Oficina de Testes de Segurança em Aplicações Android

Leonardo Oliveira Silva

Agenda

- OWASP
  - OWASP *Mobile Top* 10
  - Projeto OWASP MAS
- Ferramentas
  - MobSF
  - adb
  - *Burp Suite*
- Prática
  - Prática 1 - análise de *logs*
  - Prática 2 - análise do armazenamento de dados
  - Prática 3 - análise da comunicação de rede

Open Web Application Security Project (OWASP)

A OWASP é um projeto de código aberto que visa estudar e prevenir as diversas vulnerabilidades possíveis em software, disponibilizando metodologia, documentação e tecnologia apropriadas para cumprir o seu objetivo.

A comunidade é formada por adeptos da área de segurança da informação no mundo inteiro.

OWASP *Mobile Top* 10

Identifica e lista as dez principais vulnerabilidades encontradas em aplicativos móveis.

Esses riscos de vulnerabilidades de aplicativos foram determinados pela equipe do projeto a partir de diversas fontes, incluindo **relatórios de incidentes**, **bancos de dados de vulnerabilidades** e **avaliações de segurança**.

OWASP *Mobile Top* 10

1. ***Improper Credential Usage* (Uso Incorreto de Credenciais)**
   Este risco ocorre quando as credenciais de autenticação (por exemplo, senhas, tokens de acesso) são utilizadas de forma inadequada, expondo-as a ataques.

2. ***Inadequate Supply Chain Security* (Segurança Inadequada da Cadeia de Fornecimento)**
   Refere-se a vulnerabilidades introduzidas no aplicativo móvel devido a falhas na cadeia de fornecimento, como bibliotecas de terceiros não confiáveis ou componentes maliciosos.

OWASP *Mobile Top* 10

3. **Insecure Authentication/Authorization (Autenticação/Autorização Insegura)**
   Este risco envolve práticas de autenticação ou autorização fracas, como senhas fracas, falta de autenticação multifatorial ou falhas na validação de permissões.

4. ***Insufficient Input/Output Validation* (Validação Insuficiente de Entrada/Saída)**
   Refere-se à falha na validação adequada dos dados de entrada e saída do aplicativo, o que pode levar a vulnerabilidades como injeção de código, XSS (Cross-Site Scripting) ou manipulação de dados.

OWASP *Mobile Top* 10

5. ***Insecure Communication* (Comunicação Insegura)**
   Envolve a transmissão não segura de dados sensíveis entre o aplicativo móvel e servidores, tornando-os vulneráveis a interceptação por terceiros.
6. **Inadequate Privacy Controls (Controles de Privacidade Inadequados)**
   Refere-se à falta de proteção da privacidade dos dados do usuário no aplicativo móvel, como coleta excessiva de dados pessoais ou compartilhamento inadequado de informações.

OWASP *Mobile Top* 10

7. **Insufficient Binary Protections (Proteções Binárias Insuficientes)**
   Este risco envolve a falta de proteções adequadas no código binário do aplicativo, tornando-o vulnerável a técnicas de análise e exploração por parte de atacantes.
8. **Security Misconfiguration (Configuração de Segurança Incorreta)**
   Refere-se à configuração inadequada de recursos de segurança no aplicativo móvel, deixando-o vulnerável a ataques como acesso não autorizado ou vazamento de dados.

OWASP *Mobile Top* 10

9. **Insecure Data Storage (Armazenamento Inseguro de Dados)**
   Envolve o armazenamento inadequado de dados sensíveis no dispositivo móvel, como credenciais de login, informações pessoais e dados confidenciais.

10. **Insufficient Cryptography (Criptografia Insuficiente)**
    Refere-se ao uso inadequado ou insuficiente de técnicas de criptografia para proteger dados sensíveis armazenados ou transmitidos pelo aplicativo.

Projeto OWASP MAS

Um projeto interessante da OWASP é o *Mobile Application Security* (MAS), que fornece uma avaliação de segurança bastante completa e com resultados convincentes em relação a aplicativos móveis.

Esse projeto é formado pelos guias *Mobile Application Security Verification Standard* (MASVS) e *Mobile Application Security Testing Guide* (MASTG).

MASVS

O MASVS estabelece requisitos básicos de segurança para aplicativos móveis que são úteis em muitos cenários.

