Security assessments for software that matters.

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

Pentest-Report EventVPN Architecture & Software 02.-03.2026

Cure53, Dr.-Ing. M. Heiderich, C. Lüders, Dr. N. Kobeissi, M. V. S. Lima

Index

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

Introduction

This report describes the results of a penetration test and source code audit, as well as a design and cryptographic review conducted against the EventVPN architecture and software.

To give some context regarding the assignment's origination and composition, EventVPN contacted Cure53 in January 2026. The test execution was scheduled for late February / early March 2026, namely from CW09 - CW10. A total of twenty-three days were invested to reach the coverage expected for this project, and a team of four senior testers was assigned to its preparation, execution, and finalization.

The methodology conformed to a white-box strategy, whereby assistive materials such as source code for the EventVPN software, various design documents for the cryptographic trust assumptions, as well as all further means of access required to complete the tests were provided to facilitate the undertakings. The work was split into two separate work packages (WPs), defined as:

- **WP1**: Design & cryptographic trust-assumption reviews against EventVPN architecture
- **WP2**: Dynamic white-box pen.-tests & source code audits against EventVPN software

All preparations were completed in February 2026, specifically during CW08, to ensure a smooth start for Cure53. Communication throughout the test was conducted through a dedicated and shared Slack channel, established to combine the teams of EventVPN and Cure53. All personnel involved from both parties were invited to participate in this channel. Communications were smooth, with few questions requiring clarification, and the scope was well-prepared and clear. No significant roadblocks were encountered during the test. Cure53 provided frequent status updates, shared its findings, and offered live reporting through the aforementioned Slack channel.

The Cure53 team achieved good coverage over the scope items, and identified a total of thirteen findings. Of the thirteen security-related findings, ten were classified as security vulnerabilities, and three were categorized as general weaknesses with lower exploitation potential.

It is advised that the results of this audit left a mixed impression on the Cure53 team. This is mainly due to the fact that while the core VPN and backend systems appeared to maintain a robust security posture, the application's trust boundaries were found to suffer from several impactful issues and weaknesses. Most notably, Cure53 observed a scenario in which it was possible to obtain EventVPN Premium using a fake transaction (see EXP-21-001). As this may lead to significant financial losses, Cure53 recommends addressing this vulnerability as soon as possible.

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

All-in-all, this audit revealed several areas within the EventVPN software which it is advised require further attention and improvement. Moreover, it is advised that it may be beneficial to retest the software following the successful implementation of the fixes recommended in this report, in order to ensure a good level of security.

The report will now shed more light on the scope and testing setup, and will provide a comprehensive breakdown of the available materials. Next, the report will detail the *Test Methodology* used in this exercise. This is intended to show the client which areas of the software in scope have been covered, and which tests have been executed. Following this, the report will list all findings identified in chronological order, starting with the *Identified Vulnerabilities* and followed by the *Miscellaneous Issues* unearthed. Each finding will be accompanied by a technical description, Proof-of-Concepts (PoCs) where applicable, plus any fix or preventative advice to action.

In summation, the report will finalize with a *Conclusions* chapter in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the EventVPN software and architecture.

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

Scope

- **Design review, architecture review & secure code auditing against EventVPN architecture & software**
  - **WP1:** Design & cryptographic trust-assumption reviews against EventVPN architecture
    - **Focus areas:**
      - Detailed design and architecture review of the EventVPN system
      - Client-side architecture, authentication and authorization flows (including SRT/CAT handling), cryptographic assumptions, and interaction between application components
    - **Source code:**
      - eventvpn_source.zip
    - **Documentation:**
      - KPL-Freemium Entitlements - Clients-200226-030655.pdf
      - KPL-Introducing JSON Web Token (JWT)-200226-031144.pdf
      - KPL-JWT Token Types-200226-031403.pdf
      - KPL-Subscription Receipt Service (SRS)-200226-031044.pdf
      - KPL-The Entitlement Claim-200226-031459.pdf
      - KPL-XV Access Token and Subscription Receipt Tokens-200226-031331.pdf
  - **WP2:** Dynamic white-box pen.-tests & source code audits against EventVPN software
    - **Focus areas:**
      - Identifying vulnerabilities in the EventVPN iOS and macOS client applications
      - Verifying robustness of input validation, secure state handling, memory safety properties, and resistance against Denial-of-Service conditions, including malformed configuration data, malicious local inputs, and adversarial network conditions
    - **Target platforms:**
      - iOS 17+
      - macOS 15 (Apple Silicon)
    - **Source code & documentation:**
      - See WP1
  - **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

Fix Verification Status

In the period following the audit, the EventVPN team demonstrated a strong commitment to security by promptly remediating the identified findings. This effort resulted in the successful resolution of every identified vulnerability and weakness. To validate these improvements, the fixes were submitted to Cure53 for a formal verification process. The comprehensive results and current status of this verification exercise are detailed in the following table:

