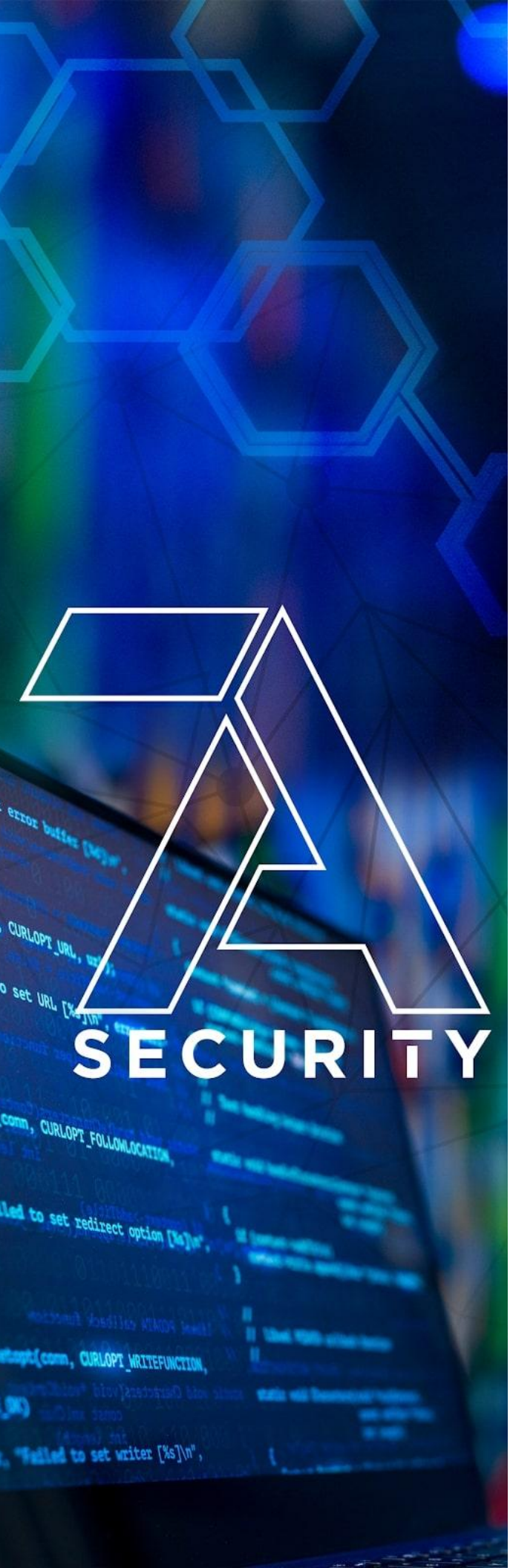

Test Targets:

ArgoVPN Mobile App
ArgoVPN Servers
ArgoVPN Crypto & Design
ArgoVPN Privacy Analysis

Pentest Report

Client:
ArgoVPN

7ASecurity Test Team:

- Abraham Aranguren, MSc.
- Miroslav Štampar, PhD.
- Óscar Martínez, MSc.
- Tarunkant Gupta, BTech.

7ASecurity
Protect Your Site & Apps
From Attackers
sales@7asecurity.com
7asecurity.com

INDEX

Introduction

"ArgoVPN is a free VPN with an unlimited bandwidth that is developed for Android devices. ArgoVPN allows users to visit blocked websites, online services, social media and messaging apps. ArgoVPN is developed according to Iranian citizens' needs to bypass Internet censorship in Iran with premium features, which users can find in commercial VPNs."

From: https://argovpn.com/en/

This document outlines the results of a penetration test and *whitebox* security review conducted against the ArgoVPN platform. The project was solicited by the ArgoVPN Team, funded by the *Open Technology Fund* (OTF), and executed by *7ASecurity* in March 2023. The audit team dedicated 38 working days to complete this assignment. Please note that this is the third penetration test for this project, following two prior audits by Cure53 in 2020. Consequently, identification of new security weaknesses was expected to be particularly challenging during this assignment, as less vulnerabilities are typically found after multiple testing and fixing cycles.

The main purpose of ArgoVPN is to provide unrestricted access to Iranian citizens, as the Internet is otherwise censored by the Iranian government. Hence, during this iteration the goal was to review the tool as thoroughly as possible, to ensure ArgoVPN users can be provided with the best possible security and privacy.

The methodology implemented was *whitebox*: 7ASecurity was provided with access to test servers, documentation, source code and a staging Android binary. A team of 4 senior auditors carried out all tasks required for this engagement, including preparation, delivery, documentation of findings and communication.

A number of necessary arrangements were in place by February 2023, to facilitate a straightforward commencement for 7ASecurity. In order to enable effective collaboration, information to coordinate the test was relayed through email, as well as a shared Signal channel. The ArgoVPN team was helpful and responsive throughout the audit, even during out of office hours and weekends, which ensured that 7ASecurity was provided with the necessary access and information at all times, thus avoiding unnecessary delays. *7ASecurity* provided regular updates regarding the audit status and its interim findings during the engagement.

This engagement split the scope items in the following work packages, which are referenced in the ticket headlines as applicable:

- WP1: Whitebox Tests against ArgoVPN Android app & ArgoVPN Authenticator lib

- WP2: Whitebox Tests against ArgoVPN Servers, Infrastructure & Configuration
- WP3: Whitebox Tests against ArgoVPN Cryptography & Design, Bridge & Falcon
- WP4: Privacy tests against ArgoVPN Android app & Servers

The security audit sections of this report attempt to answer the following question:
Q1: Does the ArgoVPN Android app or the backend servers contain any security flaws that might put users at risk?

The findings of the security audit (WP1-3) can be summarized as follows:

Identified Vulnerabilities	*Hardening Recommendations*	*Total Issues*
2	*11*	*13*

Regarding the privacy audit (WP4), *7ASecurity* directly answers 12 privacy-related questions with a confidence level ranging from *Unclear* to *Proven*. These are described in the Privacy Analysis Findings section of this report.

Moving forward, the scope section elaborates on the items under review, and the findings section documents the identified vulnerabilities followed by hardening recommendations with lower exploitation potential. Each finding includes a technical description, a proof-of-concept (PoC) and/or steps to reproduce if required, plus mitigation or fix advice for follow-up actions by the development team.

Finally, the report culminates with a conclusion providing detailed commentary, analysis, and guidance relating to the context, preparation, and general impressions gained throughout this test, as well as a summary of the perceived security posture of the ArgoVPN solution.

Scope

The following list outlines the items in scope for this project:

- **WP1: ArgoVPN Android app & ArgoVPN Authenticator lib**
  - Audited Versions
    - Android 2.1-audit 5690 (com.filtershekanha.argovpn)
  - Audited Source Code:
    - https://github.com/filtershekanha/ArgoVPN-Android
- **WP2: ArgoVPN Servers, Infrastructure & Configuration**
  - Audited IP addresses:
    - xx.xxx.xx.xxx
    - xx.xxx.xx.xxx
  - Audited Hostnames:
    - corex.redacted.com
    - bridge.redacted.com
    - addr1.redacted.com (ArgoVPN Public Network)
    - addr2.redacted.com (ArgoVPN Public Network)
    - brgtestaddr.redacted.com (ArgoVPN Bridge Network)
    - falcon.redacted.com (ArgoVPN Falcon Network)
    - argovpn.s3.amazonaws.com
  - Audited Source Code:
    - https://github.com/filtershekanha/BotFox/
    - https://github.com/filtershekanha/ArgoBridgeBot/
    - https://github.com/filtershekanha/BridgeDB-Corex/
    - https://github.com/filtershekanha/BridgeDB/
    - https://github.com/filtershekanha/v2ray-core/
- **WP3: ArgoVPN Cryptography & Design, Bridge & Falcon**
  - Audited Source Code:
    - https://github.com/filtershekanha/ArgoVPN-Android
    - https://github.com/filtershekanha/ArgoAuthenticator
- **WP4: Privacy tests against ArgoVPN Android app & Servers**
  - As above

Identified Vulnerabilities

This area of the report enumerates findings that were deemed to exhibit greater risk potential. Please note these are offered sequentially as they were uncovered, they are not sorted by significance or impact. Each finding has a unique ID (i.e. *ARG-01-001*) for ease of reference, and offers an estimated severity in brackets alongside the title.

ARG-01-007 WP1: Possible Phishing via Task Hijacking on Android *(Medium)*

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid. ArgoVPN 2.1.1-audit (5700) was found to implement the proposed mitigation.

Testing confirmed that the Android app is currently vulnerable to a number of task hijacking attacks. The *launchMode* for the app-launcher activity is currently set to *singleTask*, which mitigates task hijacking via *StrandHogg 2.0*[1] while leaving the app vulnerable via older techniques such as *StrandHogg*[2] and other techniques documented since 2015[3].

A malicious app could leverage this weakness to manipulate the way in which users interact with the app. More specifically, this would be instigated by relocating a malicious attacker-controlled activity in the screen flow of the user, which may be useful to perform Phishing, Denial-of-Service or capturing user-credentials. This issue has been exploited by banking malware trojans in the past[4].

Malicious applications typically exploit task hijacking using one or more of the following techniques:

- ***Task Affinity Manipulation***: The malicious application has two activities M1 and M2 wherein *M2.taskAffinity* = *com.victim.app* and *M2.allowTaskReparenting* = *true*. If the malicious app is opened on M2, once the victim application has initiated, M2 is relocated to the front and the user will interact with the malicious application.
- ***Single Task Mode***: If the victim application has set *launchMode* to *singleTask*, malicious applications can use *M2.taskAffinity* = *com.victim.app* to hijack the victim application task stack.

