



SAMPLE REPORT

REDPOINT SECURITY

MOBILE APPLICATION SECURITY ASSESSMENT

Date xx/xx/2024

rdpt.io

EXECUTIVE SUMMARY

OVERVIEW

Client engaged Redpoint Security (“Redpoint”) to conduct a mobile application security assessment of Client’s mobile applications (iOS and Android) by analyzing potential security flaws within the application in its running environment as well as its related backend web services.

The purpose of the assessment is to provide assurance that the applications and their infrastructure has a strong security posture and to identify technical vulnerabilities which may expose the business and/or its customers to risk. To assess the security posture of the application, a security assessment was conducted from anonymous and authenticated perspectives, using multiple application roles (authenticated user and administrative user). Redpoint was provided with credentials and access to the application to evaluate various security controls.

SCOPE

Redpoint assessed the application for software vulnerabilities that expose the business and/or its customers to risk. This assessment focused primarily on application vulnerabilities and did not include an assessment of network vulnerabilities. Redpoint tested the applications within the environments specified:

- iOS Application: Version x.x.x
- Android Application: Version x.x.x-test

PRESENTATION OF RESULTS

This document includes a summary of findings and recommendations, a description of the methodology used, and a detailed findings matrix which includes:

- Reported finding;
- The severity and difficulty of exploit and the likelihood of exploit occurring for each finding;
- Description of the finding;
- The impact to security posture resulting from each finding;
- Mitigation recommendations and any other compensating controls;

FINDINGS SUMMARY

ANALYSIS

Redpoint identified a total of [#] findings during the assessment, ranging from high to low and informational severity. These findings relate to the confidentiality, integrity, and availability of the application, the environment, and the data contained within it.

The issues with critical and high severity risks which Redpoint found during the assessment included a Broken Access Control, Sensitive Data Exposure and Insecure Direct Object Reference (IDOR) finding. Some medium risk vulnerabilities were also found, including some sensitive data exposure as well as broken authentication which permitted the ability to enumerate valid usernames through the application responses to locked accounts.

The low severity issues were the use of outdated components that have known vulnerabilities and background snapshot data disclosure. It is worth noting that, while outdated libraries may not be used in an insecure manner, inclusion of this software increases the attack surface, invites attacks, and indicates an undisciplined approach to patching software.

RATING

Overall, the application rates as insecure in an objective comparison to other applications assessed by Redpoint. Mitigation of the identified Broken Access Control, sensitive data exposure, IDOR, and username enumeration flaws and upgrades to known vulnerable components would strengthen the overall posture of the application. Additional implementation of strict-transport-security headers would add additional layers of security to the application.

FINDINGS CHART

ID	Severity	Category	Summary
1	CRITICAL	Broken Access Control	Account Takeover. The application does not properly restrict user requests based on permission level.
2	HIGH	Sensitive Data Exposure	PIN Enumeration. Valid PIN numbers for gift cards were brute-forced through the backend API
3	HIGH	Broken Access Control	Insecure Direct Object Reference. The application is exposing a reference to an internal object to users.
4	MEDIUM	Sensitive Data Exposure	Sensitive Data Disclosure. The iOS application stores sensitive data, including email address and corresponding userID on the device after normal usage.
5	MEDIUM	Broken Authentication	Account Enumeration. The application is responding to authentication requests in a manner that allows for the determination of valid accounts on the system.
6	LOW	Security Misconfiguration	Using Components with Known Vulnerabilities. Application used outdated and unpatched components that contain published CVEs.
7	LOW	Sensitive Data Exposure	Background Snapshot Data Disclosure. The application allowed sensitive data to be kept in background snapshots.
8	INFORMATIONAL	Encryption	TLS Stripping. The API permitted the client browser to initiate requests for TLS-protected resources over HTTP, allowing them to be intercepted and forwarded as HTTPS requests and enabling man-in-the-middle attacks against resources requiring TLS encryption.

FINDINGS

BROKEN ACCESS CONTROL – ACCOUNT TAKEOVER

CRITICAL

DESCRIPTION

The application does not properly restrict user requests based on permission level. This allows for users to access application functionality and data for which they should not have permission.

Specifically, use of enumerable member IDs within the Forget Password flow allows for anonymous users to reset any valid account password.

EVIDENCE

```
POST /api/user/updatepassword HTTP/1.1
Host: www.c2c-prod.examplecompany.com
Content-Type: application/json
....
User-Agent: Examplecompany-RAC/18 CFNetwork/889.9 Darwin/17.2.0
Connection: close

{"locale":"en-us","correlationId":"484d8a8c-16a0-4f6d-8597-
e522f6e4388b","user_password":"Guide#1234","brand":"Examplecompany","login":"95701872"
}
```

Reset Password HTTP Request

```
HTTP/1.1 201
Date: Wed, 06 Mar 2019 15:45:45 GMT
Content-Type: application/json; charset=UTF-8
Connection: close
....
X-Frame-Options: DENY
Set-Cookie: nlbi_1611668=Kw2OZGoiYFzT27lYQ9iOLQAAAAC+ciN/7jUbW9Dw0idctWfu; path=/;
Domain=.c2c-prod.examplecompany.com
X-Iinfo: 1-3129982-3129984 NNNN CT(0 0 0) RT(1551887143934 32) q(0 0 0 -1) r(5 5) U6
X-CDN: Incapsula
Content-Length: 129

{"correlationId":"484d8a8c-16a0-4f6d-8597-
e522f6e4388b","data":{"response":{"success":"Password has been updated
successfully"}}}
```

HTTP Response

Endpoint	Parameter
api/user/updatepassword	login

STEPS TO REPRODUCE

1. Configure the mobile application to use an intercepting proxy, such as Burp Suite.
2. Navigate through the Forgot-Password flow.
3. Review the request for the final update password request.
4. Replay this request using another valid member ID.
5. Log in to the other user's account using this member ID/password combination.

