Fine penetration tests for fine websites

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

Pentest-Report ExpressVPN Router Firmware 06.-07.2022

Cure53, Dr.-Ing. M. Heiderich, Dipl.-Inf. G. Kopf, B. Sc. M. Münch, B. Sc. L. R. König, BSc. F. Heiderich

Index

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

Introduction

"A VPN router lets you protect all devices with unlimited connections to ExpressVPN. Now with our Device Groups feature, you can connect to more than one VPN location."

From https://www.expressvpn.com/vpn-software/vpn-router

This report - titled EXP-07 - details the scope, results, and conclusory summaries of a penetration test and source code audit against the ExpressVPN router, with a specific focus on its UI, interfaces, and firmware. The work was requested by ExpressVPN in March 2022 and initiated by Cure53 in June and early July 2022, namely between CW25 to CW27. A total of thirty-two days were invested to reach the coverage expected for this project.

The testing conducted for EXP-07 was divided into two separate work packages (WPs) for execution efficiency, as follows:

- **WP1**: Source-code-assisted penetration tests against ExpressVPN router UI/interfaces
- **WP2**: Source-code-assisted penetration tests against ExpressVPN router firmware

In context, this review marks the seventh collaboration between the Cure53 and ExpressVPN teams. However, the ExpressVPN router has not yet been included in scope for any previous test iterations. This audit, therefore, constitutes the inaugural assessment against the router specifically by Cure53.

Cure53 was provided with sources, binaries, pertinent test-assisting documentation, as well as any alternative means of access and information required to complete the audit. For these purposes, the methodology chosen was white-box and a team comprising five senior testers was assigned to the project's preparation, execution, and finalization.

All preparatory actions were completed in June 2022, namely in CW24, to ensure that the testing phase could proceed smoothly. Communications were facilitated via a dedicated, shared Slack channel deployed to combine the workspaces of ExpressVPN and Cure53, thereby allowing an optimal collaborative working environment to flourish. All participatory personnel from both parties were invited to partake throughout the test preparations and discussions.

One can denote that communications proceeded smoothly on the whole. The scope was well-prepared and clear, no noteworthy roadblocks were encountered throughout testing, and cross-team queries were kept to a minimum as a result.

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

ExpressVPN delivered excellent test preparation and assisted the Cure53 team in every respect to procure maximum coverage and depth levels for this exercise. Cure53 gave frequent status updates concerning the test and any related findings, whilst simultaneously offering prompt queries and receiving efficient, effective answers from the development team. Live reporting was offered and subsequently achieved via the aforementioned Slack channel.

Regarding the findings, the Cure53 team achieved comprehensive coverage over the WP1 and WP2 scope items, identifying a total of six. Three of the findings were considered security vulnerabilities, whilst the remaining three were deemed general weaknesses with lower exploitation potential. Generally speaking, the overall yield of findings is relatively minimal for a scope of this magnitude, which certainly correlates to a positive indication of the components' security posture.

One can also positively acknowledge that all findings - irrespective of constituting a security vulnerability or general weakness - were ranked with a *Medium* severity or lower, highlighting that the vast majority of significant threats or attack surfaces were effectively deterred.

All in all, Cure53 is pleased to report that the ExpressVPN team has established a first-rate security level for the components in focus following the completion of this audit. Nevertheless, the testing team also observed some leeway for improvement; the guidance offered in this report should be adhered to and implemented to achieve an exemplary security posture for the VPN router.

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 detailed definition of all possible severity markers, as well as 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 detected vulnerabilities and followed by the general weaknesses unearthed. Each finding will be accompanied by a technical description and Proof of Concepts (PoCs) where applicable, plus any relevant mitigatory or preventative advice to action.

In summation, the report will finalize with a conclusion in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the ExpressVPN router components in focus, giving high-level hardening advice where applicable.

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

Scope

- **Code audits and security assessments against ExpressVPN's router firmware & UI**
  - **WP1**: Source-code-assisted penetration tests against ExpressVPN router UI/interfaces
    - The source code archives (*.zip*) for the Vue.js based frontend code were shared with Cure53.
    - The backend code was shared as part of the archive (*.zip*) containing the router firmware.
    - The team was provided with devices running the firmware version subject to test (4.0.2.9567) and VPN credentials to activate the VPN functionality of the router.
    - To evaluate the update functionality in more detail, parts of the *xvclient* were provided as a source code archive (*.zip*) containing the update-related implementation.
  - **WP2**: Source-code-assisted penetration tests against ExpressVPN router firmware
    - Source code packages (*.zip*) files were shared with Cure53.
    - Test devices running the firmware version subject to test (4.0.2.9567) with SSH access and VPN credentials to activate the VPN functionality of the router were provided to Cure53.
  - **In-scope items:**
    - Remotely exploitable attack vectors.
    - Attack vectors that can remotely compromise (retrieve or use) the authentication token to gain access to the router's administration.
    - Web application vulnerabilities in the router administration portal accessible from the local network.
    - Exploitable vulnerabilities reported in the auto update feature. A partial bypass of defense-in-depth measures for verifying the firmware is not considered an issue.
    - IP address and DNS leakage of client devices when connected via VPN tunnels.
    - VPN configurations, recovery mode if triggered remotely.
  - **Out-of-scope items encompass the following items and vulnerabilities:**
    - Proprietary 3rd party QSDK firmware including bootloader and binaries, not developed by ExpressVPN.
    - Vulnerabilities of which their exploitation requires knowledge of the API key.
    - Vulnerabilities that require sending requests via the HTTP protocol.
    - Manually-uploaded firmware images (no verification is applied).
    - ExpressVPN API servers and APIs.
    - Attack vectors that require physical access to the Aircove router.
    - VPN protocol implementations.
    - Recovery mode if triggered using physical access.
    - Known security issues.
- **Test-supporting material & necessary devices were available to Cure53**
- **All relevant sources were made available for Cure53**

