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

Pentest-Report Silence Labs MPC SDK & GCP Infra 11.2024

Cure53, Dr.-Ing. M. Heiderich, A. Belkahla, BA M. Haunschmid, V. Gwehenberger

Index

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

Introduction

"Silence Laboratories offer a versatile toolkit for Multi-Party Computation Threshold Signatures and Privacy-Preserving Compute"

From https://www.silencelaboratories.com/

This report describes the results of a security assessment of the Silent Network MPC SDK, as well as the GCP infrastructure configuration. The project, which included a penetration test and a dedicated source code audit, was conducted by Cure53 in November 2024.

The audit, registered as *SIL-04*, was requested by SILENCE LABORATORIES PTE. LTD. in October 2024 and carried out by Cure53 in the following month. In terms of the exact timeline and specific resources, Cure53 has completed the research in CW45. In order to achieve the expected coverage for this task, a total of nine days were invested. In addition, it should be noted that a team consisting of five senior testers was formed and assigned to the preparation, execution, documentation, and delivery of this project.

For optimal structuring and tracking of tasks, the assessment was divided into two separate work packages (WPs):

- **WP1**: White-box penetration tests & source code audits against MPC SDK codebase
- **WP2**: White-box penetration tests & reviews of MPC SDK GCP infrastructure & configuration

As the titles of the WPs indicate, the white-box methodology was used. Cure53 was provided with the relevant URLs, documentation, test-user accounts, as well as all further means of access required to complete the tests. In addition, all sources corresponding to the test targets were shared to ensure that the project could be executed in accordance with the agreed framework.

The project was completed without any major issues. To facilitate a smooth transition into the testing phase, all preparations were completed in CW44. Throughout the engagement, communications were conducted through a private, dedicated, and shared Slack channel. Stakeholders - specifically the Cure53 testers and the internal staff responsible for Silence Labs MPC SDK and GCP infrastructure - were able to participate in discussions in this space.

Cure53 did not need to ask many questions, and the quality of all project-related interactions was consistently excellent. The continuous exchange contributed positively to the overall results of this project. Significant roadblocks were avoided thanks to clear and careful preparation of the scope. While Cure53 provided frequent status updates on the examination and emerging findings, no live reporting was requested in the frames of *SIL-04*.

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

The Cure53 team achieved good coverage of the WP1-WP2 objectives. Only four security-related discoveries were made and all often have been classified as general weaknesses with low exploitation potential. In other words, this *SIL-04* revealed no actually exploitable vulnerabilities within the targets.

There should be no doubt that the outcomes showcase the security strength of the inspected MPC SDK. However, Cure53 was able to identify several general weaknesses and proposals concerning hardening advice. This advice should not be disregarded, but rather swiftly followed in order to fully ensure the upkeep of the already very good level of security.

The following sections first describe the scope and key test parameters, as well as how the work packages were structured and organized.

Then, what the Cure53 team did in terms of attack attempts, coverage, and other test-related tasks is explained in a separate chapter on test methodology. This is to show to the customer which areas of the software in scope have been covered and what tests have been executed. This transparency is particularly helpful with only a few findings present in this report-document.

Next, all findings are discussed in the miscellaneous category of flaws in a chronological order. In addition to technical descriptions, PoC and mitigation advice is provided where applicable.

The report ends with general conclusions relevant to this November 2024 project. Based on the test team's observations and the evidence collected, Cure53 elaborates on the overall impressions and reiterates the verdict. The final section also includes tailored hardening recommendations for the Silent Network MPC SDK, as well as the GCP infrastructure configuration.