IMPACT

Exploit Difficulty	Likelihood
EASY	HIGH

An attacker can gain access to sensitive information by bypassing authorization mechanisms.

RECOMMENDATION

The application should enforce a strong authorization scheme that is validated on the server side. The scheme should also ensure that the user making the request for a resource has the appropriate permission level.

Since attackers can manipulate client-side values, the authorization scheme should not rely on client-side data when making authorization decisions. Values from the client should be properly validated on the server side to ensure they have not been tampered with.

If possible, use the authorization scheme that is included with the platform framework in use. Framework authorization mechanisms typically have had more scrutiny tied to them and can prove more robust than ones that are built from scratch.

REFERENCES

- OWASP - Broken Access Controls - (https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html)

PIN ENUMERATION

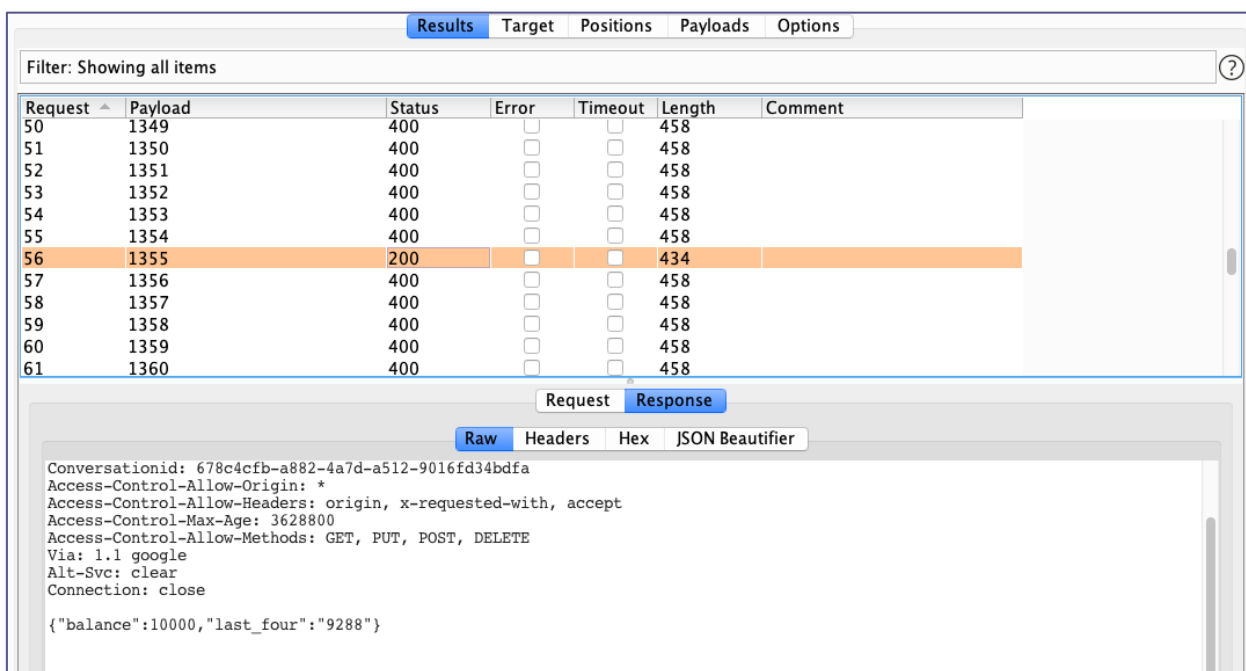
HIGH

DESCRIPTION

The application allowed the enumeration of valid PIN numbers for gift cards through brute-forcing of endpoints associated with checking a gift card balances and associating these cards with specific accounts. The following backend API URL was identified as vulnerable to this enumeration during the review:

- `url.url/api/reward_card/balanceinquiry?key={}`

The following screenshots show enumeration of the provided gift card number PIN using automated tooling against the above endpoint.



Request	Payload	Status	Error	Timeout	Length	Comment
50	1349	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
51	1350	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
52	1351	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
53	1352	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
54	1353	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
55	1354	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
56	1355	200	<input type="checkbox"/>	<input type="checkbox"/>	434	
57	1356	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
58	1357	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
59	1358	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
60	1359	400	<input type="checkbox"/>	<input type="checkbox"/>	458	
61	1360	400	<input type="checkbox"/>	<input type="checkbox"/>	458	

```
Conversationid: 678c4cfb-a882-4a7d-a512-9016fd34bdfa
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: origin, x-requested-with, accept
Access-Control-Max-Age: 3628800
Access-Control-Allow-Methods: GET, PUT, POST, DELETE
Via: 1.1 google
Alt-Svc: clear
Connection: close

{"balance":10000,"last_four":"9288"}
```

Balance endpoint returned HTTP Status 200 on valid PIN number after multiple failed attempts.

