RADICALLY OPEN SECURITY

Penetration Test Report

SMSWithoutBorders

V 1.3
Amsterdam, May 5th, 2023
Confidential

Document Properties

Client	SMSWithoutBorders
Title	Penetration Test Report
Targets	SMSWithoutBorders Android app SMSWithoutBorders back-end code
Version	1.3
Pentester	Abhinav Mishra
Authors	Abhinav Mishra, Marcus Bointon
Reviewed by	Marcus Bointon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	March 22nd, 2023	Abhinav Mishra	Initial draft
0.2	March 24th, 2023	Marcus Bointon	Review
1.0	March 25th, 2023	Marcus Bointon	1.0
1.1	April 17th, 2023	Abhinav Mishra	Retest Report
1.2	April 19th, 2023	Abhinav Mishra	Retest Report
1.3	May 5th, 2023	Marcus Bointon	Retest Review

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1 Executive Summary

1.1 Introduction

Between March 2, 2023 and March 22, 2023, Radically Open Security B.V. carried out a penetration test for SMSWithoutBorders.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the penetration test was limited to the following targets:

- SMSWithoutBorders Android app
- SMSWithoutBorders back-end code

The scoped services are broken down as follows:

- Penetration testing: 4-5 days
- Code review of the back end: 1-2 days
- Reporting: 1 days
- Scoping: 0.5 days
- PM/Review: 0.5 days
- Retest: 1-2 days
- **Total effort: 8 - 11 days**

1.3 Project objectives

ROS will perform a penetration test of the SMSWithoutBorders Android app, and review the SMSWithoutBorders back-end code with SWOB in order to assess their security. To do so ROS will use the Android app and inspect the back-end code and guide SWOB in attempting to find vulnerabilities, exploiting any such found to try and gain further access and elevated privileges.

1.4 Timeline

The security audit took place between March 2, 2023 and March 22, 2023. We spent around 48 hours in the penetration testing of the mobile app, and reviewing the back-end code.

1.5 Results In A Nutshell

During this crystal-box penetration test we found 1 High, 1 Elevated, 6 Moderate and 4 Low-severity issues.

The penetration test focused on the SMSWithoutBorders Android app and its back-end code. The high-severity vulnerability in SWB-012 (page 13) concerns a lack of input validation in one of the endpoints, leading to a reflected cross site scripting vulnerability. In SWB-006 (page 15), we found that the Android app does not require re-authentication when a user disables biometric security controls.

The moderate severity security issues concern a lack of randomly generated initialisation vectors during encryption in both Python SWB-014 (page 26) and Java SWB-001 (page 27); sessions are not synchronised between web and mobile app for a user account in SWB-011 (page 25), password policy is not enforced on the back end in SWB-010 (page 22), missing security headers in SWB-004 (page 20), the Android app leaking URLs in logs in SWB-003 (page 18) and use of TLS 1.0 and 1.1 in the backend domain in SWB-002 (page 16). The low severity findings are about improper CORS configuration in SWB-009 (page 31), clear text traffic allowed in Android app in SWB-007 (page 29), and the Android app not having a logout or delete option in SWB-005 (page 29).

By exploiting these issues, an attacker might be able to target the application users, steal user information from device, or leak information. Fixing these issues will considerably improve the security of application and code.

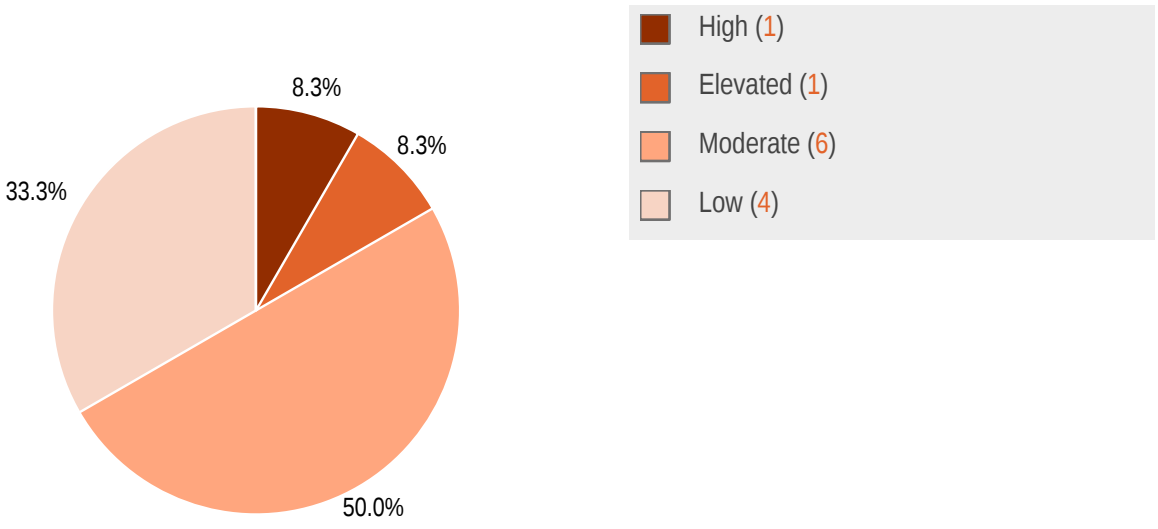
Update:
In follow-up retests on April 13th – 17th, all findings have been resolved.