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

Severity Glossary

The following section details the varying severity levels assigned to the issues discovered in this report.

Critical: The highest possible severity level. Categorizes issues that allow attackers to achieve extensive access to sensitive areas, such as critical systems, applications, data or other pertinent components in scope.

High: Categorizes issues that allow attackers to achieve limited access to sensitive areas in scope. This also includes issues with limited exploitability that can facilitate a significant impact upon the target in scope.

Medium: Categorizes issues that do not incur major impact on the areas in scope. Additionally, issues requiring a more limited exploitation are graded as *Medium*.

Low: Categorizes issues that have a highly limited impact on the areas in scope. Mostly does not depend on the level of exploitation but rather on the minor severity of obtainable information or lower grade of damage targeting the areas in scope.

Info: Categorizes issues considered merely informational in nature. They are mostly considered as hardening recommendations or improvements that can generally enhance the security posture of the areas in scope.

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

Table of Findings

Identified Vulnerabilities

ID	Title	Severity
EXP-07-003	OOS: String matching in firewall rules facilitates DNS infoleak	*Medium*
EXP-07-004	WP2: WAN packet forwarding to LAN interface	*Medium*
EXP-07-006	WP2: DoS via unauthenticated file uploads	*Medium*

Miscellaneous Issues

ID	Title	Severity
EXP-07-001	WP2: Symlink attack in implemented TAR extraction	*Medium*
EXP-07-002	WP2: Key reuse between GCM and CBC API keys	*Low*
EXP-07-005	WP2: Lack of return value check on setsockopt	*Low*

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

Test Methodology

In this next section, the threat model and testing methodologies applied during this engagement are documented, whilst an overview of the areas investigated by Cure53 - including areas that did not yield any results - is provided. Generally, all tests were performed in a manual fashion without relying on automatic tools such as vulnerability scanners or static code-analysis tools. As a result, the provided source code was statically audited and the solution was dynamically tested in addition.

Threat Model

Before covering the test methodology steps, Cure53 is keen to provide clarification regarding the threat model. ExpressVPN defined all threat denominators as follows:

- Nation-state actors exploiting vulnerabilities in the router firmware or OS from the WAN to compromise the router or connected devices through a variety of different attacks, such as "Man in the Middle" (MitM) attacks or TLS downgrading attacks, as well as weaknesses including static keys, weak encryption or weak random number generation. Notably, however, threat actors with the ability to obtain valid X.509 certificates for arbitrary domains were considered out of scope.
- Exploiting flaws in (older) protocol versions or weak cryptographic algorithms.
- Threat actors exploiting a vulnerability in the router firmware or OS from the LAN to compromise the router or connected devices by remote code execution (RCE) or exposure of sensitive user data via logs or other means, which would violate the ExpressVPN privacy policy[1].
- Threat actors monitoring traffic originating from a device running ExpressVPN router firmware to gain insight on usage and traffic inside the VPN tunnel via data leakage, anomaly detection, packet size, traffic patterns, and similar means.

Notably, a number of vulnerabilities and weaknesses were considered as known issues and hence out of scope for this assessment, including:

- Outdated software in the router firmware as a result of the dependency on the QSDK. ExpressVPN confirmed no remote exploitability affects these packages.
- A number of possible DoS conditions due to potential memory-allocation failures.
- Insecure cookie flags for the router administration portal.
- Lack of binary hardening on the router firmware.
- Deviations from best practices in the router's disabled by default OpenSSH configuration.

