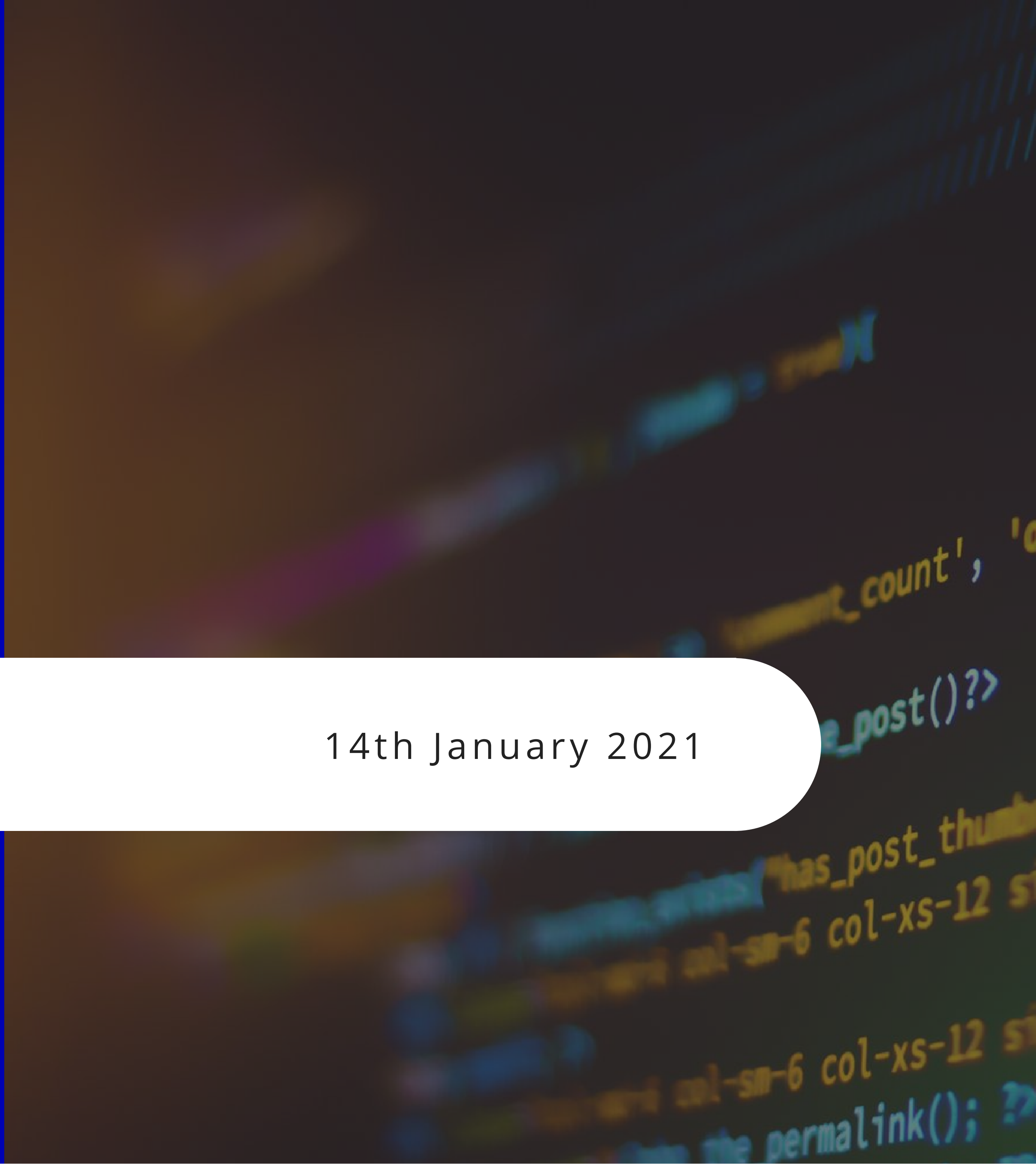
APP ASSESSMENT



VBANK

14th January 2021

verimatrix
DRIVING TRUST

OVERVIEW

Verimatrix was asked by V BANK to perform a shallow dive security analysis on the V Bank Retail Banking application