Finding	Status
EXP-21-001 WP2: EventVPN Premium obtainable with fake transactions (Critical)	Fixed
EXP-21-002 WP2: IPv6 traffic bypasses VPN on dual-stack networks (High)	Fixed
EXP-21-003 WP2: Kill switch bypass via silent tunnel failure (Medium)	Fixed
EXP-21-004 WP2: Browser privacy protections bypass (Low)	Fixed
EXP-21-005 WP2: URL spoofing via URL credentials string (Low)	Fixed
EXP-21-006 WP2: URL spoofing via invalid URL scheme (Low)	Fixed
EXP-21-007 WP2: Bypass HTTPS enforcement through redirect (Low)	Mitigated
EXP-21-008 WP2: Fingerprint Protection Disabled When JS Sandbox Is Off (Low)	Fixed
EXP-21-009 WP2: URL spoofing via invalid protocol (Low)	Fixed
EXP-21-010 WP1: SRS authentication method works as SRT issuing oracle (High)	Fixed
EXP-21-011 WP2: Browser privacy settings allows user profiling (Low)	Fixed
EXP-21-012 WP2: Darwin notifications lack authentication (Low)	Fixed
EXP-21-013 WP2: Local DoS via overwriting app group preferences (Low)	Fixed

Table: Fix verification status as of April 2026

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

Severity Glossary

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

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

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

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

Low: Denotes issues that incur considerably limited impact on the areas in scope. These mostly do not depend on the degree of exploitation, but rather on the minor severity of retrievable information or low-grade risk upon the areas in scope.

Info: Denotes issues deemed merely informational in nature. They are mostly considered hardening recommendations or best-practice improvements that will generally enhance the security posture of the areas in scope.

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 project and discusses the resulting coverage. It directly specifies how various components were examined, and provides further clarification concerning areas of investigation subjected to deep-dive assessment.

From a meta-level perspective, Cure53's methodology included both the usage of automated tools and manual testing techniques applied to the targets across the WPs. This ensured a comprehensive review that took both surface-level issues and more complex security concerns into account.

WP1: Design & cryptographic trust-assumption reviews against EventVPN architecture

The design review began with a systematic analysis of the provided architectural diagrams, threat model documentation, and the cryptographic specification governing token issuance and verification across the EventVPN ecosystem. The primary objective was to evaluate trust assumptions that underpin the separation of authentication and authorization - which is a central design decision in the EventVPN architecture.

Significant effort was directed at the JWT-based token chain, with particular attention being given to their signing mechanisms, claim validation requirements, audience restrictions, and expiration policies. The RS256 signature scheme and JWKS-based key distribution were evaluated for correctness, and no weaknesses were identified in the cryptographic primitives or their application. Token lifetimes were found to be appropriately short, with the 3-day SRT window providing a reasonable balance between usability and the ability to revoke access following chargebacks or cancellations.

The Subscription Receipt Service (SRS) authentication chain received a deep-dive assessment, as it constitutes the trust boundary between unauthenticated users and the VPN infrastructure. The chain applies authentication methods sequentially - Bearer Token, Apple Receipt, and finally the *None* fallback - until one succeeds. While the Bearer Token and Apple Receipt methods were found to implement adequate verification, the *None* method was identified as representing a significant architectural weakness. It was found that it issues server-signed SRTs to any caller without authentication, client attestation, or rate-limiting, directly enabling the threat scenario documented in the project's threat model (EXP-21-010).

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

The WireGuard server authentication mechanism was rigorously examined. The flow from CAT issuance through instance discovery to the final WireGuard handshake was traced end-to-end. Key generation, pre-shared key handling, and endpoint authentication were all found to follow established cryptographic patterns. No authentication bypass vulnerabilities or JWT misconfigurations were discovered in this path. The entitlements system - including server group assignments, feature flags, and the feature mapping rules that translate simple flags into structured JWT claims - was reviewed for logic errors or privilege escalation opportunities, and none were found.

Finally, the provided threat model scenarios were used as a structured checklist to guide testing across both WPs. Several of the documented threats - particularly those concerning unauthorized VPN access without ads, IPv6 traffic bypassing the kill switch, and data flowing outside the tunnel - were confirmed as being exploitable during this engagement, validating the threat model's accuracy while highlighting that mitigations had not yet been fully implemented at the time of testing.

WP2: Dynamic white-box pen.-tests & source code audits against EventVPN software

The application was found to use Firebase for data management and storage, requiring a thorough analysis of its configuration. The audit specifically targeted potential misconfigurations in the identity toolkit that could enable unauthorized self-registration, incorrect storage object listings, or other undesired administrative actions. Despite various exploitation attempts targeting these cloud services, all tests were unsuccessful, which confirmed the secure implementation of the Firebase environment.

Further inspection of the application’s configuration revealed the implementation of an *evpn* deeplink. An analysis was performed to validate whether this custom handler could be manipulated to perform unauthorized actions - such as terminating an active VPN connection. However, code inspection confirmed that the handler is restricted to starting the application from the widget, with no further functional surface available for exploitation. Similarly, the widget itself was audited for any misbehavior that could compromise the security of its interactions with the main application, but no relevant vulnerabilities were detected.