Scope

- **Penetration tests & source code audits against Silence Labs MPC SDK & GCP infra**
    - **WP1:** White-box penetration tests & source code audits against MPC SDK codebase
        - **Sources:**
            - https://github.com/silence-laboratories/el-services/releases/tag/audit-rc3
                - **Branch**: audit-rc3
                - **Commit-id**: d925383476d109dd3d9a03f171aa495c52f33297
            - https://github.com/silence-laboratories/walletprovider-sdk/releases/tag/audit-rc2
                - **Branch**: audit-rc2
                - **Commit-id**: 872124e526c6ed0bd7f21334e1c1e101622f81a8
            - https://github.com/silence-laboratories/el-sdk-clients/releases/tag/audit-rc2
                - **Branch**: audit-rc2
                - **Commit-id**: 23df49d3735cb3186cb73892427242213870a028
            - https://github.com/silence-laboratories/dkls23-rs/commit/d4267a42c73838e9e92b5bda634c572ec4879090
                - **Branch**: main
                - **Commit-id**: d4267a42c73838e9e92b5bda634c572ec4879090
    - **WP2:** White-box penetration tests & reviews of MPC SDK GCP infrastructure & config
        - **GCP project URL:**
            - https://console.cloud.google.com/home/dashboard?project=silent-network-426409
        - **Accounts:**
            - U: victor@rs.cure53.de
            - U: martin@rs.cure53.de
    - **Documentation**
        - https://www.notion.so/Documentation-for-Auditors-12dfe2c2b4bd80cfbe98c1e08d695b77
        - https://silencelaboratories.slack.com/archives/C044SFUMCRE/p1730363210094199
        - https://www.notion.so/Testing-Silent-Network-11bfe2c2b4bd80969b7fd16ae63424ad#12efe2c2b4bd8070a2f3f4f473e01c65
    - **Test-supporting material was shared with Cure53**
    - **All relevant sources were shared with Cure53**

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

Test Methodology

This section documents the testing methodology applied by Cure53 during this *SIL-04* project. It specifically discusses the resulting coverage, shedding light on how various components were examined. Further clarification concerning areas of investigation subjected to deep-dive assessments is offered, especially in the absence of significant security vulnerabilities across certain components.

Notably, Cure53's methodology included both automated tools and manual testing techniques, ensuring a comprehensive review that addresses both surface-level concerns and more deeply positioned security risks. Subsequent two sections refer to coverage across the WPs.

Audit of MPC SDK components

The Silent Network project implements a Multi-Party Threshold Signature Scheme (MTSS) using the DKLS protocol. The primary goal was to ensure the robustness of the key generation (DKG) and signing (DSG) processes, the security of the TEE-based execution environment, as well as the effectiveness of the economic security model implemented through *EigenLayer.*

The audit encompassed four primary repositories: *wallet-provider-sdk*, *dkls23-rs*, *el-sdk*, and *el-services*. These were analyzed in order of priority, which was determined on the basis of their potential security impact and the contexts of user interaction.

The team began with the *wallet-provider-sdk* component, which serves as the client-side interface for interacting with the Silent Network. This SDK is crucial as it handles user authentication, key generation requests and transaction signing. A thorough review of the TypeScript codebase was conducted, focusing on the correctness of cryptographic operations, data handling, and the security of the authentication flows (EOA, PassKey, and Ephemeral).

The testers verified that the SDK correctly interacts with the *el-services-audits-rc3* backend, performs necessary validations, and correctly signs challenges for both Distributed Key Generation (DKG) and Distributed Signature Generation (DSG). The SDK's handling of *ephId* and *keyId*, both generated client-side, was also scrutinized for potential manipulation vulnerabilities.

No vulnerabilities were identified in this context, with the SDK demonstrating robust data handling and secure authentication practices. However, it is recommended to add explicit documentation highlighting the importance of hardening web applications using the SDK. This concerns precautions against XSS attacks, which are warranted due to the potential sensitivity of user interactions.

The next component audited by Cure53 was *dkls23-rs*, which is a Rust-based cryptographic library implementing the DKLS protocol. The core cryptographic primitives were reviewed, focusing on the security of the DKG and DSG protocols. Particular attention was paid to the *data-envelope* crate. This crate handles data encryption and decryption, ensuring sensitive data is zeroed from memory after use.

