating any future follow-up correspondence.

### IVP-03-007 WP4: Root privilege escalation via race condition *(**Critical**)*

While analyzing the architecture of the IVPN client and service daemon, a very severe flaw was found. The IVPN service creates a listening TCP interface on a seemingly random port, which is used by OpenVPN for connections to the management interface. This means that there is a race condition wherein the daemon is already listening, but OpenVPN is still connecting. Because the daemon trusts the output of OpenVPN and parses commands originating there for execution, this *Critical* problem can be used to escalate privileges to *root*.

There are a few different recommendations regarding this issue. The IVPN service could improve the randomization of the ports, though this might also impact reliability in case the system has no free ports to allocate. In this case, the strongest mitigation is to fix the route command parsing in *mi.go*. At the moment, it liberally accepts anything that just contains the string "*route add <ip> <ip>"*. Thus, the *ovpn mi* injection almost led to a privilege escalation before (IVP-03-011). If IVPN employs a strict allow-listing for valid route commands, the issue would be fixed.

Generally, the daemon should consider all responses from the OpenVPN management interface to be potentially malicious. Handling of those messages should be designed with potential malice in mind.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

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

### IVP-03-011 WP4: OpenVPN management interface injection *(Medium)*

Using the trick to bypass the allowed clients described in IVP-03-004, the protocol was investigated further. Here it was found that the *MultihopExitSrvID* option in the *"Connect"* command can be used to inject OpenVPN management interface commands by including newlines.

While the OpenVPN management port cannot be directly used to perform very critical attacks, IVPN clearly wants to prevent arbitrary clients from connecting to it. Thus, it is recommended to encode all strings sent to the *ovpn mi* properly.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

### IVP-03-012 WP4: Firewall allows deanonymization for eavesdropper *(Medium)*

It was found that the IVPN firewall rules allow all TCP connections to the currently used IVPN entry node. This means attackers may deanonymize all IVPN users that browse an attacker-controlled website by passively observing the network traffic of all IVPN entry nodes.

An unencrypted HTTP request including a token that is unique for every visitor will be sent to all IVPN entry nodes. This request is sent from the attacker-controlled website that is browsed by the victim protected by IVPN. Due to routing and lax firewall rules, the request will not be protected by IVPN and the token coupled with the IP address of the victim can be observed in the network traffic of an entry node.

It is recommended to tighten the firewall rules in a way that only the VPN process is allowed to connect to the IVPN entry node. This could be implemented by making use of the *owner* module and supplying a process ID or user ID that only the OpenVPN process can hold. It is important to note that supplying the *root* as the user ID still leaves all users who run the browser as *root* - (i.e. old Kali Linux systems) affected by this problem.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

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

### IVP-03-013 WP4: *Root* privilege escalation via WireGuard *(**Critical**)*

It was found that the privileged daemon embeds user-input, which it receives from the unprivileged IVPN user-interface, in an unsanitized form into the WireGuard configuration file. Further, it was verified that this configuration file was loaded with *wg-quick*, letting attackers pass arbitrary commands via the *PreUp* configuration parameter that will be executed with *root* privileges. This can be abused by attackers to escalate from lower to privileges up to *root*.

It is recommended for the application to only permit the minimal base64 character set in user-input before having it embedded into the WireGuard configuration file. By doing so, attackers cannot escape from the configuration parameter that they inject and the issue is mitigated.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

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

## Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

### IVP-03-001 WP1: Lack of restricted segments for *dylib* code injection *(Info)*

While reviewing the IVPN binary on iOS, it was noted that it lacks a *__restrict* segment to ignore *Dynamic Loader* (*dyld)* environment variables which could facilitate code injection. The impact of this issue was evaluated as *Info* since no code injection could be achieved in the limited time available for this engagement.

It is likely that this kind of code injection is only feasible in a jailbroken environment or on iOS below version 10. However, the latter is not supported by the IVPN build in scope.

The absence of the *__restrict* segment can be verified on MacOS with the *size* command. The following command has to be run on the binary contained in the extracted IPA archive of the application.