Regarding data persistence, an analysis of the keychain manager found consistent adherence to security best practices. Specifically, the implementation of the k*SecAttrAccessibleWhenUnlockedThisDeviceOnly* flag ensures that stored tokens are protected with a high level of security, limiting accessibility to when the device is unlocked, and preventing the migration of secrets to other hardware. Furthermore, the email sender functionality was validated to ensure its integrity. This analysis confirmed that no user-controlled content was included in outgoing emails and that the methodology, with HTML disabled, effectively mitigated the risks of injection or malicious formatting in automated communications.

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

Most effort focused on analyzing the use of the WebView element as a secure browser. It should be noted that this component is generally discouraged for use as a browser due to its lack of modern browser security mechanisms, and here the implementation of custom anti-fingerprinting features raised further concerns. Testing revealed that these measures are not comprehensive, as evidenced by EXP-21-004, which details how device information can still be obtained. Furthermore, EXP-21-007 demonstrates that these protections can be bypassed via redirects to access restricted protocols such as HTTP. Logic errors were also identified, where disabling the JavaScript sandbox silently deactivates anti-fingerprinting protections while still reporting them as active to the user (EXP-21-008). Finally, EXP-21-009 highlights one of multiple URL-spoofing bugs inherent in the WebView architecture. While these UI issues can be mitigated by implementing custom handling and error pages, it is advised that the reliance on a WebView for secure browsing continues to present systemic security challenges.

The local attack surface was subjected to meticulous testing, with a particular emphasis on Inter-Process Communication (IPC) and the storage of sensitive data. The application utilizes multiple IPC technologies for communication with the VPN extension, and the majority of these implementations were found to be secure. However, the application employs Darwin notifications to update its UI state. Given the nature of these notifications, any application on the machine can transmit them and potentially modify the internal state of the application, which could lead to user confusion (EXP-21-012).

Furthermore, it was found that world-writable files could be leveraged to initiate a Denial-of-Service (DoS) condition on the application by continuously writing to them, thereby locking the read operation and preventing the application from loading (EXP-21-013). While these vulnerabilities are not classified as critical, and do not pose an immediate risk of data leakage or remote exploitation for the user, it is advised that their presence indicates a requirement for a more thorough comprehension of the threat modeling pertaining to local services.

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

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, all tickets are given a unique identifier (e.g., EXP-21-001) to facilitate any future follow-up correspondence.

EXP-21-001 WP2: EventVPN Premium obtainable with fake transactions *(**Critical**)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 8.1
CVSS Temporal Score: 7.7
CVSS String: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:H/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/840.html

Testing revealed that the EventVPN iOS application accepts unverified StoreKit2 transactions as valid purchases, effectively bypassing Apple's cryptographic verification of in-app purchase receipts. StoreKit 2 provides two distinct verification states: *verified* and *unverified*, precisely in order to allow applications to reject tampered or forged transactions. The EventVPN app ignores this distinction.

When a purchase is made, the result verification in *PurchaseUseCases.swift* handles the unverified case by logging a debug message and then returning the transaction as though it were valid. This means that a transaction which failed Apple's JWS signature verification is treated identically to a legitimate purchase.

Affected file:
EventVpn/EventVpn/Managers/IAP/UseCases/PurchaseUseCases.swift

Affected code:

```
case let .success(.unverified(transaction, error)):
    LoggerManager.shared.debug("Unverified purchase: \(error)")
    await transaction.finish()
    return transaction
```

The returned unverified transaction is subsequently used in the authentication flow, where its JWS representation is extracted and passed to the KAPE Client SDK via *authenticateWithOpaqueToken()*. Since the JWS is derived from an unverified transaction, the entire downstream token chain - including SRTs, and CATs - is built on a potentially forged foundation.

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

This finding is particularly concerning in the context of the threat scenario documented in the project's threat model, where a user obtains an SRT without watching ads and uses it to access VPN services for free.

To mitigate this issue, it is advised that the application should reject unverified transactions by throwing an error instead of returning them. The *unverified* case should log a security event, and should not call *transaction.finish()*, which permanently acknowledges the transaction with Apple.

EXP-21-002 WP2: IPv6 traffic bypasses VPN on dual-stack networks *(High)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 8.1
CVSS Temporal Score: 7.7
CVSS String: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/16.html

Analysis of the WireGuard tunnel configuration revealed that *allowedIPs* is set exclusively to *0.0.0.0/0* (the IPv4 catch-all), without the corresponding *::/0* (IPv6 catch-all). The tunnel interface's IPv6 settings confirm empty addresses, routes, and prefix lengths - meaning that no IPv6 traffic is routed through the tunnel.

Affected file:
EventVpn/EventVpn-PacketTunnel/VpnManager/Sources/VpnManager/Entities/WGEndpoint.swift

Affected code:
```
public var allowedIPs: String = "0.0.0.0/0"
```

DNS resolution itself is not affected. However, the issue lies in what happens after DNS resolution. When an IPv4 DNS server returns an AAAA record (IPv6 address) for a destination, applications using Happy Eyeballs (RFC 8305) may prefer the IPv6 path. Since the tunnel claims no IPv6 routes, this data traffic falls through to the physical interface's routing table.