1.6 Summary of Findings

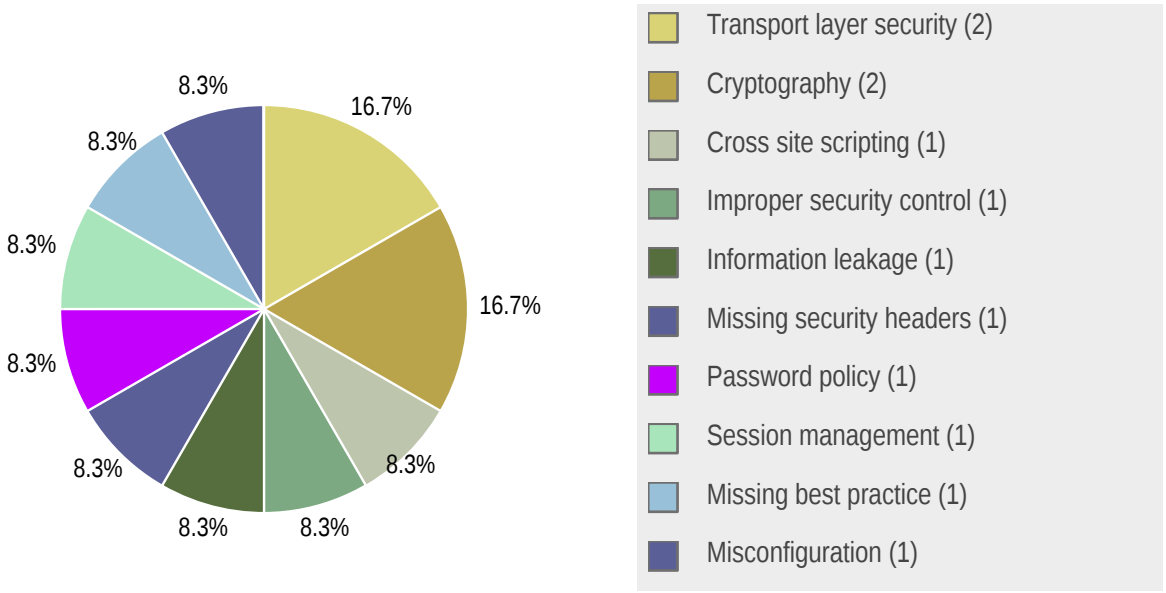
ID	Type	Description	Threat level
SWB-012	Cross site scripting	One of the endpoints used within the app is vulnerable to cross site scripting.	High
SWB-006	Improper security control	The application does not enforce re-authentication when a user disables the security settings.	Elevated
SWB-002	Transport layer security	Obsolete TLS protocol versions are supported by the host at staging.smswithoutborders.com.	Moderate
SWB-003	Information leakage	The Android app (com.afkanerd.sw0b) logs URLs in the logcat.	Moderate
SWB-004	Missing security headers	The back-end service that the Android app connects to is missing important security HTTP headers.	Moderate

SWB-010	Password policy	It is possible to set a weak password for the account, as the password policy checks are only implemented on the client side.	Moderate
SWB-011	Session Management	When a user changes the password from the web app, it says "This action will delete all currently saved tokens in your wallet and you will be logged out", however, the user is only logged out from the web app.	Moderate
SWB-014	Cryptography	The code in src/security/cookie.py and src/security/data.py does not implement a dynamically-generated random initialization vector.	Moderate
SWB-001	Cryptography	The SMSWithoutBorders Android app uses a static IV instead of a random, dynamically-generated one.	Low
SWB-005	Missing best practice	The Android app com.afkanerd.sw0b does not have any logout functionality and delete account functionality, however these are available through the web page.	Low
SWB-007	Transport layer security	The base network config of the application allows clear text traffic.	Low
SWB-009	Misconfiguration	The application implements a cross-origin resource sharing (CORS) policy that allows access from any domain.	Low

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

ID	Type	Recommendation
SWB-012	Cross site scripting	• Restrict the use of tags only to the ones that are shown in the UI. • Sanitize and validate all user input. • Perform all validation on the server side. • Optionally also perform validation on the client. • Filter input against cross-site scripting, preferably using a well-tested library.
SWB-006	Improper security control	• Require reauthentication before allowing modification of any security configuration/setting in the app.
SWB-002	Transport layer security	• Unless support for legacy browsers/devices is needed, disable TLS 1.0 and TLS 1.1 protocols. • If you must still support TLS 1.0, disable TLS 1.0 compression to avoid CRIME attacks.
SWB-003	Information leakage	• Do not write sensitive information such as username, password, URLs, tokens, etc to the Android log.
SWB-004	Missing security headers	• Implement the suggested headers, i.e. Strict-Transport-Security, Referrer-Policy, Permissions-Policy, and Content-Security-Policy, with appropriate values.
SWB-010	Password policy	• Enforce the same strong password policy in both front and back ends, and apply it to both new and existing users.
SWB-011	Session Management	• Notify users correctly about the action. • If possible, synchronise the mobile app session with that of the web app so that the data can be removed from both places.

SWB-014	Cryptography	• Use random, dynamically-generated IVs for CBC-mode encryption and decryption.
SWB-001	Cryptography	• Use random, dynamically-generated IVs for CBC-mode encryption and decryption.
SWB-005	Missing best practice	• Implement logout & delete account functions in the application, accessible via its UI.
SWB-007	Transport layer security	• Unless it is very explicitly needed by the app to work, do not allow the app to use clear text network connections. • If it is required, only allow connections to allow-listed, trusted domains.
SWB-009	Misconfiguration	• Rather than using a wildcard or programmatically verifying supplied origins, use an allow list of trusted domains.

2 Methodology

2.1 Planning

Our general approach during penetration tests is as follows:

1. **Reconnaissance**
   We attempt to gather as much information as possible about the target. Reconnaissance can take two forms: active and passive. A passive attack is always the best starting point as this would normally defeat intrusion detection systems and other forms of protection afforded to the app or network. This usually involves trying to discover publicly available information by visiting websites, newsgroups, etc. An active form would be more intrusive, could possibly show up in audit logs and might take the form of a social engineering type of attack.
2. **Enumeration**
   We use various fingerprinting tools to determine what hosts are visible on the target network and, more importantly, try to ascertain what services and operating systems they are running. Visible services are researched further to tailor subsequent tests to match.
3. **Scanning**
   Vulnerability scanners are used to scan all discovered hosts for known vulnerabilities or weaknesses. The results are analyzed to determine if there are any vulnerabilities that could be exploited to gain access or enhance privileges to target hosts.
4. **Obtaining Access**
   We use the results of the scans to assist in attempting to obtain access to target systems and services, or to escalate privileges where access has been obtained (either legitimately though provided credentials, or via vulnerabilities). This may be done surreptitiously (for example to try to evade intrusion detection systems or rate limits) or by more aggressive brute-force methods. This step also consist of manually testing the application against the latest (2017) list of OWASP Top 10 risks. The discovered vulnerabilities from scanning and manual testing are moreover used to further elevate access on the application.

2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.

- **High**

  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**

  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**

  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.
- **Low**

  Low risk of security controls being compromised with measurable negative impacts as a result.

3 Reconnaissance and Fingerprinting

We were able to gain information about the software and infrastructure through the following automated scans. Any relevant scan output will be referred to in the findings.

- Burp Suite Professional – https://portswigger.net/burp/pro
- nmap – http://nmap.org
- SSLscan – https://github.com/rbsec/sslscan
- Frida – https://github.com/frida/frida
- Objection – https://github.com/sensepost/objection
- MobSF – https://github.com/MobSF/Mobile-Security-Framework-MobSF

4 Findings

We have identified the following issues:

4.1 SWB-012 — Backend request vulnerable to reflected cross site scripting

Vulnerability ID: SWB-012	**Status:** Resolved
Vulnerability type: Cross site scripting	
Threat level: High	

Description:

One of the endpoints used within the app is vulnerable to cross site scripting.