Ele pode ser usado:

- como métrica - para fornecer um padrão de segurança com o qual os aplicativos móveis existentes possam ser comparados por desenvolvedores e proprietários de aplicativos;

MASVS

- como orientação - para fornecer orientação durante todas as fases de desenvolvimento e teste de aplicativos móveis;
- durante a aquisição - para fornecer uma linha de base para a verificação de segurança do aplicativo móvel.

MASTG

O MASTG é um manual abrangente para testes de segurança de aplicativos móveis e engenharia reversa para testadores de segurança móvel iOS e Android, apresentando o seguinte conteúdo:

- Testes de segurança no ciclo de vida de desenvolvimento de aplicativos móveis;
- Testes básicos de segurança estática e dinâmica;
- Engenharia reversa e adulteração de aplicativos móveis;

MASTG

- Avaliação de proteções de software;
- Casos de teste detalhados que mapeiam os requisitos no MASVS.

Ferramentas

MobSF

O Mobile Security Framework (MobSF) é uma ferramenta automatizada para pen-testing, análise de malware e estrutura de avaliação de segurança capaz de realizar análises estáticas e dinâmicas.

O MobSF suporta binários de aplicativos móveis (APK, XAPK, IPA e APPX) juntamente com código-fonte compactado e fornece APIs REST para integração com seu pipeline de CI/CD ou DevSecOps.

adb

O *Android Debug Bridge* (adb) é uma ferramenta de linha de comando versátil que permite a comunicação com um dispositivo. O comando adb facilita uma variedade de ações do dispositivo, como instalar e depurar apps, e fornece acesso a um *shell Unix* que pode ser usado para executar diversos comandos em um dispositivo.

Fornecido com o pacote Android SDK, preenche a lacuna entre seu ambiente de desenvolvimento local e um dispositivo Android conectado. Você geralmente o aproveita para testar aplicativos no emulador ou em um dispositivo conectado via USB ou Wi-Fi.

Burp Suite

O *Burp Suite* é uma plataforma integrada utilizada para realizar testes de segurança em aplicativos móveis e Web.

Ela contém diversas ferramentas para mapeamento de vulnerabilidades, como o *Burp Proxy*, que funciona como um *proxy* na Web e é posicionado entre o servidor e o cliente que, no caso, é o dispositivo Android e as aplicações contidas nele.

Prática!!!

Antes da prática

- Repositório com materiais
  - https://drive.google.com/drive/folders/11bK9V-Ay8_PNAElFa-Xo6UKJuZFhj-cX?usp=sharing
- Definir emulador
  - Android Studio ➔ Virtual Device Manager ➔ Create Virtual Device
- Iniciar emulador pela linha de comando
  - cd C:\Users\<nome_usuario>\AppData\Local\Android\Sdk\emulator
  - .\emulator.exe -avd nome_avd
  - .\emulator.exe -avd nome_avd -writable-system

Prática 1 - análise de *logs*

Objetivo: Analisar se alguma informação referente ao aplicativo está presente no log de mensagens de forma insegura.

Requisito do MASVS envolvido:

- **MASVS-STORAGE-2 -** O aplicativo evita o vazamento de dados confidenciais.

Teste do MASTG para averiguar o requisito:

- **MASTG-TEST-0003 -** Testando logs para dados confidenciais

Prática 1 - análise de *logs*

Aplicação utilizada:

- diva.apk

Ferramenta(s) utilizada(s):

- MobSF (análise estática)
- adb (análise dinâmica)

Prática 1 - análise de *logs*

MobSF

- Executar instância Docker
  - docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest
  - Acessar http://127.0.0.1:8000 no navegador
  - Fazer upload do arquivo .apk

Prática 1 - análise de *logs*

adb

Com o adb, você pode inspecionar facilmente o *log* de mensagens do sistema com o utilitário *Logcat*. O *Logcat* faz parte do *Dalvik Debug Monitor Server* (DDMS) no *Android Studio*.

- Acessar os *logs* do sistema
  - .\adb.exe logcat (Windows) ou ./adb logcat (Linux)

Prática 2 - análise do armazenamento de dados

Objetivo: Analisar as formas de armazenamento de dados da aplicação e verificar se os dados estão seguros.