It is recommended to consider whether the described countermeasure is required in the security model of the IVPN iOS app. The official documentation for this type of exploitation, as well as its countermeasure, is scarce for iOS and largely based on analogous code injection exploits on MacOS or app modifications on jailbroken iOS devices. Therefore, any steps to counter this on iOS can only be seen as an optional hardening measure.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

### IVP-03-002 WP4: Buffer overflow and erroneous parsing in Wifi notifier *(High)*

It was found that the IVPN daemon on Linux periodically logs unexpected crashes due to segmentation faults within the *setWifiNotifier* function. On further inspection, it can be observed that the function erroneously parses multipart netlink messages received from the kernel.

The primary issue is that the *nhl* pointer is progressed through the *buffer* for each packet processed in the inner while loop. However, once the inner while loop returns, *4096* bytes are received and written to the *nhl* pointer in the middle of the *buffer*. This increases the risk of the received data being too large for the remaining *buffer.* resulting in a classic buffer overflow.

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

Another issue is that the application logic assumes that the beginning of *recv()*'d data is always a netlink message header. This is done without anticipating netlink messages that spread over more than 4096 bytes. This can be observed since the *recv()* function receives a pointer to a *nlmsghdr* struct. The current handling is dangerous because the payload of a netlink message could be interpreted as a header. leading to unintended behaviors.

It is recommended to use the Linux syscalls offered by Golang's built-in *syscall* module[1]. By doing so, transparency and security can be improved by sticking to the safe modules of Go, thus mitigating memory corruption bugs that arise from raw pointer arithmetics and re-using well-test code.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

### IVP-03-003 WP4: Buffer Out-Of-Bounds read in WiFi notifier *(Low)*

It was found that the *get_essid* function of the WiFi notifier on Linux uses a buffer that is too small to hold an SSID with a maximum size of a WiFi network. This translates to the risk of the buffer not being null-terminated, resulting in the daemon over-reading bytes from the memory returned to the unprivileged IVPN client. This could be used to extract sensitive memory from the daemon.

It is recommended to increase the length of the buffer. As a consequence, the null-terminating null-character is not going to be overwritten by the underlying driver. Additionally, a *strndup* call can be used to read a maximum number of 32 bytes, so as to cope with potential security flaws within the driver.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

### IVP-03-004 WP4: Trivial bypass of *allowedClients* list on Linux *(Medium)*

The IVPN daemon uses an interesting approach to validate the incoming connection. When a client connects, it will parse all existing sockets on the system, match the port to a process ID and look up the process' binary. This is then compared to a strict allow-list. However, it was found that code-injection techniques can be used to bypass this check. On Linux, the *LD_PRELOAD* environment variable can be employed to hook a library function and execute one's own code in the context of the process. This is not necessarily a vulnerability in itself: when the protocol is safely designed, then a local

[1] https://golang.org/src/syscall/netlink_linux.go

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

program cannot exploit the service. This setup also prevents attacks where a malicious website would try to send commands via HTTP requests to the local port. It points towards the strategy working well-enough, yet also means that the daemon must handle commands securely. Countering this, it was shown in IVP-03-007 and IVP-03-011, that there are still some weaknesses that attackers can capitalize on. This issue could also exist in some form on Windows or Mac.

Generally, those operating systems differ in attempting to limit code injections, though there are a lot of different techniques and it is probably impossible to prevent all approaches. However, on Linux or Mac, a new IVPN user could be created and the IVPN clients use *setuid* to run as IVPN. The daemon could then reject processes that do not run as the IVPN user, likely preventing code injections via libraries or debugging features like *ptrace*. Alternatively, the current handling could be an accepted risk, as long as more emphasis is placed on designing a safe daemon protocol.