[1] https://www.helpnetsecurity.com/2020/05/28/cve-2020-0096/
[2] https://www.helpnetsecurity.com/2019/12/03/strandhogg-vulnerability/
[3] https://s2.ist.psu.edu/paper/usenix15-final-ren.pdf
[4] https://arstechnica.com/.../...fully-patched-android-phones-under-active-attack-by-bank-thieves/

- ***Task Reparenting***: If the victim application has set *taskReparenting* to *true,* malicious applications can move the victim application task to the malicious application stack.

This issue can be confirmed by reviewing the *AndroidManifest* of the Android application, which fails to set the *android:taskAffinity* attribute at both the application and activity level:

Affected File:
AndroidManifest.xml

Affected Code:
```
<activity android:theme="@style/AppTheme.NoActionBar" android:label="@string/app_name"
android:name="com.filtershekanha.argovpn.ui.ActivityMain" android:exported="true"
android:launchMode="singleTask">
```

The issue was further validated at runtime using the *AttackerApp*[5] from the *Task_Hijacking_Strandhogg* github project[6]. Only the following change was made prior to building the app:

File:
app/src/main/AndroidManifest.xml

Contents Before:
```
android:taskAffinity="com.zombie.ssa"
```

Contents After:
```
android:taskAffinity="com.filtershekanha.argovpn"
```

To ease the understanding of this problem, an example of a malicious app was created to demonstrate the exploitability of this weakness.

PoC Demo:
https://7as.es/ArgoVPN_IbD46LbIjeO/Task_Hijacking_PoC.mp4

It is recommended to implement as many of the following countermeasures as deemed feasible by the development team:

- The task affinity should be set to an empty string. This is best implemented in the Android manifest **at the application level**, which will protect all activities and

[5] https://github.com/az0mb13/Task_Hijacking_Strandhogg/tree/main/AttackerApp
[6] https://github.com/az0mb13/Task_Hijacking_Strandhogg

ensure the fix works even if the launcher activity changes. The application should use a randomly generated task affinity instead of the package name to prevent task hijacking, as malicious apps will not have a predictable task affinity to target.

- The *launchMode* should then be changed to *singleInstance* (instead of *singleTask*). This will ensure continuous mitigation in *StrandHogg 2.0*[7] while improving security strength against older task hijacking techniques[8].
- A custom *onBackPressed*() function could be implemented to override the default behavior.
- The *FLAG_ACTIVITY_NEW_TASK* should not be set in *activity launch* intents. If deemed required, one should use the aforementioned in combination with the *FLAG_ACTIVITY_CLEAR_TASK* flag[9].

Affected File:
AndroidManifest.xml

Proposed Fix:

```
<application android:theme="@style/AppTheme" android:label="@string/app_name"
android:icon="@mipmap/ic_launcher"
android:name="com.filtershekanha.argovpn.ApplicationLoader" android:allowBackup="false"
android:supportsRtl="true" android:usesCleartextTraffic="false"
android:roundIcon="@mipmap/ic_launcher_round"
android:appComponentFactory="androidx.core.app.CoreComponentFactory"
android:taskAffinity="">
[...]
<activity android:theme="@style/AppTheme.NoActionBar" android:label="@string/app_name"
android:name="com.filtershekanha.argovpn.ui.ActivityMain" android:exported="true"
android:launchMode="singleInstance">
```

[7] https://www.xda-developers.com/strandhogg-2-0-android-vulnerability-explained.../
[8] http://blog.takemyhand.xyz/2021/02/android-task-hijacking-with.html
[9] https://www.slideshare.net/phdays/android-task-hijacking

ARG-01-013 WP1/2: ArgoVPN DoS via DNS Spoofing *(High)*

Retest Notes: Improvement Verified. The ArgoVPN team implemented an improvement and 7ASecurity verified that the improvement is valid. ArgoVPN 2.1.3-audit (5730) was found to implement the proposed mitigation.

It was found that the ArgoVPN Android app is vulnerable to DoS attacks via DNS spoofing of the domains used to establish connections (Public, Bridge, or Falcon types) to v2ray servers. Please note that Iranian ISPs block *DNS over TLS* (*DoT*) since at least 2020[10], and *DNS over HTTPS (DoH)* since at least 2022[11]. For this reason, when Iranian users utilize ArgoVPN, the application defaults to using regular DNS, which is prone to DNS spoofing attacks. Although Argo Authenticator pins a public key in the client app to prevent connections to attacker-controlled servers, DNS spoofing of ArgoVPN v2ray server domains is still sufficient for attackers, such as the Iranian government, to prevent legitimate users from using the application. Please note that the impact and reliability of this issue can be increased by verifying the identity of ArgoVPN servers (ARG-01-011) and analysis of DNS traffic patterns (ARG-01-012). As the 7ASecurity test team resides outside of Iran, confirming this issue required simulating Iranian citizens as follows:

Technique 1: Forced selection of *ir* profile via Frida

To confirm this issue the *Frida*[12] framework was used to force the app to set the *ir* profile:

Frida Javascript snippet ir.js:

```
Java.perform(() => {
        var util1 = Java.use('android.telephony.TelephonyManager');
        util1.getSimState.overload().implementation = () => {
                return 5;
         };

        var util2 = Java.use('android.telephony.TelephonyManager');
        util2.getNetworkCountryIso.overload().implementation = () => {
                return "ir";
         };
});
```

Frida Command:

```
frida -U -l ir.js -f com.filtershekanha.argovpn
```

[10] https://ooni.org/post/2020-iran-dot/
[11] https://ooni.org/post/2022-iran-blocks-social-media-mahsa-amini-protests/
[12] https://github.com/frida

dnschef[13] was then used to spoof the DNS response for a Falcon domain (identifiable via ARG-01-011, ARG-01-012):

Command:

```
dnschef.py -i 192.168.68.166 --fakeip 192.168.68.166 --fakedomains falcon.redacted.com
```

Output:

```
[...]
(11:46:05) [*] 192.168.68.111: cooking the response of type 'A' for falcon.redacted.com
to 192.168.68.166
[...]
```

Result:

This results in a fake server error message being displayed by the app, so the user is unable to connect to the v2ray server:

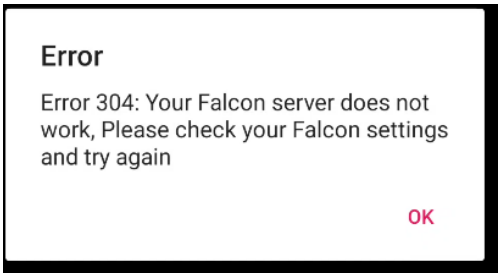

Fig.: Fake server error in Android via DNS spoofing

Technique 2: Set the *SYSTEM DEFAULT* option for DNS resolution

This issue can additionally be confirmed enabling the *SYSTEM DEFAULT* option as the *App Internal DNS Server* from the app configuration, which simulates usage of *Public Network mode* by Iranian users and the given DNS server of the network provider.

[13] https://github.com/iphelix/dnschef

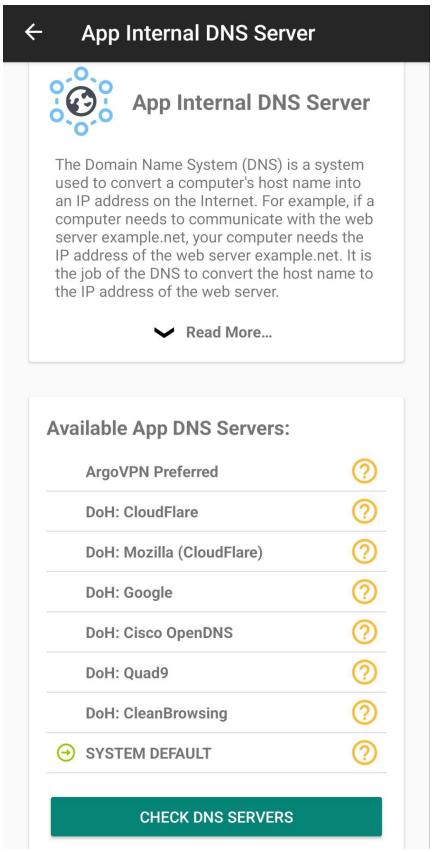

Fig.: Internal DNS Server options in ArgoVPN

Please note the *SYSTEM DEFAULT* option is recommended by the application via the following error message when DoH is actively being disrupted:

Error Message:
Error 112: ArgoVPN can't establish a connection to the server via DNS-over-HTTPS. This usually happens when a government tries to disrupt the Internet. For solving this issue you can go to "Settings" and choose another server in the "App Internal DNS Server" section. If none of those "DoH" options worked, you need to choose "SYSTEM DEFAULT" and try to connect to the ArgoVPN again.

Similar to the prior technique, DNS spoofing of *addr1.redacted.com* and *addr2.redacted.com*, while using the ArgoVPN *Public Network* connection type, resulted in the following error message:

Error Message:
Error 103: We could not find any server to connect

It is recommended to implement not-yet-blocked encrypted methods for DNS resolution. For example, the *DNS-over-QUIC (DoQ)*[14] and *DNS-over-HTTP/3 (DoH3)*[15] protocols could be considered as an alternative, as they are encrypted, resilient to packet losses[16], and require censors to employ new strategies[17,18]. More broadly, the ArgoVPN Android app ought to implement fallback mechanisms for situations where a given DNS resolution protocol fails, to move on to the next. For situations where everything else fails, potential alternatives could include DNS resolution over allowed encrypted protocols or sending clear-text DNS traffic over TCP instead of UDP, which is not prone to UDP DNS hijacking attacks[19].

