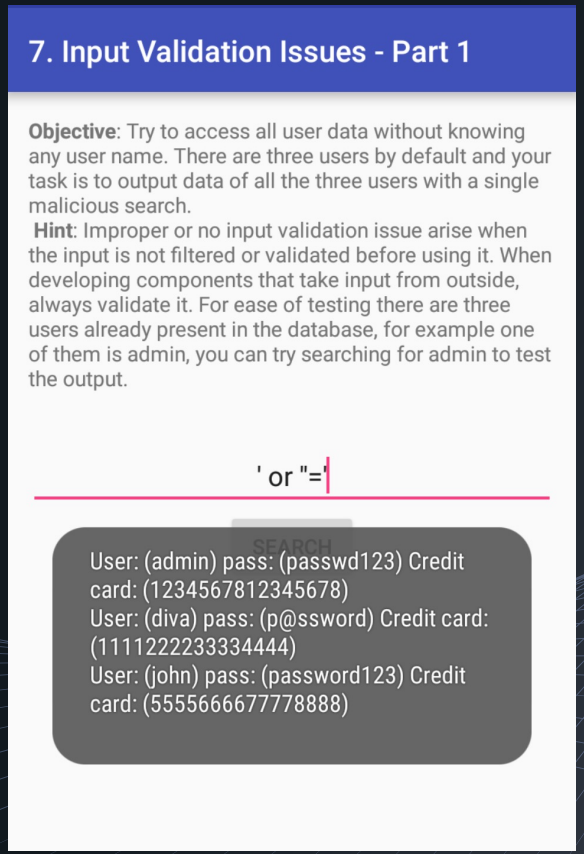
M7 – Poor Code Quality.

M7 - Poor Code Quality

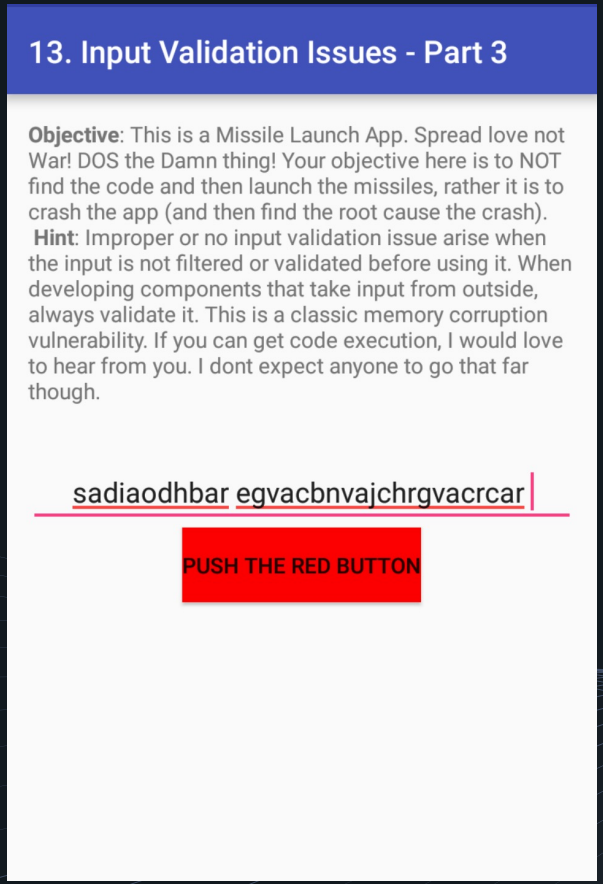
Jenis kerentanan ini lebih sulit untuk dieksploitasi dan biasanya merupakan hasil dari praktik pemrograman yang buruk dalam aplikasi seluler. Serangan yang dilakukan untuk mengeksploitasi *code quality* mungkin diantaranya buffer overflows misalnya. namun seringkali memerlukan alat khusus untuk mengidentifikasi.

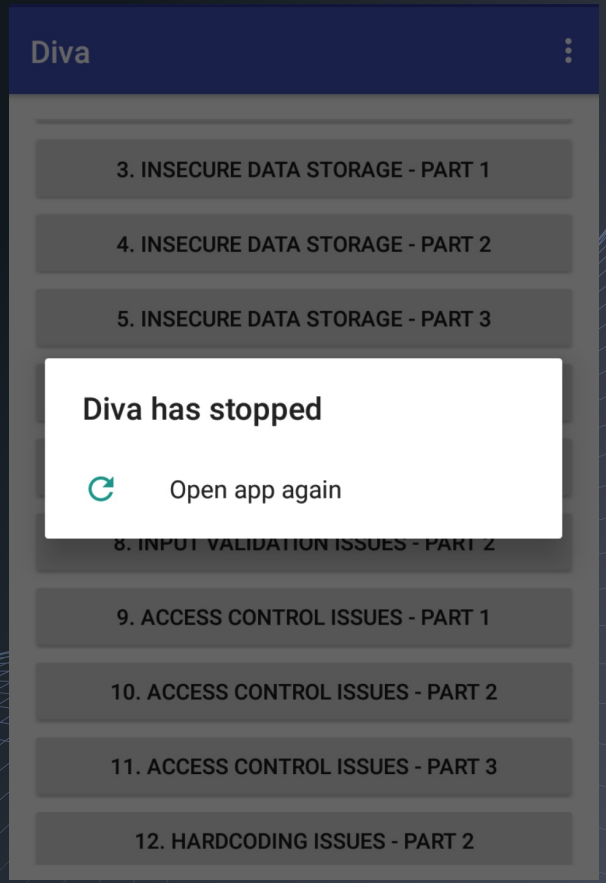
Kerentanan ini dapat dicegah dengan menerapkan praktik pemrograman yang baik dan standar seperti dokumentasi yang terorganisir. pola pengkodean yang konsisten. dan menggunakan alat analisis kode statis untuk memvalidasi penyimpanan buffer.