On a dual-stack network - where the device's WiFi or cellular interface has an IPv6 address and default route from router advertisements - this IPv6 data traffic exits the physical interface directly, bypassing the VPN tunnel entirely. The user's real IPv6 address is exposed to the destination server.

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

The tunnel interface's empty IPv6 settings (as confirmed by the client's network dump) are the expected configuration of the tunnel side, but do not prevent the physical interface from retaining its own IPv6 connectivity. The DNS address derivation logic further confirms IPv4-only design - the *transformToDot0Dot1* method exclusively handles IPv4 octets:

Affected file:
EventVpn/EventVpn-PacketTunnel/WireGuard/WireGuardProvider.swift

Affected code:
```
let dnsIPString = endpoint.dns,
let dnsIP = IPv4Address(dnsIPString) else {
    logger.error("WireGuardProvider: invalid client private key or internal
ip values.")
    return nil
}
```

This directly corresponds to the threat scenario documented in the project's threat model: when using IPv6, the user's data flows outside the tunnel because the kill switch does not block IPv6 traffic. The practical impact is limited to dual-stack networks, though these represent the majority of modern WiFi and cellular deployments.

To mitigate this issue, it is advised that *::/0* should be added to *allowedIPs* alongside *0.0.0.0/0*, so that the tunnel claims all IPv6 traffic. Alternatively, if the VPN infrastructure does not support IPv6 transit, then it is advised that the tunnel should configure an IPv6 blackhole route (empty *includedRoutes* for IPv6 with no *excludedRoutes*), to ensure that IPv6 traffic is dropped rather than leaking through the physical interface.

EXP-21-003 WP2: Kill switch bypass via silent tunnel failure *(Medium)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 7.5
CVSS Temporal Score: 7.1
CVSS String: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/693.html

Review of the *PacketTunnelProvider* revealed that when the VPN tunnel fails to start or when all endpoint connections fail, the tunnel is intentionally kept active without properly canceling or disabling on-demand reconnection. The error handling code that would perform these critical cleanup operations has been explicitly commented-out:

Affected file:
EventVpn/EventVpn-PacketTunnel/PacketTunnelProvider.swift

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

Affected code:

```
private func startNewSession() async throws {
    do {
        // ... setup code ...
        try await sessionController?.start()
    } catch {
        logger.error("PacketTunnelProvider: received a an error starting
the session: \(error),
            will NOT disable on Demand and stop the tunnel")
        self.userSettingsRepository?.updateNetworkLockApplied(to: true)
//      await disableOnDemand()
//      systemTunnel.cancelTunnelWithError(error)
    }
}
```

The same pattern also appears in the session controller delegate method that handles total endpoint failure:

```
nonisolated func sessionController(_ sessionController: any
SessionControllerType,
    didFailToConnectToAllEndpointsWith error: SessionError) {
    logger.error("PacketTunnelProvider: the session could not connect to
any endpoint
        with error: \(error), will KEEP tunnel active ")
    Task { @MainActor in
        self.userSettingsRepository?.updateNetworkLockApplied(to: true)
    }
//  Task {
//      await disableOnDemand()
//      await systemTunnel.cancelTunnelWithError(error)
//  }
}
```

Furthermore, the *startTunnel()* override swallows all errors and does not re-throw them, preventing the system from detecting that tunnel startup failed:

```
override func startTunnel(options: [String : NSObject]? = nil) async throws
{
    do {
        try await startNewSession()
    } catch {
        logger.error("PacketTunnelProvider: Start tunnel error: \(error)")
    }
}
```

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

The combined effect is that the tunnel enters a zombie state: the system believes that the tunnel is active, the kill switch blocks all traffic, but no actual VPN connection exists. The user's Internet connectivity is completely blocked with no automatic recovery mechanism. While this design choice may have been intentional, to prevent traffic leaks, it creates a DoS condition that persists until the user manually intervenes.

To mitigate this issue, it is advised that the commented-out cleanup code should be restored. At minimum, it is advised that the tunnel should either properly cancel with an error, so the system can inform the user, or it should implement a retry mechanism with exponential backoff that eventually gives up and cleanly terminates.

EXP-21-004 WP2: Browser privacy protections bypass *(Low)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 3.1
CVSS Temporal Score: 3.0
CVSS String: CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:L/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/693.html

The in-app browser window, which is designed with privacy in mind, injects multiple JavaScript functions to prevent the visited website from accessing local settings - such as window size or hardware information. It was discovered that these protective measures could be circumvented by leveraging various techniques - such as using the method *getOwnPropertyDescriptor* and accessing other objects that were not adequately addressed.

Given the complexity of the JavaScript engine and its propensity to store sensitive data across various objects and properties, numerous bypasses are achievable. Consequently, the following PoCs should not be considered to be exhaustive:

PoC #1:
```
Object.getOwnPropertyDescriptor(Navigator.prototype,'hardwareConcurrency').
get.call(navigator);
```

PoC #2:
```
const gl = new OffscreenCanvas(1,1).getContext("webgl"); // Fingerprint
device data
```

PoC #3:
```
typeof WebGLRenderingContext // Get reference, but doesn't leak any
information
```

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

It is recommended to review these protections and to implement a more comprehensive solution. If the threat modeling for such bypasses is less severe, then given that such bypasses will inherently persist to some extent, it is advised that users should be made more aware of these limitations.