Hardening Recommendations

This area of the report provides insight into less significant weaknesses that might assist adversaries in certain situations. Issues listed in this section often require another vulnerability to be exploited, need an uncommon level of access, exhibit minor risk potential on their own, and/or fail to follow information security best practices. Nevertheless, it is recommended to resolve as many of these items as possible to improve the overall security posture and protect users in edge-case scenarios.

ARG-01-001 WP1: Weaknesses via Hardcoded Keys in Git History *(Low)*

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid.

It was found that the Android release key and store passwords can be trivially retrieved from the *Git* history. Please note that after communications with the ArgoVPN team during the test, it was later established that the identified hardcoded keys are only employed for a dummy keystore and not the keystore used for official ArgoVPN releases. Hence the impact of this issue is rather low. This issue was confirmed as follows:

Commands:

```
cd ArgoVPN-Android
git diff cc0a834ce6fd36be65046043e0b907479df833be^!
```

[14] https://datatracker.ietf.org/doc/rfc9250/
[15] https://security.googleblog.com/2022/07/dns-over-http3-in-android.html
[16] https://help.nextdns.io/t/x2hmvas/what-is-...-dns-over-quic-doq-and-dns-over-https-doh-doh3
[17] https://datatracker.ietf.org/meeting/115/materials/slides-115-pearg-ooni-...
[18] https://dl.acm.org/doi/pdf/10.1145/3487552.3487836
[19] https://jhalderm.com/pub/papers/iran-foci13.pdf

Output:

```
[...]
diff --git a/gradle.properties b/gradle.properties
index f23a8fc..9a031fc 100644
--- a/gradle.properties
+++ b/gradle.properties
@@ -6,13 +6,10 @@
 # http://www.gradle.org/docs/current/userguide/build_environment.html
 # Specifies the JVM arguments used for the daemon process.
 # The setting is particularly useful for tweaking memory settings.
-RELEASE_KEY_PASSWORD=cSr[...]
-RELEASE_KEY_ALIAS=argo_vpn
-RELEASE_STORE_PASSWORD=cSr[...]
[...]
```

It is recommended to remove all hard-coded credentials, tokens and private keys from the affected repositories. Once that is done, the git history ought to be scrubbed from these sensitive secrets. This could be accomplished utilizing tools like *BFG Repo-Cleaner*[20]. It is advised to invalidate all identified credentials and generate new ones. Automated tools such as *Gitleaks*[21], *GitGuardian*[22], *TruffleHog*[23] and *Git Secrets commit hooks*[24] should be then considered for inclusion in the development process. This will drastically reduce the potential for similar issues in the future, due to repositories being scanned for secrets as developers commit code as well as regularly.

Regarding the removal of credentials from the source code, please note that while environment variables would be better than hard-coding secrets in the source code, these still have downsides and a dedicated secret management tool should be preferred[25]. Instead, applications should retrieve credentials from *AWS Secrets Manager*[26] or an equivalent secure vault that provides the application with credentials as needed at runtime but encrypts them at rest. This ensures that the applications can keep using the credentials while not being available to potential adversaries with access to leaked source code, a developer machine, or any other leak. Furthermore, credentials, secrets, and API keys should be randomly generated to mitigate the potential for brute force or password-guessing attacks. For additional mitigation guidance, please see the

[20] https://rtyley.github.io/bfg-repo-cleaner/
[21] https://github.com/zricethezav/gitleaks
[22] https://www.gitguardian.com/
[23] https://github.com/trufflesecurity/trufflehog
[24] https://github.com/awslabs/git-secrets
[25] https://security.stackexchange.com/questions/197784/is-it-unsafe-to-use-env…
[26] https://aws.amazon.com/.../aws-secrets-manager-store-distribute-and-rotate-credentials.../

OWASP Cryptographic Storage Cheat Sheet[27] and the *CWE-798: Use of Hard-coded Credentials* page[28].

More broadly, it is important to emphasize the importance of having appropriate processes to:

- Regularly rotate credentials
- Revoke and replace credentials in the event of a compromise

ARG-01-002 WP2: Possible Leaks via Debug Mode on BridgeDB *(Low)*

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid.

During the code review of the BridgeDB component, it was found that debug mode is currently enabled. Consequently, PHP error reporting is turned on, hence users will be presented with detailed debugging information in case of any error. While such messages can be helpful for debugging purposes, particularly during development, their content might disclose sensitive implementation details in certain scenarios. Furthermore, even when error messages do not expose substantial information, inconsistencies in such messages may still provide insight regarding how the application works internally. Such details might provide attackers important clues to assist the exploitation of potential flaws in the site. This issue was found within the following affected files and respective code:

Affected File:
BridgeDB/config.php

Affected Code:

```
<?PHP
if (count(get_included_files()) == 1) exit("Direct access not permitted.");
return array(
    "DEBUG" => true
);
```

Affected File:
BridgeDB/BridgeRequest.php

Affected Code:

```
<?php
```

[27] https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html
[28] https://cwe.mitre.org/data/definitions/798.html

```
$configs = require('config.php');
require_once(__DIR__ . '/../vendor/autoload.php');

if ($configs["DEBUG"]) {
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);
} else {
    error_reporting(0);
}
```

It is recommended to use an environment variable for enabling debugging mode, where the default value should be *false*. More broadly, the server ought to save detailed error messages on the server-side and only provide a correlation ID on the client-side. This allows developers to retain debugging capabilities by looking up the correlation ID on the server, without leaking any sensitive information to API clients. For additional mitigation guidance, please see the *OWASP Error Handling Cheat Sheet*[29].

ARG-01-003 WP2: Possible Weaknesses via Absent Security Headers *(Medium)*

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid.

A selection of HTTP security headers was confirmed to be absent from a number of servers related to the ArgoVPN platform. Even though this does not imply a vulnerability at present, this lack of header integration may assist an attacker to exploit certain weaknesses. This issue can be confirmed with the following commands, which show all HTTP security headers are missing on at least some endpoints:

Affected Hosts:
bridge.redacted.com
corex.redacted.com
falcon.redacted.com
addr1.redacted.com

Command:

```
curl -I https://bridge.redacted.com
```

Output:

```
HTTP/2 200
date: Sun, 05 Mar 2023 16:13:22 GMT
```

[29] https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html

```
content-type: text/html; charset=UTF-8
vary: Accept-Encoding
strict-transport-security: max-age=63072000
cf-cache-status: DYNAMIC
report-to:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=oRdXnEskhfXy7kTcVHtZ
fpdZ44nITKq6HcrPhUbnkTVVL2yz5fZlvK4I6kJcNNOnrhprsorrzrmP2RcxXH9qywN8rAoq7Mxnzv%2F4Qcpg%
2FXFzWnA6gAkgJS1hOQBrTAj7cLaAO8YbMWUCGoMj6JDxvEatInc%3D"}],"group":"cf-nel","max_age":6
04800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 7a33a550d993f3f9-BOM
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
```

To prevent a host of associated flaws, the following headers require careful review from the developer team:

- *X-Frame-Options*: Defines if framing is permitted. While effective to protect from clickjacking attacks, a framable web page can facilitate many other attack scenarios[30]. SAMEORIGIN or DENY are appropriate values in most cases.
- Some *X-Frame-Options* limitations may be offset by leveraging the CSP framework, which offers comparable protective guarantees. It is proposed to implement a simultaneous deployment of the *Content-Security-Policy: frame-ancestors 'none';* header to safeguard users of both modern and older browsers.
- *X-Content-Type-Options*: Defines if resource MIME sniffing should be initiated by the browser. Omitting this header is widely known to assist a specific attack scenario that manipulates the browser into rendering a resource as an HTML document, which ultimately incurs Cross-Site-Scripting (XSS).
- *Strict-Transport-Security (HSTS)*: When missing, this allows adversaries to downgrade HTTPS traffic to clear-text HTTP, hence facilitating MitM attacks using widely available tools, like *sslstrip*[31]. It is advised to deploy HSTS as follows:

  Strict-Transport-Security: max-age=31536000; includeSubDomains;

  It is recommended to avoid the HSTS *preload* due to its DoS potential[32].

In general, integrating security headers is considered a best security practice and should be actioned by the developer team where appropriate. The aforementioned headers

[30] https://cure53.de/xfo-clickjacking.pdf
[31] https://moxie.org/software/sslstrip/
[32] https://www.tunetheweb.com/blog/dangerous-web-security-features/

should be inserted into each and every server response, including error responses such as 4xx. Since these headers should be set consistently, 7ASecurity would like to underline the significance of retaining all HTTP headers in a specific, shared, and central area. A load balancing server or similar could be deployed to achieve this, though if this is deemed infeasible, mitigation could be achieved by utilizing the web server configuration in conjunction with a matching module.

ARG-01-004 WP2: TLS Hardening Recommendations *(Medium)*

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid.

