BLAZE

MOBILE APPLICATION PENETRATION TESTING

ACME FINANCE

CLIENT

BLAZE SAMPLES

DATE

26/11/2022

SUMMARY

1.0 Document Control

1.1 Version Control

AUTHOR	DELIVERY DATE	PAGES	VERSION	STATUS
Alan Turing	21/11/2022	51	0.7	First draft
Ada Lovelace	23/11/2022	51	0.8	Technical QA
Donald Knuth	24/11/2022	51	0.9	QA
Alan Turing	25/11/2022	51	1.0	Final

1.2 Document Distribution

NAME	TITLE	ORGANIZATION
Alan Turing	Lead Security Engineer	Blaze Information Security
Ada Lovelace	Security Engineer	Blaze Information Security
Donald Knuth	Project Manager	Blaze Information Security
Dade Murphy	Director of Information Security	ACME Ltd.
Kate Libby	Information Security Manager	ACME Ltd.

2.0 Introduction

This document presents the results of a Mobile Application Security Assessment for ACME Ltd.. This engagement aimed to identify security vulnerabilities that could negatively affect the systems under scope per se, the data they handle, and consequently the business. Blaze Information Security simulated in a systematic way, attacks that were specifically tailored for the engagement's scope to test the resilience against real-life attack scenarios.

The main objectives are presented below.

The analysis focused on vulnerabilities especially related to implementation, and on issues caused by architectural or design errors.

For each vulnerability discovered during the assessment, Blaze Information Security attributed a risk severity rating and, whenever possible, validated the existence of the vulnerability with a working exploit code. The issues' severity classification is based on the potential it presents to provide means for fraud, data leakage, and other harmful events that may bring a direct adverse impact to the business.

A remediation priority suggestion can be found in Appendix B of this document.

3.0 Scope

The mobile application under the scope, ACME Financial, was subjected to a security-focused test.

The scope of the assessment comprised of:

APPLICATION	PLATFORM
ACME Financial	iOS
ACME Financial	Android

URL	IP
https://dev-acme.com	123.123.123.123

4.0 Engagement Summary

The engagement was performed in a period of **8** business days, including report writing. The mobile application security assessment commenced on **November 14th, 2022** and ended on **November 23rd, 2022**, finishing with the final version of this report. The calendar below illustrates the allocated days by Blaze for this project.

ACTIVITY CALENDAR

NOVEMBER 2022

S	M	T	W	T	F	S
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14 ✓	15 ✓	16 ✓	17 ✓	18 ✓	19
20	21 ✓	22 ✓	23 ✓	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

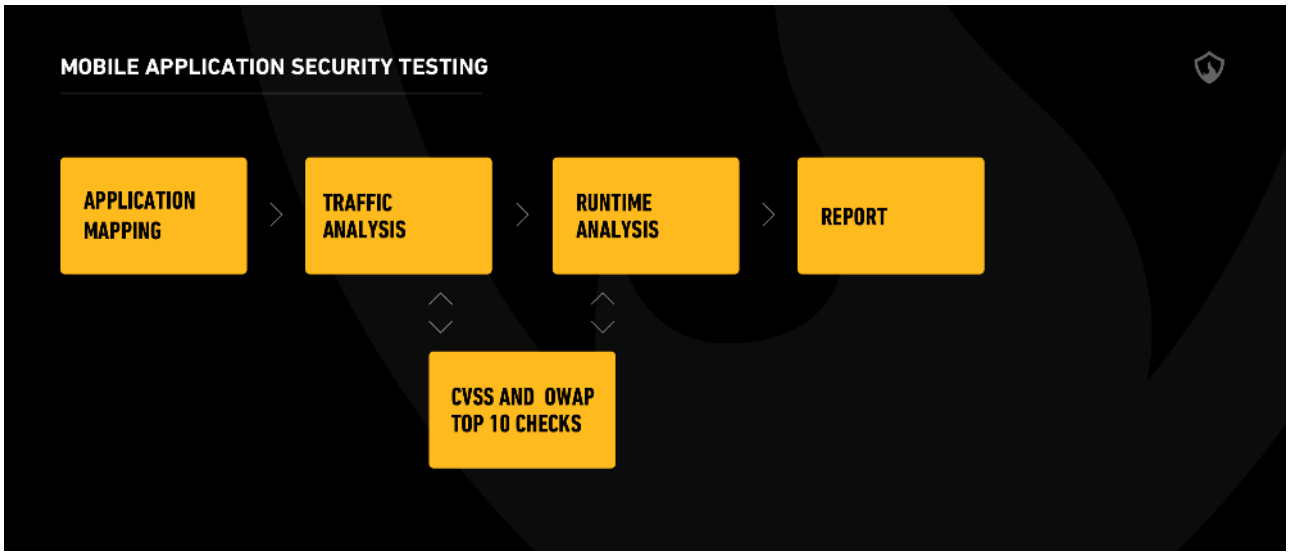
All testing activities took place against the development environment.

The mobile application was analyzed with the assistance of automated scanning tools as well as subjected to manual review.

All work was carried out remotely from the offices of Blaze Information Security.

5.0 Methodology - Mobile Application Security Testing

Blaze Information Security works with its own methodology while concurrently running the necessary tests to identify and classify vulnerabilities according to OWASP Top 10 and CVSS (Common Vulnerability Scoring System) which provides standardization of the severity of each vulnerability, adding a defensible logic behind each security flaw risk classification.



- Application Mapping
  This stage consists in observing the application's behavior to determine its features, states, protocols, and use of frameworks, and APIs. This mapping will aid in the understanding of the application under the scope and will serve as input for the correct targeting of the security assessment and vulnerability exploitations.
- Traffic Analysis
  This is the phase where Blaze will check for informative error messages, cacheable information, injections, privacy policies, cryptography, authentication resilience, business logic, data storage security, and others. In sum, scrutiny through the Client-Side checking for multiple vulnerabilities regarding traffic and data processing of information by the application.
- Runtime Analysis
  Consists in the attempt to successfully reverse engineer the application, analyzing its interaction with the operating system, checking for buffer overflows, injections in runtime, and others.
- CVSS and OWASP Top 10 Checks
  In this stage, Blaze uses as a reference the OWASP Top 10 Mobile 2016. This list reflects the main vulnerabilities commonly identified in Mobile environments.

- Report
  This is the final step of the project where all detected vulnerabilities are documented along with screenshots to confirm their existence. In this documentation process, all processes, tools, and techniques that led to the successful exploitation of vulnerabilities found during the project are detailed. Furthermore, mitigation measures for each identified security issue are presented.

6.0 Executive Summary

This topic summarizes the results of the technical security consultancy provided by Blaze Information Security for Blaze Samples concerning the assessment of a mobile application named ACME Financial, as mentioned above.

Blaze Samples a company that provides payment and banking services to the cannabis and associated sectors, requires special attention on the security of its information systems and defenses that could help mitigate every type of cyber security attack and potential fraud. Hence, Blaze Information Security was involved in the penetration test of ACME Financial mobile application for iOS and Android. First and foremost, all of the attack surface of the systems under scope was mapped in order to identify possible attack vectors. Then, attacks were specifically tailored and targeted against every piece of identified functionality.

The penetration test and security analysis of the ACME Financial mobile application had as its main goal the identification and exploitation of the maximum number of vulnerabilities in order to assess its security posture when faced with skilled and determined attackers. Thus, the applied methodology attempted to enumerate all the cases of sensitive information exposure and unauthorized access possibilities that would lead a would-be attacker into compromising the application itself.

This project was conducted during a period of 8 work days by one security engineer and all testing activities took place against the development environment. The ACME Financial mobile application and its underlying infrastructure were analyzed with the assistance of automated scanning tools as well as subjected to manual review.

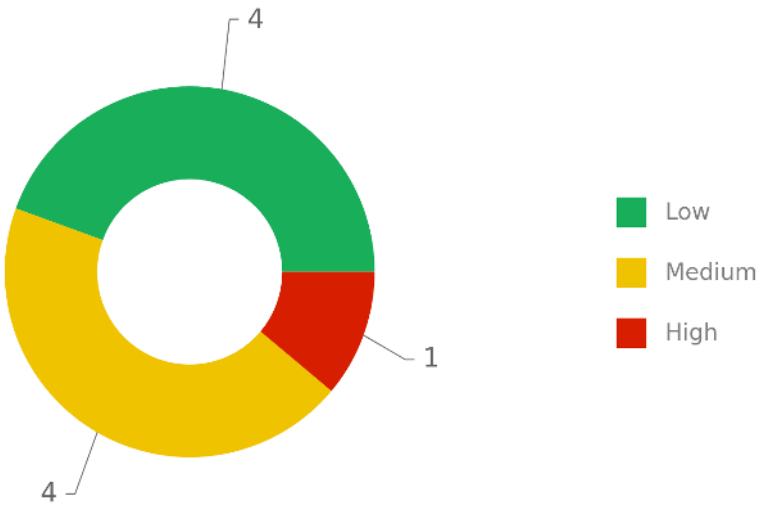
The result of this project offers a broad and clear overview of the risks to which the company is exposed through the assessed scope, granting a solid foundation with evidence and means of remediation for confident decision-making, optimizing the efficiency of the security posture of the environment under the scope.

6.1 Vulnerable Surface

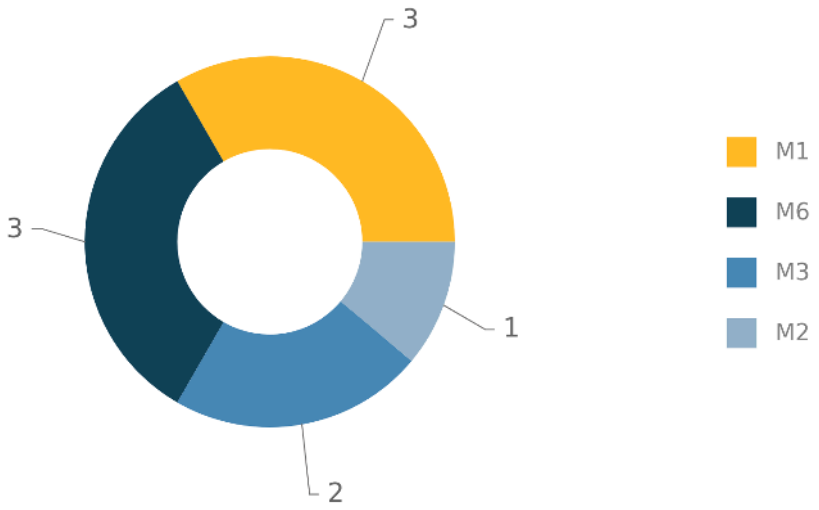
The mobile applications under scope presented a security posture considered insufficient, given the size of its attack surface for outsider/remote attackers, and the results of our security tests revealed the existence of a considerable amount of security vulnerabilities. This, of course, considers their severity and the sensitive nature of the application under the scope. Additionally, due to its attractiveness for potential cyber attacks, it was also noted the possibility to further increase the robustness of already existent security mechanisms and the opportunity to employ new security measures to prevent potential future attack attempts.

