Uma avaliação do cenário de detecção e evasão do acesso root no Android

Vinicius Moraes, Jéssyka Vilela

Centro de Informática, Universidade Federal de Pernambuco (UFPE)
CEP: 50.740-560 – Recife – PE – Brazil

{vrm, jffv}@cin.ufpe.br

Abstract. *Android is now the most used operating system in the world making it a target for cyber criminals. Aiming the security of the devices, one of the challenges faced by developers is the detection and evasion of root access. This work investigates root evasion and detection techniques used in practice through a survey, evaluates the effectiveness of applications that aim to detect root, and also of the most downloaded applications on Google Play. Finally, improvements to some of the detection techniques are presented. It is hoped that this study will contribute to the understanding of the dispute between detection and evasion and allow more secure applications to be developed.*

Resumo. *O Android é hoje o sistema operacional mais utilizado no mundo tornando-o alvo por criminosos digitais. Visando a segurança dos dispositivos, um dos desafios enfrentados pelos desenvolvedores é a detecção e evasão do acesso root. Este trabalho investiga as técnicas de evasão e detecção de root utilizadas na prática por meio de um survey, avalia a eficácia dos aplicativos que têm o objetivo de detectar o root e também dos aplicativos mais baixados no Google Play. Finalmente, apresenta-se aprimoramentos para algumas das técnicas de detecção. Espera-se que este estudo contribua para a compreensão da disputa entre detecção e evasão e permita que aplicações mais seguras possam ser desenvolvidas.*

1. Introdução

O Android é o sistema operacional para dispositivos móveis mais usado no mundo (POLISCIUC et al., 2020). Estima-se que aproximadamente 74% dos mais de 3.5 bilhões de smartphones ao redor do mundo utilizam o Android como seu sistema operacional (STATISTA, 2020). Em paralelo a esse crescimento do sistema, uma prática que se tornou comum entre alguns usuários do Android é a de realizar o procedimento de *rooting*, ou seja, ativar o acesso de superusuário (*root*) em seus aparelhos. Estima-se que em 2017 o número de dispositivos rooteados na Indonésia chegava a ser de 12%. Esse número aumenta quando olha-se para a Moldávia com 15% ou ainda para a Venezuela com 26% (KASPERSKY, 2020).

Alguns dos motivos que levam os usuários a desejarem o *root* são utilizar um determinado aplicativo, personalizar o Android (através de novos temas e funcionalidades) ou ainda desinstalar *bloatwares*, que são aplicativos indesejados que podem vir instalados no sistema e, às vezes, não são completamente removíveis por métodos tradicionais (KOTIPALLI, 2016).

Contudo, muitos desses usuários acabam não percebendo as complicações de segurança que essa prática pode causar. Quando um aplicativo pode ser executado normalmente em dispositivos rooteados surgem dois principais problemas. O primeiro é em relação a segurança do próprio usuário. Como o acesso *root* permite que um

programa obtenha privilégios muito elevados no sistema, existe o risco de que vazamentos de dados aconteçam, pois se torna possível ler arquivos privados de outros aplicativos, além da possibilidade de acesso aos dados da memória e da rede deles. Dessa forma, um malware pode obter tokens de sessão de outros aplicativos e executar operações indesejadas neles, como roubar mensagens privadas de redes sociais ou ainda realizar operações financeiras sem a autorização do usuário (CASATI, 2018). O segundo problema se refere a segurança do próprio serviço entregue pelo aplicativo, já que existem ferramentas para fraude que necessitam de privilégios de *root*, como por exemplo a possibilidade de burlar compras internas dentro dos aplicativos.

Por conta das ameaças que os aplicativos sofrem ao serem executados em aparelhos rooteados, alguns deles começaram a implementar técnicas de detecção de *root*. No contexto deste trabalho, detecção é *o ato de se utilizar de técnicas para procurar sinais que indicam a presença do root em um dispositivo Android.*

Entretanto, o acesso *root* não possui apenas desvantagens. Existem benefícios para pesquisadores, desenvolvedores e até mesmo usuários avançados que desejam ter um maior controle sobre seu próprio aparelho. Com esse alto nível de privilégios, é possível modificar parâmetros do *kernel* do Android, em busca de mais desempenho ou economia de energia ou ainda realizar monitoramentos avançados de rede (TERMUX, 2020). Em virtude dessas vantagens, surgiram métodos de evasão. Neste trabalho, a evasão se refere a *ação de esconder o acesso root existente em um aparelho Android*.