[1] https://www.expressvpn.com/privacy-policy

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

Test Coverage for WP1: ExpressVPN router UI/interfaces

The source code repositories provided by the customer contained the full implementation of the ExpressVPN router's administration interface. The interface consisted of a *Vue.js* frontend and a Lua-based backend. The backend was mostly implemented in MoonScript, which translates to Lua code. During the manual code review of the interface, the following (non-exhaustive) list of tasks was executed:

- The threat model and the provided information of the available API, as well as the scope and previously-known issues, were reviewed. This allowed for a greater understanding of the scope and helped to create a concise overview of the solution's general architecture.
- Following this, a general overview of the provided source code was obtained by briefly reviewing the code base with regard to structure, organization, and the general coding style. By doing so, Cure53 was able to gain an initial outline of the implementation and prioritize areas of the code by their potential impact, based on code patterns that could incur significant issues.
- Subsequently, the relevant identified areas were subject to an in-depth code review with a specific focus on common web-application flaws. This included the enumeration of potential XSS sinks, such as v-html, in the Vue.js frontend as well as command injections in the backend components.
- As one of the primary components of a solution's security posture, the authentication and authorization implementation was subject to comprehensive auditing. The general API-key handling was inspected in relation to a plethora of facets, from key generation with a focus on cryptographic issues over the session handling and storage, to the management and validation of the created API keys. Available routes were inspected for any potential unauthenticated or unauthorized access.
- The web-server implementation was reviewed for commonly-found issues such as path traversal.
- In typical fashion for a modern web-based interface, data was processed as JSON. The JSON parsing implementation that allows generating Lua objects was inspected for any potential vulnerabilities. Furthermore, due to usage of a schema-based validation of user-provided data, this area was particularly pertinent since a misinterpretation of the data type of provided inputs may facilitate problems in the backend. For example, if input is interpreted as a string rather than a number, the use of such data to create a command string for execution may lead to command-injection vulnerabilities.
- Since some backend components eventually call C code with potentially user-controlled data, the C code was inspected for common memory-related flaws such as buffer overflows. The RPC implementation utilized by the solution was also reviewed for memory corruption issues, as well as for more subtle issues

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

such as logical flaws that may result in data interpretation via unintended methods.
- Another sensitive area for the vast majority of web applications in general constitutes the handling of files. Thus, code paths performing file writes/reads were inspected for potential file handling issues, such as arbitrary file writes/reads or injection vulnerabilities.
- The update functionality was audited with a focus on cryptographic issues, such as signature bypasses, as well as issues that may arise by handling archive formats. Issues of this nature could constitute flawed handling of symbolic links, for example.
- Additionally, the solution was tested for common Denial-of-Service scenarios, such as file uploads that could facilitate out-of-storage situations.

The Cure53 team was provided with devices running the firmware version subject to review. Additionally, the team received multiple ExpressVPN accounts to allow for dynamic testing of the VPN services. The test setup for the dynamic tests of the frontend consisted of a device providing WAN access to the router, the router itself, and devices that were placed “behind” the ExpressVPN router - either in the Wi-Fi network or on the router’s Ethernet ports. The device providing WAN access was configured to intercept all outgoing traffic, while internal devices were configured to use a proxy, which allowed the testing team to analyze the administration interface dynamically by interception and manipulation of the HTTP traffic.

This environment, on the one hand, allowed for the analysis and verification of potential findings that arose during the manual code audit. On the other hand, the environment allowed for dynamic testing tasks such as the following to be performed during this proportion of the assessment:

- The traffic was reviewed for correct transmission of data of the configured VPN tunnel, as well as to determine the presence of any potentially unencrypted API calls performed by the router.
- Furthermore, the interface was dynamically tested for potential web-application flaws such as Cross-Site-Request-Forgery (CSRF), Cross-Site-Scripting (XSS), or other injection-related issues.
- Additionally, headers and cookies were inspected for typical issues such as absent security-related cookie flags.

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

Test Coverage for WP2: ExpressVPN router firmware

The installed firmware consists of a base image based on an OpenWRT 15.05.1 distribution, including ExpressVPN specific add-ons to configuration as well as additional binaries. As part of the review of the configuration and firmware structure, the following tasks were executed. Please note that the enumerated list offered below is non-exhaustive:

- The testing team commenced this assessment with a review of the overall router firmware structure, including available executables and program versions, in order to understand the possible implications of vulnerabilities with respect to remote code execution or shell exploitation, as well as other potential vulnerabilities.
- Furthermore, mount points, running services, as well as their respective open ports and network configuration (routes and firewall rules) were inspected in order to increase familiarity with the target device and possible attack vectors.
- Following this initial review, a highly-focused assessment of the firewall configuration was performed since it constituted one of the most promising attack vectors.
- Additionally, the DHCP client utilized and DNS resolve logic were analyzed, due to their specific exposure to the WAN interface.
- Finally, an extensive in-depth review of potential exploitation strategies and deanonymization scenarios was performed in order to identify the volume of information that could be gathered via the detected vulnerabilities.

The review itself included extensive assessments to determine any presence of the following vulnerability types:

- Information leakage in firewall
- Open services
- Access of internal networks
- Denial of Service attacks against the router
- Command injection
- Insecure configuration
- Memory corruption

The provided repositories contain code of third-party projects, binary blobs, and ExpressVPN product-specific code. This covered most of the router's firmware relevant to this assignment. The code includes Shell, C, Lua, and various scripting languages, as well as configuration files and code archives. Additionally, another repository for the updating logic *xvclient* was provided during the assessment period that contained updating logic in C++.