The vulnerable surface was thereby considered large, with 9 vulnerabilities in total. Of all of the vulnerabilities reported, 1 of them was ranked as of high severity, 4 considered as of medium, and the remaining vulnerabilities were considered of low severity.

The chart below illustrates the severity x quantity of the issues detected:



The chart below illustrates the OWASP ratings:



- **M1 – Improper Platform Usage**
- **M2 – Insecure Data Storage**
- **M3 – Insecure Communication**
- **M6 – Insecure Authorization**
- **M8 – Code Tampering**

6.2 Main Threats

Blaze Information Security encountered several issues in the application which may allow for the following real-world scenarios to materialize:

- **Unauthorized transfer of funds**

During the tests performed by Blaze Information Security, one of the functionalities identified in the mobile application under scope allowed its users to fund their ACME accounts using linked bank accounts. However, due to the absence of access control checks and verifications employed on this functionality, a potentially malicious user was able to fund his/her ACME account with the bank account of another registered user leading to an unauthorized debit on the victim's account.

- **Application does not enforce certificate pinning**

Certificate pinning (or SSL pinning) aims essentially to reduce the risk of Man-in-the-Middle

(MITM) attacks that involve the compromise of a trusted CA. The technique works by getting a hash of the original valid certificate, and hardcode it into the application. The application was not benefiting from this protection in some endpoints that handle sensitive information such as user's credentials. It is recommended to enable this mitigation in every area of the application to assure the best coverage possible against this kind of attack.

- **User's session tokens stored in application's storage**

The application stores a user's sensitive data, such as credentials, in unencrypted form in a directory on Android. Since the application executes normally on an insecure execution environment (rooted device), other installed applications may be able to access such sensitive information, among other attack scenarios such as an adversary possessing physical access. Following the best security practices, the credentials should be stored on the native Android Keystore. Developers can use the Keystore to store cryptographic keys and other sensitive material in a container to make them more difficult to extract from the device.

6.3 Vulnerabilities Table

The following table summarizes the vulnerabilities found in the application, in line with CWE whenever possible, presenting a reference regarding the impact of each vulnerability.

POINT	TITLE	CWE-ID	CWE CATEGORY	SEVERITY
1	API: Insufficient Access Control Allows for Unauthorized Funds Transfer	CWE-284	Improper Access Control	HIGH
2	API: Absence of PIN entry to execute transactions	CWE-657	Violation of Secure Design Principles	MEDIUM
3	API: Insecure Direct Object Reference (IDOR) in the Purchase functionality	CWE-639	Authorization Bypass Through User-Controlled Key	MEDIUM
4	iOS: Mobile application does not enforce certificate pinning	CWE-295	Improper Certificate Validation	MEDIUM
5	Android: Application does not enforce certificate pinning	CWE-295	Improper Certificate Validation	MEDIUM
6	Android: Background screen caching	CWE-1021	Improper Restriction of Rendered UI Layers or Frames	LOW
7	iOS: Background screen caching	CWE-1021	Improper Restriction of Rendered UI Layers or Frames	LOW
8	iOS: Absence of jailbreak detection	CWE-250	Execution with Unnecessary	LOW

			Privileges	
9	Android: Cleartext Storage of Sensitive Information	CWE-312	Cleartext Storage of Sensitive Information	**LOW**

7.0 Technical Summary - Mobile

The present topic describes the assumptions, tests, and attack attempts that took place during the security assessment of the ACME Financial mobile applications under the scope, for both Android & iOS.

First and foremost, all of the attack surfaces of the ACME Financial mobile applications were mapped in order to identify all of its possible attack vectors. During this time, the security engineer interacts with the available features so that they can familiarize themselves with the application and thus identify attack paths and possible points of vulnerable functionality.

After feature mapping and functionality enumeration, the security analyst focuses on attack vectors that may arise from an unauthenticated perspective, i.e., adversaries that do not have valid security credentials and/or authorization tokens for ACME Financial 's underlying API.

Focusing on attacks that can be conducted by authenticated users, the security analyst attempted to identify attack vectors that could negatively impact the system under the scope, thus assuming a malicious registered and authorized user attacker/threat model. This phase is crucial in an attempt to uncover possible access control & privilege escalation vulnerabilities, whereby a malicious user A is able to obtain and/or alter sensitive data of a victim user B. The malicious user A either seeks to elevate its privileges horizontally (victim user B possesses the very same privilege level as malicious user A) or vertically (victim user B possesses an increased privilege level than malicious user A).

Additionally, due to the sensitive nature of the mobile application under the scope, its money transfer functionalities were thoroughly examined and tested against race condition vulnerabilities, as well as validating the proper verifications of the specified amount when compared to the user's actual balance.

Additionally, platform-related issues are also tested for both applications installed on either Android & iOS. This process includes decompiling the applications & reverse engineering them, in an attempt to infer their implementation details, possibly identify insecure code constructs which may lead to security vulnerabilities, bypass possible software protections and hunt for hardcoded keys and/or credentials.

Thus, for both authenticated & unauthenticated threat models, the following tests on the API attempted to uncover:

- ✓ Business Logic issues
- ✓ SQL/NoSQL Injection issues
- ✓ Cross-site Scripting (XSS) issues
- ✓ Bruteforce Attempts and/or Absence of Rate Limiting, Account Lockouts, and Username Enumerations
- ✓ Insecure Direct Object Reference (IDOR) – Access Control issues
- ✓ Open Redirect issues
- ✓ Outdated Software Components
- ✓ Path fuzzing & Sensitive unprotected Directory/File identification issues
- ✓ Server Side Request Forgery (SSRF) issues
- ✓ Insufficient Input Validation issues
- ✓ Security Misconfigurations

Alternatively, the following platform-related issues were examined:

- ✓ Hardcoded API keys and/or credentials
- ✓ Bypass Software Protections
- ✓ Insecure Communications with API
- ✓ Overly Permissive Permissions
- ✓ Task Hijacking
- ✓ Plaintext Storage of Sensitive Files
- ✓ Insecure use of Deeplinks and/or URL Schemes
- ✓ Insecure Intents

8.0 Vulnerabilities

8.1 API: Insufficient Access Control Allows For Unauthorized Funds Transfer

SEVERITY: HIGH	**CWE-ID:** CWE-284	**CVSS SCORE:** 7.1
AFFECTED POINTS	https://acme.test/api/transfer_funds	
OWASP TOP 10	M6 - Insecure Authorization	