Esse conflito de interesses, entre aqueles que desejam detectar o acesso *root* e aqueles que desejam escondê-lo, gera uma disputa incessante entre dois lados, que é o contexto deste trabalho (detecção vs evasão). Este trabalho tem como objetivo realizar uma investigação do estado da arte e da prática das técnicas para detecção de *root*, avaliar a eficácia delas contra os métodos de evasão e propor técnicas para aperfeiçoar a detecção.

Esse artigo está estruturado da seguinte forma: na Seção 2 é apresentada revisão da literatura e trabalhos relacionados, na Seção 3 é descrita a metodologia adotada neste trabalho, na Seção 4 são discutidos os resultados obtidos e, por fim, na Seção 5, são apresentados as conclusões e trabalhos futuros.

2. Revisão da literatura e trabalhos relacionados

Root é o nome do superusuário de diversos sistemas operacionais incluindo o Android. O termo “root” também pode se referir ao acesso *root*, que é quando os privilégios do usuário root são obtidos, normalmente de forma temporária apenas para realizar uma determinada ação, como por exemplo instalar um programa. Como detalhado em Nguyen-Vu et al. (2017), existem muitos caminhos para alcançar o acesso *root* no Android. Atualmente, a maneira mais genérica e popular envolve o *bootloader* e o *recovery* do dispositivo (KOTIPALLI, 2016).

Segundo Nguyen-Vu et al. (2017), existem 6 classificações para as técnicas de detecção do acesso *root*: Checagem de Aplicativos Instalados, Checagem de Arquivos, Checagem de Processos, Serviços e Tarefas, Checagem de Execução de Comandos, Checagem de Propriedades do Sistema e Checagem de Permissões dos Diretórios.

Além disso, o Google fornece um conjunto de serviços e APIs (do inglês *Application Programming Interface*), denominado *SafetyNet*, com o objetivo de fornecer determinadas soluções de segurança para aplicativos do Android. Uma dessas APIs é a *SafetyNet Attestation* que pode ser utilizada por desenvolvedores para avaliar a integridade de um aparelho que estará executando seu aplicativo. Essa API possui o parâmetro *basicIntegrity* que informa o resultado de uma avaliação que, dentre outras coisas, falha caso detecte que o aparelho tem acesso *root* ou que a execução está ocorrendo em um emulador. Como a API considera o acesso *root*, ela é uma opção adotada por alguns aplicativos para dificultar a sua execução em aparelhos rooteados.

Na literatura existem alguns trabalhos que tratam sobre a detecção de acesso *root* no Android. A Tabela 1 apresenta as principais similaridades e diferenças entre este trabalho e outros que foram investigados.

Tabela 1. Comparação entre este trabalho e outros relacionados.

Critério\Trabalho	**EVANS et al., 2015**	**SUN et al., 2015**	**NGUYEN-VU et al., 2017**	**Este Trabalho (2020)**
Foco da pesquisa	Evasão e detecção de *root*	Procedimento de *rooting*, Evasão e detecção de *root*	Evasão e detecção de *root*	Evasão e detecção de *root*
Métodos de Evasão utilizados	AndroPoser, RootCloak, Hide My Root	RDAnalyzer	RootCloak, Renomear binário su	Magisk
Fonte de informação sobre as técnicas de detecção e evasão encontradas	Pesquisa na literatura, Análise estática	Pesquisa na literatura, Análise estática	Pesquisa na literatura, Análise estática	Pesquisa na literatura, Survey com usuários e desenvolvedores, Análise de eficácia da detecção dos aplicativos

Critérios de seleção para análise de aplicativos	Não divulgado.	Aplicativos listados pelo RootCloak.	Busca pelo termo "root checker" no Google Play.	Buscas por "root detect", "root check", "root verity", "magisk check", "magisk detect" no Google Play, Aplicativos com maior número de downloads.
Quantidade de aplicativos avaliados	16 aplicativos de antivírus e outros 19 de gerenciamento de dispositivos móveis.	30 aplicativos com detecção listados pelo RootCloak	92 aplicativos de detecção de *root*	136 aplicativos de detecção de *root* e 853 aplicativos mais baixados
Melhoria para a detecção	Não apresenta melhorias	Sugere duas abordagens contra evasão por *hooking*	Cria aplicativo que reúne técnicas encontradas	Sugere duas abordagens contra Magisk

Um outro trabalho relacionado é o de Polisciuc et al. (2020) que propõe a ferramenta DroiDiagnosis, que utiliza chamadas de sistemas e permissões de acesso para identificar aplicativos maliciosos por meio de características dinâmicas e estáticas.