***Note**: This issue was not yet fixed by the IVPN team but is seen as a WIP and will be addressed with in future releases.*

### IVP-03-005 WP4: Invalid pointer conversion via unsafe package *(Medium)*

It was found that the *doGetPortOwnerPID* function on Windows casts an *unsafe.Pointer* to a *uintptr* pointer and stores it into a variable before casting the *uintptr* back to a *unsafe.Pointer* several expressions later. This behavior is an invalid pointer conversion, as documented in the official Golang documentation[2]. It could lead to memory corruption bugs due to internal behavior of the Garbage Collector in Go. The bugs could be coupled by attackers with heap spraying, resulting in the potential for Local Privilege Escalation and Remote Code Execution vulnerabilities.

It is recommended to store the pointer held by the *rowsStartPtr* variable as an *unsafe.Pointer,* thus strictly adhering to the documentation. Conversions to a *uintptr* should only be done in the same expression, as specified in the Golang documentation.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

### IVP-03-006 WP1: Enabled *NSURLCache* logs login credentials *(Low)*

It was discovered that the *NSURLCache* is enabled for some API communications of the iOS app. This could accidentally expose sensitive data such as user-credentials, *authentication* tokens or PII. The impact of this issue was evaluated as *Low.* While these

[2] https://golang.org/pkg/unsafe/#Pointer

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

credentials are logged in clear-text, leaking them would require physical access to the device.

Client-side caching should be disabled to prevent automatic recording of API communications in the cache. The *Secure Mobile Development* guide[3] can be reviewed for further instructions regarding the aforementioned cache getting disabled. It should be noted that the default *NSURLCache* does not support altering the protection level of its store, as advised in *IVP-03-008*.

Consequently, this means that all requests and responses will still be cached and left unprotected at rest via the *NSURLCache,* even when an application implements *Data Protection* at the application level. If the *URLCache* is required, this can be avoided with a custom *NSURLCache* subclass, thus storing responses on an SQLite *DB* file with the *NSFileProtectionComplete* attribute set.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

### IVP-03-008 WP1: Incomplete iOS filesystem protections *(Low)*

It was found that the iOS app does not take full advantage of the native iOS filesystem protections and fails to fully protect some of its data files at rest. The affected files are only protected until the user authenticates for the first time after booting the phone. The problem is that the key to decrypt these files will remain readable in memory even while the device is locked. The impact of this issue was evaluated as *Low* as some login credentials are exposed.

This issue requires physical access to an iDevice set to a locked screen and a method of accessing the local storage, for instance, through an SSH connection established via a jailbreak. While being locked, the files represented below remain protected whenever they are not flagged with "*Operation not permitted"*.

In order to solve the issues related to file-access, it is recommended to implement the *NSFileProtection-Complete* entitlement at the application level[4] for all files, as well as considering implementing changes to the *NSUrlCache* described in IVP-03-006.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

[3] https://github.com/nowsecure/secure-mobile-development/blob/master/en/io...-requests-responses.md
[4] https://developer.apple.com/library/ios/documentation/iP...App/StrategiesforImplementingYourApp.html

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

### IVP-03-009 WP2: Lack of *FORTIFY_SOURCE* for third-party shared objects *(Info)*

While analyzing the IVPN Android app, it was noticed that some of the included shared objects are not compiled using the *FORTIFY_SOURCE*[5] compiler option. The *FORTIFY_SOURCE* macro provides basic support for detecting buffer overflows within various functions that perform memory and string operations, including the following list of functions: *memcpy, mempcpy, memmove, memset, strcpy, stpcpy, strncpy, strcat, strncat, sprintf, vsprintf, snprintf, vsnprintf* and *gets.*

It is recommended to consider compiling the referred shared objects and other libraries with *FORTIFY_SOURCE* enabled. This should be accomplished by using the compiler option *-D_FORTIFY_SOURCE=2*[6].

