[*] Sisyphus: um organizador de informações relacionadas a vulnerabilidades e correções para dispositivos Android

Ewerton Andrade[1,2], Hudson Franca[1], Wesllen Lima[1], Davi Barbosa[1]

[1]Sidia Instituto de Ciência e Tecnologia – SIDIA, Porto Velho / RO – Brasil

[2]Universidade Federal de Rondônia – UNIR, Porto Velho / RO – Brasil

{ewerton.andrade,hudson.franca,wesllen.lima,davi.barbosa}@sidia.com

Resumo. *Catalogar Registros de Vulnerabilidades e Exposições Conhecidas (CVE) referentes a dispositivos Android e compreender como a indústria corrige tais falhas de segurança não é uma tarefa trivial. Isso porque cada fabricante de smartphone disponibiliza seus relatórios em um formato distinto e a base do CVE contém registros que não dizem respeito a smartphones Android. Por isto, este trabalho apresenta uma ferramenta baseada em camadas, onde cada uma das camadas realiza parte do trabalho repetitivo e cansativo de coletar, tratar, relacionar e disponibilizar essas informações em uma nova base. Para que, então, esses dados possam ser consumidos por outros serviços e aplicações.*

Abstract. *It is not a trivial effort to index and organize Common Vulnerabilities and Exposures (CVEs) related to Android smartphones and comprehend how the industry addresses these security issues. This is due to the fact that each smartphone vendor makes its reports available in a different format, and the CVE database contains records that are not related to Android devices. For this reason, this study proposes a layer-based tool in which each layer handles a portion of the repetitious and exhausting task. Extracting, transforming, standardizing, and loading this information into a new database for these data to be utilized by other services and applications.*

1. Introdução

É inegável que os *smartphones* são indispensáveis na sociedade atual. Um indício dessa importância está no fato desses dispositivos terem se tornado a plataforma de tecnologia da informação mais consumida desde a segunda metade de 2019 [STATCOUNTER 2023]. Nesse sentido, destaca-se o fato de que a maioria dos dispositivos móveis são equipados com o Sistema Operacional Android – cerca de 70% do mercado atual [STATCOUNTER 2023]. Contudo, junto com a sua popularidade, também cresce a preocupação quanto a privacidade dos usuários e a segurança da informação nos equipamentos Android; especialmente na academia e entre usuários corporativos [Meng et al. 2018]. Pois sua predominância também o torna o principal alvo de *malwares* e usuários mal-intencionados.

Um passo essencial para aprimorar o processo de correção desses problemas de segurança é a compreensão de suas principais características. Para isso, é possível utilizar

[*]Sisyphus é um personagem da mitologia grega que ficou conhecido por ser condenado a executar trabalhos repetitivos e cansativos, assim como a ferramenta proposta neste trabalho.

classificações e bases públicas como os Registros de Vulnerabilidades e Exposições Conhecidas (*Common Vulnerabilities and Exposures* – CVE) [CVE 2023], pois trata-se de um repositório confiável e mantido por uma instituição independente.

Além disso, alternativamente, podem ser utilizados os relatórios de correções de segurança dos fabricantes desses dispositivos [Huawei 2023, Oppo 2023, Samsung 2023, Vivo 2023], bem como os boletins publicados pela própria equipe de desenvolvimento da versão base do Android, o *Android Open Source Project* (AOSP) [AOSP 2023].

No entanto, é necessário ter em mente que: (I) A base de dados do CVE é genérica [CVE 2023], ou seja, não registra apenas vulnerabilidades e problemas de segurança que afetam dispositivos equipados com Android. Logo, é necessário estabelecer uma boa estratégia de busca para se extrair informações relevantes para um contexto específico. (II) Os relatórios de correções dos fabricantes de *smartphones* e da própria equipe do Android não obedece a uma padronização [AOSP 2023, Huawei 2023, Oppo 2023, Samsung 2023, Vivo 2023]. Assim, pode variar em sua forma de apresentação (*e.g.*, PDFs, páginas Web estáticas, páginas Web dinâmicas e interfaces de programação) e dados divulgados; ou seja, podem trazer apenas dados básicos como data da correção e código identificador (ID) da vulnerabilidade, como também podem trazer dados complementares como severidade, componentes afetados, entre outras informações.