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
49	1348	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
50	1349	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
51	1350	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
52	1351	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
53	1352	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
54	1353	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
55	1354	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
56	1355	400	<input type="checkbox"/>	<input type="checkbox"/>	501	
57	1356	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
58	1357	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
59	1358	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
60	1359	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
61	1360	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
62	1361	400	<input type="checkbox"/>	<input type="checkbox"/>	471	
63	1362	400	<input type="checkbox"/>	<input type="checkbox"/>	471	

Request Response

Raw Headers Hex JSON Beautifier

```

Access-Control-Max-Age: 3628800
Access-Control-Allow-Methods: GET, PUT, POST, DELETE
Via: 1.1 google
Alt-Svc: clear
Connection: close

{"error_code":"0348","message":"The card you are trying to save already exists in your list."}

```

The tokenized API endpoint displayed a different message if the card is already added to the current account

Response

Raw Headers Hex JSON Beautifier

```

HTTP/1.1 201 Created
Date: Fri, 30 Nov 2018 21:13:36 GMT
Content-Type: application/json
Content-Length: 82
Conversationid: 3d8dfd2f-947f-4dfb-ab14-222b1dc1d0fb
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: origin, x-requested-with, accept
Access-Control-Max-Age: 3628800
Access-Control-Allow-Methods: GET, PUT, POST, DELETE
Via: 1.1 google
Alt-Svc: clear
Connection: close

{"type":"giftcard","metadata":{"balance":10000,"number":"9288"},"payment_id":"2"}

```

Successfully adding a gift card to an account after enumerating the PIN number

Endpoint	Parameter
/api/reward_card/balanceinquiry?key={}	number, last_four

IMPACT

Exploit Difficulty	Likelihood
MODERATE	HIGH

This vulnerability would allow an attacker to associate gift cards with their accounts as long as they could decipher valid gift card numbers. Depending on how these gift card numbers are displayed on the cards themselves, the number could be photographed pre-sale and then used by a malicious actor after a customer has purchased it.

RECOMMENDATION

The application should only allow a small number of attempts to validate gift card balances or associate gift cards with an account before timing out the card. If this is not possible then anti-automation defense should be applied like a Recaptcha with every request for gift card transaction.

REFERENCES

- [Recaptcha](#)

INSECURE DIRECT OBJECT REFERENCE

HIGH

DESCRIPTION

The application is exposing internal-object references to users. The object references could be a variety of things, such as files, directories, or numbers. These values can be modified in order to gain unauthorized access to resources.

Redpoint discovered multiple instances of direct object references during the review that could be manipulated to obtain sensitive information. This included the API endpoints for discount codes, account reservations, and the user account summary.

EVIDENCE

```
https://www.c2c-prod.examplecompany.com/api/DiscountService/v1/validateCDP?brand=examplecompany&cdpValue=1392782&correlationId=9d197934-5690-4935-be18-dc7eb3b58f9c&deviceToken=iOS&locale=en-us
```

Http request URL with IDOR, CDP value can be manipulated to obtain different discount codes

```
GET /api/accountSummary/v2/account-summary/95701872?brand=examplecompany&correlationId=7b604791-1c39-4573-a322-b46632a487f2&locale=en-us HTTP/1.1
Host: www.c2c-prod.examplecompany.com
Accept: */*
Connection: close
...
Accept-Encoding: gzip, deflate
```

Account Summary Endpoint – changing the path corresponding to a member's ID exposed order data.

Endpoint	Parameter
api/ DiscountService/v1/validateCDP?	cdpValue
/api/accountSummary/v2/account-summary/	member ID

STEPS TO REPRODUCE

1. Configure the mobile application to use an intercepting proxy, such as Burp Suite.
2. Authenticate to the application.
3. Review the request for account summary information.
4. Replay this request using a valid member ID that has a pending order.
5. View the order information.

IMPACT

Exploit Difficulty	Likelihood
EASY	HIGH

An attacker can gain unauthorized access to resources by manipulating the reference values.

RECOMMENDATION

Access to resources should not be based solely on user-controlled values read directly from the request. Proper authorization should be implemented for user requests to ensure that users can only access resources they have permission to access.

REFERENCES

- OWASP - Broken Access Controls - https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html)
- Insecure Direct Object Reference Prevention - OWASP Cheat Sheet Series - https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

SENSITIVE DATA EXPOSURE

MEDIUM

DESCRIPTION

The iOS application stores sensitive data, including email address and corresponding userID on the device after normal usage. Sensitive information is written to application preferences file stored under the "Preferences" folder. This is an unencrypted PLIST file containing encoded name/value pairs and is easily examined using standard development tools.