***Note**: This issue was not yet fixed by the IVPN team but is currently treated as acceptable risk, which was also confirmed with Cure53.*

### IVP-03-010 False Alert: Unencrypted login credentials in local storage *(Info)*

This issue of unencrypted login credentials appearing in local storage affects all desktop operating systems. For instance, in Windows, sensitive information such as the configuration files which include the *accountID* or OpenVPN credentials are being stored as plain-text files in the application directory. A better practice is to encrypt such information in order to offer additional protections against attackers who have reading capabilities for local files. A file example drawn from the *etc* directory is shown below, together with its sensitive contents.

An attacker with local file read capabilities could be considered too powerful already. Nevertheless, taking it into account can foster an additional hardening step and increase the defense-in-depth. For Windows, the *Data Protection* API[7] is recommended to facilitate the encryption of either another custom decryption key or the encryption of the files themselves. Similar strategies are recommended for other operating systems.

***Note**: After discussing this issue with the IVPN development team it was concluded that the issue can be marked as a false alert. The cause was a testing artifact with the chosen environment.*

[5] https://man7.org/linux/man-pages/man7/feature_test_macros.7.html
[6] https://github.com/hashbang/hardening#source-fortification
[7] https://docs.microsoft.com/en-us/dotnet/standard/security/how-to-use-data-protection

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

### IVP-03-014 WP4: Potential CSRF allows stealing VPN credentials *(Medium)*

It was found that the daemon listens on a local TCP port for the OpenVPN management interface without leveraging authentication and protocol checks. This is dangerous as the daemon ignores all messages that do not match a specific format, making [protocol downgrade attacks possible. In turn, these could allow malicious websites to send a HTTP *GET* request to the listener, extracting the VPN credentials remotely.

The OpenVPN management interface should be accessed via pipes on Windows and via UNIX domain sockets on Linux systems, respectively. It should require privileged access instead of TCP network sockets. Revisions in this realm will preclude the local browser from connecting to the OpenVPN management interface, mitigating this vulnerability.

***Note**: This issue was fixed by the IVPN team and the fix has been verified by Cure53. The problem as reported no longer exists.*

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

## Conclusions

This project broadly focused on mobile and desktop applications, as well as demon software, offered within the IVPN complex. After eighteen days dedicated to examining the scope in early 2021, five members of the Cure53 team could observe both strengths and weaknesses across various WPs/targets. To facilitate better understanding of the pervasive shortcomings and specific efforts that are advised per component, the following paragraphs will discuss the impressions in two blocks: the first dedicated to Android and iOS applications (WP1 and WP2), and the second honing in on the IVPN Electron and demon software (WP3 and WP4).

Both WP1 and WP2 enveloped code-assisted penetration testing of the respective Android and iOS branches of the IVPN application. On Android, the exposed activities, broadcasts, content providers and services were audited for manipulation via intents, as well as in relation data leakage. However, none of the exported features were found to be exploitable, demonstrating strengths of this app.

In addition, the local storage on Android was found to take advantage of encryption for protecting its secrets at rest. Common Android hardening measures such as *FLAG_SECURE* and *filterTouchesWhenObscured* were found to be employed. This further reduces the potential attack surface. Still focusing on Android, an error was spotted in the shared objects, which could be improved to be compiled with an option preventing buffer overflows (IVP-03-009). Notably, this is merely a hardening suggestion and does not signify an exploitable flaw.

More mistakes were clearly pinpointed on the iOS app branch. The IVPN iOS app was analyzed for insecure storage which could lead to information leaks and here further improvements should be considered in terms of restricting filesystem permissions (IVP-03-008) and disabling client-side caching (IVP-03-006). While it was positively noted that the iOS app takes advantage of the most common compiler and linker flags such as PIE, ARC or the Stack Canary flag, additional enhancements concern restricted segments (IVP-03-001), which would harden the app against *DyLib* code injections.

Summing up, neither of the app branches revealed information via insecure logging practices and, more broadly, both IVPN mobile applications make a robust impression and aptly minimize the attack surface. The few mobile issues documented mainly concern hardening measures and improving security for local storage, especially in the context of unauthorized physical access. The proposed measures should be interpreted as suggestions rather than necessary steps. At the same time, they will help harden the IVPN mobile apps further.