It was found that the TLS configuration of several ArgoVPN servers has a number of minor weaknesses that could be resolved. While these issues do not constitute any significant security finding at present, they might become serious as new attacks continue to be discovered and fall into the public domain. Furthermore, these misconfigurations may facilitate MiTM attacks against outdated clients. Please note that many more servers are likely affected by this issue, only the list below is provided for brevity purposes. The TLS shortcomings can be summarized as follows:

1. Support of TLS protocols with known vulnerabilities: TLS 1.0, TLS 1.1
2. Support of weak ciphers

Affected Hosts:
bridge.redacted.com
corex.redacted.com
addr1.redacted.com
falcon.redacted.com

PoC URLs:
https://www.ssllabs.com/ssltest/analyze.html?d=bridge.redacted.com&hideResults=on
https://www.ssllabs.com/ssltest/analyze.html?d=corex.redacted.com&s=172.67.220.146&hideResults=on&latest
https://www.ssllabs.com/ssltest/analyze.html?d=addr1.redacted.com&s=172.67.220.146&hideResults=on
https://www.ssllabs.com/ssltest/analyze.html?d=falcon.redacted.com&hideResults=on&latest

It is recommended to deploy TLS correctly to solve these problems. This should be done on all servers, including those that were out of scope during this assignment. The

OWASP TLS Cheat Sheet[33] is a valuable resource to do this. Ultimately, the *SSL Labs* website[34] can be helpful to verify the configuration when the website is reachable online. The goal should be to obtain an "A ranking" from the SSL Labs website. Alternatively, the *OWASP O-Saft* tool[35] may facilitate testing the TLS configuration of servers that are not reachable via the Internet.

ARG-01-005 WP1: Support of Insecure v1 Signature on Android *(Info)*

Note: The ArgoVPN team accepted the risk and decided to maintain support of older Android versions for now. The reason for this is that most users in Iran cannot afford new devices due to poor economic conditions, which leads to the continued popularity of outdated Android devices in the country[36].

It was found that the Android build currently in production is signed with an insecure *v1 APK signature*. Using the *v1* signature makes the app prone to the known Janus[37] vulnerability on devices running Android < 7. The problem lets attackers smuggle malicious code into the APK without breaking the signature. At the time of writing, the app supports a minimum SDK of 17 (Android 4.2), which also uses the v1 signature, hence being vulnerable to this attack. Furthermore, Android 4.2 devices no longer receive updates and are vulnerable to many security issues, it can be assumed that any installed malicious app may trivially gain root privileges on those devices using public exploits[38][39][40].

The existence of this flaw means that attackers could trick users into installing a malicious attacker-controlled APK which matches the v1 APK signature of the legitimate Android application. As a result, a transparent update would be possible without warnings appearing in Android, effectively taking over the existing application and all of its data.

It is recommended to increase the minimum supported SDK level to at least 24 (Android 7) to ensure that this known vulnerability cannot be exploited on devices running older Android versions. In addition, future production builds should only be signed with v2 and greater APK signatures.

---

[33] https://cheatsheetseries.owasp.org/.../Transport_Layer_Protection_Cheat_Sheet.html
[34] https://www.ssllabs.com/ssltest/
[35] https://owasp.org/www-project-o-saft/
[36] https://newsblog.cafebazaar.ir/[...]-blwiz74cckyd
[37] https://www.guardsquare.com/en/blog/new-android-vulnerability-allows-atta….affecting-their-signatures
[38] https://www.exploit-db.com/exploits/35711
[39] https://github.com/davidqphan/DirtyCow
[40] https://en.wikipedia.org/wiki/Dirty_COW

ARG-01-006 WP1: Android Binary Hardening Recommendations *(Info)*

It was found that a number of binaries embedded into the Android application are currently not leveraging the available compiler flags to mitigate potential memory corruption vulnerabilities. This unnecessarily puts the application more at risk for such issues.

Issue 1: Binaries missing usage of -D_FORTIFY_SOURCE=2

Missing this flag means common *libc* functions are missing buffer overflow checks, so the application is more prone to memory corruption vulnerabilities. Please note that most binaries are affected, the following is a reduced list of examples for the sake of brevity.

Example binaries (from decompiled production app):
lib/armeabi-v7a/libgojni.so
lib/x86_64/libgojni.so
lib/x86/libgojni.so
lib/arm64-v8a/libgojni.so
lib/armeabi-v7a/libtun2socks.so
lib/x86/libtun2socks.so

Issue 2: Missing stack canaries on some binaries

A number of binaries do not have a stack canary value added to the stack. Stack canaries are used to detect and prevent exploits from overwriting return addresses.

Affected Binaries:
lib/armeabi-v7a/libgojni.so
lib/x86_64/libgojni.so
lib/x86/libgojni.so
lib/arm64-v8a/libgojni.so

It is recommended to compile all binaries using the *-D_FORTIFY_SOURCE=2* argument so that common insecure *glibc* functions like *memcpy*, etc. are automatically protected with buffer overflow checks. Regarding stack canaries, the *-fstack-protector-all* option can be leveraged to enable stack canaries.

ARG-01-008 WP1: Increased User Risk via Missing Root Detection *(Info)*

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid. ArgoVPN 2.1.1-audit (5700) was found to implement the proposed mitigation.

It was found that the ArgoVPN Android app does not implement any form of root detection features at the time of writing. Hence, the application fails to alert users about the security implications of running the app in such an environment[41]. This issue can be confirmed by installing the application on a rooted device and validating the complete lack of application warnings.

It is recommended to implement a comprehensive root detection solution to address this problem. Please note that, since the user has root access and the application does not, the application is always at a disadvantage. **Mechanisms like these should always be considered bypassable** when enough dedication and skill characterize the attacker.

The freely available *rootbeer* library[42] for Android could be considered for the purpose of alerting users on rooted devices, while bypassable, this would be sufficient for alerting users of the dangers of running the app on rooted devices.

ARG-01-009 WP2: Missing Content Security Policy *(Info)*

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid.

It was found that the ArgoVPN backend applications do not currently leverage the protections offered by the *Content Security Policy* (CSP)[43]. This unnecessarily makes the websites more prone to Cross Site Scripting (XSS) issues. Malicious attackers might leverage this weakness to exploit XSS issues with significantly less difficulty. This weakness can be verified with the following command:

Affected Hosts:
corex.redacted.com
bridge.redacted.com
falcon.redacted.com
addr1.redacted.com

[41] https://www.bankinfosecurity.com/jailbreaking-ios-devices-risks-to-users-enterprises-a-8515
[42] https://github.com/scottyab/rootbeer
[43] https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

Command:

```
curl -I https://bridge.redacted.com
```

Resulting HTTP headers:

```
HTTP/2 200
date: Sun, 05 Mar 2023 16:13:22 GMT
content-type: text/html; charset=UTF-8
vary: Accept-Encoding
strict-transport-security: max-age=63072000
cf-cache-status: DYNAMIC
report-to:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=oRdXnEskhfXy7kTcVHtZ
fpdZ44nITKq6HcrPhUbnkTVVL2yz5fZlvK4I6kJcNNOnrhprsorrzrmP2RcxXH9qywN8rAoq7Mxnzv%2F4Qcpg%
2FXFzWnA6gAkgJS1hOQBrTAj7cLaAO8YbMWUCGoMj6JDxvEatInc%3D"}],"group":"cf-nel","max_age":6
04800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 7a33a550d993f3f9-BOM
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
```

It is recommended to implement a CSP Configuration using the *report-only*[44] mode first to ensure all functionality remains working. The *Google CSP Evaluator* website[45] can then be used to verify potential risks in the CSP settings. A possible CSP configuration that might be considered as a starting point for testing could be the following:

Proposed Fix:

```
Content-Security-Policy: default-src self; img-src https: data:; font-src 'self' data:;
object-src: 'none'; frame-ancestors 'none'
```

[44] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy-Report-Only
[45] https://csp-evaluator.withgoogle.com/

ARG-01-010 WP2: Weaknesses via Vulnerable jQuery Usage *(Info)*

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid.

During the code review, it was found that the BridgeDB component utilizes jQuery v1.9.0, which was released in 2012 and is now obsolete and vulnerable to multiple XSS and prototype-pollution vulnerabilities[46]. While most of these weaknesses are likely not exploitable under the current implementation, this is still a bad practice that could result in unwanted security vulnerabilities and highlights room for improvement in the current software patching processes. These issues can be confirmed reviewing the following files:

Affected File:
https://bridge.redacted.com/jquery.min.js

Affected Code:
```
/*! jQuery v1.9.0[...]
```

Steps to reproduce:

1. Open the login page via https://bridge.redacted.com/
2. Open Inspect Element using Ctrl+Shift+I in the browser
3. Navigate to the Console tab and enter the command *jQuery().jquery* - this will display the outdated version number of the JQuery library

It is recommended to upgrade the *jQuery* library to its latest version, which at the time of writing is jQuery v3.6.4[47]. To avoid similar issues in the future, an automated task and/or commit hook should be created to regularly check for vulnerabilities in dependencies. Some solutions that could help in this area are the *Snyk* tool[48] and the *OWASP Dependency Check* project[49]. Ideally, such tools should be run regularly by an automated job that alerts a lead developer or administrator about known vulnerabilities in dependencies so that the patching process can start in a timely manner.

[46] https://www.cybersecurity-help.cz/vdb/jquery/jquery/1.9.0/
[47] https://releases.jquery.com/jquery/
[48] https://snyk.io/
[49] https://owasp.org/www-project-dependency-check/

ARG-01-011 WP2/3: Possible Server Fingerprinting via Probing (*Medium*)

Retest Notes: Improvement Verified. The ArgoVPN team implemented an improvement and 7ASecurity verified that the improvement is valid.