Apesar de existirem estudos que categorizem e abordem os principais aspectos dos CVEs relacionados a dispositivos móveis equipados com Android [Jimenez et al. 2016, Joshi and Parekh 2016, Meng et al. 2018, Tiwari and Velayutham 2019], esses trabalhos não fazem nenhuma correlação entre essas vulnerabilidades conhecidas e as correções que vêm sendo implementadas pela indústria.

Desta forma, este estudo tem como objetivo definir um método e criar uma ferramenta para coletar e padronizar os dados divulgados pelos fabricantes, bem como extrair informações dos CVEs que afetem apenas dispositivos Android. Com isso, será possível dirimir disparidades entre os relatórios públicos e coletar informações relevantes somente para o contexto de interesse (*i.e.*, dispositivos móveis equipados com Android). Assim, além de compreender as características predominantes nos problemas de segurança que afetam esses dispositivos, será possível investigar aplicações e soluções para essas fragilidades e, ainda, viabilizar o desenvolvimento de dispositivos equipados com correções para as classes de vulnerabilidades mais críticas e frequentes.

2. Materiais e métodos

Para atingir os objetivos traçados, foram utilizados os seguintes softwares e ferramentas: o editor de código-fonte Visual Studio Code; a linguagem de programação Python para desenvolvimento de nossa ferramenta e de *scripts* de automação das tarefas; a plataforma Postman para projetar, testar e integrar as APIs (*Application Programming Interface*) utilizadas e desenvolvidas; e o banco de dados PostgreSQL para persistência dos dados e disponibilização de uma base de dados estruturada.

Ressalta-se, ainda, que pela natureza e volume dos dados, foi utilizada uma abordagem quantitativa [Khan and Anwar 2015]. Enquanto para o tratamento dos dados coletados foi utilizado o processo de Extração, Transformação e Carga (*Extract, Transform, Load* – ETL) [Kimball and Caserta 2011].

3. Descrição da ferramenta

Assim, para contornar os desafios descritos anteriormente, foi modelada uma ferramenta com arquitetura baseada em camadas. Cada camada desempenha um papel e se comunica com a camada adjacente, possibilitando, assim, a modularização das tarefas a serem desempenhadas. A Figura 1 apresenta uma visão geral da ferramenta.

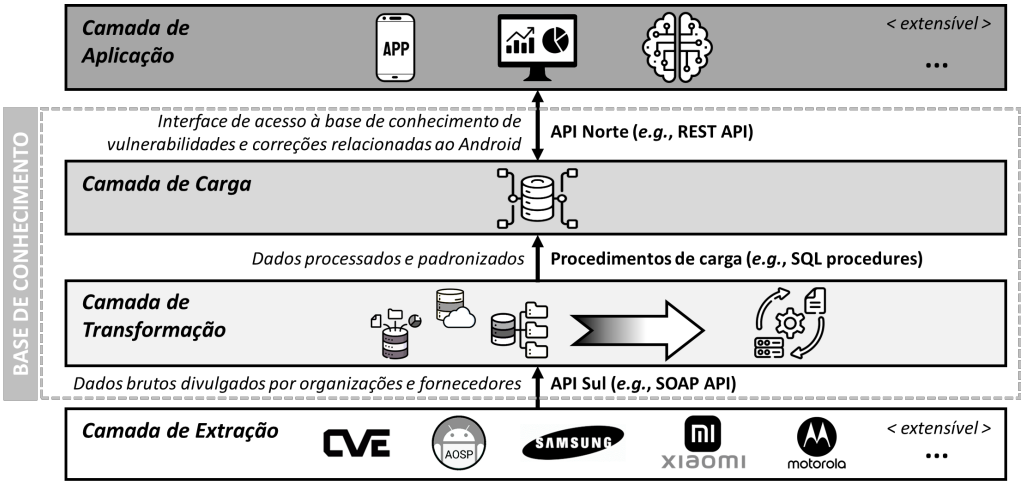

Figura 1. Visão geral do Sisyphus, a ferramenta proposta neste trabalho.