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

The assessment focused on the code and binaries specific to ExpressVPN as established in the scope provided by the customer. The review process consisted of the following tasks; please note that this enumerated list is non-exhaustive:

- General acquaintance with both the code and system in question, as well as the threat model and scope of the assessment in order to increase familiarity with the product in question.
- A cursory review of the provided code base with respect to general issues and best practices as well as project structure, with a focus on areas whereby remote exploitation may be viable. The team gained a greater understanding of the system's architecture, including internal proprietary components and the interplay with third-party components.
- Where viable and reasonable, a review of the external components exposed to the network was performed. Note that this excludes outdated and possibly exploitable binaries found on the system, such as many BusyBox tools, as long as they had not been deployed in the system.
- A specific review of the network endpoints and its publicly-reachable interfaces was performed if they had not already comprised a previous step of the configuration review.

Overall, the scope included – but was not limited to – a review against the following list of vulnerabilities:

- Remote Code Execution
- Memory corruption
- Hard-coded keys
- Unauthenticated and unauthorized access
- Injection attacks
- Denial-of-Service scenarios
- Man-in-the-Middle attacks
- Timing-related issues and race conditions
- Information leakage of sensitive and identifiable information
- Outdated cryptography

Please be aware that only the provided source code was subject to analysis by the Cure53 team. No reverse engineering or binary analysis of other components was performed during this engagement.

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

The Cure53 team received dedicated hardware for the purpose of assessing both the ExpressVPN router image and accounts to utilize the built-in VPN connection technology. All attacks instigated focused on breaching the router itself or the local network behind it. The following testing scenarios were considered:

- **LAN-internal perspective**: A quick review of attacks that may be possible for attackers that already have access to a private LAN network with full access to devices therein, as well as any possibility to send malicious packets.
- **Restricted LAN-internal perspective**: This assumes that the attacker only has restricted primitives but can send packets from/to the internal LAN. This corresponds to a classic malicious host/website browsed by a client in the LAN. This website may use dynamic content (e.g. JavaScript) to send TCP (or, even more restricted, UDP via WebRTC) packets from the client's system to other devices in the local network.
- **External perspective**: This corresponds to an attacker that has not yet gained access to the LAN, but can send packets to the WAN interface of the router. This may explicitly include a malicious host, or the ISP or state actors. Here, assessments were initiated to determine whether any escalation of primitives is possible from this position, for example via connection to the devices within the LAN directly or pretending to be part of the LAN.
- **Restricted external perspective**: This corresponds to an attacker that does not have WAN access but can send/reply to the VPN interface of the router. Testing was conducted to confirm whether one could enumerate learnings concerning the public WAN IP, as well as the same scenarios offered in the previous bullet point.

Additionally, combinations of the scenarios provided above were also considered. For instance, attackers may be able to both execute JavaScript via the browser on the client and send packets to the WAN or VPN interface.

Limitations

Ultimately, one should perceive the outcomes of this testing engagement with the following constraints in mind:

- The test did not cover a source code review of components not included in the source code archives, and only a cursory review of external components if viable.
- Only vulnerabilities that allow remote exploitation were investigated in depth.
- Furthermore, information leakage that is mitigated by using the VPN connection was considered out-of-scope and related findings marked as such.
- The assessment did not cover reverse-engineering or a binary review of any of the external components.

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

Identified Vulnerabilities

The following sections list all vulnerabilities and implementation issues identified throughout the testing period. Please 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. Furthermore, each vulnerability is given a unique identifier (e.g., *EXP-07-001*) to facilitate any future follow-up correspondence.

EXP-07-003 OOS: String matching in firewall rules facilitates DNS infoleak *(Medium)*

***Fix Note**: The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

Whilst reviewing the firewall rules, a rule was discovered allowing any UDP packets incoming on the WAN interface to be forwarded to the local DNS resolver.

The routing mechanism has a *cove_prerouting* chain in the PREROUTING table containing a rule to match any UDP packet containing the string *expressvpnrouter* and forward it to port 53, namely *dnsmasq*.

This behavior can be leveraged to send specially-crafted UDP packets to any port on the device. As long as their payload contains the string *expressvpnrouter,* they will be forwarded to the DNS resolver. An attacker may exploit this by sending malicious DNS requests to the WAN (or LAN) port of the device, which contain a legitimate request to a domain such as *www.example.com* with another field containing the magic string. Such a request will be answered by the DNS resolver and sent to the attacker's system, originating the DNS request.