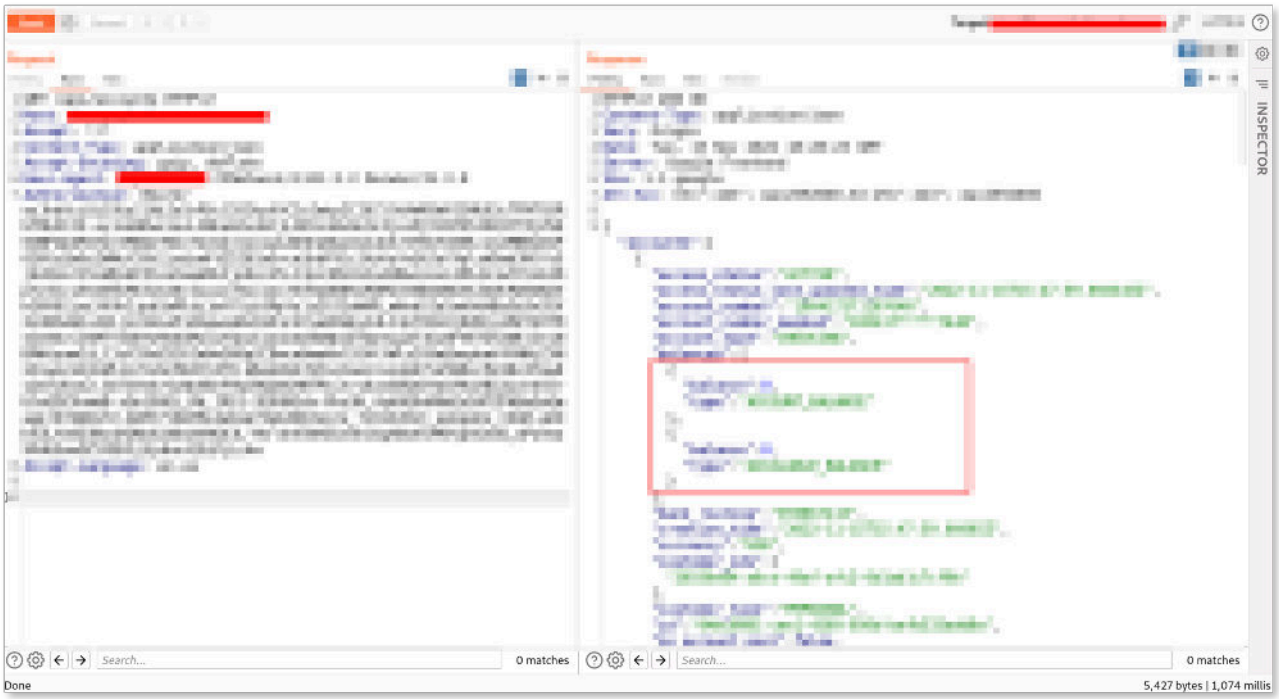
Description

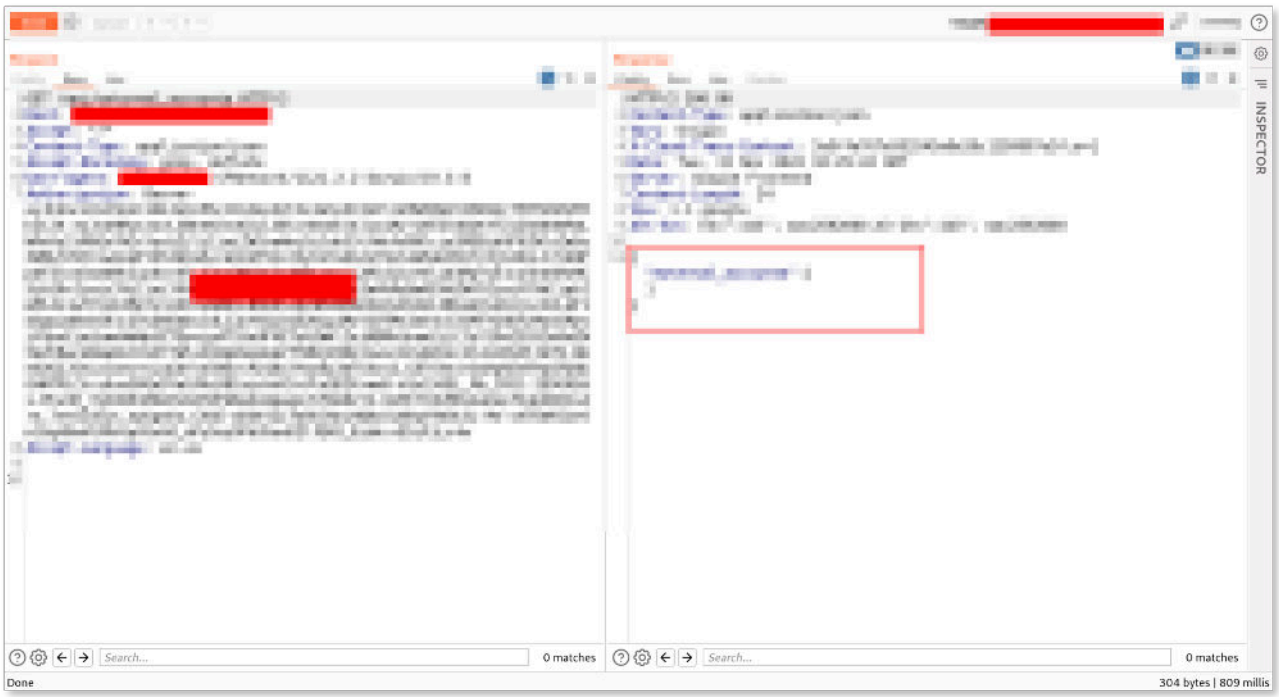
Access control is a security mechanism that allows or denies access to content and functionalities for users. Privilege checking is done after the authentication process, which will govern what authenticated users can access.

When wrongly implemented, access control becomes a security risk, allowing sensitive information to be accessed by unauthorized users and even application features to be used by agents without proper authority.

During the assessment carried out by Blaze Information Security, it was observed that a user could fund the ACME account using a bank account registered by another user.

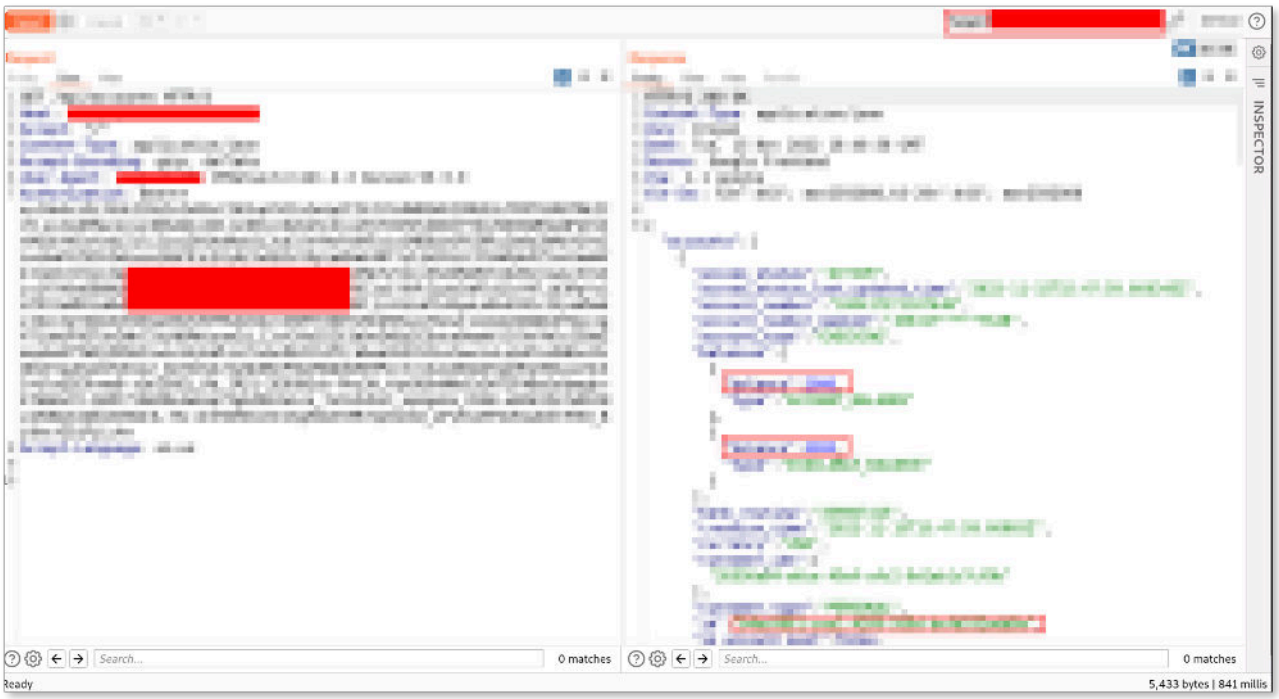
To demonstrate this vulnerability, we created a new user *normal_user+60@blazeinfosec.com*, which had a zero balance and no bank(plaid) account linked, as seen in the screenshot below:





Using the *Transfer funds* endpoint, we initiated a transfer from the registered bank account of another user *normal_user@blazeinfosec.com* (1988386e-c56d-410f-8f06-4800a50cd7d7), and a receiving account of the attacker (04bc0881-cee1-4209-950d-befb532add8a) as shown in the image below:

A 200 OK response was received, which means the funding request was successful, and the amount is now present in *normal_user+60@blazeinfosec.com* ACME account, as shown in the screenshot below:



Reference

- https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_A5-Broken_Access_Control
- https://cwe.mitre.org/data/definitions/284.html
- https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html

Solution

To solve the mentioned problem, Blaze Information Security recommends ACME conduct a survey of application access control requirements and document them in the application security policy. It is extremely important that each user can only access his/her private information. In this particular case, the API should verify if the funding account belongs to the requesting user.

8.2 API: Absence Of PIN Entry To Execute Transactions

SEVERITY: MEDIUM	**CWE-ID:** CWE-657	**CVSS SCORE:** 5.9
AFFECTED POINTS	https://acme.test/api/transfer_funds	
OWASP TOP 10	M6 - Insecure Authorization	

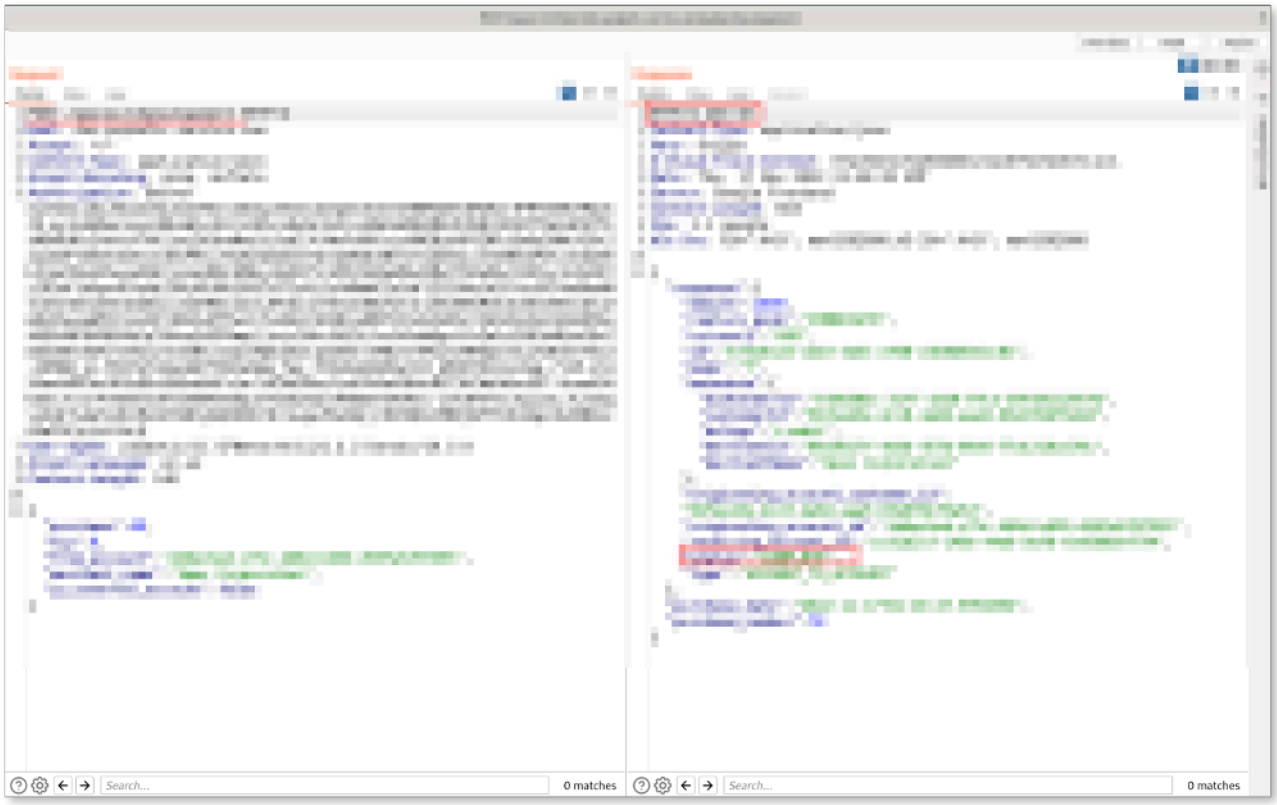
Description

A personal identification number (PIN) is a numerical code used in many electronic financial transactions. The purpose of a personal identification number (PIN) is to add additional security to the electronic transaction process by providing nonrepudiation.

