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 TrustedServer 04.-05.2022

Cure53, Dr.-Ing. M. Heiderich, Dr. A. Pirker, R. Weinberger, Dipl.-Ing. D. Gstir, D. Oberhollenzer

Index

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

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

Introduction

"The most advanced VPN server technology, independently audited to confirm essential privacy protections. It's ExpressVPN TrustedServer.

- *All data wiped with every reboot, as VPN servers run on RAM only*
- *Servers never write to the hard drive, further minimizing data risk*
- *The entire software stack is reinstalled on every server at startup*
- *We know what's running on every server, with no inconsistencies"*

From https://expressvpn.com/features/trustedserver

This report describes the results of a security assessment of the ExpressVPN complex, spanning the ExpressVPN TrustedServer setup and sources. Carried out by Cure53 in April 2022, the project included a dedicated audit of the source code and a broader look into the security posture exposed by the ExpressVPN components in scope.

Registered as *EXP-06,* the project was requested by ExpressVPN in January 2022 and then scheduled for the second quarter of 2022 to allow sufficient time for test-preparations on both sides. Even though this audit marks the sixth cooperation between ExpressVPN and Cure53, the items included in this scope were not part of any previous investigation, hence they received security-centered attention from the Cure53 testers for the first time.

As for the precise timeline and specific resources allocated to *EXP-06*, Cure53 completed the examination in CW16 and CW17 of 2022. A total of thirty-two days were invested to reach the coverage expected for this assignment, whereas a team of five senior testers has been composed and tasked with this project's preparation, execution and finalization. For optimal structuring and tracking of tasks, the work was split into two separate work packages (WPs):

- **WP1**: Source-code assisted penetration tests against primary scope items, with the full audit of the *xv_pleco_server, xv_pleco_automake* and *xv_pleco_deployer*
- **WP2**: Source-code assisted penetration tests against secondary scope items which encompassed privacy-related logging, file-writing & data processing claims made by ExpressVPN

It can be derived from above that white-box methodology was utilized. Cure53 was given access to a test-environment, server access documentation, test-user-accounts, as well as a modified client that could be used to connect to the environment/server in scope. Additionally, source code was provided to make sure the project can be executed in line

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

with the agreed-upon framework. In general, all necessary means of access and components were timely furnished to Cure53 by the ExpressVPN team. The project progressed effectively on the whole. All preparations were done in CW15 to foster a smooth transition into the testing phase. Over the course of the engagement, the communications were done using a private, dedicated and shared Slack channel. The channel discussion could be joined by all involved personnel and connected the Slack workspaces of the Cure53 and ExpressVPN teams.

The discussions throughout the test were very good and productive and not many questions had to be asked. The scope was well-prepared and clear, greatly contributing to the fact that no noteworthy roadblocks were encountered during the test. Cure53 offered frequent status updates about the test and the emerging findings. Live-reporting was not requested but Cure53 sent daily lists of new findings. The Cure53 team managed to get very good coverage over the WP1-WP2 scope items. Among twenty-nine security-relevant discoveries, four were classified to be security vulnerabilities and twenty-five to be general weaknesses with lower exploitation potential. None of the findings in fact related to WP2.

From one perspective, the number of findings is quite large and could be seen as worrisome at first glance. However, it needs to be clearly underlined that the ratio of vulnerabilities to hardening-driven items is very good. In other words, mostly general weaknesses and minor flaws were spotted. Further, most of them can be evaluated as trivial to fix and resolve. It can be positively acknowledged as well that none of the four actually identified vulnerabilities was ranked with a *High* or *Critical* severity score, showcasing an already quite robust environment exposed by the ExpressVPN TrustedServer components. More detailed evidence on the matter of the overall security posture can be consulted throughout the report.

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. A dedicated and extensive chapter on test methodology and coverage then clarifies what the Cure53 team did in terms of attack-attempts, coverage and other test-relevant tasks.

Next, all findings will be discussed in grouped vulnerability and miscellaneous categories, then following a chronological order in each group. Alongside technical descriptions, PoC and mitigation advice are supplied when applicable. Finally, the report will close with broader conclusions pertinent to this April 2022 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 ExpressVPN complex - precisely for the TrustedServer setup and server - are also incorporated into the final section.

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

Scope

- **Code audits & Security assessments of the ExpressVPN TrustedServer setup & sources**
  - **WP1**: Source-code assisted penetration tests against primary scope items
    - Source code packages (*.zip*) files have been shared with Cure53.
    - Test builds have been shared with Cure53.
    - SSH access to the testing environment has been granted.
    - A trimmed *diff* file containing new additions to OpenVPN has been provided.
  - **WP2**: Source-code assisted penetration tests against secondary scope items
    - Source code packages (*.zip*) files have been shared with Cure53.
    - Test builds have been shared with Cure53.
    - SSH access to the testing environment has been granted.
  - **Scope**
    - **Primary focus areas**
      - The primary objective for the assessment, alongside checks for general vulnerabilities, was to assess the Pleco OS / TrustedServer (but not the the deployment and activation processes) for the following issues:
      - Information disclosure or IP address leakage of client devices
      - Unauthorized access or remote code execution (RCE)
      - Vulnerabilities or weaknesses in how TrustedServer is configured
      - TrustedServer supports various VPN protocols. Each protocol was reviewed for vulnerabilities that may expose information or compromise users' privacy.
      - The protocols supported and in scope were as follows:
        - Lightway (UDP and TCP)
        - OpenVPN (UDP and TCP)
        - IKEv2
        - L2TP/IPSec
      - The assessment also focused on areas that have been written, created, customized or specifically configured by ExpressVPN.
    - **Secondary focus areas**
      - TrustedServer is supposed to be built in accordance with the general privacy policy enforced by ExpressVPN. This policy states that user logs or user information is neither collected nor stored beyond what is necessary to run a VPN service. The information that is in fact collected shall be consistently anonymized and aggregated.
      - A review was hence performed, aiming to validate that identifiable user information is never collected or written to disk on TrustedServer.
  - **Detailed test-supporting material was shared with Cure53**
  - **All relevant sources were shared with Cure53 for auditing**

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

Glossary & Clarifications

The following information is intended to help the reader understand the report wording better, especially with regard to the ExpressVPN-specific terminology. The section was added upon an explicit request made by ExpressVPN.

- *lightway* - this term refers to ExpressVPN's custom VPN protocol
- *client activities* - this term refers to processes/functions for ensuring that analytics and server load management are appropriately performed to support quality of service for users. Collection or usage of PII is explicitly not permitted in this realm.
- *username* - in the context of TrustedServer and VPN usage, this is a random alphanumeric string which is not to be confused with an identifiable username of any kind.

Additional clarification is needed in terms of the L2TP support:

- L2TP/IPsec provide weak security benefits and should only be used for anonymization or for changing locations.
- Most TrustedServer instances do not support L2TP.
- A warning is shown to users who rely on the manual L2TP configuration or who configure it in the Windows v10 application.

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-06-007	WP1: Known PSK allows MitM attack on L2TP	*Info*
EXP-06-018	WP1: Unprotected Local OpenVPN Management Interface	*Medium*
EXP-06-019	WP1: Client Activity Forgery on Connect and Disconnect	*Low*
EXP-06-027	WP1: Credential Files readable by all Service accounts	*Medium*

Miscellaneous Issues

ID	Title	Severity
EXP-06-001	WP1: Missing return value check for snprintf	*Info*
EXP-06-002	WP1: Missing return value check for clock_gettime	*Info*
EXP-06-003	WP1: Python urllib3 SSL warnings explicitly disabled	*Low*
EXP-06-004	WP1: Git SSH StrictHostKeyChecking Disabled	*Info*
EXP-06-005	WP1: Support for TLSv1 for OpenVPN	*Medium*
EXP-06-006	WP1: Default-Allow on OpenVPN Authentication Failures	*Low*
EXP-06-008	WP1: strongSwan w/o Perfect-Forward-Secrecy Possible	*Info*
EXP-06-009	WP1: Weak Cryptographic Algorithms in strongSwan	*Info*
EXP-06-010	WP1: Username enumeration for OpenVPN Authentication	*Low*
EXP-06-011	WP1: No rate-limiting for OpenVPN Authentication Plugin	*Low*
EXP-06-012	WP1: Memory Leak in Error Path	*Info*
EXP-06-013	WP1: Undefined Return Value of Function	*Info*
EXP-06-014	WP1: OpenVPN Auth Plugin Built with Compiler Warnings Disabled	*Info*
EXP-06-015	WP1: Race between Online State and Firewall Setup	*Info*
EXP-06-017	WP1: Potential weak random numbers for DH param generation	*Info*
EXP-06-020	WP1: AppArmor / Seccomp Missing	*Info*

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

EXP-06-021	WP1: Dual-Use of TLS keys for OpenVPN and Helium	*Info*
EXP-06-022	WP1: Unrestricted Outbound Connections Possible	*Info*
EXP-06-023	WP1: TrustedServer Hardening Weaknesses	*Info*
EXP-06-024	WP1: Possible MiTM attack against Ansible Workflows	*Low*
EXP-06-025	WP1: Limited security from OpenVPN tls-auth feature	*Info*
EXP-06-026	WP1: Known, unfixed vulnerability in OpenSSH	*Info*
EXP-06-028	WP1: Missing return value check for calloc()	*Info*
EXP-06-029	WP1: OpenVPN server binary lacks Stack Canaries	*Low*

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

Test Methodology

This section documents the testing methodology applied during this engagement and sheds light on various areas of the TrustedServer application complex. The tests were performed both by reviewing the provided source code and by dynamically examining the security posture of the TrustedServer testing instances. The section further clarifies which areas were investigated by Cure53 but did not yield any findings, alongside furnishing additional considerations around attack vectors.