Uma das principais diferenças no cenário deste trabalho, em relação aos que foram analisados, acontece por causa do surgimento do Magisk e a descontinuação do SuperSU. De forma que ambos os eventos alteraram o cenário de detecção ao ponto de tornar muitas implementações obsoletas, um exemplo é o aplicativo S4URC Root Checker que implementa todas as técnicas do estudo de Nguyen-Vu et al. (2017) e que atualmente não consegue detectar o Magisk.

Magisk é o nome de um conjunto de ferramentas (do inglês *toolkit*) de código aberto, criado em 2016 por John Wu. Após a descontinuação do SuperSU, o Magisk se tornou não só a maneira mais popular para obter root, mas também para escondê-lo como será evidenciado na Seção 4.2. A ferramenta, atualmente, já possui mais de 72 milhões de downloads. Ao instalar o Magisk, surge o Magisk Manager no aparelho, que atua como uma interface gráfica para outros programas que também são instalados. Um deles é o MagiskHide, que lista os aplicativos instalados, e permite marcar quais não devem detectar o root.

Nesse contexto, diferente dos trabalhos relacionados que realizaram análises estáticas para identificar os tipos de checagens de *root*, este trabalho realiza análises nos aplicativos para avaliar o quão eficazes são as atuais detecções implementadas por eles contra o Magisk. Outra distinção deste trabalho em relação aos demais é a apresentação

de melhorias para a detecção, que se mostraram eficazes contra o Magisk, como é possível observar na Seção 4.4.

Finalmente, neste trabalho, houve a condução de um *survey,* a fim de entender os mecanismos de evasão e detecção de *root* utilizados na prática. A escolha do *survey* ocorreu por ele fornecer um contato direto com usuários e desenvolvedores Android. Este método de pesquisa é interessante para reunir múltiplas visões sobre o atual cenário, além de fornecer um alcance maior de participantes do que uma entrevista.

3. Metodologia

Este estudo utiliza o método de pesquisa exploratória, visto que um dos focos é avaliar se as atuais implementações das técnicas de detecção ainda podem ser aplicadas como medidas eficazes para identificar aparelhos rooteados. A pesquisa foi desenvolvida a partir de dados coletados por meio de revisão não sistemática da literatura, um *survey* com usuários e desenvolvedores Android (Seção 3.1) e estudos empíricos com aplicativos em aparelhos rooteados (Seções 3.2 e Seção 3.3).

3.1. Survey

Em busca das técnicas de detecção mais utilizadas na prática, foi criado um *survey* através do Google Forms, que foi divulgado em *threads* de e-mail de faculdades, grupos de tecnologia em aplicativos de mensagens e fóruns de discussão sobre o Android (Reddit, Android Forum, CNET). Foram elaboradas uma versão em português, divulgada no dia 24 de junho de 2020, e uma versão em inglês, divulgada em 02 de julho de 2020. O recebimento de respostas em ambas as versões foi encerrado em 31 de agosto de 2020.

O *survey* foi estruturado em três seções: a primeira era correspondente ao perfil do respondente (2 perguntas), a segunda sobre detecção (6 perguntas) e a terceira sobre evasão (4 perguntas). Os participantes visualizaram as perguntas a depender do seu perfil. Participantes que não pertenciam a algum desses grupos foram desqualificados. As respostas obtidas, códigos desenvolvidos e outras informações relacionadas a esse trabalho podem estão disponíveis no seguinte endereço: https://github.com/sbseg/.

3.2. Análise de aplicativos de checagem de root

O objetivo desta análise foi avaliar o quão eficazes são as atuais implementações das técnicas de detecção. Por conta disso, para os testes foi escolhido um ambiente em que a detecção do *root* deveria ser mais difícil, um Xiaomi Redmi K20 (*codename davinci*) com *bootloader* desbloqueado, que passava nos testes da API *SafetyNet Attestation*, utilizando o *recovery* TWRP na versão 3.4.0, a ROM Pixel Experience na versão 10.0-20200531-14-31 (Android 10) e rooteado por meio do Magisk na versão 20.4. Além disso, o Magisk Manager teve seu nome de aplicativo e seu *package name* alterados para valores aleatórios.