Mais especificamente, cada uma das camadas desempenha as seguintes funções:

- **Camada de Extração *(extensível)*:** é a camada que coleta os dados em estágio bruto. Nela, são implementadas as automações que acessam os relatórios e bases de dados de cada um dos alvos de interesse (*e.g.*, CVE, AOSP, Samsung, Xiaomi). Como cada um desses alvos pode seguir um padrão para disponibilizar os dados, estas automações devem ser adaptadas e extensíveis. Permitindo a refatoração, caso alguma base altere o padrão de disponibilização, bem como a adição de uma nova base de dados, por meio da implementação de uma nova automação.
- **Camada de Transformação:** é a camada que recebe os dados brutos e aplica regras de negócio para transformá-los e padronizá-los. Com isso, é possível processar as informações com mais acurácia e extrair informações relevantes para o contexto de segurança em dispositivos móveis equipados com Android.
- **Camada de Carga:** é a camada que armazena os dados processados e fornece uma interface para que aplicações externas consumam essas informações.
- **Camada de Aplicação *(extensível)*:** é a camada que consome as informações disponibilizadas pela API. Nela, podem ser desenvolvidas aplicações para dispositivos móveis, plataformas de *Business Intelligence* (BI), treinados algoritmos de inteligência artificial, ou implementado qualquer tipo de aplicação que consuma as informações fornecidas pela API REST para gerar um novo conhecimento sobre a segurança de dispositivos móveis equipados com Android.

4. Resultados e discussões

4.1. Camada de Extração

Para extrair informações sobre CVEs relacionados prioritariamente aos dispositivos móveis Android, foi utilizada a API disponibilizada pelo próprio NIST [NIST 2023],

uma vez que ela possibilita a busca por diversos parâmetros como: ID do CVE, datas, severidade, palavras-chave, entre outros. Com isto, foram realizadas extrações de dados por meio das palavras-chave “Android”, “Snapdragon” e “Qualcomm” , por serem abrangentes, mas não genéricas ao ponto de trazerem vulnerabilidades que atinjam outras plataformas. Sobre esse aspecto é importante destacar que outras palavras-chave foram testadas, mas descartadas por retornarem problemas relacionados a outros dispositivos.

Complementarmente, para consultar os relatórios de segurança disponibilizados pelos principais fabricantes de *smartphones* Android, por falta de uma API padronizada ou base estruturada, foi utilizada a técnica de *web scraping* [Mitchell 2018]. Ou seja, foi realizado o download automatizado das páginas, copiando e colando o conteúdo em uma nova base de dados, para sua posterior transformação e análise. Apesar de funcional, pela falta de regulamentação ou imposição quanto a forma de divulgação desses dados, todo esse processo foi realizado de forma semiautomática, com o auxílio de *scripts*, mas variando de acordo com o fabricante. Valendo frisar que um dos objetivos desses *scripts* é facilitar o processo de atualização periódica desses dados. Sendo importante destacar que nesta primeira versão do Sisyphus foram implementadas aplicações para consultar os relatórios da Huawei, Motorola, Oppo, Samsung, Vivo e Xiaomi; além da própria implementação básica do Android, fornecida pelo projeto AOSP.

Frisa-se que esta camada é extensível pela possibilidade de desenvolvimento de novos *scripts* à medida que o interesse por outros fabricantes surja. Ou, ainda, serem atualizados em eventuais alterações no padrão de apresentação dos relatórios.

4.2. Camada de Transformação

Como resultado deste processo de extração, foi criada uma base de dados com (I) as informações públicas disponibilizadas por cada um dos fabricantes mencionados anteriormente; e (II) os CVEs e suas informações complementares, presentes no repositório. Posteriormente, foi realizada a transformação/tratamento, pois os dados extraídos estão em sua forma original e não padronizados. Ou seja, eles precisaram ser mapeados e transformados para prepará-los para o armazenamento definitivo e eventuais consultas.