Threat Model

Before covering the concrete test methodology steps, a clarification regarding the threat model is in order. ExpressVPN defined the following threat-actors:

- Nation-state actors exploiting vulnerabilities in the TrustedServer product to acquire identifying information about a user, including access to IP addresses, performing remote code execution or gaining unauthorized access through a variety of different attacks like Man-in-The-Middle or the exploitation of remote interfaces.
- Threat actors (like nation-state or cybercriminals) exploiting vulnerabilities in TrustedServer to perform remote code execution, arbitrary read and write operations and similar, critical actions on the filesystem.
- Threat actors monitoring traffic of a compromised TrustedServer to gain insights into the usage and traffic inside a VPN tunnel.

From an attacker's privileges perspective, the following two scenarios are possible:

- External attacker: An external attacker located outside of a TrustedServer instance. Such an attacker is restricted in terms of interacting with the TrustedServer through its publicly visible interfaces, namely:
  - OpenVPN
  - L2TP
  - Lightway / Helium
  - IPSec
  - NGINX (speedtest)

- Furthermore, such an external attacker may observe encrypted VPN traffic that is flowing over the wire. In order to get a foothold of the TrustedServer, an attacker must exploit a vulnerability (e.g., an RCE) within any of the exposed services.
- Attackers having restricted access to the TrustedServer: Once an attacker has been able to obtain restricted access, e.g. by exploiting a vulnerability within any of the public facing interfaces, s/he would be equipped with low-level privileges. In particular, the attacker would be restricted to the user-ID running the service

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

that has been exploited. The ultimate goal of such an attacker would be to escalate his / her privileges (either to another user-account or to the *root* user), with an ultimate goal of obtaining sensitive information pertinent to the ExpressVPN customers. This could be imagined by means of decrypting communication or leaking sensitive information.

Manual Code Review

The provided repositories are a combination of configuration files, Python scripts and a small amount of C source code. As part of the manual code review, the following tasks have been executed. Note that this list is not exhaustive.

- A review of the provided threat model and documentation of the TrustedServer complex. This was done in order to better grasp the scope of this assessment. In other words, the testers needed to become more familiar with the product subject to review.
- A review of the codebase with regard to structure, organization and best coding practices was performed. Cure53 gained a better understanding of the product's architecture and its composition, as well as the external components in use.
- A review and analysis of the external components of the TrustedServer was conducted in order to spot issues and potential vulnerabilities stemming from such external dependencies.
- A review and assessment of the publicly reachable interfaces was performed in order to identify the attack surface available to an external attacker.
- A review and analysis of the external interfaces that TrustedServer communicates to was performed in order to identify potential information leakage to external services.

The repositories in scope cover, as mentioned before, two different programming languages, namely Python and C. A review of all repositories in scope included - but was not limited to - the following vulnerabilities:

- Remote Code Execution
- Unauthenticated access
- Unauthorized access
- Impersonations
- Insecure defaults
- Insecure configurations
- Insecure permissions
- Denial-of-Service situations
- Injections to database commands
- Cross-site-scripting on template rendering

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

- Downgrade attacks
- Man-in-The-Middle attacks
- Timing attacks
- Information leakage of sensitive information
- Information leakage of identifiable information
- Memory corruptions
- Use of broken cryptography
- Use of static keys

The repositories also contain configuration files for third-party components, including OpenVPN, *strongSwan* and others. All these configuration files were reviewed for:

- Misconfigurations
- Insecure defaults
- Weak cryptography
- Outdated cryptography

Finally, the build process of the Pleco ISO server image was analyzed in order to identify issues in the build process of the TrustedServer product. As TrustedServer is based on Debian Linux, the set of essential packages installed by TrustedServer were inspected for any and all known and/or unfixed vulnerabilities.

Dynamic Testing

The Cure53 team received access to a dedicated testing environment which included three TrustedServer instances. Two of them were reachable through SSH and the third had the default configuration wherein SSH access was limited by an IP-based allow-list. Note that the server with IP-based allow list mirrors the prod set up, and that the other 2 instances specifically had SSH added to support testing. The following test scenarios were considered:

- External perspective: This corresponds to a classic attacker attempting to exploit vulnerabilities within any of the publicly available interfaces of TrustedServer. It was checked whether the attacker is able to exploit vulnerabilities on creating and establishing VPN tunnels with the array of the supported VPN providers. Also, very naive network fuzzing has been attempted for targeting OpenVPN, *lightway* and IPSec. It must be stressed though that this approach was not optimal as the fuzz testing harness was simply randomly flipping bits within packets that were sent to the referred services.
- Internal perspective: This corresponds to an attacker who already has access to a TrustedServer instance, specifically having acquired low-privileged user rights. In particular, it was verified to which information such a user has access to, which

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

interfaces s/he can reach and whether the attacker is able to escalate privileges or perform configuration changes on a TrustedServer instance.

In addition to the above-mentioned scenarios, the testing environment was also used to analyze and verify any findings made through manual code review.

Assessment Limitations

It is important to view the outcome of this testing engagement with the following constraints in mind:

- The executed test did not cover a source code review of the components listed below that run on TrustedServer.
  - OpenVPN
  - Lightway *(Note that the Lightway protocol is referred to by its internal project code-name, Helium, within the source code)*.
  - IPSec / charon daemon
- L2TP - security warning can be seen here and in the app where ExpressVPN advises users against it
- **Note**: Cure53 previously audited ExpressVPN's Lightway protocol. That report can be found here.
  The configuration provided to Cure53 was representative of a production instance with the exception of the inclusion of L2TP, which is not present on most production servers. The vast majority of TrustedServer instances do not support L2TP, so the addition of L2TP here is an exception.
- The assessment did not cover reverse-engineering or a binary review of any of the externally exposed components, as ExpressVPN and Cure53 mutually agreed that the strongest assurance would be provided by conducting a separate white-box source code review of Lightway (previous report here) and the very minimal additions to OpenVPN, which were made to improve obfuscation.
- As a result of the agreement to not do a full binary review, smart fuzzing has been performed on the aforementioned services facing the Internet of a TrustedServer instance.
- The following issues were known before the start of the audit and are, therefore, not included as separate findings here.:
  - An outdated version of *pycrypto* is currently used due to a requirement for reproducible builds. No security weaknesses affected TrustedServer through use of this version of *pycrypto*
  - Lightway and OpenVPN services share the same service account. This was a design decision taken to best support authentication to the same database, and bears very little security risk.

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

- No certificate pinning for the HTTPS requests made to mirror servers, however, very minimal risk is associated here.
- Logging of Ansible playbook commands. Ansible is used to configure elements of TrustedServer before any users are able to connect, and these initial configuration commands are logged in the system journal, but do not contain any PII or configuration secrets.
- Python dependencies with known, non-exploitable weaknesses found as dependencies of other applications:
  - *ansible 4.8.0* - The steps that are run by Ansible are run from infrastructure that is fire-walled off from the world.
  - *pip 20.3.4* - this is the version of pip currently used by Debian Bullseye
- A client's public IP address is written to the in-memory file system temporarily as part of connect and disconnect events, and is scrubbed immediately. No IP addresses or PII can be written to physical disk due to the in-ram operating system environment.

- ExpressVPN has provided attestation that these named issues are currently being remediated.

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. *EXP-06-001*) for the purpose of facilitating any future follow-up correspondence.

EXP-06-007 WP1: Known PSK allows MitM attack on L2TP *(Info)*

***Note from ExpressVPN**: ExpressVPN is currently in the process of deprecating L2TP, which will address this issue. We expect the full deprecation of L2TP to be complete by 31 October 2022.*

During a source code review of the *xv_pleco_automake* repository it was noticed that the L2TP *strongSwan* configuration uses a Pre-Shared-Key (PSK) for authentication. According to the official security recommendation[1], a PSK with low entropy for IKEv2 is vulnerable to active Man-in-the-Middle (MitM) attacks.