https://play.google.com/store/apps/details?id=not.on.play.store

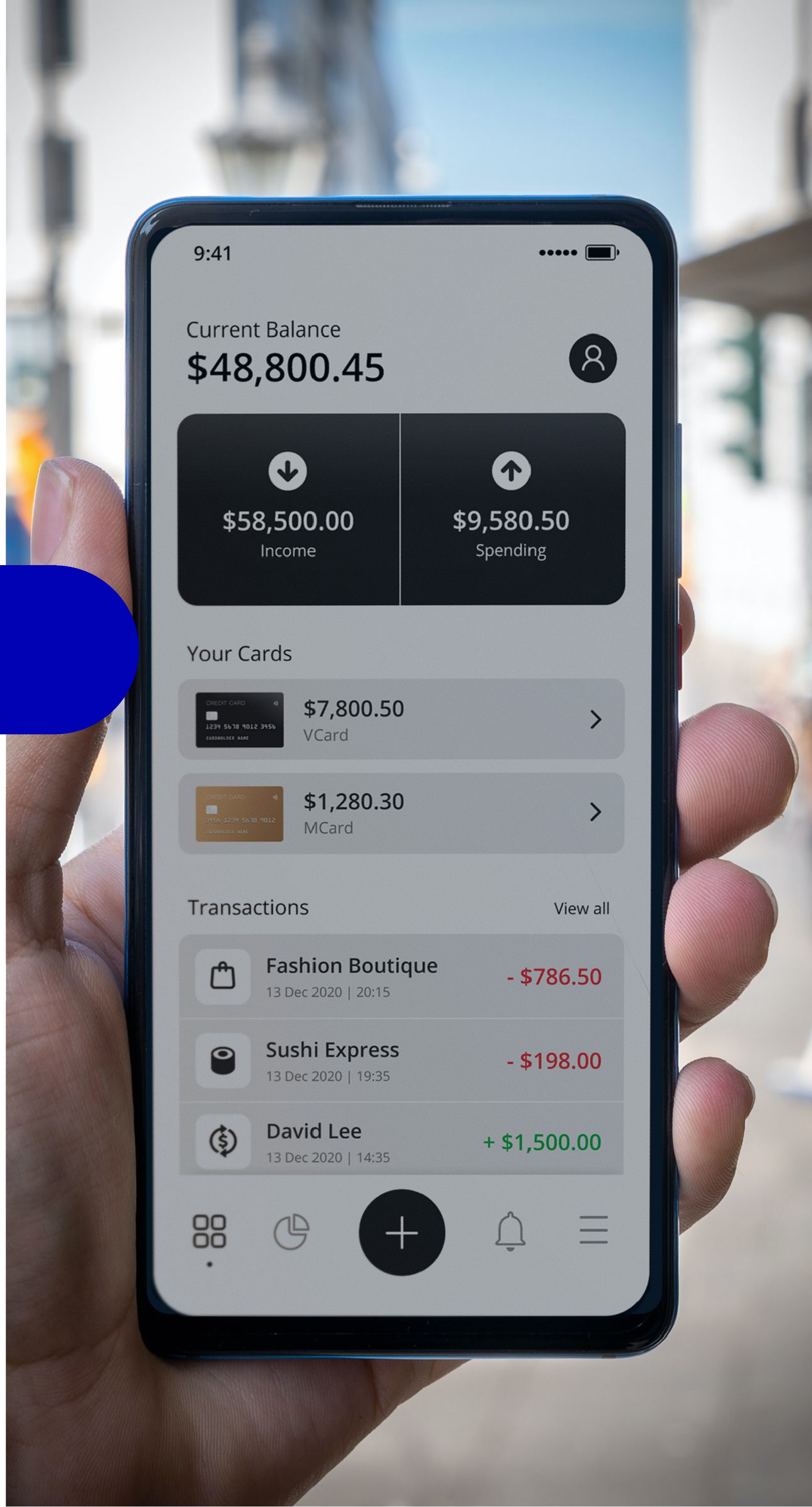
The analysis was carried out as to the Terms and Conditions found **here** and agreed to by Bob Smith on 4th of January 2021.

This document captures the methodology and findings.

The scope of the review was a “shallow dive” investigation limited to 240 minutes.

The environment used for testing: rooted and non-rooted Nexus 4 with Android 4.4 and 5.0.

verimatrix
DRIVING TRUST



METHODOLOGY

The application was downloaded from Google Play and analyzed statically & dynamically using freely available tools.

Static analysis means that the application code was observed but was not executed.

Dynamic analysis means that the application code was executed, modified and penetration test performed.

verimatrix

DRIVING TRUST

The analysis looked for the presence of standard security measures. These are an indication of how exploitable a vulnerability would be an attacker.