The *el-sdk-clients-audits-rc2* repository, containing the applications for interacting with EigenLayer contracts, was then analyzed. An investigation of the logic for rewards and slashing was held, focusing on the correctness and security of the interaction with *EigenLayer*.

Finally, Cure53 focused on *el-services-audits-rc3*, a monorepo containing code for the Aggregator, Operator, and Wallet Provider Service backend. This repository was the most complex, requiring in-depth analysis of multiple components.

Within *el-services-audits-rc3*, the *el-aggregator-audits-rc3* crate is responsible for interacting with Operators and handling DKG/DSG requests. It was first examined in relation to the configuration parsing, *EigenLayer* integration and the handling of key-shape caching.

The *el-party-svc* crate, implementing the Operator logic, was also thoroughly reviewed. This task encompassed looking into the authentication flows, key generation and signing processes, followed by the interaction with the local key-share database.

The *remote-attestation* crate was reviewed to ensure the integrity and authenticity of the Operators running within TEEs. The correct implementation of attestation and verification procedures was verified. The *wallet-provider-service* crate, responsible for handling WebSocket connections from the *wallet-provider-service-sdk*, was also scrutinized. No security issues were identified in this realm.

Throughout the audit, dynamic testing was conducted to validate the functionality and security of the system under various scenarios. This included simulating network disruptions, operator misconduct and malicious user inputs. The correct handling of data races and concurrency issues in multi-threaded environments was also verified, particularly within the Aggregator and Operator implementations.

Audit of GCP infrastructure and architecture

This part of the audit was kicked off by running static analysis tools against the GCP project Cure53 was given access to. In parallel, manual verification of the issues found was done. The amount of resources present was rather small and they were found to be deployed via *shell* scripts. Thus, these scripts were reviewed accordingly, to allow for easier checks of both configuration of Compute VMs as well as the privileges of the created service-accounts.

In preparation for the review of the architecture as a whole, the provided documentation as well as the documentation of third-party tools involved was extensively studied. Cure53 wished to pre-identify potential risks and build a list of testing hypotheses.
Another focus area of the audit concerned the questions posed by the customer as pointers. This spanned the deployment of operators on untrusted machines via the use of Gramine and Intel's SGX. Consequently, the configuration of the key components of the operator was studied.

The manifest used to configure the creation of the *graminized* docker image was found to have a small attack surface. One concrete example was the exclusion of the *resolv.conf* file from Gramine protection (see SIL-04-002). As this makes the DNS server used by the operator attacker-controlled, potential attack scenarios were considered next, especially regarding the remote attestation process.

In addition to the configuration of the image running inside the trusted execution environment itself, the attestation of said image was examined. This was found to make heavy use of features present in Gramine, not implementing much in new ways. For this, it was made sure that attestation is done correctly on the aggregator-side.

Given an untrusted operator admin, the possibility of information disclosure was analyzed as well. This was especially featured in terms of the logging configuration inside the *operator* images. Here, two logging configurations which could lead to information disclosure in a production environment were spotted. The Gramine logging config was not configured in a production-ready way (see SIL-04-001).

An essential part of running containers is their configuration regarding users, privileges and capabilities. Applicable checks were completed in this arena.

The Google Cloud Platform (GCP) assessment centered on the *silent-network* project. As a result, Cure53 assessed Silence Labs' infrastructure layout and configurations within GCP to evaluate the security posture of their services and features. This evaluation also aimed at determining how well the current setup aligns with modern security standards and best practices.

Cure53 utilized prominent security frameworks, including the CIS Google Cloud Platform Foundation Benchmark, to obtain a different perspective on the security posture of the cloud environment. Although automated tools facilitated the implementation of these benchmarks, an extensive manual review was necessary to accurately assess and validate the results.