EVIDENCE

This snippet was pulled from the `com.examplecompany.plist` file in the iOS Example Company application.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>IBGIdentifiedEmail</key>
    <string>s@rdpt.io</string>
    ...
    <key>kIBGPoliceUserDefaultKeyCurrentEmail</key>
    <string>s@rdpt.io</string>
    ...
    <key>userID</key>
    <string>01745aef-bf9c-7e0b-7609-4e895115f895</string>
    ...
</dict>
```

STEPS TO REPRODUCE

1. Open the Example Company iOS application.
2. Log in to the application.
3. Copy the preferences file contents off the device or review it using file browsing functionality.
4. Review the file for sensitive details.

IMPACT

Difficulty	Likelihood
MODERATE	HIGH

An attacker who gained physical or remote access to the device would have the ability to retrieve the sensitive information from the application data files. The contents of User Preferences may also be included in a backup of the device data in an unencrypted format.

RECOMMENDATION

Store moderately sensitive data values (such as user identifier or authenticated session tokens) in the iOS Keychain. These protections can be defeated on a device that has been jail-broken, but the values cannot be directly copied off the device.

Encrypt highly sensitive data values (such as user password or credit card number) with a key that is not also stored on the device flash memory. Commonly, an encryption key is derived from the user's password when it is entered during the authentication process.

For any cryptographic operations, review the default values for any platform algorithm to validate appropriate values have been chosen for the scenario.

REFERENCES

- [OWASP M2: Insecure Data Storage](#)

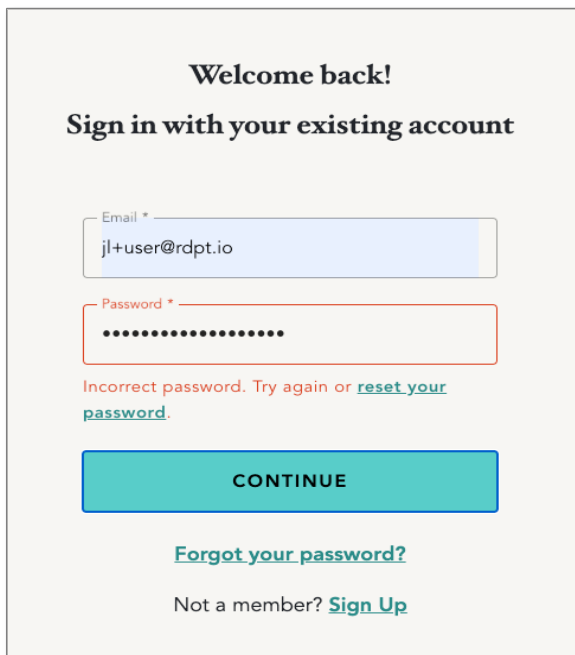
ACCOUNT ENUMERATION

MEDIUM

DESCRIPTION

The application responded to requests in a manner that allowed for the determination of valid accounts on the system. An attacker can use the discrepancies in responses between success and failure to determine valid accounts on the system. These responses can often be automated using iterating tools making attacks more effective.

EVIDENCE



Welcome back!

Sign in with your existing account

Email *

jl+user@rdpt.io

Password *

.....

Incorrect password. Try again or [reset your password](#).

CONTINUE

[Forgot your password?](#)

Not a member? [Sign Up](#)

Valid Account Returns Incorrect Password

Welcome back!

Sign in with your existing account

We couldn't find that email. If you're a [REDACTED] member, you'll need to [create a new \[REDACTED\] account](#) to access [REDACTED] clubs. This won't affect your [REDACTED] account.

Email *
jl@rdpt.io

Password *
•

CONTINUE

[Forgot your password?](#)

Not a member? [Sign Up](#)

Invalid Account Returns Different Response

STEPS TO REPRODUCE

1. Go to login page (<https://url.url/web/login>).
2. Sign in with jl+user@rdpt.io for the email and any string of characters for the password.
3. Submit.
4. View the error message stating "Incorrect password."
5. Go to login page (<https://url.url/web/login>).
6. Sign in with jl@rdpt.io for the email ID and any string of characters for the password.
7. Submit.
8. View the error message stating, "We couldn't find that email."

IMPACT

Difficulty	Likelihood
MODERATE	HIGH

An attacker can use application responses to conduct attacks against valid users to gain unauthorized access.

RECOMMENDATION

Modify application responses to display generic messages that do not leak information about valid user accounts on the system. The same generic responses should be implemented in all locations where user account interaction happens. This includes login pages, password reset, and new user registration functionality.