Since the L2TP implementation uses IKEv2 (as a default value for the *keyexchange* setting) together with PSKs, the TrustedServer L2TP implementation is potentially vulnerable to MitM attacks. Furthermore, it was noticed that the PSK is *12345678*[2] and is publicly available, specifically being shared between all clients. This setup makes it possible for an attacker to actively man-in-the-middle L2TP connections of ExpressVPN.

It means attackers could read and modify information exchanged as part of the client's communication. One can pertinently note that this issue has also been discussed with the customer during this security assessment. ExpressVPN confirmed that this risk is known and accepted. Officially, ExpressVPN discourages using this kind of communication setup for security but rather for anonymization[3], given that an attacker can potentially decrypt the VPN session[4]. See also Glossary.

Affected file:
xv_pleco_automake/etc/ipsec.d/expressvpn_l2tp.conf

Affected code:

```
conn l2tp
        [...]
        authby = psk
```

[1] https://docs.strongswan.org/strongswan-docs/5.9/howtos/securityRecommendations.html
[2] https://www.expressvpn.com/support/troubleshooting/log-items/windows-pre-shared-key/#manual
[3] https://www.expressvpn.com/what-is-vpn/protocols/l2tp
[4] https://www.expressvpn.com/what-is-vpn/protocols/ikev2

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

It is recommended to switch to EAP-based authentication, which would be aligned with the official recommendations issued for *strongSwan*[5].

EXP-06-018 WP1: Unprotected Local OpenVPN Management 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.*

During a source code audit of the *xv_pleco_automake* repository it was noticed that the OpenVPN component of TrustedServer exposes its management interface in an unprotected form. This essentially means that it is visible without enforcing password authentication via *localhost* using a TCP/IP socket.

The management interface of OpenVPN allows users to interfere with the OpenVPN processes, with the opportunities to alter behavior or query information. Therefore, any functionality exposed by the management interface can be considered sensitive and important to the operability of the OpenVPN communication.

An attacker who has low-privileged access to a TrustedServer instance (e.g., having gained it by exploiting a vulnerability within any of the external exposed interfaces), could leverage this to connect to the unprotected management interface of OpenVPN. As a consequence, they could introduce changes to OpenVPN during runtime.

Proof-of-Concept (PoC)

For this PoC, the *strongSwan* user of a TrustedServer instance was used to simulate a low-privileged user. As shown below, the *strongSwan* user is able to connect to the management interface of OpenVPN without providing credentials, thereby accomplishing a route to issuing commands.

```
strongswan@nforce-nl-pntest-01:/root$ whoami
strongswan
strongswan@nforce-nl-pntest-01:/root$ telnet 127.0.0.1 50080
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
>INFO:OpenVPN Management Interface Version 1 -- type 'help' for more info
pid
SUCCESS: pid=4382
```

[5] https://docs.strongswan.org/strongswan-docs/5.9/howtos/securityRecommendations.html

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

It is recommended to adhere to best practices outlined in the official documentation of OpenVPN[6][7]. This would include the enforcement of authentication with a password. Furthermore, it should be considered to expose the management interface only through a UNIX domain socket rather than over a network socket.

EXP-06-019 WP1: Client Activity Forgery on Connect and Disconnect *(Low)*

***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.*

During a source code review of the *xv_openvpn_auth* repository it was observed that the authentication plugin of OpenVPN supports three kinds of events emitted by OpenVPN, namely the authentication of a user, connect and disconnect. The connect and disconnect events contain information about the event, and the C plugin invoked by OpenVPN formats the string provided to the authentication plugin Python script. This format corresponds to a comma-separated string which encodes all information for the client activity component, like for example IP address or username.

One of the information items in the string is the username, which gets sent by the client application to the TrustedServer. The username gets sanitized for authentication events by checking that it contains only alphanumeric values, however, the connect and disconnect event handlers do not sanitize the provided username.

This allows an attacker to provide a specially crafted username containing commas which tamper with the format of the string expected by the OpenVPN authentication's Python script. Essentially, an attacker can provide arbitrary information through the username, thereby logging forged client activities.

Affected file:
xv_openvpn_auth/c/ovpnauth.c

Affected code:

```
int openvpn_plugin_func_v3(const int struct_version, struct
openvpn_plugin_args_func_in const *arguments, struct
openvpn_plugin_args_func_return *retptr)
{
	if(arguments->type == OPENVPN_PLUGIN_CLIENT_CONNECT_V2){
		log(PLOG_DEBUG, PLUGIN_NAME, "Handling on connect event.");

		// Get the data we need from the environment
		const char *client_ip = get_env_safe("trusted_ip", arguments-
		>envp);
```

[6] https://openvpn.net/community-resources/management-interface/
[7] https://fossies.org/linux/openvpn/doc/man-sections/management-options.rst

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

```
                const char *client_local_ip  =
                get_env_safe("ifconfig_pool_remote_ip", arguments->envp);
                const char *username = get_env_safe("username", arguments->envp);
                [...]
                // Create the CSV string and write to disk
                snprintf(context->line_buffer, LINE_BUFFER_SIZE, "on_connect,%s,
                %s,%s,%s,%s,%s,%s,%s,%s,%s,0,0,%s", client_ip, username,
                common_name, server_bind_ip, server_bind_port, platform, protocol,
                context->obfuscation_id, context->ca_version, context-
                >xor_enabled, client_local_ip);
                [...]
        }
        if(arguments->type == OPENVPN_PLUGIN_CLIENT_DISCONNECT){
                log(PLOG_DEBUG, PLUGIN_NAME, "Handling on disconnect event.");

                // Get the data we need from the environment
                const char *client_ip = get_env_safe("trusted_ip", arguments-
                >envp);
                const char *client_local_ip  =
                get_env_safe("ifconfig_pool_remote_ip", arguments->envp);
                const char *username = get_env_safe("username", arguments->envp);
                [...]
                // Create the CSV string and write to disk
                snprintf(context->line_buffer, LINE_BUFFER_SIZE, "on_disconnect,
                %s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s", client_ip, username,
                common_name, server_bind_ip, server_bind_port, platform, protocol,
                context->obfuscation_id, context->ca_version, context-
                >xor_enabled, bytes_in, bytes_out, client_local_ip);
                [...]
        }
}
```

It is recommended to sanitize the username similar to what is already done for the authentication event of OpenVPN, In essence, checks for alphanumeric usernames are necessary.

EXP-06-027 WP1: Credential Files readable by all Service accounts *(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.*

During an audit of the provided TrustedServer instances, it was found that several files containing credentials and keys are readable by all service accounts. Specifically, the following files within the */etc* directory contain either a password or a key:

client_activities.yml
icinga.yml
smartdns_creds.yml

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

telegraf.yml
vpn_auth.yml
zsync.yml
rollbar.key
strongswan.conf
dnsdist/conf.d/0100-console.conf

An attacker having low--privileged access to a TrustedServer instance could leverage this and read the content of these files, causing a secret information leakage.

Proof-of-Concept (PoC)

For this PoC, the *strongSwan* user of a TrustedServer instance was leveraged. As shown below, the *strongSwan* user is able to read the contents of the mentioned *yml* files containing sensitive information:

```
strongswan@nforce-nl-pntest-01:/etc$ whoami
strongswan
strongswan@nforce-nl-pntest-01:/etc$ ls -la *.yml
-rw-r--r-- 1 root root  121 Apr 15 00:06 aptsign.yml
-rw-r--r-- 1 root root  132 Apr 15 04:27 censorship_block_checks.yml
-rw-r--r-- 1 root root  856 Apr 15 04:27 client_activities.yml
-rw-r--r-- 1 root root   42 Apr 15 04:22 icinga.yml
-rw-r--r-- 1 root root   43 Apr 15 00:06 mirror_sync.yml
-rw-r--r-- 1 root root 1373 Apr 15 00:06 pleco.yml
-rw-r--r-- 1 root root   73 Apr 15 00:06 server_inventory.yml
-rw-r--r-- 1 root root   40 Apr 15 04:25 smartdns_creds.yml
-rw-r--r-- 1 root root   42 Apr 15 04:20 telegraf.yml
-rw-r--r-- 1 root root  326 Apr 15 04:21 vpn_auth.yml
-rw-r--r-- 1 root root   42 Apr 15 04:21 zsync.yml
strongswan@nforce-nl-pntest-01:/etc$ cat icinga.yml
username: test-value
password: <REDACTED>
strongswan@nforce-nl-pntest-01:/etc$
```

Furthermore, an attacker is able to read the *rollbar.key* file, which also contains sensitive information:

```
strongswan@nforce-nl-pntest-01:/$ whoami
strongswan
strongswan@nforce-nl-pntest-01:/$ cat /etc/rollbar.key
<REDACTED>
strongswan@nforce-nl-pntest-01:/$
```

The same applies to the *strongswan.conf* file.