EXP-21-005 WP2: URL spoofing via URL credentials string *(Low)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 3.7
CVSS Temporal Score: 3.5
CVSS String: CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/451.html

Assessment of the in-app browser window revealed that the address bar displays the full URL of the page - including authentication credentials. This condition introduces a URL spoofing vulnerability, enabling a malicious website to alter its displayed URL to impersonate another website - such as *google.com*. This deceptive practice could mislead the user into believing that they are on a legitimate site, thereby potentially facilitating phishing attacks.

PoC using basic HTTP basic authentication:

```
<script>
location=`https://www.google.com:123@example.com/`;
</script>
```

Steps to reproduce:

1. Serve and open the PoC using private browsing.
2. Observe that the URL shown is *www.google.com…* with the ellipsis. Depending on the size of the window, more data can be prepended, so as to force the ellipsis.

To mitigate this issue, Cure53 recommends refraining from displaying the full URL within the URL bar. Instead, it is advised that the display should commence with the domain accessed, thereby resolving such spoofing vulnerabilities where additional data could be prepended to the URL.

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

EXP-21-006 WP2: URL spoofing via invalid URL scheme *(Low)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 3.7
CVSS Temporal Score: 3.5
CVSS String: CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/451.html

A further URL spoofing vulnerability was identified concerning the in-app browser's handling of invalid URL schemes. Research demonstrated that the address bar presents the complete URL of the page, even when the URL is invalid. This condition also introduces a susceptibility to URL spoofing.

PoC using invalid URL scheme:

```
<script>
window.location.href=`www.google.com:something`;
</script>
```

Steps to reproduce:

1. Serve and open the PoC using private browsing.
2. Observe that the URL shown is *www.google.com…* with the ellipsis. Depending on the size of the window, more data can be prepended, so as to force the ellipsis.

To mitigate this issue, Cure53 advises that the address bar should only be updated subsequent to the receipt of the response. As an alternative, the EventVPN team could also mandate the immediate replacement of the previously-rendered page with a blank space, following navigation.

EXP-21-007 WP2: Bypass HTTPS enforcement through redirect *(Low)*

***Fix Note**: This issue has been mitigated by the EventVPN team and verified by Cure53 to be working as expected.*

CVSS Score: 3.7
CVSS Temporal Score: 3.5
CVSS String: CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/693.html

Testing confirmed that the application's embedded navigation feature fails to enforce HTTPS when the *forceHTTPS* flag is set. While the protocol is checked when the URL is loaded directly from the URL bar through the *loadWithPrivacySettings* method, it does not prevent pages from redirecting users to HTTP pages, effectively bypassing the application's controls.

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

However, in updated versions of the application, the WebView no longer loads most HTTP pages and only allows requests to localhost and internal addresses. This behavior is consistent with the default restrictions enforced by iOS App Transport Security (ATS), which blocks plain HTTP connections unless explicitly permitted by the application. As a result, the likelihood of users accessing external HTTP pages through the embedded browser is reduced.

Affected file:
kp_event_vpn/EventVpn/EventVpn/Views/Browser/Coordinator/WebViewCoordinator.swift

Affected code:

```
 private func loadWithPrivacySettings(url: URL) {
    // Create a privacy-enhanced URLRequest
    var finalURL = url

    // Force HTTPS if enabled
    if settings.forceHTTPS && url.scheme == "http" {
      if var components = URLComponents(url: url, resolvingAgainstBaseURL:
       false) {
        components.scheme = "https"
        finalURL = components.url ?? url
      }
    }
```

PoC code:

```
<h1>HTTP Redirect</h1>
<script>
location = "http://localhost:3000/";
</script>
```

While this issue could be addressed by moving the check into the *WKNavigationDelegate* implementation, Cure53 recommends avoiding WebViews for open navigation. Webviews do not implement the full set of modern browser security features, and their use for browsing can lead to numerous security issues - directly impacting the safety of VPN users.

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

EXP-21-011 WP2: Browser privacy settings allows user profiling *(Low)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 3.7
CVSS Temporal Score: 3.5
CVSS String: CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:U/C:L/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/693.html

The Private Browsing feature incorporates multiple privacy-focused modifications, and alters the default WebView functionality. These measures are intended to prevent the profiling of the browser and, consequently, the user.

However, the current implementation inadvertently facilitates profiling. By disabling several API functionalities and modifying the error responses to a custom format, it becomes trivial to infer that the user is utilizing EventVPN Private Browsing.

Affected file:
/kp_event_vpn/EventVpn/EventVpn/Views/Browser/Coordinator/WebViewCoordinator.swift

Affected code:

```
    // Block AudioContext fingerprinting (if enabled)
    if settings.blockAudioFingerprinting {
      scriptParts.append("""
        if (window.AudioContext) {
          window.AudioContext = function() {
            throw new Error('Audio context blocked for privacy');
          };
        }
        if (window.webkitAudioContext) {
          window.webkitAudioContext = function() {
            throw new Error('Audio context blocked for privacy');
          };
        }
      """)    }
```

It is recommended not to utilize custom error messages or mock default data. It is advised that it would be more appropriate to introduce random data, so that no discernible difference exists between the default WebView and Private Browsing.