It was found that censors can accurately fingerprint ArgoVPN v2ray servers sending a number of crafted HTTP requests. A malicious attacker, censor or government could leverage this weakness to fingerprint ArgoVPN servers as a prior step to permanently block all traffic to the IP address of the identified server. Fingerprinting may be trivially accomplished by running the following commands to suspected ArgoVPN servers:

Fingerprint 1: Unusual 520 HTTP response code

Command:

```
curl -i 'https://falcon.redacted.com/'
```

Output:

```
HTTP/2 520
date: Sat, 18 Mar 2023 18:59:32 GMT
content-length: 0
cache-control: no-store, no-cache
cf-cache-status: DYNAMIC
report-to:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=0qS%2BwdeuVZvcqbukLc
nLv4rp01yqt5SdtKDAiQJtKQU1tPaKvNSPZtpuWSudy9jB2zn3aBBFOK1Ty3tM8%2BttIe1gz0f4q5zEjrC1MDi
oxFoxeEBVe1JNivTIPuFco%2B20b5FCbHRxQq0%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
strict-transport-security: max-age=15552000
server: cloudflare
cf-ray: 7a9fb69e2bcc6d61-LIM
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
```

Fingerprint 2: 500 error response via invalid XXXXX headers

Command:

```
curl -i -H 'XXXXX: any' 'https://falcon.redacted.com/'
```

Output:

```
HTTP/2 500
date: Sat, 18 Mar 2023 18:59:52 GMT
content-type: text/plain;charset=UTF-8
content-length: 1336
strict-transport-security: max-age=15552000
x-frame-options: deny
```

```
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
referrer-policy: same-origin
cf-cache-status: DYNAMIC
report-to:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=Rhqm0h9wl1e6kc%2B0na
F%2BKGUAQOxoRYLS%2FN1Yl%2B80bN3mIzM8eUQAlESt%2F6qbzAUJa02FlbD7XSgzr2VW6LsuFwtbvORhHjWxZ
SM3LuUYQ0B8%2F6DkJj0M3WSuva5oo6doeyAeDuxZ%2F1c%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 7a9fb71ae9e66d65-LIM
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400

nGJZJKz3Obhf-av3q[...]
```

Fingerprint 3: Predictable 200 response to valid XXXXX headers

The following XXXXX header value can be sent to any v2ray server for fingerprinting purposes:

XXXXX header before being encrypted:

```
{"request":"auth","protocol":"A","version":2,"secret":"Zuas0IVFHyTs2Y304AutiG0vxghpwIrZ
IvkJQ0P9fDE\u003d","salt":"EW+MlGMpzRZa715RTQBelRN3oKudQCFQUBMyyWWQDUc\u003d"}
```

Command:

```
curl -i -H 'XXXXX: H4sIAA[...]' 'https://falcon.redacted.com/'
```

Output:

```
HTTP/2 200
date: Sat, 18 Mar 2023 19:00:16 GMT
content-type: text/plain;charset=UTF-8
content-length: 188
strict-transport-security: max-age=15552000
x-frame-options: deny
x-xss-protection: 1; mode=block
x-content-type-options: nosniff
referrer-policy: same-origin
cf-cache-status: DYNAMIC
report-to:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=sxv4AntY8fyQgc2f1O%2
B3yU0mFVhkBKlR5XXMoJBJ07x%2FNDOEiBFBVauJjxvidN2o8LS8wKk14GeW6EFrR%2F5c4X88cbgOk6nlYB8fR
2i%2FyjWbwQg8hkfoF4%2BptRcnL8enJz%2BzDxNUntM%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 7a9fb7b1eb826d62-LIM
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
```

```
0uCK0pX3bhYuxaOxJSOOY5m0CNqlK7eha8hldtlBXCUk95DvKh+Z65XJo4JmeiV5FKmGxz6YADBRIAs8b0Hn4Cu
l0ibY0GOq+YxlTgDA6i1L9vx8v4SmKEQNmGeYeY6fHtqGoY+yP9N6LmHFOmUouzyI88/aMURaS7uZQ6Emsytwhx
mrqiK3g7ZUj1o=
```

It is recommended to implement some level of server-side obfuscation and unpredictability to increase the difficulty to identify ArgoVPN servers. For example:

- Respond with 200 HTTP status codes to all incoming requests, the client determines what to do based on the HTTP response body.
- Replace the *XXXXX* header with a more generic name such as: *Authorization*, *X-Auth* or *X-Authorization*.
- Respond with randomly generated responses where possible.
- Consider using generic responses such as the Apache *"It works!"* page.

ARG-01-012 WP1: Possible Fingerprinting via DNS Traffic Patterns (*Medium*)

Retest Notes: Improvement Verified. The ArgoVPN team implemented an improvement and 7ASecurity verified that the improvement is valid. ArgoVPN 2.1.3-audit (5730) was found to implement the proposed mitigation.

One of the tasks for this engagement was to pinpoint unique network traffic patterns that might allow fingerprinting ArgoVPN clients and servers. It was found that the DNS traffic of the ArgoVPN Android app, which is the default for Iranian users due to network traffic restrictions, reliably reveals this information. A malicious adversary, with access to network communications between the ArgoVPN client and servers (i.e. public Wi-Fi MitM, ISP MitM, BGP hijacking, etc.), might leverage this weakness to precisely fingerprint ArgoVPN clients and servers, as a potential preliminary step prior to blocking their traffic. This issue was confirmed simulating DNS Man-In-The-Middle (MitM) as follows:

Step 1: Simulate DNS MitM

Change the DNS settings on the Android device so that they point to the attacker-controlled DNS server.

Step 2: Launch the fake DNS server

Launch a tool like *dnschef*[50] and run it on the DNS IP address selected in the prior step:

[50] https://github.com/iphelix/dnschef

Command:

```
dnschef -i 192.168.68.166
```

Step 3: Observe the resulting DNS traffic

The ArgoVPN app can now be launched to observe the resulting DNS traffic.

Result:
The following DNS requests were captured, please note how Falcon connections are established later on:

Output:

```
[...]
(06:40:37) [*] DNSChef started on interface: 192.168.68.166
(06:40:37) [*] Using the following nameservers: 8.8.8.8
[...]
(10:22:09) [*] 192.168.68.111: proxying the response of type 'AAAA' for
raw.githubusercontent.com
(10:22:09) [*] 192.168.68.111: proxying the response of type 'A' for
raw.githubusercontent.com
(10:22:09) [*] 192.168.68.111: proxying the response of type 'AAAA' for get.geojs.io
(10:22:09) [*] 192.168.68.111: proxying the response of type 'A' for get.geojs.io
(10:22:09) [*] 192.168.68.111: proxying the response of type 'AAAA' for
time.cloudflare.com
(10:22:09) [*] 192.168.68.111: proxying the response of type 'A' for
time.cloudflare.com
[...]
(10:45:19) [*] 192.168.68.111: proxying the response of type 'AAAA' for
falcon.redacted.com
(10:45:19) [*] 192.168.68.111: proxying the response of type 'A' for
falcon.redacted.com
```

As can be seen above, DNS traffic reveals a specific pattern that could help the Iranian government to fingerprint the IP addresses of both the AgoVPN clients and the ArgoVPN v2ray servers.

Given the deliberate blocking of *DNS over TLS* (*DoT*) and *DNS over HTTPS* (*DoH*) in Iran, mitigating this issue is particularly challenging. Nevertheless, it is advised to investigate the potential resolution of DNS queries utilizing other mechanisms that avoid revealing hostnames over clear-text traffic. For a number of mitigation strategies to consider in this realm, it is suggested to extrapolate the mitigation guidance offered under ARG-01-013.

Privacy Analysis Findings

This section covers the privacy-related analysis results that attempt to answer 12 questions for *WP4: Privacy tests against ArgoVPN Android app & Servers*. For this portion of the engagement, the *7ASecurity* team utilizes the following classification to specify the level of certainty regarding the documented findings. Given that this research occurred on the basis of reverse-engineering, and source code analysis, it is necessary to classify the findings to address the level of confidence that can be assumed for each discovery:

- Proven: Source code and runtime activity clearly confirm the finding as fact
- Evident: Source code strongly suggests a privacy concern, but this could not be proven at runtime
- Assumed: Indications of a potential privacy concern was found but a broader context remains unknown.
- Unclear: Initial suspicion was not confirmed. No privacy concern can be assumed.

ARG-01-Q02: Files & Information gathered by ArgoVPN (*Unclear*)

Retest Notes: Improvement Verified. The ArgoVPN team only saves hashes of user data on the server-side and 7ASecurity verified that the improvement is valid. User information such as the IP address is no longer saved in ArgoVPN servers.

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q2: What files/information are gathered by the ArgoVPN app and servers?

During the code review and dynamic analysis of the ArgoVPN backend servers and mobile application, it was found that ArgoVPN does not collect sensitive information from its users or the devices they use. Furthermore, in the only instance where some data is gathered, this is deleted every 24 hours. It should be noted that ArgoVPN server access logs are explicitly disabled. The following sections elaborate on these findings:

Part 1: Data Collected by the ArgoVPN Android application

The ArgoVPN Android application was not found to send any user information to ArgoVPN servers during this assignment.