To be clear, the analysis did not look for vulnerabilities within the app code. The complexity of modern mobile apps means that there is a “guaranteed vulnerability” in their code.

The analysis also did not try to breach any back-end systems. It was focused solely on the mobile application.

Once analysed, the apps were ranked against the Verimatrix scale.

GRADE

V Bank’s Android application scored a

D

GRADE

APPENDIX – VERIMATRIX SCALE[1]

A

HIGHLY SECURE

- All code handling sensitive data and algorithms is developed in a language that compiles to processor native machine code (i.e. C/C++)
- Strong control flow obfuscation[2] of the majority of code including all business logic
- No sensitive data (including cryptographic keys) visible in static analysis of code
- Cryptographic keys protected by whitebox[3] (or equivalent technology)
- Network traffic encrypted using TLS 1.3[4] and downgrade not possible
- Certificate pinning[5] applied to networking
- Unable to attach a debugger or hooking framework to application (either on start-up or at any time while executing)
- Application preventing from running under emulation or virtual machine
- Application signed as required by target OS
- Application resigning prevented
- Anti-tamper[6] protection of the application package and code

B

SECURE

- Strong control flow obfuscation of the majority of code including all business logic
- No sensitive data (including cryptographic keys) visible in static analysis of code
- Network traffic encrypted using TLS 1.3 and downgrade not possible
- Certificate pinning applied to networking
- Unable to attach a debugger or hooking framework to application (either on start-up or at any time while executing)
- Application preventing from running under emulation or virtual machine
- Application signed as required by target OS
- Application resigning prevented
- Anti-tamper protection of the application package and code

verimatrix DRIVING TRUST

[1] The Verimatrix scale is an updated to the grading system proposed by UL and Verimatrix here
[2] Obfuscation means scrambling computer code to make it less-intelligible to a human.
[3] Whitebox technology protects cryptographic operations and keys.
[4] TLS (Transport Layer Security) is the standard encryption protocol of the internet.
[5] Certificate pinning validates that the end point of communication is the intended end point.
[6] Anti-tamper technology provides a means to ensure the code being run is the intended code.

APPENDIX – VERIMATRIX SCALE CONTINUED

C

STANDARD

- Control flow obfuscation of all business logic
- No sensitive data (including cryptographic keys) visible in static analysis of code
- Network traffic encrypted using TLS 1.3 and downgrade not possible
- Certificate pinning applied to networking
- Unable to start application with debugger or hooking framework attached
- Application signed as required by target OS
- Application resigning prevented

D

BASIC

- Symbol obfuscation of business logic
- Network traffic encrypted
- Application signed as required by target OS

E

LITTLE OR NO SECURITY

- None

verimatrix™
DRIVING TRUST

FINDINGS: APPLICATION

6054 classes
Bytecode

1 lib
NDK Code

34/100
MobSF Security Score[1]

6.0/10
CVSS Rating[2]

Security Measure	Finding	Risk
APK is signed	V1 signature: True V2 signature: True V3 signature: False	Application is signed with v1 signature scheme, making it vulnerable to Janus vulnerability on Android <7.0.
Resigning protection	None	Application can be resigned by an attacker allowing them to repackage the application.
Protection against malicious code insertion	None	Application can be repackaged with malware inserted or security measures removed.
Network traffic encryption	TLS 1.3	

[1] https://mobsf.github.io/Mobile-Security-Framework-MobSF/
[2] https://nvd.nist.gov/vuln-metrics/cvss

FINDINGS: APPLICATION CONTINUED

Security Measure	Finding	Risk
Prohibit network protocol downgrade	Not prevent	Possible for an attacker to remotely make the application use a less secure version of TLS.
Network traffic pinned	No	Man in the middle attack
Detect rooted device	Shows warning but continues to execute	Running on a rooted device can be considered higher risk but does depend on the security policy of the app owner.
Stop debugger attaching	Debugger can attach	Attaching a debugger enables an attacker to dynamical analyze the application.
Prevent running under emulation or virtual machine	Executes under emulation and VM	Running on an emulator enables an attacker to dynamical analyze the application and to run an attack at scale.
Prevent application being traced with Frida	Frida can trace and control executing app	Utilizing a hooking framework enables an attacker to dynamical analyze the application.

verimatrix™
DRIVING TRUST

FINDINGS: BYTECODE

Security Measure	Finding	Risk
Obfuscation	Symbol obfuscation - Proguard	Weak or no obfuscation makes it easy for an attacker to statically analyze the code.
Obfuscate sensitive values in code	Yes (see below)	Private data can be found through static analysis.
Binary integrity checks	No	An attacker can modify the code as they desire.
Do not expose cryptographic keys	Visible in code	Exposed cryptographic keys can be used to expose encrypted data.