Technical description:

The application sends users to `https://staging.smswithoutborders.com` to log in or sign up. When a user clicks on the `sync` option to sync saved tokens, a request is sent to:

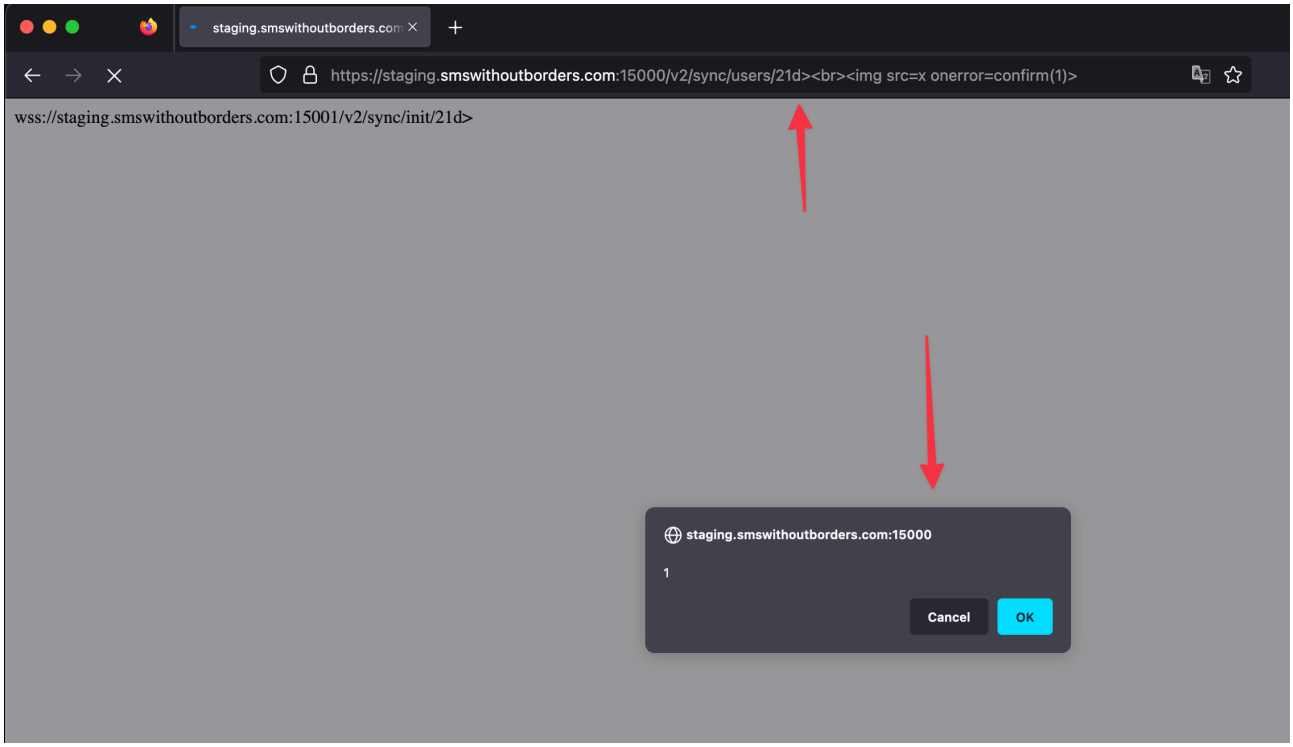
```
https://staging.smswithoutborders.com:15000/v2/sync/users/[id]
```

The `id` value in this request is reflected back with a web socket URL. However, it is possible for an attacker to inject a cross-site scripting payload in the `id` value.

Exploit example

```
/v2/sync/users/21d%3E%3Cbr%3E%3Cimg%20src=x%20onerror=confirm(1)%3E
```

Proof of concept



Update :

In the retest performed on 17th April 2023, this finding was resolved.

Impact:

Attacks against the user's browser can be launched by just using the application. A successful attack could lead to session hijacking, credential theft, or the client's system getting infected with malware.

Recommendation:

- Restrict the use of tags only to the ones that are shown in the UI.
- All user input as well as output to users must be strictly filtered. Within these checks it is necessary to implement filter mechanisms that operate on an allow-list basis instead of a block-list. Validation of parameters or input fields that can only consist of numerical values should only be accepted by the server if they are in fact numeric.
- All validation checks must be performed on the server, but may also be implemented on the client.

- To avoid cross-site scripting it is necessary to substitute special characters like [;()”´`,<>/] with their HTML entity equivalents. It is not sufficient to only filter special HTML tags like "script" because there are countless ways to successfully exploit cross-site scripting vulnerabilities; it's a good idea to use a well-tested library for this kind of filtering.

More information can be found at: https://www.owasp.org/index.php/Cross_Site_Scripting

4.2 SWB-006 — Disabling security options does not require re-authentication

Vulnerability ID: SWB-006	**Status:** Resolved
Vulnerability type: Improper security control	
Threat level: Elevated	

Description:

The application does not enforce re-authentication when a user disables the security settings.

Technical description:

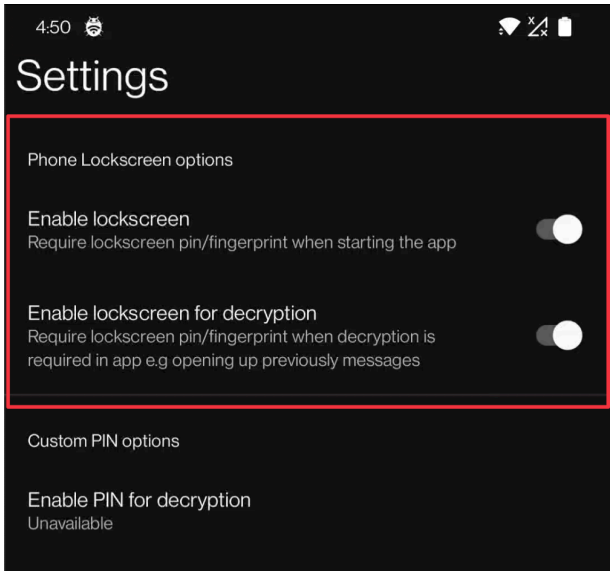
The app allows users to enable and disable the security options:

- Enable lockscreen
- Enable lockscreen for decryption

Both of these options, when enabled, require the user to authenticate using biometric or PIN every time the app is opened, or when decryption is requested. This is a major security feature in the application.