Desta forma, inicialmente todos os dados coletados foram agregados em uma única entidade, sendo criado um novo atributo para distinguir os dados provenientes dos diferentes fabricantes. Em seguida, foram removidos os registros vazios, para que posteriormente eles pudessem ser validados e corrigidos. Ou seja, foram tratadas todas as distorções de coletas nulas (*e.g.*, quando o *script* mapeia linhas vazias, porém, declaradas no código-fonte), erros de conversão e padronização, concatenações desnecessárias, e demais equívocos que eventualmente ocorreram durante o processo de extração automatizada de informações de páginas web. Assim, foi possível criar uma nova base transformada/tratada acessível e alimentada com dados confiáveis e padronizados.

4.3. Camada de Carga

Nesta camada os dados são carregados em uma nova base de dados para que possam ser acessados e disponibilizados a aplicações externas por meio de uma API REST. Além disso, vale destacar que essa nova base de dados possui dois tipos de tabelas: Tabelas dimensão (`dim_`) e tabelas fato (`fato_`); sendo este um padrão amplamente utilizado em estruturas de dados de BI. Complementarmente, para garantir a escalabilidade, este

banco foi construído sobre uma arquitetura de modelo *Star*, onde todas as dimensões necessárias têm apenas chaves estrangeiras nas tabelas fatos. Essa abordagem é versátil e permite que novas funcionalidades sejam desenvolvidas à medida que a base aumente. No momento, os seguintes serviços estão implementados e respondendo a chamadas:

Fabricantes cadastrados (`/vendors`)	**Informações sobre correção** (`/fix_details`)
P: ∅ (`void`) **R:** Lista com nomes e informações complementares dos fabricantes cadastrados na base (`list`)	**P:** Código identificador do CVE (`cve_id`) **R:** Lista com tempo médio, datas e demais informações sobre as correções divulgadas pelos fabricantes (`list`)
Correções do fabricante (`/cve_fixes`)	**Relatório Mensal** (`/monthly_report`)
P: Fabricante (`vendor`) **R:** Lista de todas as vulnerabilidades corrigidas pelo fabricante passado como parâmetro, com informações complementares que foram coletadas (`list`)	**P:** Mês e Ano (`month, year`) **R:** Lista com informações das vulnerabilidades e/ou correções, divulgadas no mês e ano passados como parâmetro (`list`)
P:Parâmetro(s) de entrada / **R:**Resposta	

Tabela 1. Chamadas implementadas na API REST.

4.4. Camada de Aplicação

Camada de mais alto nível, onde os dados extraídos, tratados e carregados podem ser utilizados pelos mais diversos tipos de aplicação. Por exemplo, a base de dados pode ser analisada em uma perspectiva temporal, examinando a divulgação das correções desenvolvidas pelos fabricantes ao longo dos anos. Uma ilustração dessa análise pode ser vista na figura abaixo, onde a API REST foi consumida utilizando Postman e os dados foram tabulados e organizados por meio de uma planilha eletrônica.

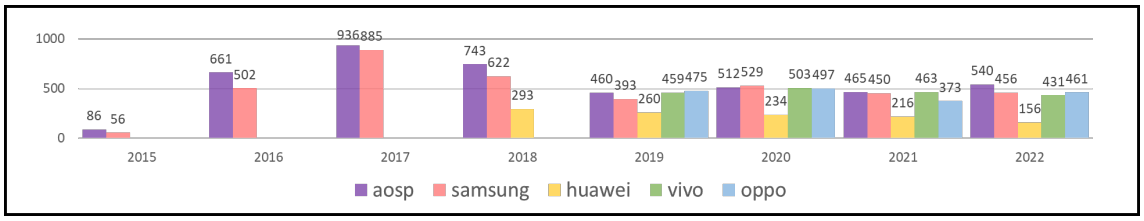

Figura 2. Correções desenvolvidas pelos fabricantes ao longo dos anos.

Destaca-se que esta camada é extensível pela possibilidade da API REST ser consumida por diversos tipos de aplicação, o que torna esta ferramenta ainda mais poderosa.