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

It is recommended to properly restrict UNIX file permissions of these files. This should be done with the least-privilege principle in mind, only granting access to users that must be able to reach these files.

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.

EXP-06-001 WP1: Missing return value check for *snprintf* (*Info*)

***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.*

During a source code review of the *xv_openvpn_auth* repository, it was found that the plugin uses the *snprintf* function on several occasions. According to the official documentation, this function returns an integer which constitutes the number of characters that have been written, or a negative number indicating that the operation was not successful[8].

In fact, *snprintf* can internally use a dynamic buffer and return -1 instead of the string length if *snprintf* runs out of memory. Even though the described behavior only occurs under specific and rare circumstances, the buffer used for storing the value could potentially be cut off in the discussed cases.

Affected file:
xv_openvpn_auth/c/helpers.c

Affected code:

```
void create_auth_file_name(char *auth_file_name, const char *auth_directory,
const char *prefix) {
        [...]
        snprintf(auth_file_name, AUTH_FILENAME_MAX_SIZE, "%s/%s.%ld%.9ld",
        auth_directory, prefix, current_time.tv_sec, current_time.tv_nsec);
}
```

Affected file:
xv_openvpn_auth/c/helpers.c

[8] https://www.cplusplus.com/reference/cstdio/snprintf/

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

Affected code:

```
int openvpn_plugin_func_v3(const int struct_version, struct
openvpn_plugin_args_func_in const *arguments, struct
openvpn_plugin_args_func_return *retptr)
{
[...]
       if(arguments->type == OPENVPN_PLUGIN_AUTH_USER_PASS_VERIFY) {
              [...]
              snprintf(context->line_buffer, LINE_BUFFER_SIZE, "auth,%s,%s,%s\
              n", username, password, auth_control_file);
              [...]
       }

       if(arguments->type == OPENVPN_PLUGIN_CLIENT_CONNECT_V2){
              [...]
              snprintf(context->line_buffer, LINE_BUFFER_SIZE, "on_connect,%s,
              %s,%s,%s,%s,%s,%s,%s,%s,%s,0,0,%s", client_ip, username,
              common_name, server_bind_ip, server_bind_port, platform, protocol,
              context->obfuscation_id, context->ca_version, context-
              >xor_enabled, client_local_ip);
              [...]
       }

       if(arguments->type == OPENVPN_PLUGIN_CLIENT_DISCONNECT){
              [...]
              // Create the CSV string and write to disk
              snprintf(context->line_buffer, LINE_BUFFER_SIZE, "on_disconnect,
              %s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s", client_ip, username,
              common_name, server_bind_ip, server_bind_port, platform, protocol,
              context->obfuscation_id, context->ca_version, context-
              >xor_enabled, bytes_in, bytes_out, client_local_ip);
              [...]
       }
}
```

It is recommended to properly check the returned value of *snprintf* and ensure that cases when *snprintf* fails are detected.

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

EXP-06-002 WP1: Missing return value check for *clock_gettime* *(Info)*

***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.*

While reviewing the source code of the *xv_openvpn_auth* repository, it was noticed that the OpenVPN authentication plugin uses the time of the *CLOCK_MONOTONIC_RAW* clock, relying on the *clock_gettime* function returning an integer as an indicator for the success of the operation[9]. It was observed that the plugin ignores the return value of this function, thereby continuing with code execution despite failures. This could lead to unpredictable issues in the TrustedServer.

Affected file:
xv_openvpn_auth/c/helpers.c

Affected code:

```
void create_auth_file_name(char *auth_file_name, const char *auth_directory,
const char *prefix) {
	// Get a timestamp
	struct timespec current_time;
	// Use monotonic raw clock to ensure time always flows in the right
	direction
	// and that NTP can't fiddle with it
	clock_gettime(CLOCK_MONOTONIC_RAW, &current_time);
	// Combine the filename with the directory and the prefix (i.e
	/run/openvpn/auth/147878.66836249125932)
	snprintf(auth_file_name, AUTH_FILENAME_MAX_SIZE, "%s/%s.%ld%.9ld",
	auth_directory, prefix, current_time.tv_sec, current_time.tv_nsec);
}
```

It is recommended to check the return value of *clock_gettime* and perform error handling in case that the function returns an error.

EXP-06-003 WP1: Python urllib3 SSL warnings explicitly disabled *(Low)*

***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.*

During a source code review of the *xv_pleco_automake* repository, it was discovered that several Python scripts within the legacy folder explicitly disable all SSL-related warnings raised by the Python package *urllib3*[10]. This is done with the directive called *requests.packages.urllib3.disable_warnings()*.

[9] https://linux.die.net/man/3/clock_gettime
[10] https://urllib3.readthedocs.io/en/stable/

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

As also mentioned within the official *urllib3* documentation, making unverified HTTPS requests is strongly discouraged, since it may allow malicious actors to intercept and eavesdrop on communications.

Affected files:
xv_pleco_automake/legacy/report_blocked_ips.py
xv_pleco_automake/legacy/report_portspeed.py

Affected code:
```
[...]
requests.packages.urllib3.disable_warnings()
[...]
```

It is recommended to remove the *disable_warnings()* directives in order to make sure that the SSL-related warnings are issued and reported by the *urllib3* library.

EXP-06-004 WP1: Git SSH *StrictHostKeyChecking* Disabled *(Info)*

***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.*

While auditing the source code repositories in scope, it was discovered several scripts explicitly disable the SSH feature *StrictHostKeyChecking*[11]. Setting this value to *"no"* results in accepting SSH host keys from remote servers that are not on the list of known hosts.

When cloning a *git* repository, the SSH daemon automatically checks and maintains a database of identities for all hosts which have ever been used. For servers where the host key has changed or is unknown, the *ssh_config* keyword *StrictHostKeyChecking* decides on whether to check the provided host key or not. A changed host key may be an indicator that the remote server has been compromised.

Affected files:
xv_pleco_automake/bin/update_automake.sh
xv_pleco_server/modules/automake/hooks/pre-build/2010-clone-pleco-automake.hook
xv_pleco_server/modules/ip_reputation/hooks/pre-build/2010-clone-ip-reputation.hook
xv_pleco_server/modules/vpn-related/hooks/pre-build/4010-xv-client-activities.hook
xv_pleco_server/modules/vpn-related/hooks/pre-build/4020-xv-freeradius-auth.hook

[11] https://linuxhint.com/ssh-stricthostkeychecking/

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

Affected code:

```
[...]
GIT_SSH_COMMAND='ssh -i [...] -o StrictHostKeyChecking=no' \
    git pull
[...]
```

In order to eliminate the minor risk of cloning from a compromised *git* server, it is recommended to set the *StrictHostKeyChecking* configuration flag to "*yes*". It must be emphasized that the risk is minimal if not negligible here, as also reflected by the severity level *Info*.

EXP-06-005 WP1: Support for TLSv1 for OpenVPN *(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.*

While reviewing the source code of the *xv_pleco_automake* repository, it was discovered that the OpenVPN configuration sets the minimal TLS version required for the control channel of the OpenVPN tunnels. The TrustedServer configuration uses as a minimal TLS version 1.0, which is not in accordance with security best practices[12].

Affected file:
xv_pleco_automake/etc/templates/openvpn.conf.j2

Affected code:

```
[...]
tls-version-min 1.0
[...]
```

It is recommended to enforce TLS version 1.2 or newer for the control channel of OpenVPN.

[12] https://community.openvpn.net/openvpn/wiki/Hardening

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

EXP-06-006 WP1: Default-Allow on OpenVPN Authentication Failures *(Info)*

***Note from ExpressVPN**: This is the intended behavior in order to support a strong user experience. ExpressVPN and Cure53 agree there is no security impact as a result of this finding.*

During a review of the *xv_pleco_automake* repository, it was noticed that the OpenVPN plugin authenticates users via username / password. The Python script handling the password verification, however, follows a default-allow approach in case the password check logic raises an error. Under normal circumstances, the implementation should never raise an error unless the authentication database is corrupted.

One can pertinently note that this issue has also been discussed with the customer during the course of this security assessment. It was confirmed that it was a business decision to apply a default-allow approach in case of errors in the password check functionality.

Affected file:
xv_pleco_automake/pleco/cli/ovpnauth_inotify.py

Affected code:

```
def do_auth(creds_line, database_path, rate_limiter):
	[...]
	try:
		auth_response = authenticate(username, password, database_path)
	except Exception:
		write_control_file_access_granted(control_file)
		if rate_limiter.allow_call():
			rollbar.report_exception()
		return
	[...]
```

From a security perspective, it is recommended to follow a default-deny approach when performing authentication and authorization checks.