Part 2: Data Collected by ArgoVPN Servers

The ArgoVPN servers generally collect no user information and even disable access logs. However, a couple of scenarios exist where minimal user data is gathered:

When users navigate to *bridge.redacted.com* to obtain a bridge address for ArgoVPN, the BridgeDB server collects the *user_ip*, *user_agent*, *user_country*, and *user_asn* in the *web_requests* table of the *db_bridge* database. This information is deleted every 24 hours and used as a method to slowly reveal v2ray servers to users, depending on the distribution method. 7ASecurity could find no evidence of misuse of this data during this assignment. The data captured can perhaps be best illustrated running the following command on the database:

Command:

```
MariaDB [db_bridge]> select * from web_requests;
```

Output:

```
| req_id    | req_datetime              | bridge_id | user_ip         | user_agent
| user_country | user_asn | is_argovpn | is_tor | new_bridge |
+---------+--------------------+-----------+-------------+--------------------------
-------------------------------------------------------------------------------+------
--------+----------+------------+--------+------------+
| 5212242 | 2023-03-26 20:59:33 |       2009 | xx.xxx.xx.xxx | Mozilla/5.0 (Windows NT
10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36 | GB
| M247 Ltd |          0 |      0 |          1 |
| 5212243 | 2023-03-26 20:59:44 |       2009 | xx.xxx.xx.xxx | Mozilla/5.0 (Windows NT
10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36 | GB
| M247 Ltd |          0 |      0 |          0 |
```

The code capturing the above information is located in the following code path:

Affected File:
BridgeDB/BridgeRequest.php

Affected Code:

```
[...]
public function getBridge($ip, $ua)
{
        $this->ip = $this->conn->real_escape_string($ip);
        $this->ua = $this->conn->real_escape_string($ua);
        $this->country = $this->conn->real_escape_string($this->getCountry());
        $this->asn = $this->conn->real_escape_string($this->getAsn());
[...]
private function saveRequest($bridgeId, $isNew)
```

```
{
        $sql = "INSERT INTO `web_requests` (req_datetime, bridge_id, user_ip,
user_agent, user_country, user_asn, is_argovpn, is_tor, new_bridge) VALUES
(now(),'$bridgeId','$this->ip' ,'$this->ua', '$this->country', '$this->asn', '" .
($this->isArgoUser ? 1 : 0) . "' ,'" . ($this->isTorUser ? 1 : 0) . "', '" . ($isNew ?
1 : 0) . "' );";

        return ($this->conn->query($sql) === true);
}
[...]
```

Another scenario occurs when an admin user sends requests to *corex.redacted.com*. In this case, the BridgeDB server collects the *user_ip* and *user_agent* in the *api_access* table of the *db_bridge* database. The *user_ip* is used to decide if the user is blocked or not. This issue can be confirmed with the following commands:

Command:

```
MariaDB [db_bridge]> select * from api_access limit 2;
```

Output:

```
| access_id | access_time         | user_ip        | user_agent             | success |
comment |
+-----------+---------------------+----------------+------------------------+---------+-
--------+
|       3132 | 2023-02-13 21:06:28 | xx.xxx.xx.xxx | PostmanRuntime/7.30.1 |       1 |
NULL    |
|       3133 | 2023-02-13 21:09:29 | xx.xxx.xx.xxx | PostmanRuntime/7.30.1 |       1 |
NULL    |
```

The above data is collected by the following code snippet:

Affected File:
BridgeDB-Corex/src/main/java/com/filtershekanha/corex/service/AccessService.java

Affected Code:

```
public void addApiAccess(String ip, String userAgent, boolean success) {
        ApiAccess apiAccess = new ApiAccess();
        apiAccess.setUserIp(ip);
        apiAccess.setUserAgent(userAgent);
        apiAccess.setSuccess(success);

        accessRepository.save(apiAccess);
}
```

Part 3: Regular deletion of data

It is important to note that the ArgoVPN servers contain multiple code and configurations in charge of deleting the minimal data gathered every 24 hours. The following examples illustrate this:

Affected File:
BridgeDB-Corex/src/main/resources/application.properties

Affected Code:
```
conf.LogRemoverCron=0 0 0/1 * * ?
```

Affected File:
BridgeDB-Corex/src/main/java/com/filtershekanha/corex/utils/LogRemover.java

Affected Code:
```
    @Scheduled(cron = "${conf.LogRemoverCron}")
    @Transactional
    public void logRemoverTask() {
        if (appConfig.getDebug()) {
            logger.info("Log Remover Task :: Execution Time - {}",
dateTimeFormatter.format(LocalDateTime.now()));
        }

        // Take care of the last 24H stats
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.HOUR, -24);
        Date date = calendar.getTime();

        statsService.saveStats(date);
        statsService.deleteOldStats(date);
```

Affected File:
BridgeDB-Corex/src/main/java/com/filtershekanha/corex/service/StatsService.java

Affected Code:
```
    public void deleteOldStats(Date date) {
        webRequestsRepository.cleanRequests(date);
    }
```

Affected File:
BridgeDB-Corex/src/main/java/com/filtershekanha/corex/db/repo/WebRequestsRepository.java

Affected Code:

```
@Query("delete from WebRequests where reqDatetime <= :theTime")
void cleanRequests(@Param("theTime") Date theTime);
```

Affected File:

ArgoBridgeBot/src/main/java/com/filtershekanha/bridgebot/service/LogRemoverService.java

Affected Code:

```
@Scheduled(cron = "0 0 0/1 * * ?") // Every Hour
@Transactional
public void logRemoverTask() {
    requestService.doLogClean();
}
```

Affected File:

ArgoBridgeBot/src/main/java/com/filtershekanha/bridgebot/service/RequestService.java

Affected Code:

```
public void doLogClean() {
    Calendar calendar = Calendar.getInstance();
    calendar.add(Calendar.HOUR, -appConfig.getMinRequestHour());
    Date date = calendar.getTime();

    long count = requestRepository.deleteAllByReqTimeBefore(date);
    System.out.println(count + " records deleted");
}
```

Affected File:

ArgoBridgeBot/src/main/java/com/filtershekanha/bridgebot/sql/repo/RequestRepository.java

Affected Code:

```
public interface RequestRepository extends CrudRepository<Request, Integer> {

    Optional<Request> findByUserId(Long userId);

    Long deleteAllByReqTimeBefore(Date date);
}
```

Part 4: Access logs are disabled

While reviewing the Argo VPN servers in scope, it was also discovered that even the access logs are disabled, hence ArgoVPN user IPs will generally not be leaked in the server file system either:

Affected File (ArgoVPN server):
/usr/local/etc/v2ray/config.json

Affected Code:

```
  "log": {
    "loglevel": "none"
  },
```

Affected File (ArgoVPN & BridgeDB servers):
/etc/nginx/nginx.conf

Affected Code:

```
error_log /var/log/nginx/error.log crit;
[...]
    access_log off;
```

In conclusion, ArgoVPN collects almost no user data and disables access logs. Furthermore, in the few instances where some data is collected, this is deleted after 24 hours. The following minimal potential improvements to user privacy could be considered for future releases:

1. Hashing of user data (i.e. User IP, User Agent, etc.) prior to storage would ensure that, even during the 24 hour window prior to deletion, user data would be non-trivial to recover even for attackers with access to ArgoVPN servers.
2. Further reduction of the amount of information gathered by the servers to the minimum possible necessary for the solution to work.

ARG-01-Q03: Where & How ArgoVPN transmits Data (*Unclear*)

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q3: Where and how are the files/information gathered transmitted?
What information can the mobile operators and ISP see, if a user is using the app in a high risk scenario?

As stated in ARG-01-Q02, the ArgoVPN mobile app and servers collect almost no user information. The remaining privacy-relevant areas are user traffic protection and DNS resolution, as users of the ArgoVPN Android app connect to the Internet and proxy their traffic through ArgoVPN servers. The following sections elaborate on the 7ASecurity impressions gathered in this regard during the code review, server audit and runtime analysis of the ArgoVPN solution:

Part 1: How general user traffic is protected

When ArgoVPN Android app users connect to the service, a VPN tunnel is created and hence all data is protected with two layers of encryption:

1. The first layer uses TLS between the Mobile app and Cloudflare servers at the transport layer of the network TCP/IP stack.
2. The second layer leverages *VMESS* (V2Ray Modified Encryption Secure Messaging Protocol) at the application level, which encrypts communications between the ArgoVPN server running the V2Ray proxy solution and the ArgoVPN mobile application. Hence, ArgoVPN servers receive encrypted *VMESS* proxy requests, encapsulated within the *WebSockets* or *GRPC* protocols.

The above implementation means that both layers of encryption would need to be defeated to intercept network communications.

Please note that, similar to Tor exit nodes, the ArgoVPN servers proxy user traffic and may have access to clear-text HTTP traffic, in situations where users access clear-text HTTP websites. However, this is simply a result of how proxying works and no suspicious code or functionality could be identified during the code review and server audit of ArgoVPN servers.

In short, ArgoVPN correctly secures network communications and therefore does not allow mobile operators or ISPs to inspect network traffic. TLS and *VMESS* validation was found to be implemented correctly, and minor TLS (ARG-01-004) and HTTP

security header improvements (ARG-01-003) were implemented by the ArgoVPN team during the audit.

Part 2: How user DNS traffic protected

In most configurations, DNS traffic is protected using DNS over HTTPS (DoH), when this is the case, the DNS request is not sent to ArgoVPN servers, but rather third parties like the following:

DNS over HTTPS (DoH) query endpoints

- https://cloudflare-dns.com/dns-query
- https://mozilla.cloudflare-dns.com/dns-query
- https://dns.google/dns-query
- https://dns11.quad9.net/dns-query
- https://doh.opendns.com/dns-query
- https://doh.cleanbrowsing.org/doh/security-filter

ARG-01-Q04: ArgoVPN protects PII at rest & in transit (Unclear)

Retest Notes: Improvement Verified. The ArgoVPN team only saves hashes of user data on the server-side and 7ASecurity verified that the improvement is valid. User information such as the IP address is no longer saved in ArgoVPN servers.

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q4: Is sensitive PII insecurely stored or easily retrievable from the app or servers?

ArgoVPN does not store any Personal Identifiable Information (PII) at the time of the assessment. However, since IP addresses can be considered PII (i.e. US CCPA, EU GDPR), a possible improvement in this area would be to save only a one-way hash of the client IP in the few scenarios that ArgoVPN servers store them, as recommended in ARG-01-Q02.