The absence of PINs makes it easier for an attacker who has compromised a user's account to access the user's account and perform transactions.

During the tests performed by Blaze Information Security, it was observed that the mobile application under scope did not require the user to input a pin to authorize/approve a transaction.

This issue is highlighted in the below screenshot:



This is made more threatening due to the presence of vulnerabilities such as cleartext storage of session tokens as well as the ability of the application to run in an insecure environment (root/jailbreak).

Reference

- https://www.cryptomathic.com/news-events/blog/why-banks-need-non-repudiation-of-origin-and-non-repudiation-of-emission

Solution

Blaze Information Security recommends the use of PINs to authorize/approve transactions.

8.3 API: Insecure Direct Object Reference (IDOR) In The Purchase Functionality

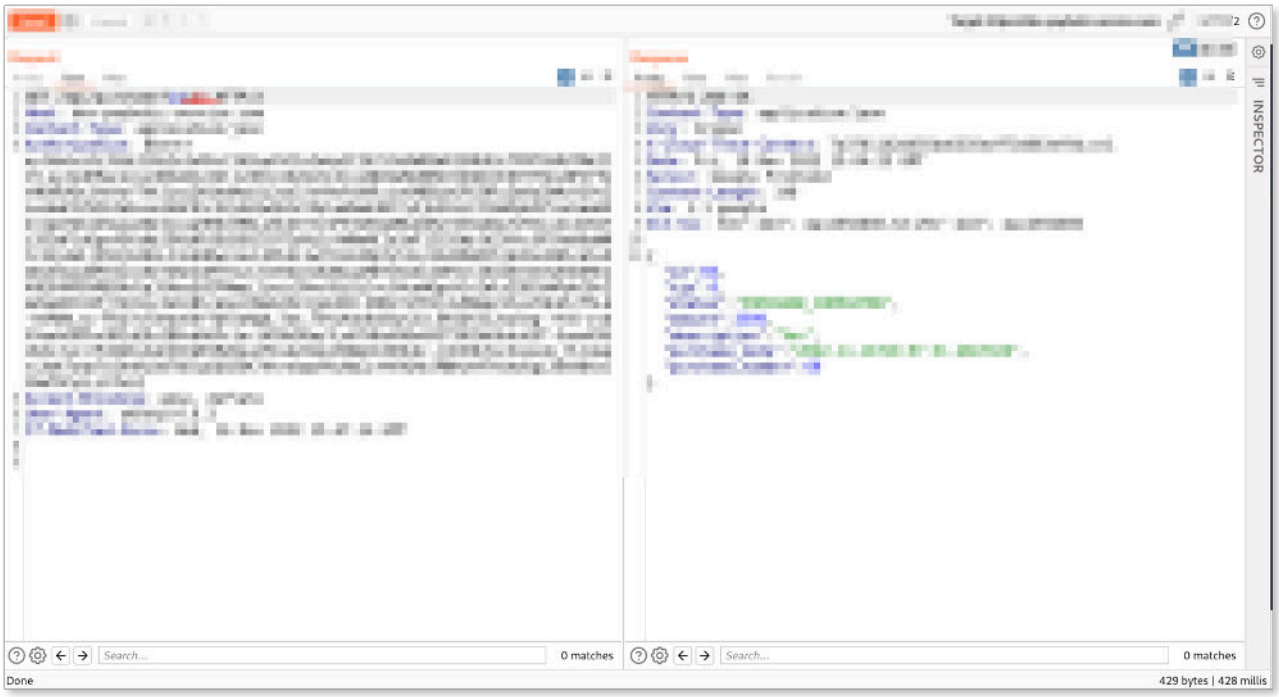
SEVERITY: MEDIUM	**CWE-ID:** CWE-639	**CVSS SCORE:** 4.3
AFFECTED POINTS	https://dev-acme-financial.test/api/purchase?id=	
OWASP TOP 10	M6 - Insecure Authorization	

Description

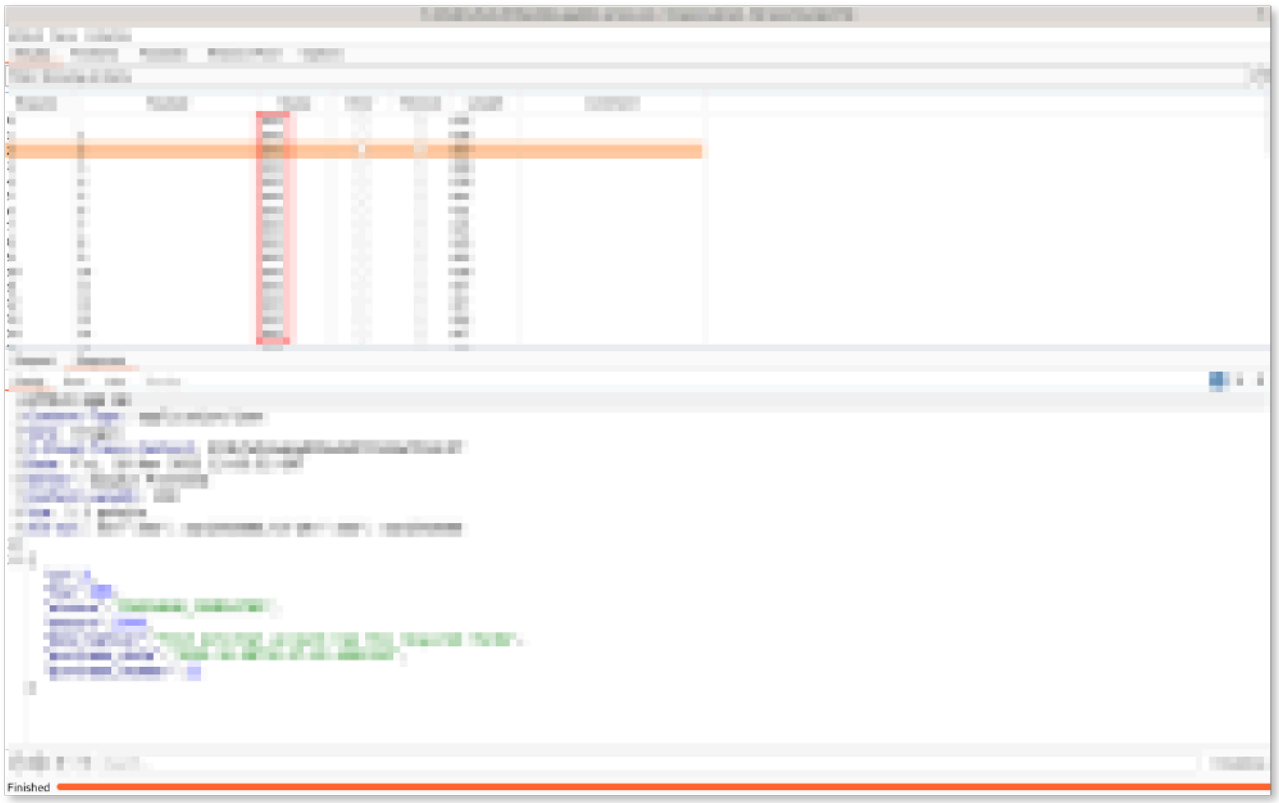
Insecure Direct Object Reference (IDOR) is an access control vulnerability that arises when an application uses user-supplied input to access internal objects directly. Without proper validation of input and access control mechanisms, an attacker may be able to access resources, leak or even alter sensitive information about other users.

During the security assessment, it was observed that the purchase ID followed a sequential pattern and didn't implement any sort of restriction, making it possible to view the status of all transactions/purchases.

The screenshot below represents a sample request generated by a user:



By iterating through the IDs, we could view all transactions/purchases performed by other users:



Reference

- https://www.acunetix.com/blog/web-security-zone/what-are-insecure-direct-object-references/
- https://portswigger.net/web-security/access-control/idor

Solution

In order to solve the mentioned problem, Blaze Information Security recommends ACME to conduct a survey of application access control requirements and document them in the application security policy, as it is extremely important that each user can only access his own private information. In this particular case, the API should verify if the purchase ID belongs to the requesting user.

8.4 IOS: Mobile Application Does Not Enforce Certificate Pinning

SEVERITY: MEDIUM	**CWE-ID:** CWE-295	**CVSS SCORE:** 5.4
AFFECTED POINTS	iOS	
OWASP TOP 10	M3 - Insecure Communication	

Description

The security model brought by TLS/SSL, widely used across web and mobile applications to ensure confidentiality and integrity to data in transit, is based on the concept of trusted third parties.

Operating systems and browsers have one or more Certificated Authority (CA) marked as trusted. This way, certificates generated by those trusted CAs are automatically accepted as valid. These certificates have fields that identify the domains to which the certificate was generated for, issue date, fingerprint for unique identification, and more.

In case an attacker compromises a trusted CA, the attacker could issue their own malicious certificates for arbitrary domains. Consequently, the adversary could perform Man-in-the-Middle (MITM) attacks, impersonating the server for which they have a valid, yet malicious certificate.

The technique known as certificate pinning (or SSL pinning), aims to reduce the risk of such attacks, that involve the compromise of a trusted CA. The technique works by getting a hash of the original valid certificate, and hardcode it into the application.

For future requests, the application will not only verify the expiration date of the certificate but also will match the hashes, because even valid certificates, but issued in different dates by different CAs, will have different fingerprints.

In order to prove the absence of such mechanism, a purposely made certificate was generated and its CA marked as trusted in the mobile phone. The following screenshots displays the traffic interception without any need to tamper with the application or the device, underscoring the absence of certificate pinning in the app under test:

Reference

- https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
- https://carvesystems.com/news/cert_pin/

Solution

In order to solve the mentioned problem, Blaze Information Security advises to implement certificate pinning in the whole application to increase the cost against active man-in-the-middle attacks.

