FIAP

Curso de Defesa Cibernética

Relatório Anual: Desenvolvimento de um RASP para Aplicativos Android

Alunos: João Pedro Rodrigues Bessa, Yuri Torres da Costa França, Bruno Eduardo Dias Souza

São Paulo

2024

Grupo: E404

Projeto: Yatta

João Pedro Rodrigues Bessa - RM550688

Yuri Torres da Costa França - RM98206

Bruno Eduardo Dias Souza - RM551944

Resumo

Este relatório descreve as atividades realizadas pelo grupo no desenvolvimento de um Runtime Application Self-Protection (RASP) para aplicativos Android. O objetivo do trabalho foi implementar e demonstrar um RASP em formato de SDK para ser integrado a aplicativos Android, com verificações de segurança críticas, tais como detecção de dispositivo rooteado, modo debug, utilização de VPN, emulador, overlay de tela e a presença do Frida.

Palavras-chave: RASP, Segurança Móvel, Android, SDK.

Sumário

1 Introdução

Este relatório descreve as atividades realizadas pelo grupo no desenvolvimento de um *Runtime Application Self-Protection* (RASP) para aplicativos Android, parte do desafio anual denominado "Challenge". O objetivo do trabalho foi implementar e demonstrar um RASP em formato de SDK para ser integrado a aplicativos Android, com verificações de segurança críticas, tais como detecção de dispositivo rooteado, modo debug, utilização de VPN, emulador, overlay de tela e a presença do Frida. Este relatório está organizado em quatro capítulos, um para cada sprint realizado ao longo do projeto.

2 Primeiro Sprint

No primeiro sprint, o grupo focou em realizar um estudo inicial sobre o que é um RASP, como ele poderia ser desenvolvido, quais tecnologias seriam necessárias e as ferramentas que iríamos utilizar. Após esse estudo, foi decidido que o RASP seria desenvolvido em Kotlin, utilizando o Android Studio como plataforma de desenvolvimento. As ferramentas escolhidas para as provas de conceito (POCs) foram Frida, Bluestacks e o sistema de debug do próprio Android Studio.

Foram discutidas as verificações que o RASP realizaria por meio de um SDK, abrangendo, inicialmente, a detecção de dispositivos rooteados, modo debug ativo, uso de VPN e a execução em emuladores.

3 Segundo Sprint

No segundo sprint, desenvolvemos as quatro verificações mencionadas: verificação de dispositivo rooteado, verificação de modo debug ativo, verificação de VPN e verificação de emulador. Neste estágio, essas verificações foram implementadas diretamente em um aplicativo básico de *Hello World* para demonstração, ainda não em um formato de biblioteca SDK.

Foi feita uma prova de conceito (*POC*) na qual o aplicativo era fechado caso uma dessas condições fosse detectada. Utilizamos o Bluestacks para rodar o aplicativo e o código foi importado no formato `.aar`, demonstrando a eficácia das verificações.

4 Terceiro Sprint

No terceiro sprint, aprimoramos as verificações implementadas anteriormente e as transformamos em uma biblioteca SDK, permitindo que fossem chamadas em qualquer aplicativo Android por meio da importação das classes. Além disso, foi desenvolvida uma nova verificação que detecta a presença de overlay de tela.

A POC foi realizada novamente utilizando o Bluestacks para simular o ambiente do cliente, e conseguimos demonstrar que todas as verificações funcionavam conforme o esperado, fechando o aplicativo caso alguma condição fosse identificada.

5 Quarto Sprint

No quarto e último sprint, foi implementada a verificação da presença do Frida, uma ferramenta popular usada por atacantes para realizar engenharia reversa em aplicativos. Além disso, foi aplicada ofuscação no código-fonte do SDK, dificultando a análise por invasores.

A POC final será realizada no dia 30/09/2024, quando o RASP será exibido ao cliente para validação final, demonstrando sua eficácia na proteção do aplicativo.

6 Conclusão

O desenvolvimento do RASP ao longo das quatro sprints nos permitiu adquirir um conhecimento aprofundado sobre segurança de aplicações móveis, além de nos familiarizar com as práticas de desenvolvimento de SDKs em Kotlin. As verificações implementadas abrangem algumas das principais ameaças a que aplicativos Android estão expostos, fornecendo uma camada adicional de segurança por meio de um SDK que pode ser facilmente integrado a diferentes projetos.

Esperamos que o trabalho realizado contribua para a evolução das práticas de segurança em aplicações móveis e que futuras melhorias possam incluir verificações adicionais ou otimizações nas já existentes.

Referências

[1] ANDROID Developers. *Developer guides*. Disponível em: https://developer.android.com/guide. Acesso em: 24 set. 2024.

[2] FRIDA. *Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers*. Disponível em: https://frida.re/. Acesso em: 24 set. 2024.

[3] BLUESTACKS. *App player*. Disponível em: https://www.bluestacks.com/. Acesso em: 24 set. 2024.

[4] GENYMOTION. *The Android Emulator for all your needs*. Disponível em: https://www.genymotion.com/. Acesso em: 24 set. 2024.

[5] IRSHAD, V. *Runtime Application Self-Protection techniques (RASP) in Android apps*. Medium, 2022. Disponível em: https://medium.com/@irshad.vit/runtime-application-self-protection-techniques-rasp-in-android-apps-c54536b29f
Acesso em: 24 set. 2024.

[6] APPDOME. *Protect Android apps from overlay attacks and malware*. Disponível em: https://www.appdome.com/how-to/mobile-fraud-detection/ato-prevention/protect-android-apps-from-overlay-attacks-malware/. Acesso em: 24 set. 2024.