ARG-01-Q05: ArgoVPN protects Data at Rest & In Transit (*Unclear*)

Retest Notes: Improvement Verified. The ArgoVPN team only saves hashes of user data on the server-side and 7ASecurity verified that the improvement is valid. User information such as the IP address is no longer saved in ArgoVPN servers.

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q5: Do the app and servers protect the data appropriately at rest and in transit?

The ArgoVPN servers were not found to store or cache user traffic (images, web pages, etc.), disable access logs, store little data, and the only data collected is removed every 24 hours (ARG-01-Q02), additionally data in transit is protected by two layers of encryption as explained in ARG-01-Q03. Implementing the recommendations provided in the aforementioned tickets will further improve the protection of data at rest and in transit.

ARG-01-Q06: ArgoVPN does not gather more Data than necessary (*Unclear*)

Retest Notes: Improvement Verified. The ArgoVPN team only saves hashes of user data on the server-side and 7ASecurity verified that the improvement is valid. User information such as the IP address is no longer saved in ArgoVPN servers.

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q6: Is there any data gathered on the app & servers beyond what is necessary for the service?

During the code review and runtime analysis of the ArgoVPN app, 7ASecurity did not find any evidence to suggest that ArgoVPN is collecting excessive amounts of data beyond what is required for the solution to operate.

As explained in ARG-01-Q02 and ARG-01-Q05, ArgoVPN does not collect user PII, disables server access logs, does not cache user traffic artifacts (images, web pages, etc.), and in the only scenarios where minimal user data such as IP addresses are collected, these are removed after 24 hours. The minimal information collected is necessary in the current implementation as it is used for the distribution method (i.e. ArgoVPN server discovery process), as well as blocking users after a certain number of tries, This can be confirmed in the following code snippets:

Affected File:
BridgeDB/BridgeRequest.php

Affected Code:

```
private function isMinTimePassed()
    {

        if ($this->isTorUser && $this->unique_tor_request) {
            $condition = "`is_tor` = 1";
        } elseif ($this->isArgoUser && $this->unique_argo_request) {
            $condition = "`is_argovpn` = 1";
        } else {
            switch ($this->distribute_method) {
                case 'ip':
                    $condition = "`user_ip` = '$this->ip'";
                    break;
                case 'ua':
                    $condition = "`user_agent` = '$this->ua'";
                    break;
                case 'country':
                    $condition = "`user_country` = '$this->country'";
                    break;
                case 'asn':
                    $condition = "`user_asn` = '$this->asn'";
                    break;
                case 'global':
                default:
                    $condition = "1 = 1";
            }
        }

        $sql = "SELECT * FROM `web_requests` WHERE  $condition  AND TIMEDIFF(now(),
`req_datetime`) <= '$this->min_req_time' AND `new_bridge` = 1 ORDER BY `req_id` DESC
LIMIT 1;";
        $this->last_query_result = $this->conn->query($sql);

        return !$this->hasRow();
}
```

Affected File:
BridgeDB-Corex/bin/main/application.properties

Affected Code:

```
conf.maxLoginTry=3
```

Affected File:
BridgeDB-Corex/src/main/java/com/filtershekanha/corex/service/Authenticator.java

Affected Code:
```
public AuthResult getAuthenticated(String user, String key, String ip, String
userAgent) {
    if (isBlocked(ip)) {
        return AuthResult.BLOCKED;
}
[...]
private boolean isBlocked(String ip) {
    ArrayList<ApiAccess> apiAccess = accessService.getFailedAttempts(ip);
    return apiAccess.size() >= appConfig.getMaxLoginTry();
}
```

This being said, implementing the recommendations provided in ARG-01-Q02 will further improve the privacy of ArgoVPN users.

ARG-01-Q07: ArgoVPN does not appear to track Users (Unclear)

Retest Notes: Improvement Verified. The ArgoVPN team only saves hashes of user data on the server-side and 7ASecurity verified that the improvement is valid. User information such as the IP address is no longer saved in ArgoVPN servers.

This ticket summarizes the 7ASecurity attempts to answer the following question

Q7: Does the app implement any sort of user tracking function via location or other means?

The audited ArgoVPN Android app and servers were not found to contain face recognition artifacts, collect user location information or track user activity such as visited websites or DNS requests.

The only exception to this is explained in ARG-01-Q02 and ARG-01-Q06, where the ArgoVPN server must temporarily collect minimal information related to the *user_ip*, *user_agent*, *user_country*, and *user_asn* as a method to slowly reveal v2ray servers to users, depending on the distribution method. However, even in this case, the data is deleted after 24 hours.

Similarly, the ArgoVPN Android application only gathers information related to the country where the device is located, language, time zone, and IP address, to decide which profile to use, but this information is not sent to the server.

For minor potential improvements to further reduce user tracking potential, it is advised to extrapolate the mitigation guidance offered under ARG-01-Q02.

ARG-01-Q08: ArgoVPN does not seem to weaken Crypto intentionally (Unclear)

Retest Notes: Fix Verified. The ArgoVPN team resolved this issue and 7ASecurity verified that the fix is valid.

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q8: Does the app intentionally weaken cryptographic procedures to ensure third-party decryption?

7ASecurity identified a number of minor cryptographic weaknesses during this assignment, as described in ARG-01-004. Nevertheless, these did not appear to be intentional security weaknesses introduced to facilitate third party decryption.

ARG-01-Q09: ArgoVPN does not use the SD Card insecurely (Unclear)

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q9: Is data dumped in the SD Card from where it could be retrieved later without even entering the PIN to unlock the device?

During the code review and dynamic analysis of the ArgoVPN Android mobile app, no evidence could be identified to suggest that the mobile application uses the SD Card.

ARG-01-Q10: ArgoVPN seems free from RCE Vulnerabilities (Unclear)

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q10: Does the app or servers contain vulnerabilities or shell commands that could lead to RCE in any way?

7ASecurity did not identify any vulnerability that could lead to RCE either directly or indirectly during this engagement. Specifically, no RCE weaknesses were found in the source code provided by ArgoVPN, the embedded binaries in the mobile app or in any third-party dependencies in use.

ARG-01-Q11: ArgoVPN does not appear to contain Backdoors (Unclear)

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q11: Does the app or servers have any kind of backdoor?

The 7ASecurity team was unable to identify any backdoors in the source code provided by ArgoVPN, as well as the underlying dependencies and binaries reviewed during this engagement. In short, no backdoor signs were found within any ArgoVPN component at runtime or at rest. Specifically, all common backdoor mechanisms were checked, including suspicious file access, unexpected back-connect attempts, execution of operating system commands, and exfiltration attempts of obfuscated content, to name a few.

ARG-01-Q12: ArgoVPN seems free from Root PrivEsc Artifacts (Unclear)

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q12: Does the app attempt to gain root access through public Android vulnerabilities or in other ways?

In a similar fashion to the approach followed to answer ARG-01-Q11, 7ASecurity performed multiple attempts to identify code, binary artifacts and dependencies that might result in the ArgoVPN app gaining root privileges.

Once again, the audit team was unable to find any potential root privilege escalation either via direct prompts in a rooted environment or through the exploitation of system

vulnerabilities. Instead, the ArgoVPN application was found to respect its limited privileges in the expected manner.

ARG-01-Q13: ArgoVPN implements Obfuscation (Assumed)

This ticket summarizes the 7ASecurity attempts to answer the following question:

Q13: Does the app use obfuscation techniques to hide code and if yes for which files and directories?

During the audit, 7ASecurity was provided with access to audit the source code of all ArgoVPN components (i.e. mobile app & server backend). This source code was not found to be obfuscated in any way and no attempts to hide functionality could be identified.

For the building process of the Android APK, it was confirmed that the ArgoVPN Android application uses the *ProGuard*[51] code obfuscation tool, setting the *minifyEnabled*[52] option to *true*. This hides the package, class and function names, renaming them to single character strings. This means that identifier renaming is enabled, while string encryption is not used and Java reflection is not present. Usage of the aforementioned settings can be found in the following configuration file:

Affected File:
ArgoVPN-Android/app/build.gradle

Affected Code:

```
buildTypes {
        [...]

        release {
            minifyEnabled true
            signingConfig signingConfigs.release
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
}
```

The following example illustrates the difference from the original file to the obfuscated file:

[51] https://www.guardsquare.com/proguard
[52] https://developer.android.com/studio/build/shrink-code

Original File:
ArgoVPN-Android/app/src/main/java/com/filtershekanha/argovpn/utils/AppUtils.java

Original Code:

```
package com.filtershekanha.argovpn.utils;
[...]
public static boolean isDebuggerPresent() {
        if (BuildVars.DEBUG_VERSION) return false;

        // Battery Temperature Check
        Intent intent = ApplicationLoader.applicationContext.registerReceiver(null, new
IntentFilter(Intent.ACTION_BATTERY_CHANGED));
        if (intent != null) {
            float temp = ((float) intent.getIntExtra(BatteryManager.EXTRA_TEMPERATURE,
0)) / 10;
            if (temp == 0.0f) return true;
        } else {
            return false;
        }

        // TODO: CPU Temperature Check

        // Emulator Check
        String product = Build.PRODUCT;
        if (product != null) {
            if (product.contains("sdk") || product.contains("vbox")) {
                return true;
            }
        }

        // Check if apk patched for debugging
        if
((ApplicationLoader.applicationContext.getApplicationContext().getApplicationInfo().fla
gs & ApplicationInfo.FLAG_DEBUGGABLE) != 0) {
            return true;
        }

        // Check for debugger attached
        if (Debug.isDebuggerConnected()) {
            return true;
        }

        return false;
}
```

Decompiled File:
331eae3d4b0d7c75282e7904d6436764-java/v/d.java

Decompiled Code:

```
package v;
[...]
public static boolean n() {
        Intent registerReceiver = ApplicationLoader.f2702a.registerReceiver(null, new
IntentFilter("android.intent.action.BATTERY_CHANGED"));
        if (registerReceiver != null) {
            if (registerReceiver.getIntExtra("temperature", 0) / 10.0f == 0.0f) {
                return true;
            }
            String str = Build.PRODUCT;
            if ((str != null && (str.contains("sdk") || str.contains("vbox"))) ||
(ApplicationLoader.f2702a.getApplicationContext().getApplicationInfo().flags & 2) != 0
|| Debug.isDebuggerConnected()) {
                return true;
            }
        }
        return false;
}
```

Result:
Developer comments are removed, package names and function names are replaced by random single letter characters, etc.