Depending on the respective configuration, this may lead to information leakage: suppose a user is running a DNS server in their internal network, which provides internal IP addresses and/or host names. Such information could be queried by attackers on the router's WAN interface. Furthermore, it may be possible to deduce the router's DNS cache contents by measuring the response timing. Notably, the direct exposure of the router's dnsmasq server to the Internet increases the system's overall attack surface - for instance, in the eventuality an exploitable vulnerability in dnsmasq is identified.

This rule comprises the following MoonScript file:

Affected file:
xv_router_firmware/meta-app/cove-apps/files/firewall/iptables.moon

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

Affected code:

```
@_acc "-A cove_prerouting -p udp -m string --string expressvpnrouter --algo bm -j REDIRECT --to-port 53"
```

As indicated by the firewall rule provided above, the system allows an attacker to perform DNS lookups on the router's WAN port. The behavior can be verified by using the dig tool, as outlined in the excerpt below:

```
$ dig @$WAN_IF -r -p 1234 +nocmd +noquestion +nostats +noedns
expressvpnrouter.com
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56283
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; ANSWER SECTION:
expressvpnrouter.com.    0    IN    A    192.168.132.1
$ dig @$WAN_IF -r +nocmd +noquestion +nostats +noedns expressvpnrouter.com
;; communications error to 192.168.178.39#53: connection refused
```

Notably, on destination port 1234 the DNS request for *expressvpnrouter.com* is indeed answered. However, on port 53 the connection is refused. This is consistent with the firewall rule that will redirect all packets (independently of their destination port) to port 53 if they contain the string *expressvpnrouter*. However, packets directly targeting port 53 are dropped.

In order to be able to exploit this issue, an attacker would likely want to be able to perform queries for arbitrary domain names (i.e., domain names not including the string *expressvpnrouter*). To achieve this goal, the *expressvpnrouter* string can be placed outside the DNS query section in a DNS packet. One possible area for this constitutes the Additional Records section.

Utilizing the PoC attached, one can send DNS queries of this nature to the WAN interface UDP 1234 (or LAN UDP 53 or other ports) and observe that these are indeed processed and answered:

PoC:

```
$ ./dns.py
Sending DNS packet:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 0
;; flags: rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; QUESTION SECTION:
;www.secfault-security.com.      IN     A
;; ADDITIONAL SECTION:
expressvpnrouter.          1337   IN     A      13.37.13.37
-------------------------------------------
```

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

```
Received DNS packet:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 0
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; QUESTION SECTION:
;www.secfault-security.com.       IN     A
;; ANSWER SECTION:
www.secfault-security.com. 5687   IN     A      78.47.254.29
```

tcpdump trace:

```
14:40:26.590683 IP 192.168.178.67.43967 > 192.168.178.39.1234: UDP, length 75
14:40:26.591020 IP 192.168.178.39.1234 > 192.168.178.67.43967: UDP, length 59
```

A PCAP trace of this exchange, captured from the client, is also attached, next to the code to reproduce this behavior.

Notably, the testing team also assessed whether this works through the VPN. To achieve this, the testers hosted a server that continuously sends a previously-captured DNS request to *expressvpnrouter.com* with a client connected via the VPN. However, in this case the tester's server only receives the messages sent by the client and no DNS response, while the DNS request to the client is not intercepted by the router in this setup. This behavior can be confirmed from tcpdump again, with the initial connection from the LAN to the external server which is routed over the VPN entry point, as well as the server's answer to the VPNs public IP back to our client in the LAN. This behavior does not alter, regardless of the strings the UDP packet contains. As such, this finding remains unexploitable via VPN.

There are several methods by which one can address and mitigate this issue. If the firewall rule should not be considered a business requirement, one could remove it entirely. In the eventuality it is required, this issue can be mitigated by matching only packets originating from the router's LAN interface. If matching packets from the WAN interface should be required in addition, a thorough matching logic could be implemented to ensure that only DNS requests containing one (and only one) query for the *expressvpnrouter.com* domain are processed.

EXP-07-004 WP2: WAN packet forwarding to LAN interface *(Medium)*

***Fix Note**: The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

Whilst reviewing the system's firewall rules, the discovery was made that the default policy of the *FORWARD* chain is set to *ACCEPT*. Whilst the *zone_wan_dest_REJECT* rule in the *zone_wan_forward* chain rejects all previously-unmatched packets on the WAN port on the basis of their outgoing interface, packets that are incoming on the WAN port but route out to the *br-lan* interface will be forwarded.