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

Moving on to WP3 and WP4, here Cure53 focused on how IVPN implements UI, CLI and demon for different operating systems via one shared codebase. The idea keeps the complexity low and makes security reviews easier. As many VPN applications, IVPN also uses a privileged service to handle the networking configuration part.

To begin with the communication between the user application and the privileged port, IVPN implements it via a local TCP server and protects it through a client binary verification. However, it was found that this client verification can easily be bypassed by a local attacker who can then directly communicate with the daemon (IVP-03-004). On its own, this does not constitute a security issue because the mechanism prevents malicious websites from attempting to communicate with the daemon via maliciously crafted HTTP requests, despite not stopping a fully-local attacker.

From a meta-level perspective, the protocol with the daemon is designed safely, meaning that no easy privilege escalations are possible. For example, commands to execute arbitrary programs or update the daemon with a malicious file do not exist. At the same time, the communication between the IVPN daemon and the OpenVPN management interface was revealed to be prone to both local (IVP-03-011) and remote (IVP-03-014) attack-vectors. This is an indicator that the transport protocol should be reconsidered as a whole. Cure53 advises looking into pipes and unix domain sockets as a means to prevent such attacks.

While specifically looking for privilege escalation issues allowing to execute arbitrary commands, it was found that the daemon trusts responses from the OpenVPN management interface when the latter specifies the commands for execution. This could be triggered by a local attacker to gain *root* level position on the system IVP-03-013.

Many memory corruption bugs represent a security hazard in relation to privileged processes. IVP-03-002 shows that many challenges can be solved without *unsafe* and CGO, which are typically vulnerable. This could be a general indicator to refactor unsafe and CGO code into safe alternatives. The firewall rules were found to be overly lax, allowing potential deanonymization of the remote peers against strong/state-funded attackers. As described in IVP-03-012, no connections to outer peers should be allowed. With the exception of the IVPN daemon and VPN process.

Finally, the Electron UI has been audited for common vulnerabilities. The templates and JavaScript code has been checked for all potential sinks which could lead to XSS. Here, user-input is being handled in a safe manner, partly due to the correct and secure usage of the Vue.js framework. In some instances, the JavaScript code makes use of the potentially dangerous *innerHTML* property, which however relies on data stored on a user's filesystem or returned via the demon, hence requiring a powerful attacker.

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

Electron has been configured in a secure manner by employing all state-of-the-art security flags and not enabling additional, potential harmful, features. Accessing critical Electron-specific APIs, like *openExternal* or more implicit things like redirects of the main window, was found secure. The IVPN team did a good job by either allow-listing the arguments, or disabling the problematic features.

The general file handling has been reviewed. The Linux client, for instance, writes its settings into files which are guarded by *0664* access permissions, which are readable by anyone, however rely on the fact that the *.config* directory is *0600*. This is the case for a default Ubuntu installation while other distributions could not adhere to this. It is recommended to specify the file permission explicitly when it comes to potentially sensitive files.

Across all desktop platforms, some of the most sensitive files have been found unencrypted, as shown by full *accoundIDs* or OpenVPN credentials (see IVP-03-010). Moreover, some of the JavaScript dependencies were found to be quite old, hence a quick check on a potentially impactful lib, *deepmerge*, has been conducted to see if it was vulnerable to prototype pollution. Cure53 strongly advises to keep the dependencies as current as possible. The main business logic was found to be very lightweight, meaning that there is not much room to exploit potential features. This design decision contributes greatly to this rather good security posture of the UI component.

It needs to noted that all relevant issues have been addressed by the IVPN team before this final report document was authored and released. It needs to further be noted that the IVPN daemon software inspected in WP3 was not released to the public before the IVP-03 security audit was finished. This means, that none of the WP3- and WP4-related issues were exposed to regular IVPN users.

Cure53 would like to thank Nick Pestell, Alexandr Stelnykovych, Iain Douglas as well as the rest of the IVPN team for their excellent project coordination, support and assistance, both before and during this assignment.