8.5 Android: Application Does Not Enforce Certificate Pinning

SEVERITY: MEDIUM	**CWE-ID:** CWE-295	**CVSS SCORE:** 5.4
AFFECTED POINTS	Android	
OWASP TOP 10	M3 - Insecure Communication	

Description

The security model brought by TLS/SSL, widely used across web and mobile applications to ensure confidentiality and integrity to data in transit, is based on the concept of trusted third parties.

Operating systems and browsers have one or more Certificated Authority (CA) marked as trusted. This way, certificates generated by those trusted CAs are automatically accepted as valid. These certificates have fields that identify the domains to which the certificate was generated for, issue date, fingerprint for unique identification, and more.

In case an attacker compromises a trusted CA, they could issue their own malicious certificates for arbitrary domains. Consequently, the adversary could perform Man-in-the-Middle (MITM) attacks, impersonating the server for which they have a valid, yet malicious certificate.

The technique known as certificate pinning (or SSL pinning), aims to reduce the risk of such attacks, that involve the compromise of a trusted CA. The technique works by getting a hash of the original valid certificate, and hardcode it into the application.

For future requests, the application will not only verify the expiration date of the certificate but also will match the hashes, because even valid certificates, but issued in different dates by different CAs, will have different fingerprints.

In order to prove the absence of such mechanism, a purposely made certificate was generated and its CA marked as trusted in the mobile phone. The following screenshot displays the traffic interception without any need to tamper with the application or the device, underscoring the absence of certificate pinning in the app under test:

Reference

- https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning
- https://carvesystems.com/news/cert_pin/

Solution

In order to solve the mentioned problem, Blaze Information Security advises to implement certificate pinning in the whole application to increase the cost against active man-in-the-middle attacks.

8.6 Android: Background Screen Caching

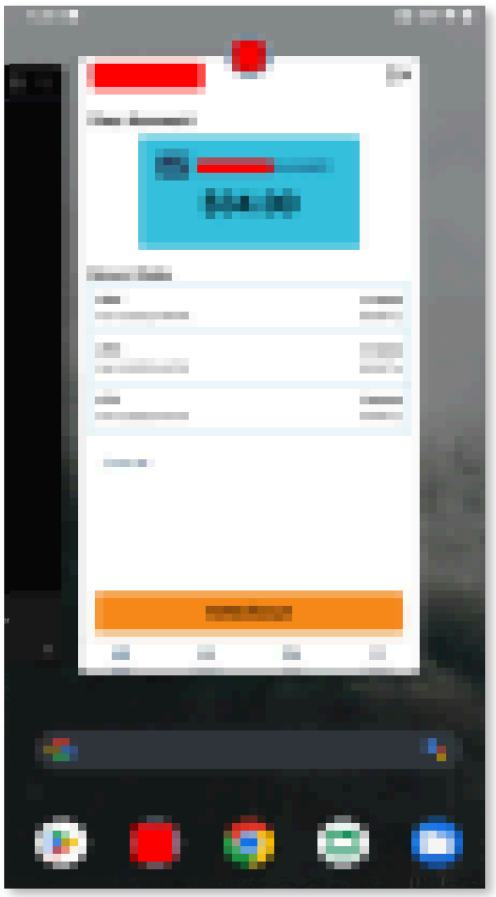
SEVERITY: LOW	**CWE-ID:** CWE-1021	**CVSS SCORE:** 3.3
AFFECTED POINTS	Android	
OWASP TOP 10	M1 - Improper Platform Usage	

Description

When the App Overview key of an Android device is pressed, a screenshot of the application's current state is taken before it goes to background execution.

In case sensitive or confidential information was being displayed on the screen at the time of the screenshot, this sensitive data may be stored unencrypted in the device.

The following screenshot shows a screen capture of the application when it was backgrounded while displaying sensitive information:



Reference

- https://developer.android.com/reference/android/view/WindowManager.LayoutParams#FLAG_SECURE
- https://www.tutorialspoint.com/how-do-i-prevent-android-taking-a-screenshot-when-my-app-goes-to-the-background
- https://cwe.mitre.org/data/definitions/1021.html

Solution

There are three popular solutions to prevent a screenshot from being taken when an application goes to background execution:

Explicitly mark the fields hidden via the View Controller.
Overlay another image immediately before the application goes into background state, so the

screenshot will instead capture this image and not one of the application.
Using Android API method setFlags of Windows' class as described below.
This last method handover developed the possibility of changing the application exhibition configuration. One of the possible configurations is FLAG_SECURE, a flag that prevents the application content to appear in screenshots or non-secure views:

The code below is an example of not making use of FLAG_SECURE:

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

The code below is an example of the same application using FLAG_SECURE:

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(
                WindowManager.LayoutParams.FLAG_SECURE,
                WindowManager.LayoutParams.FLAG_SECURE
        );
        setContentView(R.layout.activity_main);
    }
}
```

Blaze Information Security recommends the use of FLAG_SECURE in all screens that expose sensitive information.

8.7 IOS: Background Screen Caching

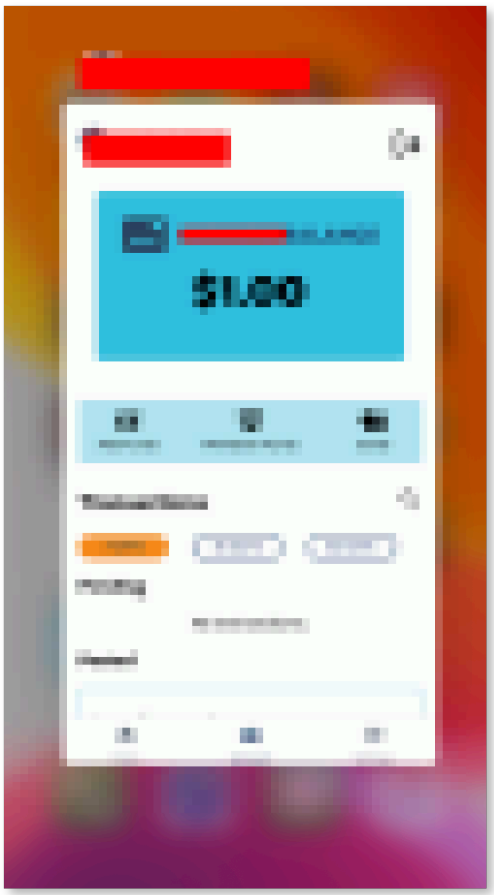
SEVERITY: LOW	**CWE-ID:** CWE-1021	**CVSS SCORE:** 3.3
AFFECTED POINTS	iOS	
OWASP TOP 10	M1 - Improper Platform Usage	

Description

When the Home key of an iPhone or iPad is pressed, a screenshot of the application's current state is taken before it goes to background execution.

In case sensitive or confidential information was being displayed on the screen at the time of the screenshot, this sensitive data may be stored unencrypted in the device.

The following screenshot shows a screen capture of the application when it was backgrounded while displaying sensitive information:



Reference

- https://www.virtuesecurity.com/blog/ios-background-screen-caching/
- https://developer.apple.com/documentation/uikit/uiapplication/1623097-ignoresnapshotonnextapplicationl
- https://medium.com/nomtek/screenshot-preventing-on-mobile-apps-9e62f51643e9
- https://blog.mindedsecurity.com/2021/05/mobile-screenshot-prevention-cheatsheet.html
- https://cwe.mitre.org/data/definitions/1021.html

Solution

There are two popular solutions to prevent a screenshot from being taken when an application goes to background execution:

Explicitly mark the fields hidden via the View Controller.
Overlay another image immediately before the application goes into background state (using 'sceneWillResignActive' and 'sceneDidBecomeActive' methods from AppDelegate/ SceneDelegate, for example), so the screenshot will instead capture this image and not one of the application.

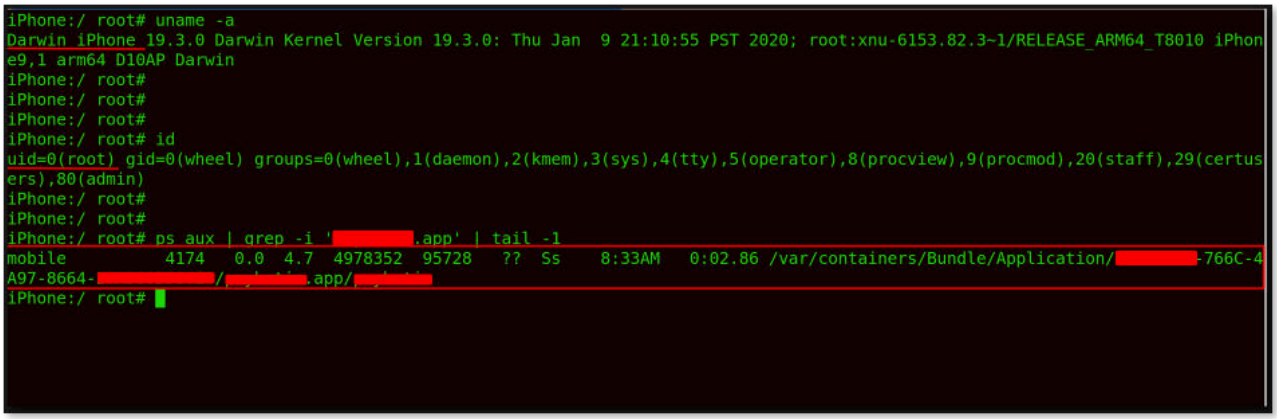
8.8 IOS: Absence Of Jailbreak Detection

SEVERITY: LOW	**CWE-ID:** CWE-250	**CVSS SCORE:** 3.8
AFFECTED POINTS	iOS	
OWASP TOP 10	M1 - Improper Platform Usage	

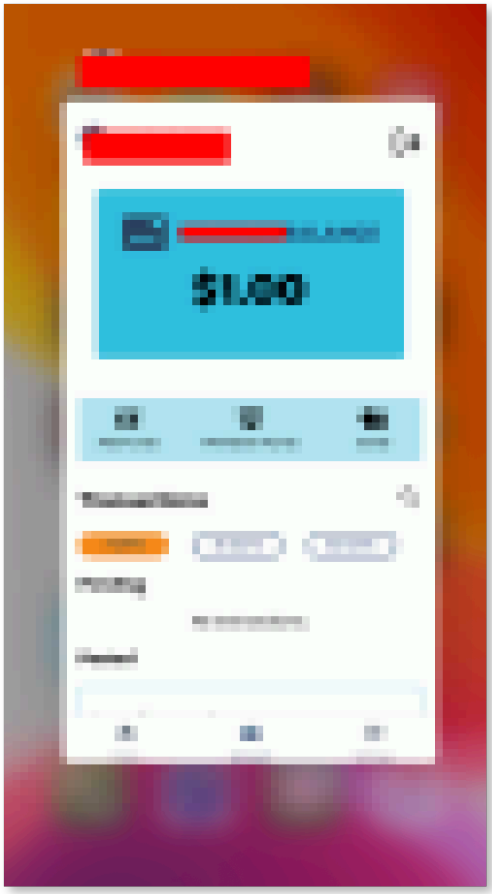
Description

During the tests performed by Blaze Information Security, it was found that the application under testing could be executed on jailbroken/rooted devices. When a device is jailbroken or rooted, a series of security mechanisms that the platform provides become disabled, thus increasing the probability of an attack against the installed application.

The following image demonstrates the application under testing executing on an insecure device:



The following image demonstrates the actual application running on the insecure device:



Reference

- https://support.apple.com/en-us/HT201954
- https://www.guardsquare.com/en/blog/protecting-apps-checkra1n-ios-jailbreak-exploit
- https://cwe.mitre.org/data/definitions/250.html

Solution

In order to solve the mentioned problem, it is recommended that security mechanisms that attempt to detect whether a device is insecure (Jailbroken/rooted) should be implemented. Even though these attempts can always be bypassed by determined attackers, this solution prevents that non-tech-savvy users install the application in such insecure devices, thus not leaking sensitive information to potentially malicious applications.

Examples of such verification techniques include: verifying the existence of files and/or folders

that would normally only exist in insecure devices, verify if the application is being debugged, etc.

8.9 Android: Cleartext Storage Of Sensitive Information

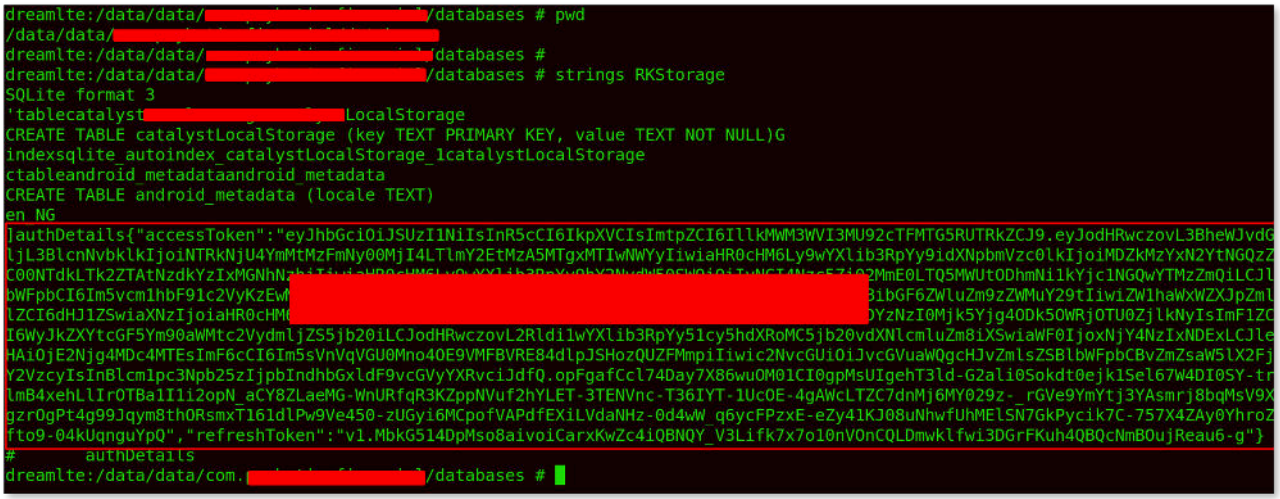
SEVERITY: LOW	**CWE-ID:** CWE-312	**CVSS SCORE:** 3.8
AFFECTED POINTS	Android	
OWASP TOP 10	M2 - Insecure Data Storage	

Description

During the assessment, Blaze Information Security found that the application stores client's session and refresh token in RKStorage in unencrypted form in a directory on Android.

Since the application executes normally on an insecure execution environment (rooted device), other installed applications may be able to access such sensitive information, among other attack scenarios such as an adversary possessing physical access, increasing the severity of this vulnerability.

The following image demonstrates the type of sensitive information stored in the files, such as the authentication token:



Reference

- https://owasp.org/www-project-mobile-top-10/2016-risks/m2-insecure-data-storage
- https://developer.android.com/training/articles/keystore
- https://cwe.mitre.org/data/definitions/312.html

Solution

Since version 4.3 (API Level 18), the Android offers a secure way to store cryptographic keys. The feature is known as Android Keystore and developers can use it to store cryptographic keys and other sensitive material in a container to make them more difficult to extract from the device.

9.0 Conclusion

The ultimate goal of a security assessment is to bring the opportunity to better illustrate the risk of an organization and help make it understand and validate its security posture against potential threats to its business. Having that in mind and into consideration, Blaze Information Security was able to conclude what follows.

The security posture of the ACME Financial Application was considered insufficient, taking into account the size of its attack surface, the number of vulnerabilities found, their corresponding impact, probability and severity, the sensitive nature of the application under the scope, and its attractiveness for potential cyber-attacks and fraud. During the security testing period, 11 vulnerabilities were found in total, where 1 of them was considered of high severity, 8 considered of medium severity, and the remaining two considered as low severity issues.

One of the vulnerabilities that were found resulted from the *Transfer Funds* functionality which allowed users to fund their ACME accounts directly from their bank account. The problem exists due to the lack of access control checks on the API endpoint exposing this feature, which allows potentially malicious users to fund their ACME account from other registered user bank accounts leading to unauthorized debit on the victim account.

Regarding the medium-severity vulnerability identified, was related to the application not enforcing certificate pinning in all the places. Certificate pinning aims essentially to reduce the risk of Man-in-the-Middle (MITM) attacks that involve the compromise of a trusted CA. The application was found to not benefit from this protection. Alternatively, other medium-severity issues were also found, such as the absence of root/jailbreak detection, the cleartext storage of sensitive information, the absence of PIN when executing a transaction, as well as the insecure direct object reference in the Purchase functionality.

Other types of vulnerabilities were also found, such as the background screen caching on iOS and Android.

With that in mind, Blaze Information Security provides the following recommendations that we believe should be adopted as the next steps to further enhance the security posture of Blaze Samples:

- Understand why the vulnerabilities were introduced and what caused them
- Implement access control verifications in every security-sensitive functionality
- Educate the development team with secure coding practices
- Perform annual black-box security testing of the application due to its level of importance to the business

- ✓ Establish a Secure Development Lifecycle (SDL) program in the software development lifecycle of the organization
- ✓ Perform regular security-focused code audits, both internally and by third parties, especially before rolling out to production major versions of the software

Blaze Information Security would like to thank the team of ACME Ltd. for their support and assistance during the entire engagement.

10.0 Appendix A - Vulnerability Criteria Classification

Below is the risk rating criteria used to classify the vulnerabilities discussed in this report:

SEVERITY	DESCRIPTION
CRITICAL	Leads to the compromise of the system and the data it handles. Can be exploited by an unskilled attacker using publicly available tools and exploits. Must be addressed immediately.
HIGH	Usually leads to the compromise of the system and the data it handles.
MEDIUM	Does not lead to the immediate compromise of the system but when chained with other issues can bring serious security risks. Nevertheless, it is advisable to fix them accordingly.
LOW	Do not pose an immediate risk and even when chained with other vulnerabilities are less likely to cause serious impact.
INFO	Does not pose an immediate risk, but requires continuous surveillance so that it doesn't become a liability through ill-use or future modifications in the system.

11.0 Appendix B - Remediation Priority Suggestion

For this assessment it was defined an order of prioritization for remediation of the vulnerabilities discovered. The criteria used to define this order took into consideration the severity and the perceived effort required to fix the issues.

The following table lists the order in which the vulnerabilities should be fixed:

SEVERITY	DESCRIPTION
HIGH	API: Insufficient Access Control Allows for Unauthorized Funds Transfer
MEDIUM	API: Absence of PIN entry to execute transactions
LOW	Android: Cleartext Storage of Sensitive Information
MEDIUM	API: Insecure Direct Object Reference (IDOR) in the Purchase functionality
MEDIUM	Android: Application does not enforce certificate pinning
MEDIUM	iOS: Application does not enforce certificate pinning
LOW	Absence of jailbreak detection
LOW	iOS: Background screen caching
LOW	Android: Background screen caching

12.0 Appendix C - Tls/Ssl Ciphers Security Scanning

Host: dev-acme-financial-service.test
Port: 443

```
 Testing protocols via sockets except NPN+ALPN

 SSLv2      not offered (OK)
 SSLv3      not offered (OK)
 TLS 1      not offered
 TLS 1.1    not offered
 TLS 1.2    offered (OK)
 TLS 1.3    offered (OK): final
 NPN/SPDY   grpc-exp, h2, http/1.1 (advertised)
 ALPN/HTTP2 h2, http/1.1, grpc-exp (offered)

 Testing cipher categories

 NULL ciphers (no encryption)                      not offered
(OK)
 Anonymous NULL Ciphers (no authentication)        not offered
(OK)
 Export ciphers (w/o ADH+NULL)                     not offered
(OK)
 LOW: 64 Bit + DES, RC[2,4], MD5 (w/o export)      not offered
(OK)
 Triple DES Ciphers / IDEA                         not offered
 Obsoleted CBC ciphers (AES, ARIA etc.)            offered
 Strong encryption (AEAD ciphers) with no FS       not offered
 Forward Secrecy strong encryption (AEAD ciphers)  offered (OK)