REFERENCES

- OWASP Broken Authentication - https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication.html

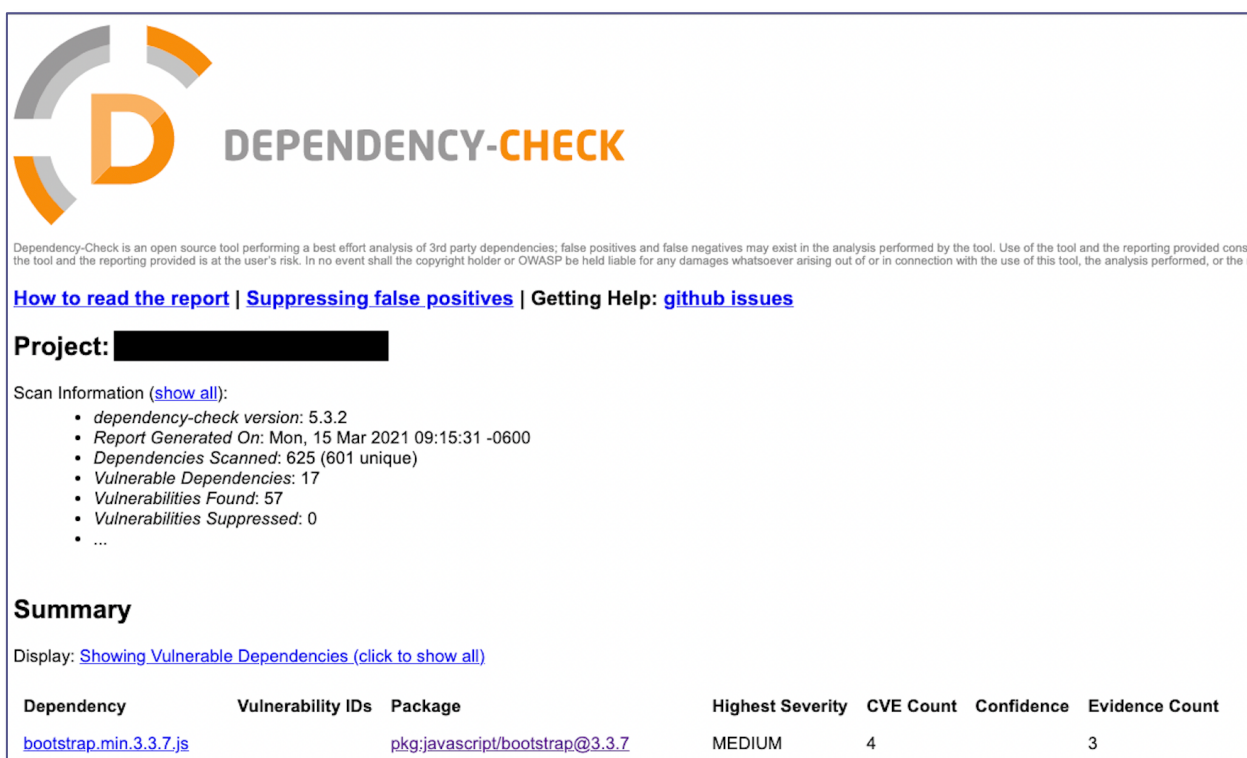
USING COMPONENTS WITH KNOWN VULNERABILITIES

LOW

DESCRIPTION

Outdated or weak components are in use by the application. These components may be part of a programming library or underlying platform. These weaknesses are commonly targeted by attackers because of the publicly available information on these vulnerabilities.

EVIDENCE



Dependency-Check is an open source tool performing a best effort analysis of 3rd party dependencies; false positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the reporting provided.

[How to read the report](#) | [Suppressing false positives](#) | [Getting Help: github issues](#)

Project: ██████████

Scan Information ([show all](#)):

- *dependency-check version:* 5.3.2
- *Report Generated On:* Mon, 15 Mar 2021 09:15:31 -0600
- *Dependencies Scanned:* 625 (601 unique)
- *Vulnerable Dependencies:* 17
- *Vulnerabilities Found:* 57
- *Vulnerabilities Suppressed:* 0
- ...

Summary

Display: [Showing Vulnerable Dependencies \(click to show all\)](#)

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
bootstrap.min.3.3.7.js		pkg:javascript/bootstrap@3.3.7	MEDIUM	4		3

Vulnerable Components

CVE.1.2021.PleasUpdate

CVE.2.2018.

IMPACT

Difficulty	Likelihood
MODERATE	HIGH

Weak and outdated components can be exploited by an attacker to gain unauthorized access to the application and its environment.

RECOMMENDATION

Update the component to a current, non-vulnerable version.

Implement security policies governing the use, tracking, and updating of platforms, libraries, and components. This will ensure that when future security updates are published, they can be worked into the application and environment.

REFERENCES

- OWASP Top 10 2017 A9:2017 – Using Components with Known Vulnerabilities - https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities

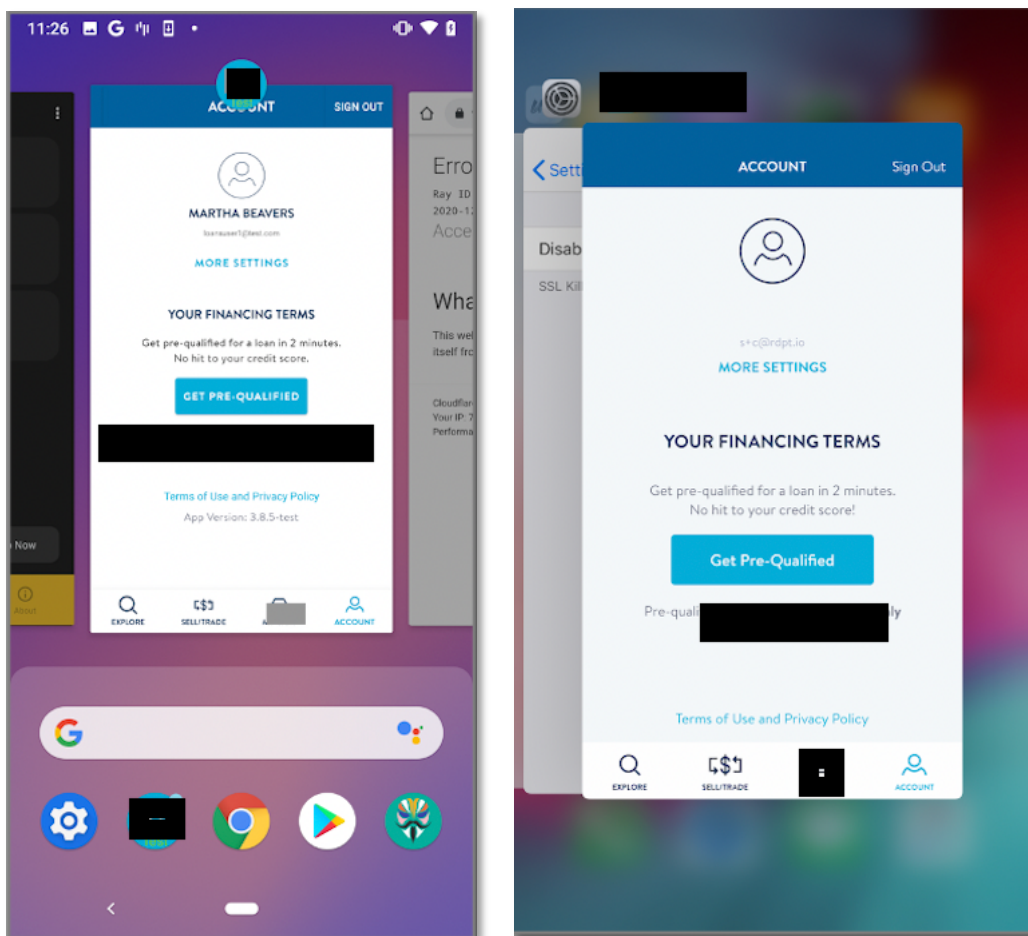
BACKGROUND SNAPSHOT DATA DISCLOSURE

LOW

DESCRIPTION

The application stores sensitive data on the device during normal usage. A screenshot of the application is automatically taken when the application is sent to the background (such as by selecting the “Home” button). Any sensitive information captured in the screenshot will be stored in an unencrypted format. The screenshot is written to image files located beneath the application’s “Snapshots” folder when the application is sent to the background.

The iOS version was vulnerable on the Login and Account pages.



Account pages backgrounded for both iOS/Android applications

IMPACT

Difficulty	Likelihood
------------	------------

MODERATE

MODERATE

An attacker with physical or remote access to the device would have the ability to retrieve any sensitive information exposed in the background images. On some versions of iOS, the file contents can be directly copied off the device.

In these instances, the data on these pages isn't overly sensitive but the data displayed on these pages could change in the future to include more sensitive information on users.

RECOMMENDATION

Clear sensitive data from the screen, blank the screen contents, or overwrite the screen with an image that does not contain sensitive data when the application is sent to the background or transitions from an active to inactive state.

For iOS, utilize the `applicationDidEnterBackground` function to generate a generic background image.