Please note that given the nature of ArgoVPN, which aims to provide uncensored access to the Internet, it is difficult to find a balance between user trust by making everything open source vs. obfuscation to make censorship of ArgoVPN more difficult by the Iranian government:

On one hand, user trust could be improved by making certain ArgoVPN components open source. However, making all ArgoVPN components open source would substantially ease censorship attempts. A possible compromise could be to make parts of the solution open source, while components that might facilitate censorship remain obfuscated.

Conclusion

The ArgoVPN solution defended itself well against a broad range of attack vectors. Being its third audit, the relatively few findings identified during this iteration proves the value of regular penetration testing. The platform will become increasingly difficult to attack as additional cycles of security testing and subsequent hardening continue.

From a security perspective, ArgoVPN provided a number of positive impressions during this assignment that must be mentioned here:

- First of all, the ArgoVPN team was very responsive during the test and promptly fixed the vast majority of security and privacy weaknesses identified. By the time the report was sent, most fixes and privacy improvements were already implemented.
- The mobile app and backend servers offer relatively little attack surface, which drastically reduces chances for security vulnerabilities. Similarly, no evidence could be found to suggest that either the mobile or backend components store sensitive data or PII. While ArgoVPN servers are configured to prevent port exposure, dictionary attacks and unnecessary leaks via log storage. Another positive impression in this regard is that secrets are encrypted at rest in Java properties using *Jasypt*[53].
- The code audit performed on all components provided a positive impression whereby the source code appears to be mature, previously audited, professionally written, appropriately commented and offering little opportunity to security or privacy concerns.
- The platform is based on solid foundations, for example, the infrastructure and servers are adequately configured, patched and hardened. While trust on popular cloud providers (e.g. Cloudflare) is successfully leveraged for circumvention of Internet restrictions posed by the local government.
- The mobile application was found to be safe from Denial-of-Service (DoS), redirect vulnerabilities, Deeplink attacks, as well as leaks via Android backups, log messages or screenshots. Additionally, it correctly protects DNS traffic via DNS over HTTPS (DoH) by default, except for Iranian users, where this type of traffic is blocked. All in all, ArgoVPN user traffic is sufficiently protected with two layers of encryption, as well as an adequate selection of encryption algorithms. Furthermore, once the VPN connection is established, network traffic generated by the ArgoVPN app is indistinguishable from regular HTTPS traffic
- The ArgoVPN backend servers were found to be robust against many traditional web application security attack vectors. For example, no Command Injection,

[53] http://www.jasypt.org/

SQLi, XSS, CSRF, SSRF, or RCE issues could be identified during this assignment. Furthermore, strict output encoding is in place for every user input parameter and the implementation for banning users was found to be well implemented.

- Access Control seems to be generally well implemented, where unauthorized users are unable to query APIs. Additionally, the servers implement authentication based on credentials and whitelist IP addresses where possible.

The ArgoVPN mobile application was found to be affected by a number of common misconfigurations. Its security posture will improve substantially with a focus on the following areas:

- **Denial-of-Service (DoS):** It is recommended to implement appropriate fallback mechanisms to better protect users against DoS attacks. The application should adapt faster to new restrictions where DNS resolution mechanisms are always one step ahead of the censors (ARG-01-012, ARG-01-013).
- **Hijacking Attacks:** The Android application should mitigate well-known Task Hijacking attacks (ARG-01-007).
- **General Hardening:** Other less important hardening recommendations include implementing a root detection mechanism to alert users about security risks prior to using the application (ARG-01-008), a number of settings that could be improved to better protect users on older supported devices (ARG-01-005) and rolling out binary protections to mitigate potential memory corruption vulnerabilities (ARG-01-006).

The security of the backend server components will improve substantially with a focus on the following areas:

- **Software Patching:** The solution should implement appropriate software patching procedures which regularly apply security patches in a timely manner (ARG-01-010). In a day and age when most lines of code come from underlying software dependencies, regularly patching these becomes increasingly important to avoid unwanted security vulnerabilities. Possible automation for this could include tools like *Snyk.io*[54] or *Renovate Bot*[55].
- **Secret Management** should be improved to ensure application secrets are not disclosed via hardcoded credentials or the commit history (ARG-01-001). Instead, these ought to be stored outside of the source code to reduce the potential for leaks and privilege escalation throughout the infrastructure. Special care should be taken to ensure credentials are also removed from the github history. The development team should then perform global searches and educate

[54] https://snyk.io/
[55] https://github.com/renovatebot/renovate

developers to avoid similar issues in the future. More broadly, adequate IT security and DevSecOps procedures are needed at the infrastructure level.

- **TLS Hardening:** A number of servers support insecure TLS protocols with publicly known security vulnerabilities (ARG-01-004). An effort should be made to address these issues and ensure the TLS configuration is hardened to protect users from Man-In-The-Middle (MitM) attacks.
- **Information Leaks:** It is also important to avoid leaking information (ARG-01-002) that facilitates the exploitation of other vulnerabilities. Similarly, appropriate protections should be in place to make fingerprinting of ArgoVPN servers more difficult (ARG-01-011).
- **Modern Browser Security Features**: The platform would benefit from improving its usage of modern web technologies such as the Content Security Policy (CSP) (ARG-01-009) and appropriate HTTP Security headers (ARG-01-003). An adequate implementation of these security controls will reduce the potential for XSS and other client-side attacks, hence protecting users in edge-case scenarios.

Regarding the ArgoVPN privacy audit, the following positive impressions should be mentioned first:

- ArgoVPN stores no PII (ARG-01-Q04), disables access logs and deletes client IP addresses every 24 hours (ARG-01-Q02). Furthermore, the servers do not collect any more data than strictly necessary for the service to work (ARG-01-Q06).
- ArgoVPN implements two layers of encryption to protect user communications (ARG-01-Q03, ARG-01-Q05) and does not appear to intentionally weaken cryptography to facilitate third party decryption (ARG-01-Q08).
- 7ASecurity was unable to find any evidence of user location, face recognition, user traffic or user DNS resolution tracking during this assignment (ARG-01-Q07).
- No insecure usage of the Android SD Card (ARG-01-Q09), No RCE vulnerabilities (ARG-01-Q10) and no backdoors (ARG-01-Q11) could be found in the ArgoVPN mobile app or backend servers in scope during this iteration.
- The ArgoVPN Android app does not attempt to gain root privileges (ARG-01-Q12).

The privacy posture of the ArgoVPN solution will improve significantly with a focus on the following areas:

- **Data Gathering:** ArgoVPN collects almost no user information, which is deleted every 24 hours. Minimal potential improvements to this could include attempts to reduce the amount of data collected further and/or hash the remaining information prior to storage for additional privacy protection (ARG-01-Q02).

Please note that one-way hashing of user data was implemented at the end of this assignment. Hence, ArgoVPN no longer stores any user data, such as client IP addresses, on its servers.

- **Potential for an Open Source Implementation:** Once all recommendations from this report are applied, ArgoVPN could consider making certain components open source. This would make the source code available to any third party, proving ArgoVPN has “nothing to hide” and making its implementation fully transparent and open to scrutiny (ARG-01-Q13). At the same time, some components should likely be kept closed source, in order to increase the effort required to censor ArgoVPN users.

It is advised to address all issues identified in this report, including informational and low severity tickets where possible. This will not just strengthen the security posture of the application significantly, but also reduce the number of tickets in future audits.

Once all issues in this report are addressed and verified, a more thorough review, ideally including another code audit, is highly recommended to ensure adequate security coverage of the platform. This provides auditors with an edge over possible malicious adversaries that do not have significant time or budget constraints.

Please note that future audits should ideally allow for a greater budget so that test teams are able to deep dive into more complex attack scenarios. Some examples of this could be third party integrations, complex features that require to exercise all the application logic for full visibility, authentication flows, challenge-response mechanisms implemented, subtle vulnerabilities, logic bugs and complex vulnerabilities derived from the inner workings of dependencies in the context of the application. Additionally, the scope could perhaps be extended to include other Internet-facing ArgoVPN resources or fuzzing.

It is suggested to test the application regularly, at least once a year or when substantial changes are going to be deployed, to make sure new features do not introduce undesired security vulnerabilities. This proven strategy will reduce the number of security issues consistently and make the application highly resilient against online attacks over time.

7ASecurity would like to take this opportunity to sincerely thank Nariman Gharib and the rest of the ArgoVPN team, for their exemplary assistance and support throughout this audit. Last but not least, appreciation must be extended to the *Open Technology Fund (OTF)* for sponsoring this project.