 Testing server's cipher preferences

 Has server cipher order?     yes (OK) -- only for < TLS 1.3
 Negotiated protocol          TLSv1.3
 Negotiated cipher            TLS_AES_256_GCM_SHA384, 253 bit
ECDH (X25519)
 Cipher per protocol

Hexcode  Cipher Suite Name (OpenSSL)       KeyExch.
Encryption  Bits     Cipher Suite Name (IANA/RFC)
-----------------------------------------------------------------
-----------------------------------------------------------
SSLv2
 -
```

```
SSLv3
 -
TLSv1
 -
TLSv1.1
 -
TLSv1.2 (server order)
 xcca8   ECDHE-RSA-CHACHA20-POLY1305       ECDH 253
ChaCha20    256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
 xc02f   ECDHE-RSA-AES128-GCM-SHA256       ECDH 253
AESGCM      128
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
 xc030   ECDHE-RSA-AES256-GCM-SHA384       ECDH 253
AESGCM      256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
 xc013   ECDHE-RSA-AES128-SHA              ECDH 253
AES         128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 xc014   ECDHE-RSA-AES256-SHA              ECDH 253
AES         256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLSv1.3 (no server order, thus listed by strength)
 x1302   TLS_AES_256_GCM_SHA384            ECDH 253
AESGCM      256
TLS_AES_256_GCM_SHA384
 x1303   TLS_CHACHA20_POLY1305_SHA256      ECDH 253
ChaCha20    256
TLS_CHACHA20_POLY1305_SHA256
 x1301   TLS_AES_128_GCM_SHA256            ECDH 253
AESGCM      128
TLS_AES_128_GCM_SHA256


 Testing robust forward secrecy (FS) -- omitting Null
Authentication/Encryption, 3DES, RC4

 FS is offered (OK)           TLS_AES_256_GCM_SHA384
                              TLS_CHACHA20_POLY1305_SHA256
                              ECDHE-RSA-AES256-GCM-SHA384 ECDHE-
RSA-AES256-SHA
                              ECDHE-RSA-CHACHA20-POLY1305
                              TLS_AES_128_GCM_SHA256
                              ECDHE-RSA-AES128-GCM-SHA256 ECDHE-
RSA-AES128-SHA
 Elliptic curves offered:     prime256v1 X25519
```

```
 Testing server defaults (Server Hello)

 TLS extensions (standard)    "renegotiation info/#65281"
                              "EC point formats/#11" "session
ticket/#35"
                              "next protocol/#13172" "key
share/#51"
                              "supported versions/#43"
                              "extended master secret/#23"
                              "application layer protocol
negotiation/#16"
 Session Ticket RFC 5077 hint 100800 seconds but: FS requires
session ticket keys to be rotated < daily !
 SSL Session ID support       yes
 Session Resumption           Tickets no, ID: no
 TLS clock skew               -218 sec from localtime
 Signature Algorithm          SHA256 with RSA
 Server key size              RSA 2048 bits (exponent is 65537)
 Server key usage             Digital Signature, Key Encipherment
 Server extended key usage    TLS Web Server Authentication
 Serial / Fingerprints        D885BA47BABAEA4A09DB44AB36174BC7 /
SHA1 3D79DBFCD88F798631E4F13F6E7E9AA157ADFA98
                              SHA256
1CCF080B5D0BE0882F373109BE61E355223199248D70F0A5850A405605369182
 Common Name (CN)             dev-acme-financial-service.test
 subjectAltName (SAN)         dev-acme-financial-service.test
 Trust (hostname)             Ok via SAN and CN (same w/o SNI)
 Chain of trust               Ok
 EV cert (experimental)       no
 Certificate Validity (UTC)   expires < 60 days (59) (2022-10-18
22:04 --> 2023-01-16 22:04)
 ETS/"eTLS", visibility info  not present
 Certificate Revocation List  "http://crls.pki.goog/gts1d4/
SuXeq4F0tuc.crl":http://crls.pki.goog/gts1d4/SuXeq4F0tuc.crl
 OCSP URI                     "http://ocsp.pki.goog/s/gts1d4/
akaGT057LNg":http://ocsp.pki.goog/s/gts1d4/akaGT057LNg
 OCSP stapling                not offered
 OCSP must staple extension   --
 DNS CAA RR (experimental)    not offered
 Certificate Transparency     yes (certificate extension)
 Certificates provided        3
 Issuer                       GTS CA 1D4 (Google Trust Services
LLC from US)
 Intermediate cert validity   #1: ok > 40 days (2027-09-30
01:00). GTS CA 1D4 <-- GTS Root R1
                              #2: ok > 40 days (2028-01-28
```

```
01:00). GTS Root R1 <-- GlobalSign Root CA
 Intermediate Bad OCSP (exp.) Ok


 Testing HTTP header response @ "/"

 HTTP Status Code               404 Not Found (Hint: supply a path
which doesn't give a "404 Not Found")
 HTTP clock skew                -216 sec from localtime
 Strict Transport Security      not offered
 Public Key Pinning             --
 Server banner                  Google Frontend
 Application banner             --
 Cookie(s)                      (none issued at "/") -- maybe
better try target URL of 30x
 Security headers               X-Content-Type-Options: nosniff
 Reverse Proxy banner           via: 1.1 google


 Testing vulnerabilities

 Heartbleed (CVE-2014-0160)                not vulnerable (OK),
no heartbeat extension
 CCS (CVE-2014-0224)                       not vulnerable (OK)
 Ticketbleed (CVE-2016-9244), experiment.  not vulnerable (OK)
 ROBOT                                     Server does not
support any cipher suites that use RSA key transport
 Secure Renegotiation (RFC 5746)           supported (OK)
 Secure Client-Initiated Renegotiation     not vulnerable (OK)
 CRIME, TLS (CVE-2012-4929)                not vulnerable (OK)
 BREACH (CVE-2013-3587)                    no gzip/deflate/
compress/br HTTP compression (OK)  - only supplied "/" tested
 POODLE, SSL (CVE-2014-3566)               not vulnerable (OK),
no SSLv3 support
 TLS_FALLBACK_SCSV (RFC 7507)              No fallback possible
(OK), no protocol below TLS 1.2 offered
 SWEET32 (CVE-2016-2183, CVE-2016-6329)    not vulnerable (OK)
 FREAK (CVE-2015-0204)                     not vulnerable (OK)
 DROWN (CVE-2016-0800, CVE-2016-0703)      not vulnerable on
this host and port (OK)
                                           make sure you don't
use this certificate elsewhere with SSLv2 enabled services
                                           "https://censys.io/
ipv4?q=1CCF080B5D0BE0882F373109BE61E355223199248D70F0A5850A405605
369182":https://censys.io/
ipv4?q=1CCF080B5D0BE0882F373109BE61E355223199248D70F0A5850A405605
369182  could help you to find out
```

```
 LOGJAM (CVE-2015-4000), experimental      not vulnerable (OK):
no DH EXPORT ciphers, no DH key detected with <= TLS 1.2
 BEAST (CVE-2011-3389)                     not vulnerable (OK),
no SSL3 or TLS1
 LUCKY13 (CVE-2013-0169), experimental     potentially
VULNERABLE, uses cipher block chaining (CBC) ciphers with TLS.
Check patches
 Winshock (CVE-2014-6321), experimental    not vulnerable (OK)
 RC4 (CVE-2013-2566, CVE-2015-2808)        no RC4 ciphers
detected (OK)


 Running client simulations (HTTP) via sockets

 Android 4.4.2                 TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Android 5.0.0                 TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Android 6.0                   TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Android 7.0 (native)          TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Android 8.1 (native)          TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 253 bit ECDH (X25519)
 Android 9.0 (native)          TLSv1.3 TLS_AES_128_GCM_SHA256,
253 bit ECDH (X25519)
 Android 10.0 (native)         TLSv1.3 TLS_AES_128_GCM_SHA256,
253 bit ECDH (X25519)
 Chrome 74 (Win 10)            TLSv1.3 TLS_AES_128_GCM_SHA256,
253 bit ECDH (X25519)
 Chrome 79 (Win 10)            TLSv1.3 TLS_AES_128_GCM_SHA256,
253 bit ECDH (X25519)
 Firefox 66 (Win 8.1/10)       TLSv1.3 TLS_AES_128_GCM_SHA256,
253 bit ECDH (X25519)
 Firefox 71 (Win 10)           TLSv1.3 TLS_AES_128_GCM_SHA256,
253 bit ECDH (X25519)
 IE 6 XP                       No connection
 IE 8 Win 7                    No connection
 IE 8 XP                       No connection
 IE 11 Win 7                   TLSv1.2 ECDHE-RSA-AES128-SHA, 256
bit ECDH (P-256)
 IE 11 Win 8.1                 TLSv1.2 ECDHE-RSA-AES128-SHA, 256
bit ECDH (P-256)
 IE 11 Win Phone 8.1           TLSv1.2 ECDHE-RSA-AES128-SHA, 256
bit ECDH (P-256)
 IE 11 Win 10                  TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
```

```
 Edge 15 Win 10                TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 253 bit ECDH (X25519)
 Edge 17 (Win 10)              TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 253 bit ECDH (X25519)
 Opera 66 (Win 10)             TLSv1.3 TLS_AES_128_GCM_SHA256,
253 bit ECDH (X25519)
 Safari 9 iOS 9                TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Safari 9 OS X 10.11           TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Safari 10 OS X 10.12          TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Safari 12.1 (iOS 12.2)        TLSv1.3
TLS_CHACHA20_POLY1305_SHA256, 253 bit ECDH (X25519)
 Safari 13.0 (macOS 10.14.6)   TLSv1.3
TLS_CHACHA20_POLY1305_SHA256, 253 bit ECDH (X25519)
 Apple ATS 9 iOS 9             TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Java 6u45                     No connection
 Java 7u25                     No connection
 Java 8u161                    TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Java 11.0.2 (OpenJDK)         TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 Java 12.0.1 (OpenJDK)         TLSv1.3 TLS_AES_128_GCM_SHA256,
256 bit ECDH (P-256)
 OpenSSL 1.0.2e                TLSv1.2 ECDHE-RSA-AES128-GCM-
SHA256, 256 bit ECDH (P-256)
 OpenSSL 1.1.0l (Debian)       TLSv1.2 ECDHE-RSA-
CHACHA20-POLY1305, 253 bit ECDH (X25519)
 OpenSSL 1.1.1d (Debian)       TLSv1.3 TLS_AES_256_GCM_SHA384,
253 bit ECDH (X25519)
 Thunderbird (68.3)            TLSv1.3 TLS_AES_128_GCM_SHA256,
253 bit ECDH (X25519)