However, when a user tries to deactivate these options to remove these security controls, the application does not require the user to authenticate again.

Security options



Update :

In the retest performed on 17th April 2023, this finding was resolved.

Impact:

It is possible that a user with these security controls enabled, loses the device for some time and the attacker disables the controls. This would allow the attacker to decrypt the app and use it without needing to authenticate.

Recommendation:

- Require reauthentication before allowing modification of any security configuration/setting in the app.

4.3 SWB-002 — Deprecated TLS versions supported

Vulnerability ID: SWB-002	**Status:** Resolved
Vulnerability type: Transport layer security	
Threat level: Moderate	

Description:

Obsolete TLS protocol versions are supported by the host at `staging.smswithoutborders.com`.

Technical description:

The Android app communicates with the domain `staging.smswithoutborders.com`. The TLS implementation on this domain accepts connections over TLS 1.0 and 1.1 versions. These weaker protocol versions are considered deprecated and obsolete, and are no longer supported by modern browsers.

```
Testing SSL server staging.smswithoutborders.com on port 443 using SNI name staging.smswithoutborders.com

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   enabled
TLSv1.1   enabled
TLSv1.2   enabled
TLSv1.3   enabled
```

The PCI DSS (Payment Card Industry Data Security Standard) specifies that TLS 1.0 may no longer be used as of June 30, 2018. It also strongly suggests that you disable TLS 1.1. These protocols may be affected by vulnerabilities such as FREAK, POODLE, BEAST, and CRIME.

Update :

In the retest performed on 13th April 2023, this finding was resolved.

```
Testing SSL server staging.smswithoutborders.com on port 443 using SNI name staging.smswithoutborders.com

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   disabled
TLSv1.2   enabled
TLSv1.3   enabled
```

Impact:

Use of TLS 1.0 and 1.1 make the communication susceptible to downgrade attacks, as they rely on SHA-1 hashes for guaranteeing integrity of exchanged messages, and this hash function is considered weak and hence obsolete. Handshake authentication also uses SHA-1, which makes it easier for an attacker to impersonate a server for machine-in-the-middle attacks.

Recommendation:

- Unless support for legacy browsers/devices is needed, disable TLS 1.0 and TLS 1.1 protocols.
- If you must still support TLS 1.0, disable TLS 1.0 compression to avoid CRIME attacks.

4.4 SWB-003 — HTTP request URLs are logged

Vulnerability ID: SWB-003	**Status:** Resolved
Vulnerability type: Information leakage	
Threat level: Moderate	

Description:

The Android app (`com.afkanerd.sw0b`) logs URLs in the logcat.

Technical description:

We noticed that the application logs URLs in the Android log.

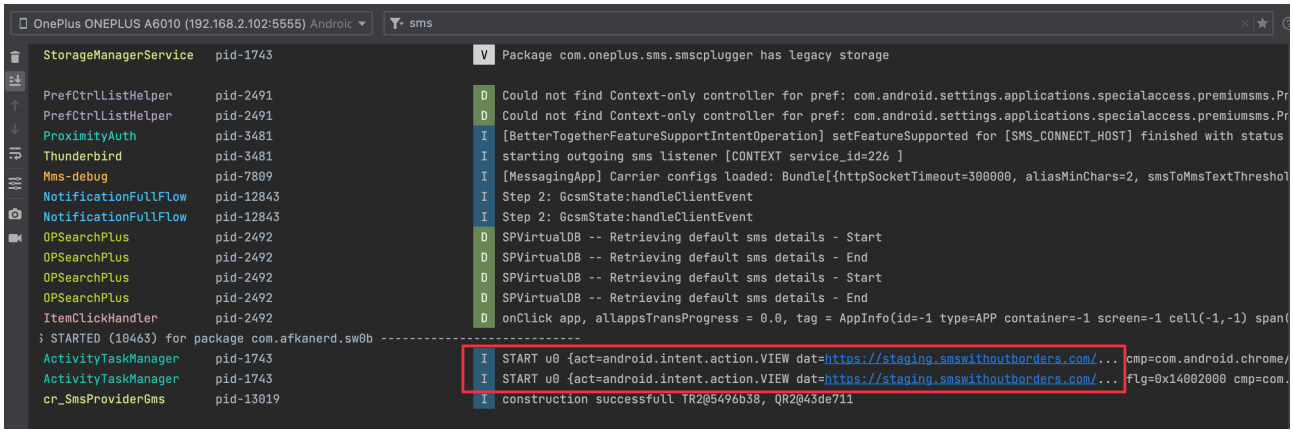
Logging URLs

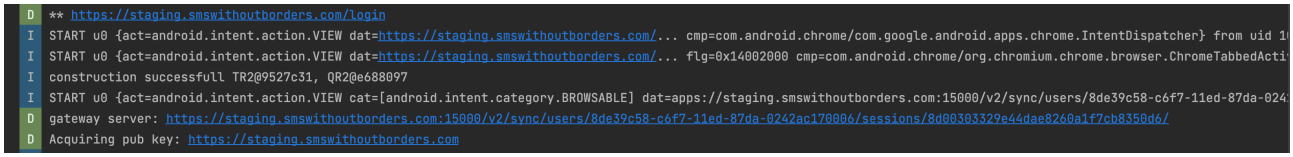
WelcomeActivity

```
public void onContinueClick(View view) {
        String smswithoutbordersHandshakeUrl =
 getString(R.string.smswithoutborders_official_site_login);
        Log.d(getLocalClassName(), "** " + smswithoutbordersHandshakeUrl);
        Uri intentUri = Uri.parse(smswithoutbordersHandshakeUrl);
        Intent intent = new Intent(Intent.ACTION_VIEW, intentUri);
        startActivity(intent);
    }
```

In the code above, the value of `smswithoutbordersHandshakeUrl` is written to the log.

Log cat





Update :

In the retest performed on 14th April 2023, we found that logs have been disabled in the app. However, some URLs are still logged (through intent from browser app). confirmed that this is required for the app's sync function to work, so we consider the finding resolved.

Impact:

Logging sensitive information in the Android log is not a recommended practice as this information can potentially be accessed by other applications on the same device.

Recommendation:

- Do not write sensitive information such as username, password, URLs, tokens, etc to the Android log.