Requisito do MASVS envolvido:

- **MASVS-STORAGE-1** - O aplicativo armazena dados confidenciais com segurança

Teste do MASTG para averiguar o requisito:

- **MASTG-TEST-0001** - Testando armazenamento local para dados confidenciais

Prática 2 - análise do armazenamento de dados

Aplicação utilizada:

- diva.apk

Ferramenta(s) utilizada(s):

- adb (análise dinâmica)

Prática 2 - análise do armazenamento de dados

Principais formas de armazenamento no SO Android:

- Preferências compartilhadas (*Shared Preferences*);
- Banco de dados SQLite;
- Armazenamento interno;
- Armazenamento externo.

Prática 2 - análise do armazenamento de dados

Preferências compartilhadas (*Shared Preferences*)

A API *SharedPreferences* é comumente usada para salvar permanentemente pequenas coleções de pares chave-valor.

Os dados armazenados em um objeto *SharedPreferences* são gravados em um arquivo XML de texto simples.

O objeto *SharedPreferences* pode ser declarado publicamente legível (acessível a todos os aplicativos) ou privado.

Prática 2 - análise do armazenamento de dados

adb

- Permitir acesso root ao emulador/dispositivo Android
  - .\adb.exe shell (Windows) ou ./adb shell (Linux)
  - su
- Acessar o conteúdo da aplicação
  - cd data/data/jakhar.aseem.diva/
- Verificar o arquivo de *Shared Preferences*
  - cd shared_prefs
  - cat jakhar.aseem.diva_preferences.xml

Prática 2 - análise do armazenamento de dados

Banco de dados SQLite

SQLite é um mecanismo de banco de dados SQL que armazena dados em arquivos “.db”.

O Android SDK tem suporte integrado para bancos de dados SQLite.

Prática 2 - análise do armazenamento de dados

adb

- Acessar o diretório dos bancos de dados SQLite
  - cd data/data/jakhar.aseem.diva/databases
- Acessar o banco de dados
  - sqlite3 ids2
  - .databases
  - .tables
  - select * from myuser;

Prática 2 - análise do armazenamento de dados

Armazenamento interno

Você pode salvar arquivos no armazenamento interno do dispositivo.

Os arquivos salvos no armazenamento interno são contidos por padrão e não podem ser acessados por outros aplicativos no dispositivo.

Quando o usuário desinstala seu aplicativo, esses arquivos são removidos.

Prática 2 - análise do armazenamento de dados

adb

- Acessar o diretório da aplicação
  - cd data/data/jakhar.aseem.diva/
- Acessar o banco de dados
  - cat uinfo...

Prática 3 - análise da comunicação de rede

Objetivo: analisar a comunicação HTTP e HTTPS da aplicação.

Requisito do MASVS envolvido:

- **MASVS-NETWORK-1** - O aplicativo protege todo o tráfego de rede de acordo com as melhores práticas atuais

Teste do MASTG para averiguar o requisito:

- **MASTG-TESTE-0020** - Testando as configurações de TLS

Prática 3 - análise da comunicação de rede

Aplicação utilizada:

- androgoat.apk

Ferramenta(s) utilizada(s):

- adb
- Burp Suite (análise dinâmica)

Prática 3 - análise da comunicação de rede

adb

- Instalar certificado no dispositivo
  - adb root
  - adb remount
  - adb push 9a5ba575.0 /sdcard/
  - adb shell
  - mv /sdcard/9a5ba575.0 /system/etc/security/cacerts/
  - chmod 644 /system/etc/security/cacerts/9a5ba575.0
  - reboot

Prática 3 - análise da comunicação de rede

Burp Suite

- Definir o proxy
  - Proxy ➔ Proxy Seetings ➔ Add
  - Definir porta e endereço IP (da sua máquina)
  - Request handling ➔ Support invisible proxying
- No emulador
  - Seetings ➔ Network & Internet ➔ Internet ➔ AndroidWifi ➔ Edit ➔ Proxy ➔ Manual
  - host e porta: as mesmas que definiu no Burp Suite

Referências

https://owasp.org/

https://owasp.org/www-project-mobile-top-10/

https://mas.owasp.org/

https://mas.owasp.org/MASVS/

https://mas.owasp.org/MASTG/

https://github.com/OWASP/owasp-mastg/

https://github.com/OWASP/owasp-masvs/

FIM!!!

leonardosilva_99@alu.ufc.br