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

Steps to reproduce:

1. Connect the router WAN port to an upstream device with a DHCP daemon running to assign it a WAN IP. Establish the router as a gateway, e.g., with *192.168.178.10* constituting the router's WAN IP and *enp0s31f6* the Ethernet interface of the upstream device running the DHCP daemon for subnet *192.168.178.1/24*:

   ```
   192.168.178.10 dev enp0s31f6 proto static src 192.168.178.1 metric 50
   ```

2. Add a rule to route all IP packets directed to the router's subnet *192.168.132.1/24* to this gateway:

   ```
   192.168.132.0/24 via 192.168.178.10 dev enp0s31f6 proto static metric 40
   ```

3. Observe that a testing device connected to the router's LAN may now be reached via the WAN port from the upstream device, assuming the IP is known:

   ```
   $ ping 192.168.132.150
   PING 192.168.132.150 (192.168.132.150) 56(84) bytes of data.
   64 bytes from 192.168.132.150: icmp_seq=1 ttl=63 time=117 ms
   64 bytes from 192.168.132.150: icmp_seq=2 ttl=63 time=34.0 ms
   …
   --- 192.168.132.150 ping statistics ---
   2 packets transmitted, 2 received, 0% packet loss, time 1002ms
   rtt min/avg/max/mdev = 33.980/75.300/116.621/41.320 ms

   $ traceroute 192.168.132.150
   traceroute to 192.168.132.150 (192.168.132.150), 30 hops max, 60 byte
   packets
    1  * * *
    2  * 192.168.132.150 (192.168.132.150)  71.466 ms  71.450 ms
   ```

   Notably, the upstream device is *not* part of the router's LAN or in any way connected to it.

This issue may be combined with the finding documented in ticket EXP-07-003, if such devices are known to the router's DNS server. To mitigate this issue, Cure53 advises reviewing the firewall rules, with particular care taken to exclude packets destined to internal networks and incoming from the eth0 interface.

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

EXP-07-006 WP2: DoS via unauthenticated file uploads *(Medium)*

***Fix Note**: The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

Whilst reviewing the implementation of the HTTP server on the target system, the discovery was made that the system accepts and handles file uploads before it authenticates the user by verifying the provided API key. This can be observed in the code excerpt offered below:

Affected file:
xv_router_firmware/meta-app/cove-apps/files/uapi/http/Request.moon

Affected code:

```
read_luahttp: (headers, stream)=>
 Writer = require "uapi.http.Writer"

 @set_luahttp_headers headers

 if "POST" == @_method
        ct = @_headers["content-type"] or ""

        if ct\match "^application/json"
               @_body = stream\get_body_as_string get_body_timeout
               status, @_body = json.decode @_body
               return false unless status

       elseif ct\match "^multipart/form%-data"
               -- hf = io.open "/tmp/raw.upload.headers", "w"
               -- for k, v in pairs @_headers
               --     hf\write "#{k}=#{v}\n"
               -- hf\close!

               file = Writer ct
               stream\save_body_to_file file, 300
               assert file\status!, file\error!

               @_body = nil
               @_file = file\filename!

               @_file_writer = file
```

A remote attacker could exploit this issue by sending repeated requests containing large files for upload to the target system. Whilst dynamically verifying this issue, the observation was made that uploading excessively large files renders the web interface

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

unresponsive. This could lead to a DoS condition, which may result in the router becoming unavailable to the user in question.

To mitigate this issue, Cure53 recommends verifying the API key before handling file uploads. During the review, no code paths were identified that would require being able to handle file uploads without prior authentication.

Miscellaneous Issues

This section covers any and all noteworthy findings that did not lead to an exploit but might 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 way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

EXP-07-001 WP2: Symlink attack in implemented TAR extraction *(Medium)*

***Note from ExpressVPN**: This issue has no impact on the strong security posture of the Aircove router. Exploitation of this issue requires an attacker to be able to provide a malicious update file, and ExpressVPN has implemented multiple checks to ensure that Aircove updates are only downloaded from a trusted ExpressVPN source. A fix will be applied in the near future for defense in depth.*

Whilst reviewing the firmware update logic, the discovery was made that it is vulnerable to a potential symlink attack in its archive extraction logic. The implementation of the signature verification functions as follows:

- A number of files are extracted from the firmware tar archive. Files of this nature include the actual firmware binary signature and a CA certificate, for instance.
- These files are then utilized to perform the firmware's signature validation.

The code offered below is utilized for the purpose of file extraction from the archive.

Affected file:
xv_router_firmware/meta-app/cove-apps/files/components/autoupdate/downloader.moon

Affected code:

```
sig_file = "#{fw_file}.sig"
  process.await "tar -xvf #{filename} #{sig_file} -C #{target_folder}"
  signature_path = "#{target_folder}/#{sig_file}"
```

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

Here, the testing team observed that the code simply extracts the tar file entry, which matches the provided file name. However, this process lacks an essential check to ensure the extracted entry constitutes an actual file. Hence, the code offered above could extract directories from the archive in addition.

Such directories could also include their respective sub-entries, which may contain symbolic links. Notably, whilst the code attempts to clean up the extracted files following the verification, this cleanup functionality relies on the *os.remove* function, which will not operate on non-empty directories. Therefore, if the extracted tar archive contained a directory, this directory would persist in the file system. As a result, one could place a directory hierarchy such as the following:

/common/firmware.sig/hacked -> /etc

In this example, *hacked* is a symbolic link to the */etc* directory.