4.5 SWB-004 — Missing security headers

Vulnerability ID: SWB-004	**Status:** Resolved
Vulnerability type: Missing security headers	
Threat level: Moderate	

Description:

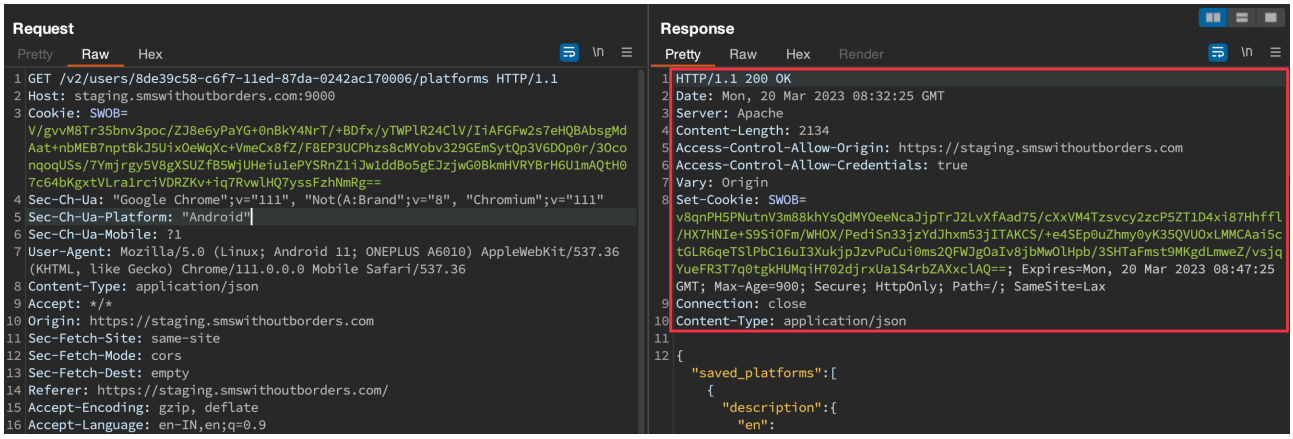
The back-end service that the Android app connects to is missing important security HTTP headers.

Technical description:

The Android app connects to `https://staging.smswithoutborders.com`. The web app at this domain does not implement some important security headers in its responses. These headers can help to prevent several attack types against users.

Header	Detail
Strict-Transport-Security	HTTP Strict Transport Security is an excellent feature to support on your site and strengthens your implementation of TLS by getting the User Agent to enforce the use of HTTPS. Recommended value: `Strict-Transport-Security: max-age=31536000; includeSubDomains`.
Referrer-Policy	Referrer Policy is a header that allows a site to control how much information the browser includes when navigating away from a document, and should be set by all sites.
Permissions-Policy	Permissions Policy allows a site to control which browser features and APIs can be used in the browser, such as location, video input, and physical movement.
Content-Security-Policy	The Content Security Policy header provides an effective set of tools to protect your site against XSS and supply-chain attacks. By allow-listing permitted content sources, browsers can be prevented from loading malicious assets from other places.

Response Headers



Take a look at the security headers project at https://securityheaders.com for further advice on security-related HTTP headers.

Update :

In the retest performed on 17th April 2023, this finding was resolved; Recommended headers have been added.

Impact:

Security headers improve the overall security of the application/endpoint. Not implementing security headers might allow attackers to target the user with different types of attacks. For example, not using HSTS could allow an attacker to conduct a machine-in-the-middle attack, in some conditions, and the attacker will be able to read and manipulate the HTTP traffic. An example of this is a user connecting via a malicious Wifi access point. HSTS is considered an essential header for secure websites. Not setting the header can result in an auditor seeing the site as not fulfilling GDPR article 5(1)(f). Similarly, the permission policy also improves defence-in-depth for the application.

Recommendation:

- Implement the suggested headers, i.e. Strict-Transport-Security, Referrer-Policy, Permissions-Policy, and Content-Security-Policy, with appropriate values.

4.6 SWB-010 — Password policy not enforced on back end

Vulnerability ID: SWB-010	**Status:** Resolved
Vulnerability type: Password policy	
Threat level: Moderate	

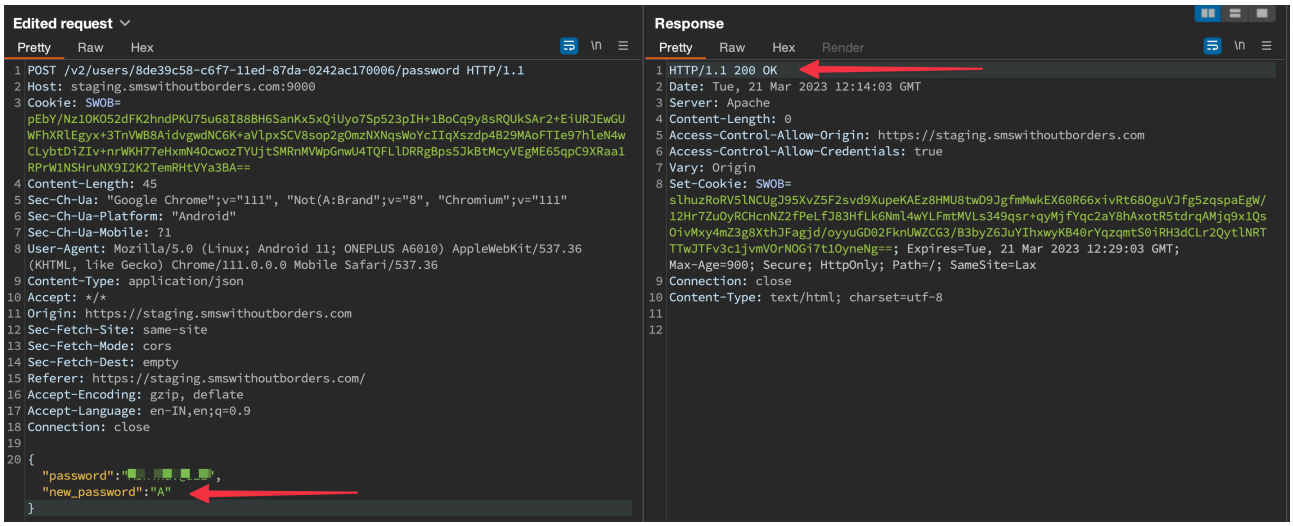
Description:

It is possible to set a weak password for the account, as the password policy checks are only implemented on the client side.