```
@property (UIImageView *)backgroundImage;
- (void)applicationDidEnterBackground:(UIApplication *)application {
    UIImageView *myBanner = [[UIImageView alloc]
initWithImage:@"overlayImage.png"];
    self.backgroundImage = myBanner;
    [self.window addSubview:myBanner];
}
```

For Android, set the `FLAG_SECURE` property on the window to obscure any sensitive content.

REFERENCES

- OWASP Mobile Top 10 2016 – M2: Insecure Data Storage - https://www.owasp.org/index.php/Mobile_Top_10_2016-M2-Insecure_Data_Storage

TLS STRIPPING

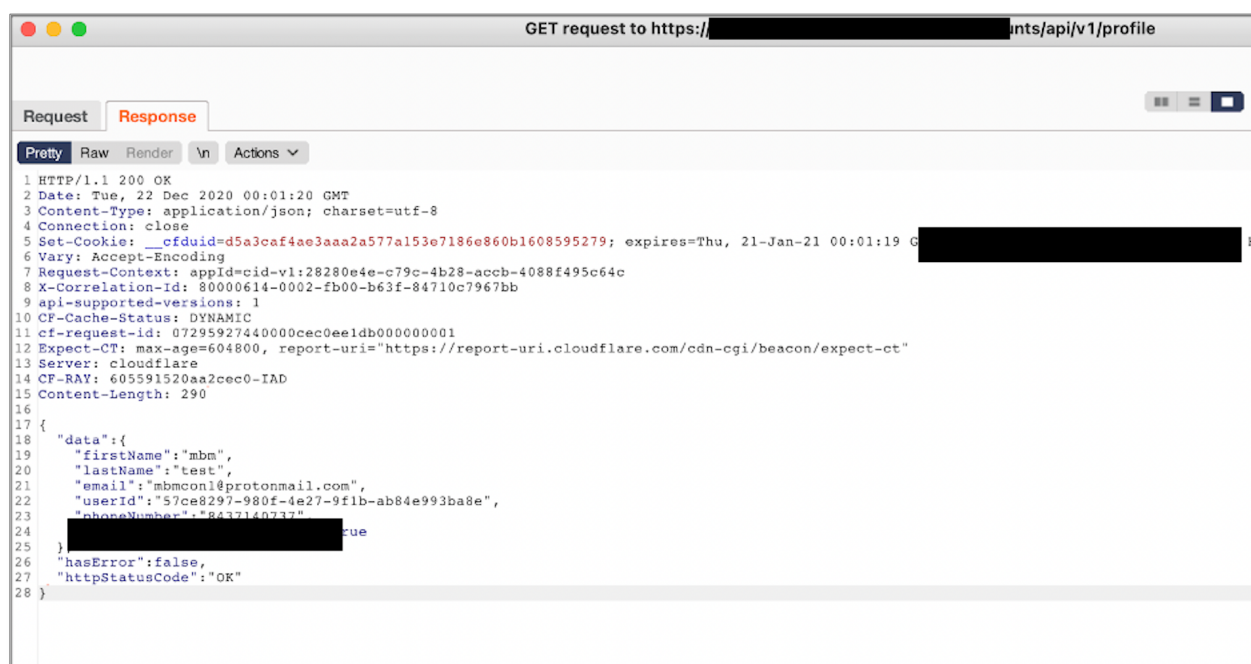
INFORMATIONAL

DESCRIPTION

The API permitted the client browser to initiate requests for TLS-protected resources over HTTP, allowing them to be intercepted and forwarded as HTTPS requests and enabling man-in-the-middle attacks against resources requiring TLS encryption.

For a mobile API, this is typically not an issue as “in the wild” clients will not be requesting resources directly as would be the case in a web application.

EVIDENCE