M7 - Poor Code Quality



M7 - Poor Code Quality





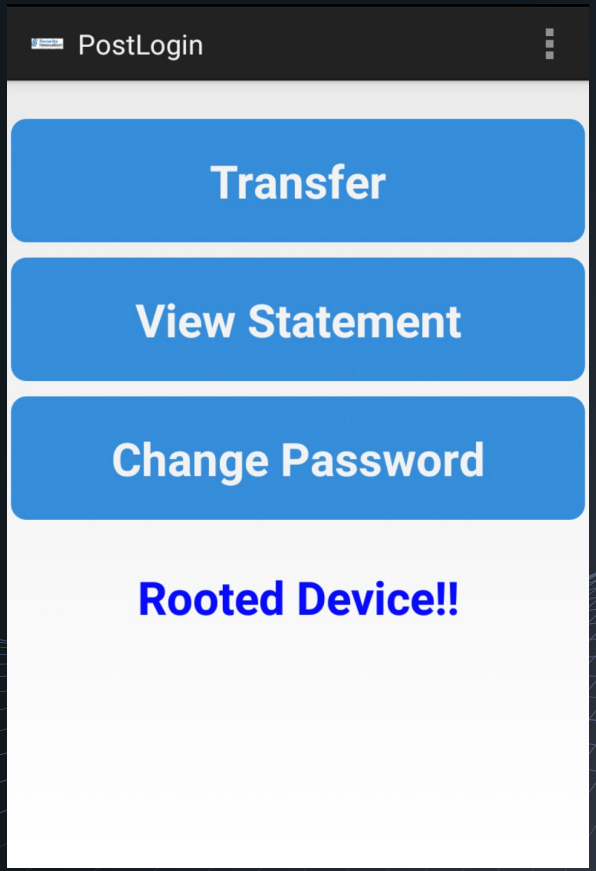
M8 - Code Tampering.

M8 - Code Tampering

Musuh dapat mengeksploitasi *code tampering* menggunakan aplikasi seluler yang ada di toko / situs pihak ketiga. Melakukannya dapat menyebabkan penambahan fitur berbahaya dan perubahan atau modifikasi pada sumber daya dan data aplikasi.

Mendeteksi kerusakan kode adalah kunci untuk mencegahnya. Ini dapat dilakukan dengan mengintegrasikan sistem yang mendeteksi segala perubahan pada kode aplikasi dan mengimplementasikan solusi reaktif jika terdeteksi.

M8 - Code Tampering



M9 - Reverse Engineering.

M9 - Reverse Engineering

Jika penyerang dapat menganalisis kode aplikasi dan membedahnya dengan menggunakan alat khusus di dalam lab mereka sendiri. dapat ditemukan bahwa aplikasi seluler rentan terhadap rekayasa balik. Jika kerentanan dapat dieksploitasi. hal itu dapat menyebabkan pengungkapan informasi lebih lanjut dan serangan terhadap sistem backend.

Seringkali alat kebingungan digunakan untuk mencegah musuh melakukan reverse engineering dan melanjutkan serangan yang meningkatkan kerentanan ini.

M10 - Extraneous Functionality.

M10 - Extraneous Functionality

Dengan juga memeriksa aplikasi mobile secara menyeluruh melalui konfigurasi dan file log. musuh mungkin dapat mengeksploitasi fungsionalitas asing di backend dan seluruh aplikasi. Ini dapat mengekspos bagaimana aplikasi benar-benar berfungsi dan berpotensi menghasilkan eksekusi anonim dari tindakan istimewa.

Dampak dari kerentanan ini dapat dikurangi melalui tinjauan kode manual sebelum rilis aplikasi. Menganalisis titik akhir API dan memeriksa file dan log juga akan membantu memastikan bahwa fungsi tersembunyi lebih sulit ditemukan dan dieksploitasi.

TERIMA KASIH

https://github.com/rootbakar/Android-Pentesting/blob/master/full-reference-link-tutorial.md