5. Considerações finais

Neste trabalho foi apresentado o Sisyphus, uma ferramenta que funciona como organizador dos CVEs e correções implementadas pelos fabricantes de *smartphones* Android. Esta ferramenta foi projetada sob uma perspectiva de camadas, onde cada uma das camadas realize parte do trabalho repetitivo e cansativo que é coletar, tratar, relacionar e disponibilizar essas informações em um banco de dados que utilize modelos alinhados às boas práticas da ciência de dados.

Outro aspecto relevante é que tanto a camada de extração de dados quanto a camada de disponibilização desses dados para outras aplicações são extensíveis. Logo, relatórios de novos fabricantes e outras bases de interesse podem ser incorporadas. Bem como seus dados padronizados e organizados podem ser utilizados em aplicações como

painéis com perspectiva temporal de evolução dos relatórios, pesquisas acadêmicas, consultas sobre a natureza das vulnerabilidades, informações quantitativas sobre vulnerabilidades corrigidas pelos fornecedores, entre diversas outras ferramentas e tecnologias.

Desta forma, acredita-se que o Sisyphus ajudou a dirimir a disparidade entre as vulnerabilidades conhecidas e os relatórios dos fabricantes de dispositivos móveis Android. Assim, será possível realizar novos trabalhos para (I) compreender as características predominantes nos problemas que afetam esses dispositivos; (II) investigar aplicações e soluções para essas fragilidades; (III) analisar as relações entre a versão do Android e as vulnerabilidades conhecidas; (IV) estabelecer correlação entre os CVEs e os boletins de segurança de diferentes fabricantes; (V) e, ainda, viabilizar o desenvolvimento de dispositivos com correções para vulnerabilidades mais críticas e frequentes.

Agradecimentos

Este artigo é o resultado do projeto de PD&I Mobile Cybersecurity, realizado pelo Sidia Instituto de Ciência e Tecnologia em parceria com a Samsung Eletrônica da Amazônia Ltda., usando recursos da Lei Federal nº 8.387/1991, estando sua divulgação e publicidade em conformidade com o previsto no artigo 39º do Decreto nº 10.521/2020.

Referências

AOSP (2023). Android Open Source Project. `https://source.android.com/`.

CVE (2023). Common Vulnerabilities and Exposures. `https://cve.mitre.org/`.

Huawei (2023). HUAWEI EMUI/Magic UI security updates. Disponível em: `https://consumer.huawei.com/en/support/bulletin/`. Acesso em: 2023-08-01.

Jimenez, M., Papadakis, M., Bissyandé, T. F., and Klein, J. (2016). Profiling Android vulnerabilities. In *2016 IEEE QRS*, pages 222–229.

Joshi, J. and Parekh, C. (2016). Android smartphone vulnerabilities: A survey. In *2016 ICACCA (Spring)*, pages 1–5. IEEE.

Khan, E. R. and Anwar, H. (2015). *Research methods of computer science*. Laxmi.

Kimball, R. and Caserta, J. (2011). *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. Wiley.

Meng, H., Thing, V. L., Cheng, Y., Dai, Z., and Zhang, L. (2018). A survey of android exploits in the wild. *Computers & Security*, 76:71–91.

Mitchell, R. (2018). *Web scraping with Python*. "O'Reilly Media, Inc.".

NIST (2023). National Vulnerability Database API. Disponível em: `https://nvd.nist.gov/developers/vulnerabilities`. Acesso em: 2023-08-01.

Oppo (2023). Oppo security response center. `https://security.oppo.com/`.

Samsung (2023). Security Updates. `https://security.samsungmobile.com/`.

STATCOUNTER (2023). GlobalStats. `https://gs.statcounter.com`.

Tiwari, P. K. and Velayutham, T. (2019). Android Vulnerabilities: Taxonomy and nextGen Ecosystem. In *2019 IEEE IBSSC*, pages 1–7. IEEE.

Vivo (2023). Android Security Updates. `https://www.vivo.com/en/security`.