```
1 HTTP/1.1 200 OK
2 Date: Tue, 22 Dec 2020 00:01:20 GMT
3 Content-Type: application/json; charset=utf-8
4 Connection: close
5 Set-Cookie: __cfduid=d5a3caf4ae3aaa2a577a153e7186e860b1608595279; expires=Thu, 21-Jan-21 00:01:19 GMT; path=/; domain=[redacted]; HttpOnly; Secure
6 Vary: Accept-Encoding
7 Request-Context: appId=cid-v1:28280e4e-c79c-4b28-accb-4088f495c64c
8 X-Correlation-Id: 80000614-0002-fb00-b63f-84710c7967bb
9 api-supported-versions: 1
10 CF-Cache-Status: DYNAMIC
11 cf-request-id: 07295927440000cec0ee1db000000001
12 Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
13 Server: cloudflare
14 CF-RAY: 605591520aa2cec0-IAD
15 Content-Length: 290
16
17 {
18   "data": {
19     "firstName": "mbm",
20     "lastName": "test",
21     "email": "mbmcon1@protonmail.com",
22     "userId": "57ce8297-980f-4e27-9f1b-ab84e993ba8e",
23     "phoneNumber": "8437140737",
24     "[redacted]": "[redacted]@ue"
25   }
26   "hasError": false,
27   "httpStatusCode": "OK"
28 }
```

HTTP response with no HSTS header.

IMPACT

Difficulty	Likelihood
HARD	LOW

An attacker with successful interception capability, such as a shared Wi-Fi Man-in-the-Middle attack scenario could intercept the traffic and decrypt the communications. This could include credentials or financial information.

Given the application in its current state, the severity on this finding has been reduced to Best Practice.

RECOMMENDATION

Configure the API to return the Strict-Transport-Security header, along with the max-age marking.