 Rating (experimental)

 Rating specs (not complete)  SSL Labs's 'SSL Server Rating
Guide' (version 2009q from 2020-01-30)
 Specification documentation  "https://github.com/ssllabs/
research/wiki/SSL-Server-Rating-Guide":https://github.com/
ssllabs/research/wiki/SSL-Server-Rating-Guide
 Protocol Support (weighted)  100 (30)
 Key Exchange     (weighted)  90 (27)
 Cipher Strength  (weighted)  90 (36)
 Final Score                  93
```

```
 Overall Grade                A
 Grade cap reasons            Grade capped to A. HSTS is not
offered.
```

13.0 Appendix C - Port Scanning Report

Below we present the port scanning reports using **Nmap**, for TCP and UDP protocols. For the TCP protocol, all ports were analyzed (1 to 65535), and for the UDP protocol, only the most common ports were scanned (1 to 1024).

Host: dev-acme-financial-service.test
Protocol: TCP

```
Starting Nmap 7.92 ( "https://nmap.org":https://nmap.org  ) at
2022-11-18 19:22 WAT
Nmap scan report for dev-acme-financial-service.test
(11.22.33.44)
Host is up (0.16s latency).
rDNS record for 11.22.33.44
Not shown: 65531 filtered tcp ports (no-response), 2 filtered
tcp ports (net-unreach)
PORT    STATE SERVICE    VERSION
80/tcp  open  tcpwrapped
443/tcp open  ssl/https  Google Frontend
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 404 Not Found
|     content-type: text/plain; charset=utf-8
|     vary: Origin
|     x-content-type-options: nosniff
|     x-cloud-trace-context: 99138a55f8f4a7cc84761d3d6e2e5f0d
|     date: Fri, 18 Nov 2022 18:33:50 GMT
|     server: Google Frontend
|     Content-Length: 19
|     via: 1.1 google
|     Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
|     page not found
|   GetRequest:
|     HTTP/1.0 404 Not Found
|     content-type: text/plain; charset=utf-8
|     vary: Origin
|     x-content-type-options: nosniff
|     x-cloud-trace-context: 4f630da1ccd6a8e6911765caf4d56b60;o=1
|     date: Fri, 18 Nov 2022 18:33:49 GMT
```

```
|     server: Google Frontend
|     Content-Length: 19
|     via: 1.1 google
|     Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
|     page not found
|   HTTPOptions:
|     HTTP/1.0 404 Not Found
|     content-type: text/plain; charset=utf-8
|     vary: Origin
|     x-content-type-options: nosniff
|     x-cloud-trace-context: b65815b0faed9b245d06bb24a952f815
|     date: Fri, 18 Nov 2022 18:33:49 GMT
|     server: Google Frontend
|     Content-Length: 19
|     via: 1.1 google
|     Alt-Svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
|     page not found
|   RTSPRequest:
|     HTTP/1.0 400 Bad Request
|     Content-Type: text/html; charset=UTF-8
|     Referrer-Policy: no-referrer
|     Content-Length: 273
|     Date: Fri, 18 Nov 2022 18:33:55 GMT
|     <html><head>
|     <meta http-equiv="content-type" content="text/
html;charset=utf-8">
|     <title>400 Bad Request</title>
|     </head>
|     <body text=#000000 bgcolor=#ffffff>
|     <h1>Error: Bad Request</h1>
|     <h2>Your client has issued a malformed or illegal
request.</h2>
|     <h2></h2>
|     </body></html>
|   tor-versions:
|     HTTP/1.0 400 Bad Request
|     Content-Length: 54
|     Content-Type: text/html; charset=UTF-8
|     Date: Fri, 18 Nov 2022 18:33:50 GMT
|_    <html><title>Error 400 (Bad Request)!!1</title></html>
|_ssl-date: TLS randomness does not represent time
|_http-title: Site doesn't have a title (text/plain;
charset=utf-8).
|_http-cors: GET POST OPTIONS
|_http-server-header: Google Frontend
| ssl-cert: Subject: commonName=dev-acme-financial-service.test
| Subject Alternative Name: DNS:dev-acme-financial-service.test
```

```
| Not valid before: 2022-10-18T21:04:17
|_Not valid after:  2023-01-16T21:04:16
1 service unrecognized despite returning data. If you know the
service/version, please submit the following fingerprint at
"https://nmap.org/cgi-bin/submit.cgi?new-
service":https://nmap.org/cgi-bin/submit.cgi?new-service  :
SF-Port443-TCP:V=7.92%T=SSL%I=7%D=11/
18%Time=6377D0E4%P=x86_64-pc-linux-gn
SF:u%r(GetRequest,15E,"HTTP/1\.0\x20404\x20Not\x20Found\r\
ncontent-type:\x
SF:20text/plain;\x20charset=utf-8\r\nvary:\x20Origin\r\nx-
content-type-opt
SF:ions:\x20nosniff\r\nx-cloud-trace-
context:\x204f630da1ccd6a8e6911765caf
SF:4d56b60;o=1\r\ndate:\x20Fri,\x2018\x20Nov\x202022\x2018:33:49\
x20GMT\r\
SF:nserver:\x20Google\x20Frontend\r\nContent-Length:\x2019\r\
nvia:\x201\.1
SF:\x20google\r\nAlt-
Svc:\x20h3=\":443\";\x20ma=2592000,h3-29=\":443\";\x2
SF:0ma=2592000\r\n\r\n404\x20page\x20not\x20found\
n")%r(HTTPOptions,15A,"H
SF:TTP/1\.0\x20404\x20Not\x20Found\r\ncontent-type:\x20text/
plain;\x20char
SF:set=utf-8\r\nvary:\x20Origin\r\nx-content-type-
options:\x20nosniff\r\nx
SF:-cloud-trace-context:\x20b65815b0faed9b245d06bb24a952f815\r\
ndate:\x20F
SF:ri,\x2018\x20Nov\x202022\x2018:33:49\x20GMT\r\
nserver:\x20Google\x20Fro
SF:ntend\r\nContent-Length:\x2019\r\nvia:\x201\.1\x20google\r\
nAlt-Svc:\x2
SF:0h3=\":443\";\x20ma=2592000,h3-29=\":443\";\x20ma=2592000\r\n\
r\n404\x2
SF:0page\x20not\x20found\n")%r(FourOhFourRequest,15A,"HTTP/1\.0\
x20404\x20
SF:Not\x20Found\r\ncontent-type:\x20text/plain;\x20charset=utf-8\
r\nvary:\
SF:x20Origin\r\nx-content-type-options:\x20nosniff\r\nx-cloud-
trace-contex
SF:t:\x2099138a55f8f4a7cc84761d3d6e2e5f0d\r\ndate:\x20Fri,\x2018\
x20Nov\x2
SF:02022\x2018:33:50\x20GMT\r\nserver:\x20Google\x20Frontend\r\
nContent-Le
SF:ngth:\x2019\r\nvia:\x201\.1\x20google\r\nAlt-
Svc:\x20h3=\":443\";\x20ma
SF:=2592000,h3-29=\":443\";\x20ma=2592000\r\n\r\n404\x20page\
```

```
x20not\x20fou
SF:nd\n")%r(tor-versions,B3,"HTTP/1\.0\x20400\x20Bad\x20Request\
r\nContent
SF:-Length:\x2054\r\nContent-Type:\x20text/
html;\x20charset=UTF-8\r\nDate:
SF:\x20Fri,\x2018\x20Nov\x202022\x2018:33:50\x20GMT\r\n\r\
n<html><title>Er
SF:ror\x20400\x20\(Bad\
x20Request\)!!1</title></html>")%r(RTSPRequest,1AD,
SF:"HTTP/1\.0\x20400\x20Bad\x20Request\r\nContent-Type:\x20text/
html;\x20c
SF:harset=UTF-8\r\nReferrer-Policy:\x20no-referrer\r\nContent-
Length:\x202
SF:73\r\nDate:\x20Fri,\x2018\x20Nov\x202022\x2018:33:55\x20GMT\r\
n\r\n\n<h
SF:tml><head>\n<meta\x20http-equiv=\"content-
type\"\x20content=\"text/html
SF:;charset=utf-8\">\n<title>400\x20Bad\
x20Request</title>\n</head>\n<body
SF:\x20text=#000000\x20bgcolor=#ffffff>\n<h1>Error:\x20Bad\
x20Request</h1>
SF:\n<h2>Your\x20client\x20has\x20issued\x20a\x20malformed\x20or\
x20illega
SF:l\x20request\.</h2>\n<h2></h2>\n</body></html>\n");
```

Host: dev-acme-financial-service.test
Protocol: UDP

```
Starting Nmap 7.92 ( "https://nmap.org":https://nmap.org  ) at
2022-11-18 20:18 WAT
Nmap scan report for dev-acme-financial-service.test
(11.22.33.44)
Host is up (0.035s latency).
rDNS record for 11.22.33.44: 11.33.44.55.googleusercontent.com
Not shown: 999 closed udp ports (port-unreach)
PORT    STATE         SERVICE
443/udp open|filtered https

Nmap done: 1 IP address (1 host up) scanned in 3.47 seconds
```

BLAZE
INFORMATION SECURITY

BRAZIL

+55 81 3071.7148
R. VISCONDE DE JEQUITINHONHA, 279,
EMPRESARIAL TANCREDO NEVES, OFFICE
701, RECIFE, PERNAMBUCO

PORTUGAL

+351 222 081 647
PRAÇA DO BOM SUCESSO, 131 - OFFICE 206,
4150-146, PORTO

WWW.BLAZEINFOSEC.COM
INFO@BLAZEINFOSEC.COM