This process can be problematic when attempting to extract further archives. Assume that a second extraction process has been initiated and operates upon a new tar file that again contains an entry such as *firmware.sig/hacked/passwd*. Extracting this entry would result in overwriting the system's */etc/passwd* file, which would likely lead to a full system compromise.

Notably, this behavior does not require a valid signature to successfully abuse. The relevant steps (extraction of tar file entries) are performed before the signature check is even initiated.

To mitigate this issue, Cure53 recommends desisting entry extraction from archives that have not been subject to comprehensive signature verification beforehand. If possible, the firmware file should be accompanied by a separate signature file, which could then be utilized without the need to perform an archive extraction.

EXP-07-002 WP2: Key reuse between GCM and CBC API keys *(Low)*

***Note from ExpressVPN**: ExpressVPN and Cure53 agree that there is no security impact here, and that this finding has no impact on the strong security posture of the Aircove router. A fix will be applied in the near future for defense in depth.*

Whilst reviewing the implementation of the API token handling code, the observation was made that two token versions are supported: an "old" format that relies on encrypting a known payload with AES in CBC mode, as well as a "new" format that relies on AES-GCM. Both implementations, however, share the same cryptographic key for

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

performing the symmetric encryption operation. This is highlighted in the code excerpt offered below:

Affected file:
xv_router_firmware/meta-app/cove-apps/files/utils/token.moon

Affected code:

```
decrypt = (a, old)->
   return "" if "string" != type a

   key   = get_key!
   bin   = base.decode a
   return "" if not bin

   -- allow old token verification for only ten minutes after reboot
   -- it works only on info API and upgrades token
   if old and utils.monotime! < 600
       iv     = bin\sub 1, 16
       ctext = bin\sub 17
       return "" if ctext\len! < 1

       cip    = cipher.new "aes-128-cbc"
       ctx    = cip\decrypt key, iv
       token  = ctx\final ctext

       return token or ""

   iv  = bin\sub 1, 12
   tag   = bin\sub 13, 28
   ctext = bin\sub 29
   return "" if ctext\len! < 1

   cip = cipher.new "aes-128-gcm"
   ctx = cip\decrypt key, iv

   ctx\tag tag
   token = ctx\final ctext
   token or ""
```

Here, one can observe that both the “old” and new token format utilize the same symmetric key. This is considered a negative practice and should be avoided if possible. This owes to the fact that CBC is generally malleable and does not include an authentication token like GCM. Therefore, if an attacker is able to construct a token that could be successfully decrypted using CBC without having access to the underlying key, this would - from a cryptographic perspective - constitute a breach of the token scheme.

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

Indeed, an attack of this nature may be possible under a number of assumptions. Assume an attacker had retrieved access to a valid GCM-encrypted token and also had prior knowledge of the token's plain text representation (which follows a fixed format primarily consisting of two timestamps).

GCM internally operates in CTR mode, which means that it encrypts incrementing NONCEs using the utilized cipher (AES in this case). The NONCE values used here are derived from the provided IV, which is part of the encrypted token. Given a valid GCM-based token and its plaintext, an attacker can therefore derive the NONCE values used in GCM's encryption, as well as the CTR keystream that results from encrypting such NONCEs using AES. This behavior would grant the attacker a pair of *(NONCE, AES_Encrypt(NONCE))* values. Pertinently, such information can directly be used to produce a valid CBC ciphertext block that decrypts to an arbitrary plaintext. This would function successfully since CBC uses its initialization vector differently to GCM. For CBC, the initialization vector is XORed to the decryption result of the first ciphertext block. If the decryption result of this block is known in advance, the IV can be adjusted to produce arbitrary plaintexts. Now, with the knowledge of a *(NONCE, AES_Encrypt(NONCE))* pair, an attacker can achieve exactly that: compute *IV = NONCE XOR Desired_Plaintext*, and provide *IV + AES_Encrypt(NONCE)* as CBC ciphertext.

Notably, however, the analyzed implementation does not appear to be affected by this issue since it will disallow tokens that have not been explicitly allow-listed by the router. Nevertheless, the analyzed construction still constitutes a negative practice and should be replaced by an implementation that solely relies on GCM if possible.

EXP-07-005 WP2: Lack of return value check on *setsockopt* *(Low)*

***Note from ExpressVPN**: ExpressVPN and Cure53 agree this issue has no impact on the security posture of the Aircove router or it's overall operation. A fix will be applied in the near future for defense in depth.*

Whilst reviewing the *110-add-dnsmasq-fwmark.patch* file, the observation was made that the code does not check the return value of the *setsockopt()* calls, which is made to add *fwmarks* to outgoing packets. This is considered a negative practice since the *setsockopt()* call may fail due to insufficient memory, for instance, according to the official documentation. Subsequently, the *setsockopt()* call may not mark the outgoing packets correctly. However, one should note that this condition may be considerably difficult for an attacker to achieve. As a result, this issue can only be assigned a *Low* miscellaneous severity rating.