The environment is relatively small, comprising four Virtual Machines (VMs). Those are accompanied by associated *compute* disks, all housed within a single Virtual Private Cloud (VPC) along with basic ingress firewall rules and several service accounts. The VM configurations were carefully examined, focusing on isolation levels, encryption practices and service-account usage.

All VM disks are secured with Google-managed encryption, while each VM has also been assigned a unique service-account, adhering to the principle of least privilege. Some VMs also run containerized workloads from the artifact registry, configured to block direct access and prevent privilege escalation by disabling privileged mode, STDIN buffering, and pseudo-TTY allocation.

While the overall security posture is solid, it could be further strengthened by enabling confidential computing and restricting project-wide SSH keys, thereby enhancing data protection and access control across the environment.

Further analysis of firewall configurations identified that ingress rules permit SSH, RDP (although closed), and ICMP traffic from any IP address, exposing the environment to potential external access. This unrestricted access could pose a security risk, as it allows inbound traffic from all sources.

Additionally, while Google-managed encryption is utilized for *Secrets Manager,* implementing Customer-Managed Encryption Keys (CMEK) would provide greater control over sensitive data and strengthen data protection measures.

Overall, the security posture of the assessed environment is reasonably strong, with foundational security measures in place. Implementing additional hardening measures, as outlined in SIL-04-003, would further enhance the resilience of the GCP resources against potential threats.

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, while a vulnerability is present, an exploit may not always be possible. Each finding has been given a unique identifier (e.g., *SIL-04-001*) to foster follow-up correspondence if required.

SIL-04-001 WP2: Verbose log levels in *operator* Gramine configuration *(Low)*

Fix note: *This issue has been addressed by the Silence Labs team and applied them to the el-services repository. Cure53 verified the corresponding fixes based on the commit 907c6a947aeef841a32890c536e47a51ae51b8b0.*

While reviewing the manifest used for *graminizing* the operator image, it was found that on two occasions logging is too verbose for the use-cases proclaimed. This can lead to the disclosure of internal data - or even secrets - to the *operator* admin.

Affected file:
el-services-audit-rc3/deploy/sgx/operator.manifest

Affected code:
```
loader.env.RUST_LOG = "debug"
[...]
loader.log_level = "warning"
```

For the *loader* log-level, only *"error"* is a suitable setting for production use per the Gramine documentation[1]. Thus, it is recommended to adapt the log-level accordingly.

Additionally, the *RUST_LOG* environment variable should also be set to an appropriate, higher log-level to avoid the disclosure of technical details to *operator* admins.

[1] https://gramine.readthedocs.io/en/stable/manifest-syntax.html

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

SIL-04-002 WP2: *Operators* on rogue DNS server *(Low)*

***Fix note:** This issue has been addressed by the Silence Labs team and applied them to the el-services repository. Cure53 verified the corresponding fixes based on the commit 907c6a947aeef841a32890c536e47a51ae51b8b0.*

During the analysis of the TEE setup using Gramine, it was found that the *resolv.conf* file is marked as *trusted* in the manifest. This leads to Gramine ignoring potential changes to the file and not pinning a hash for file contents. Thus, this element is not protected[2].

Since there are no DNS servers given when starting a Docker container, the hosts *resolv.conf* is used[3]. This makes the DNS server used by *operator* attacker-controlled. Several possible impact scenarios were identified in this context:

- The */etc/sgx_default_qcnl.conf* is pinned to a certain hash via Gramine. With a rogue DNS server, the host *pccs.el.silencelaboratories.com* in *pccs_url* used for getting the PCK certificates and verification collateral could be pointed to a rogue PCCS server. However, this could not be used to tamper with the attestation, as the *aggregator* uses a Gramine library for verification. The latter does not use the attacker-supplied PCCS.
- Redirecting *googleapis.com* might lead to the disclosure of *authorization* tokens of a given service-account or even access to key-shares being uploaded. **Note:** The customer posited that the operator instance running on GCP is not in the production setup. Thus, this attack only works if an *operator* is set up in a way that uses GCP APIs.
- The resolution of the peers for DKLS could be tampered with.