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

EXP-21-012 WP2: Darwin notifications lack authentication *(Low)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 2.6
CVSS Temporal Score: 2.5
CVSS String: CVSS:3.1/AV:N/AC:H/PR:L/UI:R/S:U/C:N/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/306.html

The application utilizes Darwin notifications[1] to communicate state changes from the VPN extension. The vulnerability described here stems from the fact that these notifications - which are not designed to include authentication - can be issued by any application on the machine. Consequently, an arbitrary application can modify the user interface's state - e.g., by misleading the user into believing that the connection is terminated, or by forcing an error popup with a disconnect option. This could lead the user to re-establish the VPN connection, or to close the application due to perceived inconsistencies.

PoC:

```
#!/usr/bin/env swift
import Foundation
notify_post("packetTunnelSessionDidFailToConnect")
```

Affected notifications:

```
packetTunnelStatusDidUpdateToConnected
packetTunnelStatusDidUpdateToConnecting
packetTunnelStatusDidUpdateToDisconnected
packetTunnelStatusDidUpdateToDisconnecting
packetTunnelSessionDidFailToConnect
```

It is advised that Darwin notifications are inappropriate in such scenarios, and that alternative IPC mechanisms are more suitable when the identity of the sending process is a requirement.

[1] https://developer.apple.com/documentation/darwinnotify/darwin-notification-api

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

EXP-21-013 WP2: Local DoS via overwriting app group preferences *(Low)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 2.0
CVSS Temporal Score: 1.9
CVSS String: CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:N/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/400.html

Analysis of shared preferences utilization in the macOS application revealed a vulnerability whereby the application can freeze during startup if the associated *.plist* file is undergoing a write operation. This condition can lead to a DoS state if another - potentially malicious - application continuously writes to this file.

PoC:

```
$ while true; do sleep 0.1 && /usr/libexec/PlistBuddy -c
"Set :isUserSubscribed bool true" "/Users/{USER}/Library/Group
Containers/group.com.ios.eventvpn-app/Library/Preferences/group.com.ios.eve
ntvpn-app.plist"; done
```

It is advised that several mitigations could be implemented here - such as performing the file read asynchronously with appropriate timeout handling. Furthermore, it is advised that the usage of app groups could be discontinued, as these are susceptible to read / write operations by any application, and that alternative IPC functionalities should be utilized for communication with the VPN extension.

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

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.

EXP-21-008 WP2: Fingerprint Protection Disabled When JS Sandbox Is Off *(Low)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 2.0
CVSS Temporal Score: 1.9
CVSS String: CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:N/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/693.html

While investigating the privacy protection settings in the application's navigation feature, it was found that the security flags from the fingerprinting protection section are ignored during navigation when the isolate JavaScript option is disabled. Since the application UI does not provide any indication that this feature is necessary in order to implement the fingerprinting measures (such as turning off the entire section if the sandbox is disabled), the user may inadvertently turn off the sandbox, believing that the fingerprinting options still cover it.

An additional conclusion from this analysis is that this option provides no real JavaScript isolation; both injected scripts and the page scripts run in the same context, except that fingerprint scripts are applied before the page script is executed.

Affected file:
kp_event_vpn/EventVpn/EventVpn/Views/Browser/Coordinator/WebViewCoordinator.swift

Affected code:

```
private func setupPrivacyEnhancements() {
  guard let webView = webView else { return }

  // Inject privacy protection scripts (if enabled)
  if settings.isolateJavaScript {
    let privacyScript = WKUserScript(
      source: getPrivacyProtectionScript(),
      injectionTime: .atDocumentStart,
      forMainFrameOnly: false
    )

webView.configuration.userContentController.addUserScript(privacyScript)
  }
```

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

```
  [...]

}

private func getPrivacyProtectionScript() -> String {
  var scriptParts: [String] = []

  scriptParts.append("(function() {")

  // Spoof screen dimensions (if enabled)
  if settings.spoofScreenDimensions {
    scriptParts.append("""
      Object.defineProperty(screen, 'width', { value: 1920, writable: false
});
      Object.defineProperty(screen, 'height', { value: 1080, writable:
false });
      Object.defineProperty(screen, 'availWidth', { value: 1920, writable:
false });
      Object.defineProperty(screen, 'availHeight', { value: 1080, writable:
false });
    """)
  }

  // Spoof timezone (if enabled)
  if settings.spoofTimezone {
    scriptParts.append("Date.prototype.getTimezoneOffset = function()
{ return 0; };")
  }

  // Spoof hardware info (if enabled)
  [...]

  // Block Canvas fingerprinting (if enabled)
  [...]

  // Block WebGL fingerprinting (if enabled)
  [...]

  // Block AudioContext fingerprinting (if enabled)
  [...]

  // Block camera access (if enabled)
  [...]

  // Block geolocation (if enabled)
  [...]
```

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

```
  // Clear storage periodically (if enabled)
  [...]

  scriptParts.append("})();")

  return scriptParts.joined(separator: "\n")
}
```

Cure53 recommends removing the sandbox option from the UI and always injecting the script, since the fingerprinting settings already configure it. This patch, however, still imposes risk on users due to the use of WebViews for open navigation. It is advised that Webviews do not implement the full set of modern browser security features, and their use for browsing can lead to numerous security issues, directly impacting the safety of VPN users.