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

EXP-06-008 WP1: *strongSwan* w/o Perfect-Forward-Secrecy Possible *(Medium)*

***Fix Note**: The issue was partly addressed by the ExpressVPN team and the fix for IKEv2 was verified by Cure53 who were able to review the related diff & PR. Only the L2TP issue remains.*

***Note from ExpressVPN**: ExpressVPN is currently in the process of deprecating L2TP, so L2TP/IPSec will be deprecated. We expect the full deprecation of L2TP to be complete by 31 October 2022. Officially, ExpressVPN discourages using this kind of communication setup for security but rather for anonymization, as described here.*

While reviewing the source code of the *xv_pleco_automake* repository, it was observed that the *strongSwan* configurations of *expressvpn_l2tp.conf* and *ipsec_express-vpn.conf.j2* both contain ciphers for the *esp* property which does not offer Perfect-Forward-Secrecy[13]. According to the official security guidelines, it is strongly recommended to use perfect forward secrecy in order to negotiate an independent session key for each IPsec or child security association[14].

Not having perfect forward secrecy in place allows an attacker to potentially decrypt previous communications when getting a hold of key material used to protect the IPSec communication.

Affected file:
xv_pleco_automake/etc/ipsec.d/expressvpn_l2tp.conf

Affected code:
```
conn l2tp
       [...]
       esp = "aes256-sha256-modp2048,\
              aes256-sha256-modp1536,\
              [...]
              aes256-sha1,\
              aes256-sha1,\
              aes128-sha256"
```

Affected file:
xv_pleco_automake/etc/templates/ipsec_expressvpn.conf.j2

Affected code:
```
conn ios-production
       [...]
       esp = "aes256-sha1-modp1536,\
```

[13] https://en.wikipedia.org/wiki/Forward_secrecy
[14] https://docs.strongswan.org/docs/5.9/howtos/securityRecommendations.html

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

```
                aes256-sha1-modp1024,\
                aes256-sha1,\
                aes128-sha256"

conn ios-production-ikev2
        [...]
        # AES_CBC_256/HMAC_SHA2_256_128
        esp = "aes256-sha256-modp2048,\
                aes256-sha256-modp1536,\
                aes256-sha256-modp1024,\
                aes256-sha1-modp2048,\
                aes256-sha1-modp1536,\
                aes256-sha1-modp1024,\
                aes128-sha1-modp1024,\
                aes256-sha1,\
                aes128-sha256"
```

It is recommended to append a Diffie-Hellman group to all ciphers in order to enable Perfect-Forward-Secrecy. This does not signify a considerably negative performance overhead but increases the security posture.

EXP-06-009 WP1: Weak Cryptographic Algorithms in *strongSwan* *(Low)*

***Note from ExpressVPN**: ExpressVPN is currently in the process of deprecating L2TP, which will address this issue. We expect the full deprecation of L2TP to be complete by 31 October 2022.*

While reviewing the source code of the *xv_pleco_automake* repository, it was noticed that several *strongSwan* configurations explicitly allow the usage of weak cryptographic primitives. According to the official recommendations regarding security best practices for *strongSwan*, the following cryptographic primitives must not be used[15]:

- Encryption: *des, 3des, cast, blowfish*
- Integrity Protection/Pseudo Random Functions: *md5, sha1*
- Diffie-Hellman Groups: *modp512, modp768, modp1024, modp1024s160, modp1536, modp2048s224, modp2048s256, ecp192*

Affected files:
xv_pleco_automake/etc/ipsec.d/expressvpn_l2tp.conf

Affected code:

```
conn l2tp
        [...]
        ike = "aes256-sha256-modp2048,\
                aes256-sha256-modp1536,\
```

[15] https://docs.strongswan.org/strongswan-docs/5.9/howtos/securityRecommendations.html

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

```
                aes256-sha256-modp1024,\
                aes256-sha1-modp2048,\
                aes256-sha1-modp1536,\
                aes256-sha1-modp1024,\
                aes128-sha1-modp1024,\
                3des-sha1-modp1024"

        esp = "aes256-sha256-modp2048,\
                aes256-sha256-modp1536,\
                aes256-sha256-modp1536,\
                aes256-sha256-modp1024,\
                aes256-sha256-modp1024,\
                aes256-sha1-modp2048,\
                aes256-sha1-modp2048,\
                aes256-sha1-modp1536,\
                aes256-sha1-modp1536,\
                aes256-sha1-modp1024,\
                aes256-sha1-modp1024,\
                aes128-sha1-modp1024,\
                aes128-sha1-modp1024,\

                aes256-sha1,\
                aes256-sha1,\
                aes128-sha256"
```

Affected file:

xv_pleco_automake/etc/templates/ipsec_expressvpn.conf.j2

Affected code:

```
conn ios-production
       [...]
       esp = "aes256-sha1-modp1536,\
              aes256-sha1-modp1024,\
              aes256-sha1,\
              aes128-sha256"
       ike = "aes256-sha1-modp1536,\
              aes256-sha1-modp1024"
[...]
conn ios-production-ikev2
       ike = "aes256-sha256-modp2048,\
              aes256-sha256-modp1536,\
              aes256-sha256-modp1024,\
              aes256-sha1-modp2048,\
              aes256-sha1-modp1536,\
              aes256-sha1-modp1024,\
              aes128-sha1-modp1024,\
              3des-sha1-modp1024"

       [...]
```

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

```
esp = "aes256-sha256-modp2048,\
       aes256-sha256-modp1536,\
       aes256-sha256-modp1024,\
       aes256-sha1-modp2048,\
       aes256-sha1-modp1536,\
       aes256-sha1-modp1024,\
       aes128-sha1-modp1024,\
       aes256-sha1,\
       aes128-sha256"
```

It is recommended to replace all weak cryptographic algorithms with strong cryptographic primitives.

EXP-06-010 WP1: Username enumeration for OpenVPN Authentication *(Info)*

***Note from ExpressVPN**: ExpressVPN and Cure53 agree it's not possible to fully remediate due to the way OpenVPN was designed. In addition, all of our usernames and passwords are 24 characters long and randomly generated. Brute forcing over the network would be extremely challenging. First OpenVPN would keep disconnecting you and then you'd have the firewall rate limiting to deal with.*

During a code review of the *xv_pleco_automake* repository it was noticed that the OpenVPN implementation uses a custom plugin written in Python for checking the authentication of a user. For this purpose, the Python script checks the username and password provided by the OpenVPN client. Specifically, in the first step the script performs a database lookup using the provided username, and in case a match in the database gets found, it compares the delivered password with the password hash stored within the database. In case the provided user was not found, the script immediately returns.

Since the computation of a password hash may take a considerable amount of time depending on the utilized password hashing function, an attacker is able to observe timing differences between users who exist and do not exist.

Affected file:
xv_pleco_automake/pleco/cli/ovpnauth_inotify.py

Affected code:

```
@retry(wait_fixed=100, stop_max_attempt_number=10)
def authenticate(username, password, database_path):
       [...]
       try:
              dbconn = sqlite3.connect(database_path)
              curs = dbconn.cursor()
```

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

```
                query = curs.execute("SELECT encrypted_credentials from
                vpn_accounts where username=? LIMIT 1", (username,))
                result = query.fetchone()
                if result and password_valid(password, result[0]):
                        return True
                return False
        finally:
                dbconn.close()
```

It is recommended to utilize a constant-time comparison when authenticating users in order to eliminate the risk of side-channel leakage.

EXP-06-011 WP1: No rate-limiting for OpenVPN Authentication Plugin *(Info)*

***Note from ExpressVPN**: ExpressVPN discussed this issue with Cure53 during the audit, and mentioned that we rate limit new connections into OpenVPN using the firewall, which maintains authentication at a manageable level. For that reason there's no need to add any special rate limiting in the auth plugin itself.*

While reviewing the *xv_pleco_automake* repository, it was observed that the OpenVPN implementation uses a custom plugin written in Python for checking the authentication of a user. For that purpose, the Python script checks the username and password provided by the OpenVPN client, deferring the password check to the *passlib* module[16], which compares the provided password with the expected password hash. In case that the passwords do not match, the module returns *false*. However, the rate-limiting of the OpenVPN plugin Python script only increments the number of the issued calls in case of errors.

Depending on the hash algorithm in place for comparing the passwords, an attacker may abuse this and try to brute-force passwords of the OpenVPN users.