Affected file:
el-services-audit-rc3/deploy/sgx/operator.manifest

Affected code:
```
# Docker swaps resolv.conf during runtime of the container, allow to do so.
allowed_files = [
    "file:/etc/resolv.conf",
]
```

Cure53 recommends hardcoding the DNS server used to a trusted server (e.g., Cloudflare's 1.1.1.1 or Google's 8.8.8.8 and 8.8.4.4). Moreover, removing the *resolv.conf* from the trusted file list when creating the Gramine image is also advised.

[2] https://gramine.readthedocs.io/en/stable/manifest-syntax.html#allowed-files-1
[3] https://docs.docker.com/engine/network/#dns-services

SIL-04-003 WP2: General GCP network hardening considerations *(Info)*

Fix note: *This issue has been addressed by the Silence Labs team and applied them to the el-services repository. Cure53 verified the corresponding fixes based on the commit 907c6a947aeef841a32890c536e47a51ae51b8b0.*

During a review of the GCP resource configuration, several potential security concerns were identified regarding the hardening of cloud resources.

First, the ingress firewall rules for the VM Instances are overly permissive, allowing SSH (TCP:22), RDP (TCP:3389), and ICMP connections from any source (0.0.0.0/0). This open configuration exposes the resources to potential network reconnaissance. While SSH access is protected by public key authentication, the unrestricted nature of the ingress rules still increases the attack surface.

Second, the configuration of the VMs is another concern. It was found that all VMs make use of project-wide SSH keys, which allow access to multiple instances across the project. Project-wide SSH keys can ease the SSH key management. However, if they are compromised, they pose a security risk which can impact all VM instances within the project.

Third, it was observed that Confidential Computing capabilities[4] are not being utilized. Confidential Computing adds another layer of protection to sensitive data in use, namely by keeping it encrypted in memory.

Cure53 recommends tightening the firewall rules to prevent direct exposure of the VM instances. It is also advised to implement instance-specific SSH key management to limit the scope of access and ensure more granular control over who can access each resource. To further bolster the security posture, enabling Confidential Computing is strongly recommended as a way of ameliorated handling for sensitive workloads.

SIL-04-004 WP1: Unbounded key-shape cache size configuration *(Low)*

Fix note: *This issue has been addressed by the Silence Labs team and applied them to the el-services repository. Cure53 verified the corresponding fixes based on the commit 907c6a947aeef841a32890c536e47a51ae51b8b0.*

It was observed that the application uses an environment variable *KEY_SHAPE_CACHE_SIZE* to configure the LRU cache size for key-shapes, doing so without proper validation of upper bounds. A malicious administrator could set this to an arbitrarily large value, potentially causing memory exhaustion on the host system. Such approaches could possibly lead to Denial-of-Service problems.

[4] https://cloud.google.com/security/products/confidential-computing

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

Affected code:
```
[...]
let key_shape_cache_size: usize = env::var("KEY_SHAPE_CACHE_SIZE")
    .ok()
    .and_then(|s| s.parse().ok())
    .unwrap_or(10);
[...]
```

To mitigate this issue, it is recommended to set a maximum allowable cache size and ensure that configurations remain safe by logging warnings or blocking startup if limits are exceeded. The recommended cache size ranges should be clearly documented and justified in the deployment guide, while memory usage should be monitored through metrics to maintain optimal performance.

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

Conclusions

Cure53 was tasked with auditing both the source code of the Silent Network MPC SDK, as well as its corresponding GCP infrastructure configuration. The project as a whole allows the implementation of Multi Party Threshold Signature Scheme into projects.

Prior to the test, Cure53 was supplied with the source code of the relevant components as well as access to one GCP project. Additional documentation was supplied via Notion and helped the testers tremendously, especially in reviewing the architecture as a whole.