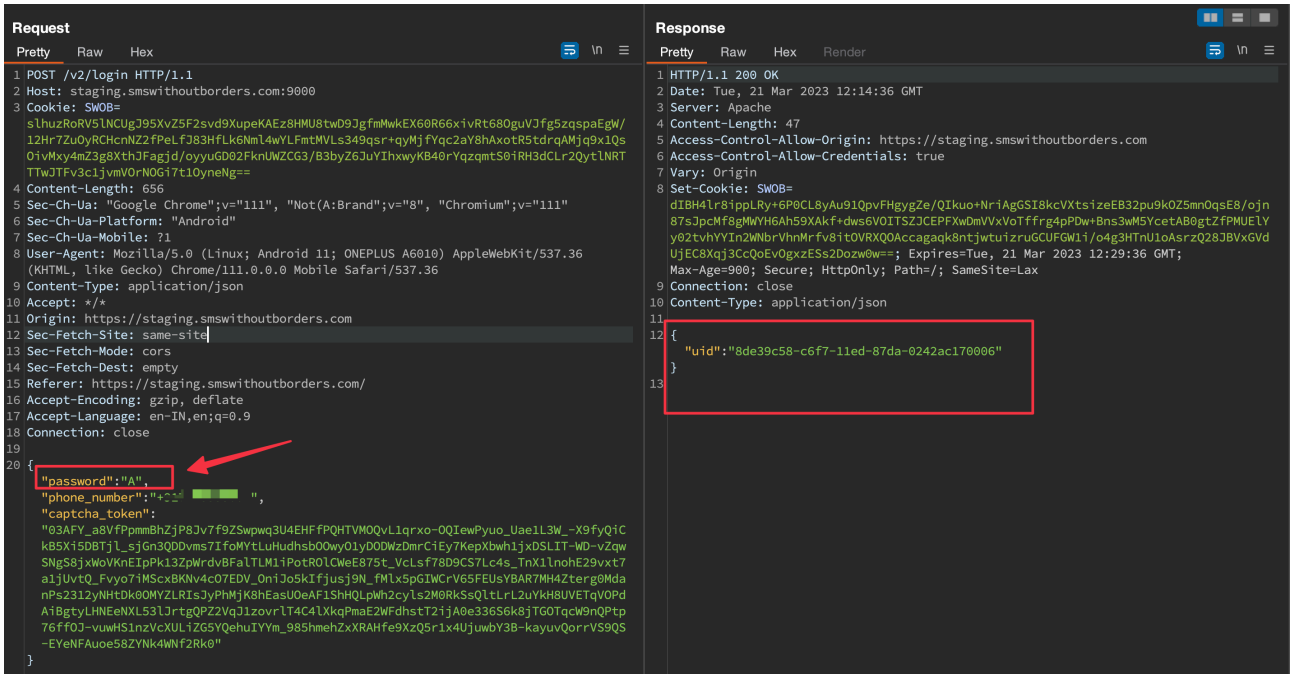
Technical description:

During the penetration test, we noticed that it is possible to trick the password change request and set a password as weak as a single character. This is because the password policy is not enforced and validated in the back end.

Setting weak password

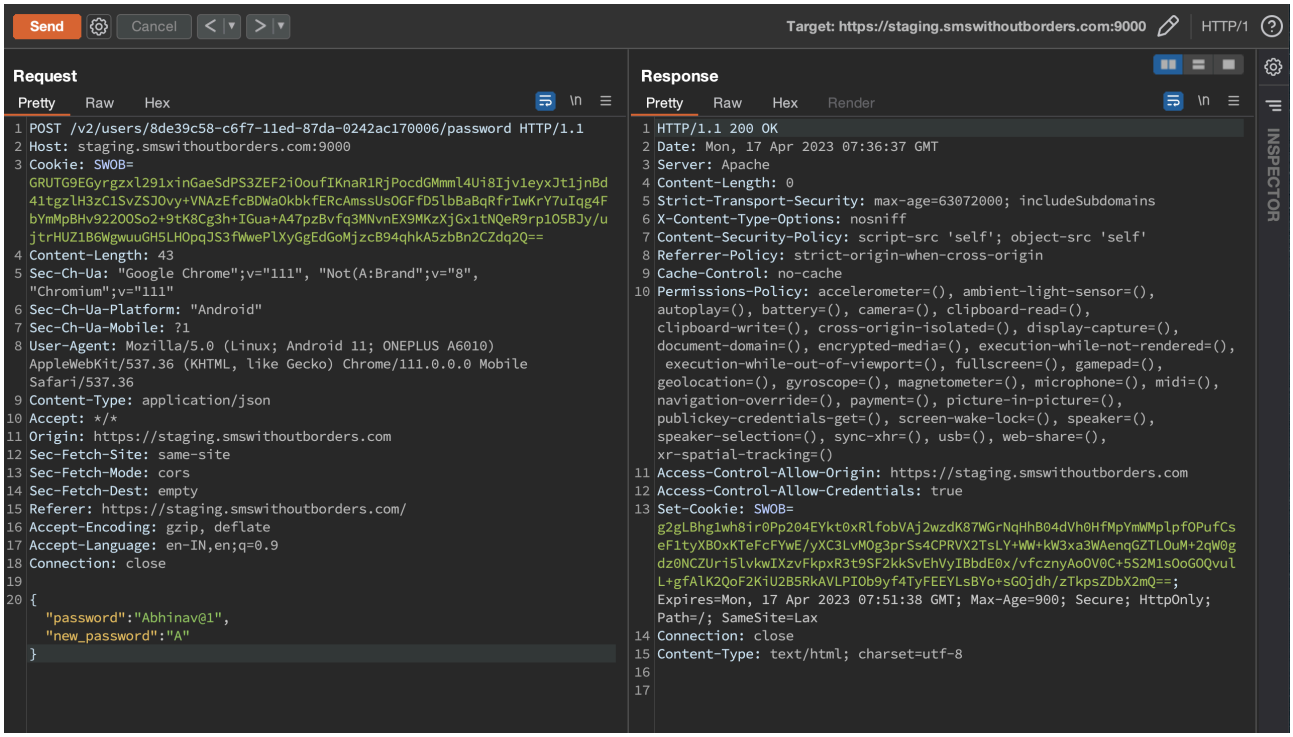


Login with single character password



Update :

In the retest performed on 17th April 2023, this finding still holds; we were still able to change the password to a single character:



Update :

In the retest performed on 19th April 2023, this finding was resolved.

Impact:

A weak password policy may allow users to choose easily guessable passwords. Attackers would then be able to perform practical brute-force or password guessing attack on user accounts. A successful attack would lead to full account takeover.

Recommendation:

- Enforce the same strong password policy in both front and back ends, and apply it to both new and existing users.