Bytecode - Possible sensitive values visible in code
“token” : “o4rq66ns23qr”
“google_api_key” : “a34a633a165357cs...”
“authentication_salt” : “jfd9qdhjsa93ej3l”

Bytecode - Possible cryptographic keys visible in code
“storage_aes_key” : “a356d23abe9f5bb2582e2d7653ee5b89”

FINDINGS: NDK CODE

Security Measure	Finding	Risk
Obfuscation	None	Symbol obfuscation not applicable to NDK code but absence of control flow obfuscation makes it easier for an attacker to statically analyze the code.
Obfuscate sensitive values in code	None found in code	
Binary integrity checks	None	An attacker can modify the code as they desire.
Do not expose cryptographic keys	None found in code	

DATA PRIVACY

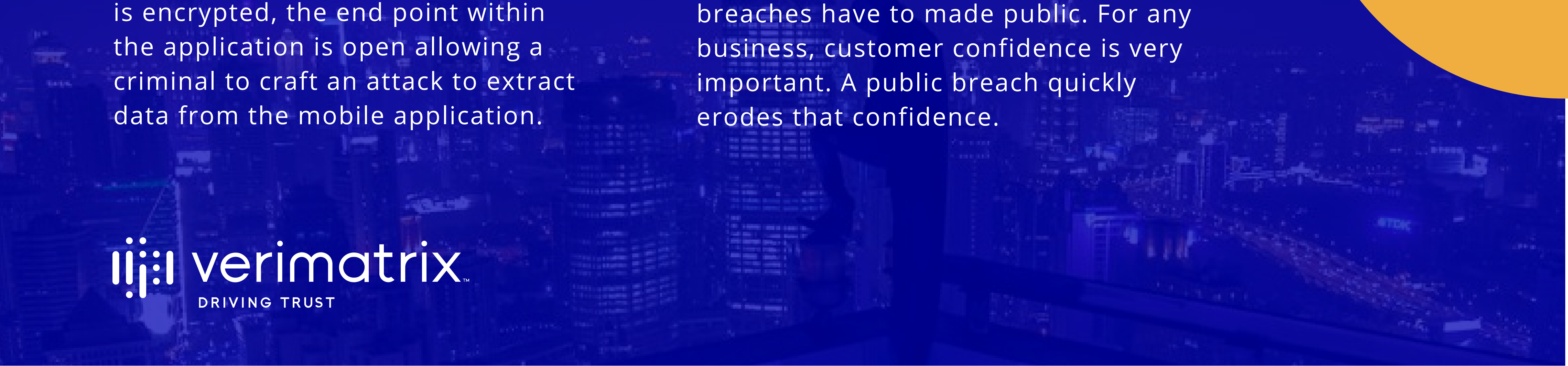
An application scoring a D grade would typically be vulnerable to personal data theft.

The application does not have the necessary protections that stop an attacker find what data the application is processing, how it is processed and how it transmitted and stored. While the network traffic is encrypted, the end point within the application is open allowing a criminal to craft an attack to extract data from the mobile application.

This means there is a risk of penalties under privacy legislation such as GDPR. It should also be noted that most data privacy regulations put a duty to disclose breach on the data controller. This means that any breaches have to made public. For any business, customer confidence is very important. A public breach quickly erodes that confidence.

D

GRADE



IMPROVEMENTS

Through this short review, it can be seen that the application would benefit from **more powerful obfuscation** including control flow. This would make it much harder for an attacker to recover meaningful and readable source code.

More powerful environmental checks can be used to thwart attempts to attach debuggers or to otherwise observe the application running; as well as remove the ability for users to create repackaged versions of the app that circumvent the root checks.

Anti-tamper technology would stop security and other features in the application from being removed by an attacker. It would also inhibit an attacker from creating a repackaged version of the application.

Care should also be taken to ensure that communication is correctly configured, and network connections are pinned.

If these changes are made, the application would grade substantially higher than it was assessed in this review.

As banking applications handle sensitive personal and financial data, as well as connecting to wider banking infrastructure through APIs, Verimatrix would recommend that consideration is given to a deeper dive Vulnerability Assessment or App Security Audit.

D

GRADE

verimatrix™
DRIVING TRUST