Affected file:
xv_pleco_automake/pleco/cli/ovpnauth_inotify.py

Affected code:

```
@retry(wait_fixed=100, stop_max_attempt_number=10)
def authenticate(username, password, database_path):
        [...]
        try:
                dbconn = sqlite3.connect(database_path)
                curs = dbconn.cursor()
                query = curs.execute("SELECT encrypted_credentials from
                vpn_accounts where username=? LIMIT 1", (username,))
                result = query.fetchone()
```

[16] https://passlib.readthedocs.io/en/stable/index.html

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

```
                if result and password_valid(password, result[0]):
                        return True
                return False
        finally:
[...]
def do_auth(creds_line, database_path, rate_limiter):
        [...]
        try:
                auth_response = authenticate(username, password, database_path)
        except Exception:
                write_control_file_access_granted(control_file)


                if rate_limiter.allow_call():
                        rollbar.report_exception()
                return
        [...]
```

It is recommended to also account for failed authentication attempts in the rate-limiting of the OpenVPN authentication module.

EXP-06-012 WP1: Memory Leak in Error Path *(Low)*

***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.*

While reviewing the source code of the *xv_openvpn_auth* repository, it was noticed that a file descriptor is kept open if the function *write_line_to_magic_file* fails. This will leak memory and may open the door for a DoS attack.

Affected files:
xv_openvpn_auth/c/helpers.c

Affected code:

```
fp=fopen(context->auth_file_name, "w");
if(fp) {
        // Write out our string
        int line_length = strnlen(context->line_buffer, LINE_BUFFER_SIZE);
        int records_written = fwrite(context->line_buffer, line_length, 1, fp);
        // Check the write succeeded - measure in 'records' written i.e. should
be 1
        if(records_written != 1) {
                log(PLOG_ERR, PLUGIN_NAME, "(%s) Error: Writing auth data file.
Cannot continue with authentication: %s", event, strerror(errno));
                return false;       }
```

It is recommended to close all opened file descriptors before leaving the function early.

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

EXP-06-013 WP1: Undefined Return Value of Function *(Low)*

***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.*

While reviewing the source code of the *xv_openvpn_auth* repository, it was noticed that the function *openvpn_plugin_func_v3* can return with an unspecified return value.

Affected files:
xv_openvpn_auth/c/ovpnauth.c

Affected code:

```
        // Even if it failed to write to disk, return success so that user is not
disconnected
        return OPENVPN_PLUGIN_FUNC_SUCCESS;
    }
        <RETURN STATEMENT MISSING>
}
```

It is recommended to explicitly return either *OPENVPN_PLUGIN_FUNC_ERROR* or *OPENVPN_PLUGIN_FUNC_SUCCESS*.

EXP-06-014 WP1: OpenVPN Auth Plugin Built with Compiler Warnings Disabled *(Info)*

***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.*

While reviewing the source code of the *xv_openvpn_auth* repository, it was noticed that no compiler warnings are enabled. The consequences might be that possibly security-relevant errors remain unnoticed.

Affected files:
xv_openvpn_auth/c/CMakeLists.txt

It is recommended to explicitly specify a minimum set of compiler warnings. On UNIX platforms *-Wall* or similar can be used.

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

EXP-06-015 WP1: Race between Online State and Firewall Setup *(Info)*

***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.*

During a source code review of the *xv_pleco_server* and *xv_pleco_automake* repositories it was discovered that arming the host firewalls happens after network initialization. This opens a narrow time window where the system is already reachable via the network but not all firewall rules are applied.

Affected file:
xv_pleco_automake/etc/systemd/system/automake.service

It is recommended to either load all firewall rules before *network-online.target* is reached or to set a default *deny* rule before going online and loading the rule-set later.

EXP-06-016 WP1: Use of deprecated "ssh-rsa" Signature Scheme *(Low)*

***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.*

Another discovery made during a source code review of the *xv_pleco_deployer* repository was that the SSH configuration used the default cryptographic algorithms. For the version of OpenSSH used by TrustedServer (v8.4), this includes the *ssh-rsa* signature scheme which still uses the SHA-1 hashing algorithm. Using SHA-1 in this setting is considered insecure, as it would be possible to create chosen-prefix hash collisions with an acceptable amount of funds. For this reason, OpenSSH disables this signature algorithm starting with release 8.8[17].

Affected files:
xv_pleco_deployer/roles/provision/host-ssh-config/files/sshd_config

PoC:

```
$ ssh -vvv -oHostKeyAlgorithms=ssh-rsa -p 1022  -l root -i <KEYPATH>
46.166.186.2
[...]
debug2: peer server KEXINIT proposal
debug2: KEX algorithms: curve25519-sha256@libssh.org,diffie-hellman-group18-
sha512,diffie-hellman-group14-sha256,diffie-hellman-group16-sha512
debug2: host key algorithms: ssh-ed25519,rsa-sha2-512,rsa-sha2-256,ssh-rsa
debug2: ciphers ctos: chacha20-poly1305@openssh.com,aes256-
gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
debug2: ciphers stoc: chacha20-poly1305@openssh.com,aes256-
gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

[17] https://www.openssh.com/txt/release-8.8

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

```
debug2: MACs ctos: hmac-sha2-512-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,umac-128-etm@openssh.com
debug2: MACs stoc: hmac-sha2-512-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,umac-128-etm@openssh.com
debug2: compression ctos: none,zlib@openssh.com
debug2: compression stoc: none,zlib@openssh.com
debug2: languages ctos:
debug2: languages stoc:
debug2: first_kex_follows 0
debug2: reserved 0
debug1: kex: algorithm: curve25519-sha256@libssh.org
debug1: kex: host key algorithm: ssh-rsa
debug1: kex: server->client cipher: chacha20-poly1305@openssh.com MAC:
<implicit> compression: none
debug1: kex: client->server cipher: chacha20-poly1305@openssh.com MAC:
<implicit> compression: none
debug1: kex: curve25519-sha256@libssh.org need=64 dh_need=64
debug1: kex: curve25519-sha256@libssh.org need=64 dh_need=64
debug3: send packet: type 30
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
```

It is recommended to either explicitly disable the *ssh-rsa* signature scheme or update to a newer version of OpenSSH which disables this by default.

EXP-06-017 WP1: Potential weak random numbers for DH param generation *(Info)*

***Note**: After the report was reviewed by ExpressVPN, it was clarified that the spotted issue is not a mistake as assumed by Cure53, but in fact the use of /dev/urandom was intentional here.*

During a source code review conducted for the *xv_pleco_deployer* repository, Cure53 discovered a mistake during the activation of a server. Specifically, the Diffie-Hellman parameters are generated using the command *openssl dhparam -rand /dev/urandom -out /etc/openvpn/dh2048.pem 2048 2>&1 >/dev/null.*

Specifying */dev/urandom* as a source for random bytes could potentially lead to using weak random numbers. This file will be readable even though the Kernel's entropy pool is not filled enough to yield sufficiently secure random numbers. The flaw happens usually right after a reboot of the machine when the Kernel has not had enough time to collect random data from various sources.

The problem explained above has already been mitigated in Pleco since the kernel configuration parameter *CONFIG_RANDOM_TRUST_CPU* was enabled. This approach activates the kernel's trust in the CPU manufacturer to properly initialize the CPU's internal CRNG, resulting in the kernel using this as an additional entropy source.

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

Since the CPU's CRNG is available from the beginning of the boot process, this prevents low entropy secure numbers being emitted from */dev/urandom*. However, this only happens if the CPU's CRNG is secure.

While it is very unlikely that insecure random numbers are used for DH parameter generation, it is nevertheless recommended to use */dev/random* instead when running *openssl* commands. Alternatively, it is also possible to remove the *-rand* parameter from the command line, since OpenSSL should fall back to using the *getrandom()* system call, which is also considered secure.

EXP-06-020 WP1: AppArmor / Seccomp Missing *(Info)*

***Note from ExpressVPN**: This is intended behavior to ensure simplicity and consistency of TrustedServer. Additional restrictions that limit the ability of varying processes to interact provides minimal improvement, while significantly increasing operational complexity which may weaken security.*

While reviewing the provided TrustedServer instances, it was discovered that the deployed operating system does not take advantage of process isolation / sandboxing techniques such as AppArmor[18] or Seccomp[19]. AppArmor is a Mandatory Access Control (MAC) mechanism, provided as a kernel security module enhancement, allowing programs to confine programs to a limited set of resources. An attacker having low-privileged access to a TrustedServer instance, e.g., by exploiting a vulnerability within any of the external exposed interfaces - namely *charon, openvpn, helium* or *x2ltpd*, could leverage this and attempt to pivot to other processes. This is due to the lack of sandboxing and isolation in place, with a potential result in a form of privilege escalation.

PoC:
The lack of loaded AppArmor profiles can be observed by running the following command on any of the deployed server instances:

```
root@nforce-nl-pntest-01:~# apparmor_status
apparmor module is loaded.
```

The lack of Seccomp filtering can be demonstrated by running the *checksec.sh*[20] utility as follows:

```
root@nforce-nl-pntest-01:/tmp/checksec/checksec.sh-master# ./checksec --proc-all
[...]
```

[18] https://wiki.ubuntu.com/AppArmor
[19] http://manpages.ubuntu.com/manpages/bionic/man2/seccomp.2.html
[20] https://github.com/slimm609/checksec.sh

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

It is recommended to strictly isolate all processes that are exposed to the outside. This reduces the risk of privilege escalation in case a malicious actor has been able to compromise any of the aforementioned services. In other words, the available attack surface for privilege escalation would be significantly reduced. Furthermore, it is encouraged to revisit Seccomp filtering for the referred processes in order to reduce the number of the allowed system calls.

EXP-06-021 WP1: Dual-Use of TLS keys for OpenVPN and Helium *(Info)*

***Note from ExpressVPN**: We consider our certificates to be secure, and adding a separate set of them for another protocol doesn't improve security.*

During a source code review of the *xv_pleco_automake* repository, it was observed that the OpenVPN configuration and the Helium configuration both rely on the same key, namely the key persisted in the file *server_ca2.key*. Reusing the same cryptographic key for multiple purposes is not in accordance with best practices[21] for cryptographic key material, since the leakage or compromise of one component immediately affects all other components which rely on the same cryptographic key.

Affected file:
xv_pleco_automake/etc/templates/openvpn.conf.j2

Affected code:
```
cert /etc/openvpn/server_ca2.crt
key /etc/openvpn/server_ca2.key
```

Affected file:
xv_pleco_automake/etc/templates/helium.conf.j2

Affected code:
```
-- Location of server certificate
server_cert = "/etc/openvpn/server_ca2.crt"

-- Location of private key
server_key  = "/etc/openvpn/server_ca2.key"
```

It is recommended that OpenVPN and Helium use individual keys in order to align with best practices for the usage of cryptographic key material.

[21] https://cheatsheetseries.owasp.org/cheatsheets/Key_Management_Cheat_Sheet.html

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

EXP-06-022 WP1: Unrestricted Outbound Connections Possible *(Info)*

***Note from ExpressVPN**: This functionality is required for the VPN server to fulfill its intended purpose, and is intended. As a result, this issue will not be addressed.*

While reviewing the provided TrustedServer instances, it was noticed that the TrustedServer does not filter or prevent outbound connections to arbitrary services and hosts. This unrestricted outbound connectivity could, for example, be abused by an attacker who exploits an RCE vulnerability and attempts to establish an outbound reverse shell to an attacker-controlled server.

PoC:
The following command has been executed on one of the TrustedServer instances, demonstrating that an outbound connection to an arbitrary server on the Internet was possible:

```
strongswan@nforce-nl-pntest-01:/tmp$ wget 134.122.69.246:8000
--2022-04-26 11:24:08--  http://134.122.69.246:8000/
Connecting to 134.122.69.246:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3364 (3.3K) [text/html]
Saving to: 'index.html'

index.html                                                          100%
[================================================================================
================================================================================
===>]   3.29K  --.-KB/s    in 0s

2022-04-26 11:24:08 (331 MB/s) - 'index.html' saved [3364/3364]
```

It is recommended to restrict outbound connectivity to a predefined set of IP addresses / hostnames, as well as to further incorporate a basic minimum local firewall rule-set on the TrustedServer instance itself. The latter needs to effectively block outbound connectivity to hosts and services that are not required by the TrustedServer.

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

EXP-06-023 WP1: TrustedServer Hardening Weaknesses *(Low)*

***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.*

Most Linux default installations have several security options disabled due to requiring individual work or possibly affecting the system's usability for the majority of the users. There are several configuration options listed below and known for improving the security of TrustedServer.

Hidepid

Every user can see all of the processes and their parameters on a Linux server. Under certain premises, this behavior might leak information or point an attacker in the right direction when it comes to escalating privileges.

PoC:

```
root@nforce-nl-pntest-01:~# mount | grep proc
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
```

It is evident from the output above that *hidepid* is missing.

Hidepid is an option that can be activated when the *procfs*[22] is mounted. This can be achieved with the following entry inside the server's *fstab*.

```
[...]
proc            /proc           proc        hidepid=2       0 0
```

If this item is enabled, a non-root user can exclusively see the processes that were started by them and not by others.

Tmpfs Mount Settings

It was discovered that the */tmp* partition is mounted without having the *noexec*[23] option set, which would ensure that users are not able to executable binaries from the */tmp* partition. Attackers often drop executable binary files to */tmp*, as this is usually a folder where all (also lower privileged) users have write permissions to.

PoC:

```
root@nforce-nl-pntest-01:~# mount | grep "/tmp"
tmpfs on /tmp type tmpfs (rw,nosuid,nodev,relatime)
```

[22] https://en.wikipedia.org/wiki/Procfs
[23] https://man7.org/linux/man-pages/man8/mount.8.html

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

It is recommended to add the mount option *noexec* to the line within the file */etc/fstab* where the */tmp* partition gets mounted.

Dmesg Restrict

Dmesg[24] is a Linux command showing messages printed by the kernel. It contains information about the boot process and hardware, which means that in some cases it might disclose certain details to an attacker. This especially holds for an adversary who already has limited privileges on the server and can now escalate to root.

PoC:

```
root@nforce-nl-pntest-01:~# cat /proc/sys/kernel/dmesg_restrict
0
```

It is recommended to restrict the access to kernel messages to root by adding the following line to the *sysctl* configuration:

```
kernel.dmesg_restrict = 1
```

YAMA ptrace scope

YAMA[25] is a Linux security module (LSM) to limit the *ptrace* scope. Usually, a user is allowed to reach *ptrace*, and therefore has the capacity to read process memory and any process with the same effective user ID. YAMA offers a mechanism to tighten this scope. A common setting is that only the parent process is allowed to *ptrace* one of its child processes.

PoC:

```
root@nforce-nl-pntest-01:~# cat /proc/sys/kernel/yama/ptrace_scope
0
```

It is recommended to restrict ptrace access to root by adding the following line to the *sysctl* configuration:

```
kernel.yama.ptrace_scope = 2
```

[24] https://en.wikipedia.org/wiki/Dmesg
[25] https://www.kernel.org/doc/html/latest/admin-guide/LSM/Yama.html

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

EXP-06-024 WP1: Possible MitM attack against Ansible Workflows *(Low)*

***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.*

During a source code review of *xv_pleco_deployer*, it was noticed that the Ansible configuration disables host key checking, meaning that Ansible will not verify the host it connects to by its SSH host key. Instead, it ignores the host key and simply connects without any verification.[26]

This can be problematic when the *activate.yml* playbook is executed because it will deploy the secrets to the host. An attacker with appropriate control over the network between the administrators PC running Ansible and the host Ansible connects to can launch a MiTM attack and extract all deployed secrets. Additionally, spoofing attacks where the attacker redirects the connection to a malicious host would be possible, with malicious hosts ultimately acting as legitimate ones.

Affected file:
xv_pleco_deployer/ansible-prod.cfg

Affected code:

```
[defaults]
inventory = ./inventory/prod/

vault_identity_list = shared@prompt

host_key_checking = False

# Use the determined python version, NOT /usr/bin/python if it exists
interpreter_python = auto
```

At least the *activate.yml* playbook the SSH host key should be verified to mitigate SSH MitM and similar attacks. Since gaining this level of control over the network requires sophisticated access, this finding was rated at as a limited risk.

[26] https://docs.ansible.com/ansible/latest/user_guide/connection_details.html#m...ing-host-key-checking

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

EXP-06-025 WP1: Limited security from OpenVPN tls-auth feature *(Info)*

***Note**: After the report was reviewed by ExpressVPN, it was clarified that in fact no Load Balancer is being used here, which renders the issue to be a false alert.*

While inspecting the OpenVPN configuration on the test systems and reviewing *xv_pleco_deployer,* it was noticed that the *tls-auth* feature of OpenVPN was activated. This feature adds an additional HMAC signature to all TLS handshake packets. To use it, every client must use the same *tls-auth* key in their OpenVPN configuration as the server. In a load balancer scenario, all servers behind the same load balancer must also share the same *tls-auth* key. The OpenVPN documentation states that this can protect against[27]:

- DoS attacks or port flooding on the OpenVPN UDP port.
- Port scanning to determine which server UDP ports are in a listening state.
- Buffer overflow vulnerabilities in the SSL/TLS implementation.
- SSL/TLS handshake initiations from unauthorized machines (while such handshakes would ultimately fail to authenticate, *tls-auth* can cut them off at a much earlier point).

The important point here is that this approach only works as long as the *tls-auth* key remains secret and is not broadly known. In the case of a VPN provider like ExpressVPN, this property cannot hold as everybody who signs up for an account will be in possession of this secret, thus eliminating any security provided by it. What is more, it looks like the *tls-auth* key is never replaced over time. As a result, once a user is in possession of the key, that key remains valid even if the user's service subscription has ended.

Cure53 wants to emphasize that it is recommended to continue using this feature, but it is important to not rely on it for security. As the key is shared, it should be assumed that this feature is compromised. To slightly improve its usefulness, one could rotate the key in regular intervals. This would, however, require all existing clients replacing the key, which might have a negative impact on user-experience.

[27] https://openvpn.net/community-resources/hardening-openvpn-security/

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

EXP-06-026 WP1: Known, unfixed vulnerability in OpenSSH *(Info)*

***Note from ExpressVPN**: ExpressVPN does not use these features, as mentioned by Cure53. We're also unable to implement a fix, as the upstream implementation has not been addressed by the OS vendor.*

While inspecting the OpenSSH package installed by *xv_pleco_server*, it was noticed that it contains a known vulnerability known under *CVE-2021-41617*[28] which allows privilege escalation in non-default configurations using the *AuthorizedKeysCommand* and *AuthorizedPrincipalsCommand* configuration options. This affects all OpenSSH versions from 6.2 through 8.x before 8.8, which includes version 8.4 used by TrustedServer.

As TrustedServer currently does not use these features, they are not vulnerable. This finding is merely intended as a reminder that this is something to be aware of when making changes to the OpenSSH configuration.

EXP-06-028 WP1: Missing return value check for calloc() *(Info)*

***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.*

While reviewing the provided diff file for OpenVPN, it was noticed that return values are not properly checked when calling *calloc*[29].

Under specific circumstances, *calloc* call can fail and return NULL to the caller, for example, when a memory request cannot be satisfied because there is no usable block of memory on the heap of the C runtime or, alternatively, when the C runtime memory management requested more memory from the operating system than available. While this situation is rare, in such a case the application might expose undefined behavior.

Affected files:
git-diff-xvpn-openvpn_public-openvpn.diff

Affected code:

```
[...]
+        xvpn_actual_ip = (char *)calloc(1,100);
+        char *port = (char *)calloc(1,100);
[...]
+        xvpn_misdirect_ip = (char *)calloc(1,100);
[...]
+        xvpn_original_port = (char *)calloc(1, 100);
[...]
+        xvpn_actual_port = (char *)calloc(1, 100);
```

[28] https://security-tracker.debian.org/tracker/CVE-2021-41617
[29] https://linux.die.net/man/3/calloc

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

```
[...]
```

From reviewing the provided *diff,* it is evident that the return values of the *calloc* function calls are not checked and subsequently used within *snprintf*, among others.

It is recommended to properly check the return value of *calloc* in order to detect failure.

EXP-06-029 WP1: OpenVPN server binary lacks Stack Canaries *(Low)*

***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.*

While reviewing the binary protection flags of the running VPN server binaries within the provided testing environment, it was discovered that the OpenVPN server binary lacks stack canaries[30] compiler mitigation security feature.

PoC:
The lack of stack canaries for the OpenVPN server binary can be observed by using the *checksec.sh*[31] utility:

```
# ./checksec --file=/usr/sbin/openvpn --format=json
{ "/usr/sbin/openvpn":
{ "relro":"full","canary":"no","nx":"yes","pie":"no","rpath":"no","runpath":"no"
,"symbols":"no","fortify_source":"yes","fortified":"10","fortify-able":"21" } }
```

It is recommended to modify the build process of the OpenVPN server binary and to enforce the usage of stack canaries, e.g., *"-fstack-protector"* or *"-fstack-protector-all"* within GCC[32].

[30] https://mcuoneclipse.com/2019/09/28/stack-canaries-with-gcc-checking-for-stack-overflow-at-runtime/
[31] https://github.com/slimm609/checksec.sh
[32] https://gcc.gnu.org/onlinedocs/gcc/Instrumentation-Options.html

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

Conclusions

It can be concluded that the ExpressVPN team already has a good understanding of the IT security measures relevant to the ExpressVPN TrustedServer. They were clearly capable of minimizing the attack surface for their elements included in the scope of this April 2022 project. Yet, given that five members of the Cure53 team managed to spot twenty-nine security-relevant flaws, there is some room for improvement to be considered. Particularly the greater preponderance for general weaknesses showcases that minor threats should be also addressed before the complex can reach very good levels of security.

The white-box approach and thirty-two days being dedicated to this examination simultaneously increase Cure53's confidence in issuing a positive verdict about the security premise of the ExpressVPN scope. In addition to the comments on findings, the processes and operations linked to the examinations put ExpressVPN in a very good light. Cure53 was in constant communication with the customer through a dedicated Slack channel, with help being provided whenever requested.

Moving on to specifics, the security assessment featured six repositories, namely *xv_client_activities, xv_freeradius_auth, xv_openvpn_auth, xv_pleco_automake, xv_pleco_deployer* and *xv_pleco_server*. For these repositories, two work packages were defined.

The first work package (WP1) covered an assessment of the TrustedServer complex, including, but not limited to, a review for the running Pleco OS. This task aimed at identifying common vulnerabilities, for example remote code execution, information leakage, unauthorized access. It further included a review of the configuration of the VPN components which TrustedServer uses.

The second work package (WP2) centered on whether TrustedServer was built in-line with ExpressVPN's privacy policy, which states that no data is collected and stored beyond what is necessary to run the VPN service.

The assessment included a code review of all provided repositories paired with a dynamic examination of the external exposed interfaces as well as the TrustedServer instance itself. For that purpose, the team got SSH access to two dedicated testing servers. It was mutually agreed that only the code within the provided repositories was in scope of this assessment, while the binary and source code review of the actual VPN components (IPSec, OpenVPN and Lightway) were excluded. The reason is that all VPN components should rather be reviewed in a dedicated source code assessment.

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

Generally speaking, the testers got the impression that the repositories and the source code are clearly organized, and that security considerations had a substantial impact on the development and design of TrustedServer. In summary, four vulnerabilities and twenty-five miscellaneous issues were identified over the course of this assessment. None of the identified issues was rated as *High* or *Critical* in terms of severity, which speaks for the overall security posture of the product.

The vulnerabilities include two findings rated with *Medium* severity, which require low-privileged access to a TrustedServer instance. One of the findings demonstrates an unprotected management interface of OpenVPN (see EXP-06-018), whereas the other revealed files readable by all service accounts on TrustedServer containing sensitive credential information stored on the TrustedServer (see EXP-06-027). The miscellaneous issues cover several configuration issues of OpenVPN and *strongSwa*n, but also shed light on the missing defense-in-depth measures for the underlying operating system.

In terms of fuzzing, only naive network fuzzing has been attempted and did not result in any crashes. This can be attributed to many reasons. First, the RateLimiter has a significant impact on the actual throughput of the fuzzing efforts. Secondly, only naive bit flipping was possible.

Moving forward, it is highly encouraged to incorporate fuzz testing against a build having features like address space sanitization (ASAN) enabled, in order to spot memory corruption vulnerabilities. Such flaws might be missed during a review such as this one. Fuzz testing should be applied to the OpenVPN, IPSec as well as Lightway daemon, using modern fuzz testing frameworks such as AFL++ or libFuzzer, in order to fully leverage the benefits of code coverage driven fuzzing.

To summarize what has been argued till now, the TrustedServer complex makes a good impression. Despite the somewhat excessive number of the identified issues, most problems were either classified as general weaknesses or required low-privileged access to a TrustedServer instance as a prerequisite for further escalation. Note that none of the findings spotted in this assessment affect WP2, all of them were found to affect WP1 only, which is worth a mention.

Besides some minor issues within the configuration, the VPN services configurations appear to be in good shape, providing appropriate security for client connections. The general host configuration would benefit from a more in-depth approach to security, for instance in leveraging more of the security features provided by Linux. AppArmor, Seccomp and similar features to consider are summarized in EXP-06-023.

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

Resolving all of the vulnerabilities and miscellaneous issues will certainly narrow the attack surface and elevate the security posture of the ExpressVPN even more, providing defense-in-depth for the TrustedServer product. In conclusion, this project predominantly drew attention to specific components that require security enhancement, rather than pointing out major problems. It is recommended to plan and continue to execute external and internal actions on security aspects, particularly before any substantial changes are deployed.

Such a strategy could help ensure that new features cannot introduce undesired security vulnerabilities. Furthermore, the TrustedServer complex could profit from recurring security assessments as well as dedicated source code reviews of its running components, for example Lightway / Helium or obfuscation techniques implemented on top of OpenVPN.

To conclude, this security review achieved good coverage over all working packages, identifying a multitude of security-relevant issues that provide ample opportunity for hardening measures.

Cure53 would like to thank Brian Schirmacher, Brendan Horan, Pete Membrey and the rest of the ExpressVPN team for their excellent project coordination, support and assistance, both before and during this assignment.