4.7 SWB-011 — Changing the password does not log the user out of the mobile app

Vulnerability ID: SWB-011	**Status:** Resolved
Vulnerability type: Session Management	
Threat level: Moderate	

Description:

When a user changes the password from the web app, it says "This action will delete all currently saved tokens in your wallet and you will be logged out", however, the user is only logged out from the web app.

Technical description:

During the penetration test, we noticed that changing the password of the account only logs the user out of their web app session, and not from the mobile app. If a user has previously logged in to a mobile app, then the data saved in the mobile app will still be available after the password change. However, this data will not be available in the web app even when they log in with new password.

This might be a confusing behaviour for users as they would think that the tokens saved in mobile app would also be deleted.

Update :

In the retest performed on 17th April 2023, this finding was resolved.

Impact:

This behaviour might confuse the users in thinking that they have successfully removed the token from their account, however someone with access to the device where they logged in before would still be able to get the tokens.

Recommendation:

- Notify users correctly about the action.
- If possible, synchronise the mobile app session with that of the web app so that the data can be removed from both places.

4.8 SWB-014 — CBC encryption used with a static IV

Vulnerability ID: SWB-014	**Status:** Resolved
Vulnerability type: Cryptography	
Threat level: Moderate	

Description:

The code in `src/security/cookie.py` and `src/security/data.py` does not implement a dynamically-generated random initialization vector.

Technical description:

When encrypting data with a cipher in Cipher Block Chaining (CBC) mode, an Initialization Vector (IV) is used to randomize the encryption. This is done so that the same plaintext doesn't always produce the same ciphertext for a given key. The IV doesn't need to be secret, but should be unpredictable in order to avoid "Chosen-Plaintext Attacks".

Affected code in `src/security/data.py`:

```
logger.debug("starting data encryption ...")

        if not data:
            result = {'e_data':None}

            logger.info("- Nothing to encrypt")
            return result
        else:
            data_bytes = data.encode("utf-8")
            iv_bytes = None if not iv else iv.encode("utf-8")
            cipher = AES.new(self.key, AES.MODE_CBC, self.iv if not iv_bytes else iv_bytes)
            ct_bytes = cipher.encrypt(pad(data_bytes, 16))
            ct_iv = cipher.iv.decode("utf-8")
            ct = ct_bytes.hex()

            result = {'iv':ct_iv, 'e_data':ct}

            logger.info("- Successfully encryted data")
            return result
```

- src/security/cookie.py

```
logger.debug("starting cookie encryption ...")
        cipher = AES.new(self.key, AES.MODE_CBC, self.iv if not iv else iv)
        data_bytes = data.encode()
        ct_bytes = cipher.encrypt(pad(data_bytes, AES.block_size))
        ct = b64encode(self.iv + ct_bytes).decode('utf-8')

        logger.info("- Successfully encryted cookie")
```

```
        return ct
```

We found the same issue in SWB-001 (page 27), but in Java.

Update :

In the retest performed on 17th April 2023, this finding was resolved.

Impact:

Static or predictable IVs make it much easier to mount chosen-ciphertext attacks on data encrypted with CBC-mode ciphers.

Recommendation:

- Use random, dynamically-generated IVs for CBC-mode encryption and decryption.

4.9 SWB-001 — CBC encryption used with a static IV

Vulnerability ID: SWB-001 **Status:** Resolved

Vulnerability type: Cryptography

Threat level: Low

Description:

The SMSWithoutBorders Android app uses a static IV instead of a random, dynamically-generated one.

Technical description:

When encrypting data with a cipher in Cipher Block Chaining (CBC) mode, an Initialization Vector (IV) is used to randomize the encryption. This is done so that the same plaintext doesn't always produce the same ciphertext for a given key. The IV doesn't need to be secret, but should be unpredictable in order to avoid "Chosen-Plaintext Attacks".

Affected code in `app/.../main/java/com/example/sw0b_001/Security/SecurityAES.java`:

```
public byte[] encrypt(byte[] iv, byte[] input, byte[] sharedKey) throws Throwable {
    byte[] ciphertext = null;
    try {
        SecretKeySpec secretKeySpec = new SecretKeySpec(sharedKey, "AES");
        IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
```

```
        Cipher cipher = Cipher.getInstance(DEFAULT_AES_ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec, ivParameterSpec);
        ciphertext = cipher.doFinal(input);
    }
    catch (Exception e) {
        e.printStackTrace();
        throw new Throwable(e);
    }
    return ciphertext;
}

public byte[] decrypt(byte[] iv, byte[] input, byte[] sharedKey) throws Throwable {
    byte[] decryptedText = null;
    try {
        SecretKeySpec secretKeySpec = new SecretKeySpec(sharedKey, "AES");
        IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);

        Cipher cipher = Cipher.getInstance(DEFAULT_AES_ALGORITHM);
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivParameterSpec);
        decryptedText = cipher.doFinal(input);
    }
```

Both the encryption and decryption use a static IV. If the encryption uses a dynamically generated IV and only the decryption uses a statically generated IV then it is not a security issue.

CBC mode eliminates a weakness of Electronic Code Book (ECB) mode by allowing identical plaintext blocks to result in different encrypted ciphertext blocks. This is possible by the XOR-ing of an IV with the initial plaintext block so that every plaintext block in the chain is XOR'd with a different value before encryption. If IVs are reused, then identical plaintexts would result in identical encrypted ciphertexts. However, even if IVs are not identical but are generated in a predictable way, then they may still break the security of CBC mode against chosen-plaintext attacks.

We found the same issue in SWB-014 (page 26), but in Python.

Update :

In the retest performed on 13th April 2023, this finding was resolved.

Impact:

Static or predictable IVs make it much easier to mount chosen-ciphertext attacks on data encrypted with CBC-mode ciphers.

Recommendation:

- Use random, dynamically-generated IVs for CBC-mode encryption and decryption.

4.10 SWB-005 — No logout and delete feature available in app

Vulnerability ID: SWB-005 **Status:** Resolved

Vulnerability type: Missing best practice

Threat level: Low

Description:

The Android app `com.afkanerd.sw0b` does not have any logout functionality and delete account functionality, however these are available through the web page.

Technical description:

As there is no logout option available in the app, the only option left for users is to delete the app. We also noticed that the app does not provide an option to delete the account along with its associated data.

Update :

In the retest performed on 17th April 2023, this finding was resolved; Logout and delete functions have been added.

Impact:

Not having a logout function might make the user's sessions susceptible to attacks.

Recommendation:

- Implement logout & delete account functions in the application, accessible via its UI.