EXP-21-009 WP2: URL spoofing via invalid protocol *(Low)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 2.0
CVSS Temporal Score: 1.9
CVSS String: CVSS:3.1/AV:N/AC:H/PR:H/UI:R/S:U/C:N/I:L/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/451.html

During the assessment of the internal WebView, a further URL spoofing exploit was found. In this case, URLs with invalid protocols are displayed in the address bar before the page loads. This happens when the previously-rendered page is displayed. The invalid protocol scheme is fully shown, but it may be unnoticeable to the user, as it is possible to include punctuation such as *https.://*.

PoC code:

```
<h1>Spoofed Page</h1>
<script>
location = "https.://www.eventvpn.com/";
</script>
```

To mitigate this issue, Cure53 recommends reordering the application's visual updates. To be precise, it is advised that requests should be resolved before any alteration to the address bar. In this case, WebView should promptly display an error page indicating that the URL is invalid.

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

EXP-21-010 WP1: SRS authentication method works as SRT issuing oracle *(High)*

***Fix Note**: This issue has been fixed by the EventVPN team and verified by Cure53 to be working as expected. The described issue no longer exists.*

CVSS Score: 6.8
CVSS Temporal Score: 6.4
CVSS String: CVSS:3.1/AV:N/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:N/E:P/RL:U/RC:C
CWE: https://cwe.mitre.org/data/definitions/287.html

Analysis of the SRS authentication chain confirmed that the terminal None method - used as a fallback for EventVPN's freemium model - issues legitimately-signed SRTs to any caller without performing any authentication. The specification states: "*This authentication type does not perform any authentication. Instead, it always returns a list of default entitlements.*"

The authentication chain applies methods top-to-bottom until one succeeds. When all preceding methods (Bearer Token, Apple Receipt) fail or are not applicable, the None method issues a server-signed SRT containing the *default_entitlement_source* configuration. This SRT is cryptographically indistinguishable from any other SRT issued by the SRS - it carries a valid RS256 signature, a proper *iss* claim of *evtvpn.srts*, and a legitimate *aud* array.

An attacker need only send a request to the SRS endpoint without any authentication token. The None method matches, a free-tier SRT is returned, and the attacker exchanges it for a CAT via CATS, then connects to the VPN infrastructure. No rate-limiting, client attestation, or proof-of-origin is specified for this path. This directly realizes the threat scenario described by EventVPN: "*Someone could create a VPN app that provides access to EventVPN infrastructure without limits or ad requirements, costing us money in capacity charges.*"

To mitigate this issue, it is advised that the SRS None path should require client attestation - such as Apple App Attest or DeviceCheck - to prove that the request originates from the legitimate EventVPN application. Further, the SRT issued via None should carry a restrictive audience limiting which CATS instance will accept, and CATS should enforce per-IP and per-device-fingerprint rate-limits on free-tier CAT issuance.

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

Conclusions

As noted in the *Introduction*, this late February / early March 2026 penetration test and source code audit - which included a design and cryptographic review - was conducted by Cure53 against the EventVPN architecture and software.

From a contextual perspective, twenty-three working days were allocated to reach the coverage expected for this project. The methodology used conformed to a white-box strategy, and a team of four senior testers was assigned to the project's preparation, execution, and finalization.

Cure53's assessment of the EventVPN iOS and macOS applications was carried out using both static and dynamic analysis, with full access to the application source code. The scope encompassed a design and cryptographic trust-assumption review of the EventVPN architecture (WP1), as well as dynamic white-box penetration tests and source code audits of the client software (WP2).

The iOS and macOS applications were found to share a substantial portion of their codebase, and consequently, the majority of identified vulnerabilities were common to both platforms. Testing covered the VPN tunnel implementation, the authentication and authorization chain, the StoreKit 2 purchase flow, the in-app Private Browsing feature, IPC mechanisms, local data storage, and third-party service integrations - including Firebase, deep links, and the widget extension.

During this engagement, the most critical findings made concerned the freemium model's trust boundaries. Testing revealed that the application accepts unverified StoreKit 2 transactions as valid purchases (EXP-21-001), enabling an attacker to obtain premium VPN access using forged receipts. Further, the architectural review (WP1) identified a complementary server-side weakness, in that the SRS None authentication fallback issues legitimately-signed SRTs to any unauthenticated caller without client attestation or rate-limiting (EXP-21-010).

Together, the two findings described above realize a threat scenario explicitly documented in the project's threat model - where unauthorized users can obtain VPN access and consume infrastructure capacity at the operator's expense. It is therefore advised that the client-side purchase validation and server-side token issuance paths should be hardened as a priority, with unverified transactions being rejected outright and client attestation - such as Apple App Attest or DeviceCheck - being required for the free-tier SRT issuance path.