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

Affected file:
xv_router_firmware/meta-open-apps/dnsmasq/files/patches/110-add-dnsmasq-fwmark.patch

Affected code:

```
--- dnsmasq-2.86.orig/src/forward.c
+++ dnsmasq-2.86/src/forward.c
@@ -451,6 +451,9 @@ static int forward_query(int udpfd, unio
       if ((fd = allocate_rfd(&forward->rfds, srv)) != -1)
         {

+             if (daemon->fwmark)
+             setsockopt(fd, SOL_SOCKET, SO_MARK, &daemon->fwmark,
sizeof(unsigned int));
+
```

In order to provide defense-in-depth and strengthen the overall security foundation of the implementation, Cure53 recommends checking the return value of the *setsockopt()* call. In the eventuality it is not successful, one should either retry or signal this error condition without actually allocating the socket.

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

Conclusions

The impressions gained during this report - which details and extrapolates on all findings identified during the CW25 to CW27 testing against the ExpressVPN router by the Cure53 team - will now be discussed at length. To summarize, the confirmation can be made that the components under scrutiny have garnered a positive impression, though some improvements can be made to further elevate the platform to an exemplary security posture.

Communication was achieved via a shared Slack channel, cross-team queries regarding certain findings and functionality were promptly answered, and the engineering team provided immediate assistance to the testing team when required. This was particularly welcome in situations whereby application flows or technical issues were initially difficult to understand.

Whilst discussing the threat model with ExpressVPN, the testing team noted that the threat model does not include attackers that are able to obtain valid X.509 certificates for arbitrary host names. For a nation-state attacker, this might generally be a possibility by taking legal action against a CA, taking control over the target's DNS zone, or by mounting BGP-based attacks, for example. Therefore, from the auditors' perspective, security against nation states may not be fully assured.

By using a modern framework (Vue.js) for the frontend of the router UI and following best practices, a number of vulnerability classes such as XSS were avoided by default.

Generally speaking, in relation to web-application flaws, evidence certainly suggests that the ExpressVPN team is acutely aware of common web-application flaws and able to successfully mitigate them. However, a handful of potentially dangerous patterns were identified during the audit. For example, string concatenation using potentially user-controlled data to generate external commands may lead to vulnerabilities in the future and should be avoided through use of dedicated libraries alternatively.

Overall, the code organization and quality garnered a positive impression, indicating that security was a high priority during development and a substantial aspect of the software's life cycle.

Regarding the networking configuration, the testing team is keen to stress that this area would benefit from hardening improvement to further bolster the solution's security, particularly concerning the firewall rules. During this engagement, a number of potential scenarios which may facilitate deanonymization of users - via exploitation of issue EXP-07-004 in part - were identified.

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

However, these could not fully be evaluated during the time frame of this project. Nevertheless, these scenarios may be of interest for future audits or internal analysis; three of which are described in the following enumerated list:

- In this first scenario, an attacker may send packets destined for the internal LAN to the WAN interface. Such packets may target “smart” devices within the LAN, or any device reacting to network requests such as MQTT for that matter. The device would subsequently open a connection to an attacker-controlled server. The notion here is that the attacker can associate multiple different VPN IPs together or VPN IPs with “public” WAN IPs, since the router is able to group multiple devices into using different VPNs (or no VPN at all).
- Similarly, the packet is sent to the internal LAN but already constitutes a connection request (or response). Rather than rely on a complex mechanic wherein an MQTT packet signals a smart device to search for “www.evil.com”, for instance, we directly tell the device that it has an incoming connection request from the malicious host, hoping that the device will respond through the VPN and thus again establish an association. An existing connection may also be used whereby the attacker responds through the WAN IP instead of the VPN IP to verify a guessed association.
- Finally, without requiring the aforementioned firewall bypass, both scenarios may be considered from a restricted internal perspective. Rather than bypass the firewall, we use a website browsed by the client to run JavaScript code from within the network and establish connections to the “outside world”.

Additionally, one could potentially leverage the issue described in EXP-07-003 for DNS spoofing of VPN-ed devices, in the eventuality an attacker can send DNS requests from the WAN with a fake sender address chosen from the LAN IP range. Aside from these aforementioned weaknesses, all identified issues thankfully do not affect the actual core feature itself (the network-wide VPN solution). Generally speaking, the code was deemed soundly composed and the configuration adheres to typical best practices. However, configuration issues do persist and old third-party components continue to be leveraged, core features notwithstanding.

Whilst these do not directly lead to vulnerabilities, an attacker may be able to exploit these associated weaknesses at a later date and as such should be avoided where possible. These components should be updated or avoided in general; likewise, the firmware should be stripped to only include a minimal amount of binaries required for the system to operate as intended.

Cure53 would like to thank Brian Schirmacher and Harsh S. from the ExpressVPN team for their excellent project coordination, support and assistance, both before and during this assignment.