```
Strict-Transport-Security: max-age=<expire-time>  
Strict-Transport-Security: max-age=<expire-time>; includeSubDomains
```

REFERENCES

- OWASP HTTP Strict Transport - https://www.owasp.org/index.php/HTTP_Strict_Transport_Security

APPENDIX: DEFINITIONS

Redpoint Security levels are determined by the evaluation of multiple factors surrounding the vulnerability and the environment the vulnerability was identified in. The rating of a vulnerability changes based on these factors. In general, the factors that make up the risk of an identified vulnerability include:

- Likelihood of vulnerability being identified
- Public availability of technical details
- Attack tools
- Requisite attacker skill
- Potential and likelihood for successful exploitation
- Attack surface and users affected

SEVERITY

Severity levels as well as specific factors documented with levels in the finding are further defined in the tables below. While these definitions encompass common scenarios, a vulnerability's Severity level may be adjusted based on the specific circumstances in which it was encountered.

CRITICAL – The exposure may be exploited resulting in outcomes such as system compromise, authentication bypass, or unauthorized data access by users without privileges or existing user access. These findings are typically exploitable without authentication and should be addressed immediately.

HIGH – The exposure may be exploited resulting in outcomes such as system compromise, privilege escalation, or unauthorized data access by users with access to the system. These findings are exploitable by existing users and should be addressed as soon as possible.

MEDIUM – The exposure may be exploited resulting in outcomes such as system compromise, or privilege escalation where some user interaction is required for the attack to be successful. These findings should be remediated once all critical and high severity findings are remediated.

LOW – The exposure may provide information or access, which, while not exposing the system to current risk, may expose the system to risk in the future. These findings should be addressed but can be remediated over a longer timeline.

BEST PRACTICE - Controls that could be implemented to further enhance the security posture of the application, or not based on business decision. Redpoint recommends a wide range of preventative controls to help stop vulnerabilities before they can be exploited. Implementing these controls with a robust SDLC program and regular reviews can greatly increase an applications security posture.

DIFFICULTY OF EXPLOIT

The Difficulty of Exploit is the level of effort that is required for a threat agent to successfully exploit vulnerabilities identified in the target. This measurement takes into account compensating controls or other conditions that may augment the effort required to exploit the vulnerability.

EASY – The level of effort or sophistication require to exploit vulnerabilities present in the target is minimal or zero. This vulnerability may be exploited by a threat with little or no intelligence or access. Compensating controls may not protect the target from attack. For example, a device that is exposed to the Internet that exhibits a vulnerability for which there is readily available exploits tools and/or techniques.

MODERATE – A moderate level of effort or sophistication is required to exploit the vulnerability. Compensating controls or conditions raise the level of effort require to attack the target. A threat must

bypass compensating security controls to successfully exploit vulnerabilities. For example, a vulnerability that is present on a device but protected by a firewall requires a higher level of effort to exploit than if the vulnerable device was exposed directly to the Internet.

HARD – Significant effort is required to attack the device. Compensating controls significantly shield the target from attack by threats. Threats must circumvent security controls in order to exploit vulnerabilities present in the target. For example, the device is only accessible by an authenticated internal user with elevated access rights.

LIKELIHOOD

Redpoint evaluates the Likelihood of exploit based on criteria regarding whether application privileges are required for the exploit, user interaction is required to successfully use the attack, or, especially, whether there is significant exploit code maturity, i.e., attackers are commonly using attack pathways that align with the vulnerability.

HIGH – The vulnerability exists in a critical attack path which attackers commonly exploit. There is a minimal need for interaction with user or application admins or maintainers or an internal privileged role within the organization to successfully perform the exploit.

MEDIUM – The exploit code maturity is less developed than is the case with HIGH rated likelihood exploits, but there still is awareness of the attack vector. There may need to be some trivial user interaction or successful phishing to perform the exploit.

LOW – The vulnerability requires innovation or edge case testing to successfully perform an exploit. The attacker would have to be internal to the organization or work to compromise a hard human target to successfully exploit the vulnerability.

OVERALL APPLICATION RATING

Redpoint evaluates the reviewed application in the context of other assessments we have performed. This rating strives to answer the common questions we receive from clients wondering how their applications compare to others. As such, the overall ranking reflects our researchers' qualitative evaluation of an application's security posture relative to the security posture of other applications we have reviewed. The following is a rough guide regarding the meaning of these qualitative evaluations.

- **Critically Insecure** – The application has several critical and high-severity vulnerabilities that are accessible to unauthenticated attackers with well-established methods and vectors for exploits. Confidentiality, Integrity, and Availability of the application are all at risk, and remediation solutions require potentially difficult implementations. Redpoint recommends immediate remediation of specified findings to protect your customers and your organization's security reputation.
- **Insecure** – The application has high-severity vulnerabilities that threaten the Confidentiality, Integrity, or the Availability of the application. However, mitigations should not present too many implementation challenges. The high-risk vulnerabilities should be remediated expeditiously to protect customers and your organization's security reputation.
- **Moderately Insecure** – The application may have one or two high-risk vulnerabilities and several medium-risk findings that threaten Confidentiality, Integrity, or Availability. Mitigation should proceed according to priority of threat.
- **Average** – The application presents vulnerabilities that are commonly found in application testing, but none are especially severe. Mitigation should proceed according to priority of threat.

- **Moderately Secure** – Redpoint did not find any vulnerabilities that were concerning, and generally the application handled authentication, authorization, user input, data transfer and storage in a secure manner. The report includes a couple mitigation recommendations to improve the application's overall security posture or to align with security best practices.
- **Secure** – Redpoint evaluates the application as demonstrating a secure posture relative to Confidentiality, Integrity, and Availability. Findings are largely related to best security practices.

APPENDIX: METHODOLOGY

APPROACH

Redpoint utilizes a standard methodology for assessments that is comprised of three phases: information gathering, vulnerability identification, and reporting. Each phase feeds the next, but any activity in later phases may inform additional research and testing. The activities are cyclical to provide the analyst with working knowledge of the targeted properties for additional threat vectors.

INFORMATION GATHERING

Redpoint starts by reviewing application endpoints based on availability, application use-cases, developer documentation, and application source code. These endpoints are analyzed for use, potential parameters, additional attack surface, and possible threats. Applications are reviewed during this phase from multiple points of view, including an anonymous, un-authenticated user, an authenticated user, and an authenticated partner.

Redpoint analyzes client-identified mobile application binaries and client-provided source code during mobile assessments for controls that affect security posture, including authentication and authorization controls, sessions management, API communications between mobile application and its supporting network, data storage, logging behavior, communication protocols, input handling, encryption settings, and other application behavior.

VULNERABILITY IDENTIFICATION

Redpoint uses the identified endpoints and controls of the identified assets to identify and explore possible security vulnerabilities across applications based on our expertise in assessing application flaws. Special attention will be paid to possible fraud and business logic flaws that could affect the Client, its partners, or its customers.

Redpoint utilizes industry-standard vulnerability lists for assessment purposes, including OWASP's Mobile Application Security Verification Standard (MASVS) 0.9.2., OWASP's Application Security Verification Standard, the OWASP Top 10 Security Risks, and the SANS CWE Top 25 Software Errors. These vulnerabilities are assessed across various security domains as they apply to the targeted application. Additional attack surfaces and weaknesses may be noted during this portion of the assessment for further research.

REPORTING

To finalize the assessment activity, Redpoint documents the assessment vulnerabilities, endpoints, and findings in a report that summarizes the results into actionable items for remediation by the Client. Each finding documents the steps required to reproduce identified vulnerabilities and includes recommendations for remediating or mitigating the threat.