Throughout the assessment Cure53 stayed in constant contact with the client's team through a dedicated Slack channel. The communication was excellent and made it possible to offer necessary refinement for the already found issues, as well as gaining insight into the components and their configurations.

The Silent Network project audit confirmed the security and effectiveness of its Multi-Party Threshold Signature Scheme (MTSS) using the DKLS protocol. This was focused on Distributed Key Generation (DKG), Distributed Signature Generation (DSG), and Trusted Execution Environment (TEE) integrity.

Initial emphasis was placed on the *wallet-provider-sdk*, which is a critical client-side SDK for user authentication, key generation, and transaction signing. This item was confirmed secure in terms of cryptographic operations, data handling and authentication flows.

Next, the *dkls23-rs* library's cryptographic primitives were confirmed to securely execute DKG and DSG. The complex *el-services-audits-rc3* monorepo containing code for Aggregator and Operator services was scrutinized for its handling of DKG/DSG requests, Operator authentication, and remote attestation in TEEs. A minor issue regarding the unbounded key shape cache size was discovered. It could potentially lead to a Denial-of-Service, as outlined in SIL-04-004.

Dynamic testing further demonstrated the system's resilience to network disruptions, malicious inputs, and concurrency challenges. To that end, it affirmed the robustness and reliability of Silent Network's MTSS implementation.

The analysis of the Silent Network project revealed a generally well-structured and security-conscious codebase. The development team demonstrates an understanding of secure coding practices and has implemented several security controls to protect against common vulnerabilities.

While the services hosted on GCP were used for testing purposes, best practices were generally adhered to. Service accounts had only privileges they needed, while the applications were deployed using a container registry. The deployment of GCP resources was done via *shell* scripts, which might make management harder if the number of resources grows.

Some hardening measures for the small amount of GCP resources could be identified, most of which stem from the resources being deployed with default configurations. Again, this is an area where utilizing a DevOps approach could help securing resources reliably. In addition, the configuration of ingress firewall rules and VM setup requires further hardening to reduce exposure and enhance security, as described in SIL-04-003.

The setup of operators inside a TEE - in this case Intel’s SGX - was found to be implemented solidly. It uses Gramine to wrap a Docker container and for remote attestation which is checked for each TLS session. However, two minor hardening measures could be identified.

First, the configuration of logging for operators was found to be unsuitable for production use but as much is said by the Gramine documentation. Additionally, even though no instances of logging of secrets was found, having Rust log debug message might lead to information disclosure to the untrusted operator admins. Details can be found in SIL-04-001.

Second, one of the files not protected by Gramine is the *resolv.conf* present inside the container. This file could be controlled by an attacker in the given setup. This leads to a range of possible scenarios using a rogue DNS server to redirect operator requests to potentially malicious hosts. For example, this could apply to the PCCS server configured in the SGX config (SIL-04-002)

In general, while the critical core components were easily identifiable, the way different services are deployed was not. Many different ways to deploy the applications were found in the supplied repositories, many of which appeared to only be present for testing purposes or deprecated. While this did not impede testing itself, it made it harder to distinguish between production and testing configurations.

Considering the meager sum total of the documented flaws, Cure53 deemed it apt to provide the Test Methodology chapter, which provides more notes and allows a complete understanding of the auditors’ pentesting techniques and coverage.

All in all, it is recommended that the areas in-scope are tested on a regular basis utilizing a white-box testing approach. Cure53 recommends recurring testing at least once a year or when significant changes are made, to ensure that new features do not introduce undesired security vulnerabilities. This is a proven strategy that will consistently reduce the number of security issues and, over time, make the platform much more resilient to potential attacks.

Cure53 would like to thank Szymon Zimnowoda and Dr. Andrei Bytes from the SILENCE LABORATORIES PTE. LTD. team for their excellent project coordination, support and assistance, both before and during this assignment.