4.11 SWB-007 — Clear text traffic is enabled in the application

Vulnerability ID: SWB-007 **Status:** Resolved

Vulnerability type: Transport layer security

Threat level: Low

Description:

The base network config of the application allows clear text traffic.

Technical description:

The network security configuration allows apps to customize their network security settings. These settings can be configured for specific domains and for a specific app, for example to customize which Certificate Authorities (CAs) are trusted for an app's secure connections, to protect apps from accidental usage of cleartext traffic etc.

The Android application (`com.afkanerd.sw0b`) includes the following line in its `AndroidManifest.xml` configuration file:

Config:

```
android:usesCleartextTraffic="true" >
```

This signals that the app intends to use cleartext network traffic, such as unencrypted HTTP. The default value for apps that target API level 27 or lower is "true", but apps that target API level 28 or higher default to "false". Note that the Google Play store is increasing its minimum API level to 33 in August 2023.

Update :

In the retest performed on 17th April 2023, this finding was resolved.

Impact:

Allowing cleartext traffic would impact the confidentiality, authenticity, and protections against tampering; a network attacker can eavesdrop on transmitted data and also modify it without being detected.

Recommendation:

- Unless it is very explicitly needed by the app to work, do not allow the app to use clear text network connections.
- If it is required, only allow connections to allow-listed, trusted domains.

4.12 SWB-009 — Cross-origin resource sharing is permitted from arbitrary origins

Vulnerability ID: SWB-009 **Status:** Resolved

Vulnerability type: Misconfiguration

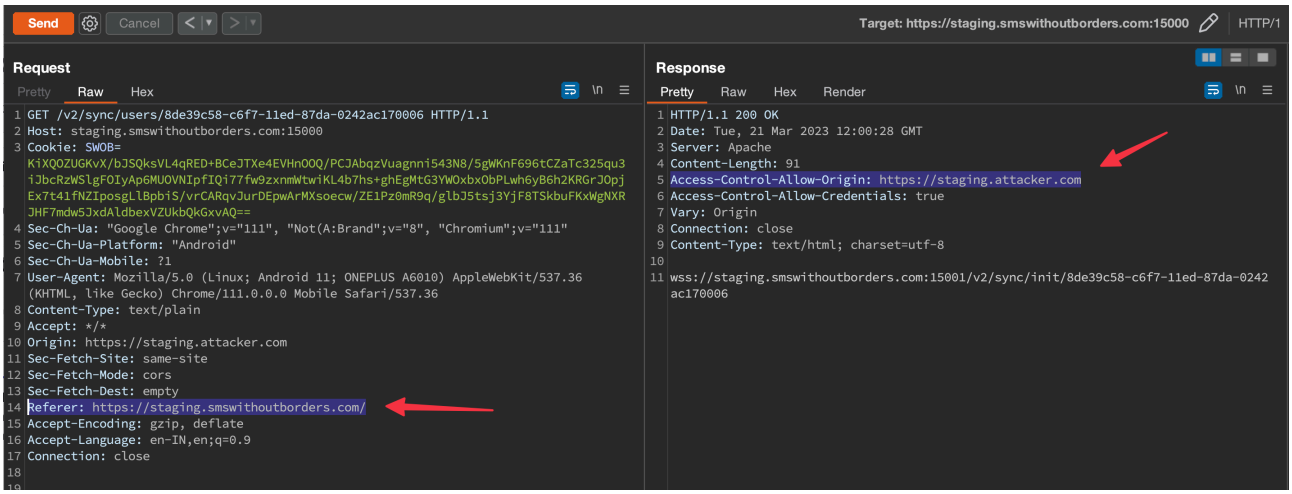
Threat level: Low

Description:

The application implements a cross-origin resource sharing (CORS) policy that allows access from any domain.

Technical description:

An HTML5 cross-origin resource sharing (CORS) policy controls whether and how content running on other domains can perform two-way interaction with the domain that publishes the policy. The policy is fine-grained and can apply access controls per-request based on the URL and other features of the request.



Update :

In the retest performed on 17th April 2023, this finding was resolved.

Impact:

Trusting arbitrary origins effectively disables the same-origin policy, allowing two-way interaction with third-party sites.

Recommendation:

- Rather than using a wildcard or programmatically verifying supplied origins, use an allow list of trusted domains.

5 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

5.1 NF-008 — Testing `SyncInitiateActivity` and schemes

During the penetration test, we performed several attempts to find security issues in `SyncInitiateActivity` and associated schemes. The application uses the following URL schemes to perform a sync from browser to the app:

- apps://
- app://
- intent://

The app opens the `SyncInitiateActivity` when a link with any of these schemes, using hostnames `developers.smswithoutborders.com`, `staging.smswithoutborders.com`, or `smswithoutborders.com`, and the path prefixes: `/v2/sync/users/`, or `/sign-up/` is opened.

We tested this sync flow for several different types of vulnerabilities, but we found the app to be secure against them.

5.2 NF-013 — Testing intent handling and local storage

During the penetration test, we tested the app for any security issues related to intent handling, however we did not discover any such issues. We also looked at how the application stores data on the device, and found it to be stored securely.

6 Future Work

- **Retest of findings**
  When mitigations for the vulnerabilities described in this report have been deployed, a repeat test should be performed to ensure that they are effective and have not introduced other security problems.
- **Regular security assessments**
  Security is an ongoing process and not a product, so we advise undertaking regular security assessments and penetration tests, ideally prior to every major release or every quarter.

7 Conclusion

We discovered 1 High, 1 Elevated, 6 Moderate and 4 Low-severity issues during this penetration test.

We found only one high-severity issue during this pentest, which allows exploitable XSS. The other issues are almost all related to a lack of best-practices surrounding service configuration such as missing security HTTP headers, or support for obsolete TLS protocols, and should be easy to fix. Inconsistent application of validation, sanitization and password policies make the app and back-end a little less robust than they could be. We also found the use of static CBC encryption initialisation vectors in both Java and python code, weakening security promises the service makes.

We recommend fixing all of the issues found and then performing a retest in order to ensure that mitigations are effective and that no new vulnerabilities have been introduced.

Finally, we want to emphasize that security is a process – this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Update:

In follow-up retests on April 13th – 17th, all findings have been resolved.

Appendix 1 Testing team

Abhinav Mishra	Abhinav has 10+ years of experience in the penetration testing of web, mobile and infrastructure. He has received numerous accolades from multiple organisations for responsible disclosure of vulnerabilities. He is also known for providing trainings on web, mobile and infrastructure security.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.