The VPN tunnel configuration also presented two notable issues. Firstly, the WireGuard *allowedIPs* setting routes only IPv4 traffic through the tunnel, leaving IPv6 traffic to exit the physical interface unprotected on dual-stack networks - which represent the majority of modern WiFi and cellular deployments (EXP-21-002).

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

Secondly, it was found that error handling in the *PacketTunnelProvider* had been deliberately weakened. Here, cleanup code that would cancel a failed tunnel and disable on-demand reconnection was seen to be commented-out, causing the tunnel to enter a zombie state where the kill switch blocks all traffic but no actual VPN connection exists (EXP-21-003). While this design choice may have been motivated by leak prevention, it trades one failure mode for another without informing the user. It is advised that adding *::/0* to *allowedIPs* and restoring the error handling with a proper retry-and-fail mechanism would address both of these gaps.

Overall, it was found that the largest client-side attack surface has arisen from the use of a WebView as a privacy-focused browser. The Private Browsing feature augments a standard WebView with custom JavaScript intended to prevent fingerprinting and tracking, but it is advised that the implementation introduces additional complexity and expands the potential attack surface, particularly in areas such as fingerprinting resistance and URL handling. Here, multiple URL spoofing vulnerabilities were identified due to deficiencies in URL handling. Authentication credentials were found to be displayed in the address bar (EXP-21-005), invalid URL schemes were rendered without validation (EXP-21-006, EXP-21-009), and HTTPS enforcement could be bypassed via redirects (EXP-21-007).

The fingerprinting protections themselves were also found to be incomplete. The team noted that spoofed values could be recovered through alternative JavaScript APIs such as *getOwnPropertyDescriptor* and *OffscreenCanvas* (EXP-21-004), the protections were silently disabled when the JavaScript sandbox option was turned off (EXP-21-008), and the custom error messages thrown by the privacy scripts made it trivial to identify EventVPN Private Browsing users (EXP-21-011).

The cumulative effect of the issues described here is that the privacy browser may not fully achieve the intended level of anonymity in all scenarios. WebViews do not implement the full set of modern browser security features, and building comprehensive privacy protections on top of this foundation is an ambitious undertaking that would require significantly more investment in order to reach a reliable state. It is advised that the EventVPN team should consider directing users to a dedicated browser application for privacy-sensitive navigation, rather than maintaining a custom in-app solution.

Analysis of the local attack surface revealed two issues related to IPC and shared data storage. It was found that the application uses Darwin notifications to synchronize VPN state between the UI and the tunnel extension, but these notifications are anonymous and can be issued by any process on the system, thereby allowing a malicious co-resident application to manipulate the displayed connection state (EXP-21-012). Similarly, app group preferences used for the storing of user settings were found to be writable by any unsandboxed application, enabling a DoS condition through continuous writes to the shared *.plist* file (EXP-21-013).

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

Both of these issues would require local access and a co-resident malicious application, which limits their practical severity, but it is advised that they still illustrate a pattern where platform mechanisms are relied upon without fully accounting for their trust assumptions. It is advised that alternative IPC mechanisms that authenticate the sending process would mitigate both concerns.

On a positive note, several areas of the application were found to demonstrate solid security practices. The Firebase integration was tested for misconfigurations - including attempts to self-register in the identity toolkit, list storage objects, and access other users' data - and all exploitation attempts were unsuccessful. The *evpn* deeplink handler was found to be appropriately limited, allowing only application launch from the widget, with no exploitable actions, and the widget extension itself presented no security concerns. Keychain usage was also found to follow best practices, with *kSecAttrAccessibleWhen-UnlockedThisDeviceOnly* being correctly applied to stored tokens.

Further, the JWT-based authorization architecture was found to be well-designed in principle. Here, the separation of authentication from authorization was sound, token lifetimes were appropriately short, and JWKS-based signature verification was correctly implemented. The WireGuard tunnel protocol implementation itself - including key generation, pre-shared key handling, and endpoint authentication - was found to follow established patterns, and no cryptographic weaknesses were identified. Finally, the authentication mechanism for the WireGuard servers was rigorously examined, and no authentication bypass vulnerabilities or misconfigurations related to its JWT implementation were discovered.

In summary, the core VPN infrastructure and backend were found to demonstrate a solid security posture, with the primary areas of concern being found to lie at the application's trust boundaries. It is advised that the most impactful improvements would involve hardening the StoreKit 2 transaction validation and the SRS None path (to protect the freemium model from unauthorized access), addressing the IPv6 and kill switch tunnel configuration gaps (to ensure consistent traffic protection), and reconsidering the WebView-based browser approach, which was found to constitute the widest attack surface in the application at the time of testing, due to the inherent complexity of implementing comprehensive privacy protections within this component.

Cure53 would like to thank Brian Schirmacher, Laura Sempre, and Sufiyan Yasa from the EventVPN team for their excellent project coordination, support and assistance, both before and during this assignment.