Esse ambiente tem um maior grau de dificuldade para ser detectado por causa do Magisk, que contém o MagiskHide, e que permite esconder, para determinados aplicativos, a existência de diretórios e binários que revelariam o Magisk, como o

/system/bin/su, se utilizando da funcionalidade *namespaces* do *kernel* Linux, presente no Android.

Ademais, a API SafetyNet Attestation da Google permite avaliar se um aparelho tem o acesso root, porém ela tem algumas limitações, como não funcionar para detectar o root fornecido pelo Magisk. Além disso, ela precisa do Google Play Services, que não está presente em celulares de regiões como a China. Ela foi indiretamente testada na Análise dos Aplicativos Mais Baixados (Seção 4.3), quando aplicativos que se utilizam dela, como o Pokémon Go, não conseguiram detectar o root.

A fim de escolher os aplicativos a serem analisados, em junho de 2020 foram reunidos os 250 primeiros resultados retornados em buscas no Google Play para cada um dos seguintes termos: “root detect”, “root check”, “root verify”, “magisk check”, “magisk detect”. Em seguida, os aplicativos foram filtrados a partir dos seguintes critérios de exclusão: Aplicativos repetidos; Aplicativos que não tinham como objetivo a detecção de *root*; Aplicativos pagos; e, Aplicativos incompatíveis com o aparelho de teste. Após a filtragem, 136 aplicativos foram selecionados (lista disponível no GitHub), para cada um deles, os seguintes passos foram realizados:

1) O aplicativo foi instalado através do Google Play;
2) O aplicativo foi adicionado na lista do MagiskHide e, em seguida, executado;
3) Os comportamentos do aplicativo foram analisados a fim de verificar se o mesmo detectou o *root* do aparelho;
4) Se fosse observado comportamento que indicasse que o aplicativo está detectando o *root*, o aplicativo era submetido a um descompilador para confirmar, em seu código, a detecção.

3.3. Análise da eficácia da detecção nos aplicativos mais baixados

Complementando a análise anterior, foi realizado outro estudo empírico, este com o objetivo de avaliar as características dos aplicativos que mais implementam a detecção, e a eficácia da mesma em aplicativos populares.

Para identificar os aplicativos que detectavam *root*, o ambiente escolhido não fazia uso de métodos para escondê-lo. O aparelho era um Moto G4 Plus (*codename athene*) que falhava em ambos os testes da API *SafetyNet Attestation*, com *bootloader* desbloqueado, utilizando o TWRP na versão 3.2.1, a ROM Resurrection Remix na versão 6.2.0-20180916 (Android 8.1.0) e com acesso *root* fornecido pelo seu próprio sistema modificado.

Em julho de 2020, através do site AndroidRank[1], foi obtida uma lista dos 20 aplicativos mais baixados de cada uma das 49 categorias do Google Play, resultando em um total de 980 aplicativos. Todos eles eram gratuitos, no entanto, o Google Play não permitiu que 127 deles fossem instalados por restrições de localidade ou incompatibilidade com o aparelho de teste. Para os 853 aplicativos restantes, os passos a seguir foram realizados:

1) Os aplicativos foram instalados pelo Google Play e, em seguida, executados;

[1] Disponível em <https://www.androidrank.org/>, acessado em 19 de julho de 2020.

2) Foram buscados comportamentos visuais que indicassem a detecção do *root*;
3) Para aplicativos que precisavam de autenticação, apenas a parte não autenticada foi considerada;
4) Todos aplicativos que não apresentavam sinais de detecção e tinham alguma funcionalidade de autenticação, manipulando credenciais, eram registrados;
5) Caso fosse considerado que um aplicativo estava detectando o *root*, o mesmo era executado contra o Magisk no ambiente da análise anterior (Seção 3.2).

A avaliação dos 853 aplicativos levou aproximadamente 6 semanas e teve o suporte de um script desenvolvido para auxiliar no processo (disponível no GitHub).

3.4 Ameaças à Validade

Esta seção apresenta ameaças à validade dos resultados deste trabalho considerando a classificação de Wohlin (2012). A fim de reduzir ameaças à *validade de constructo*, o *survey* foi construído de forma iterativa, obtendo-se assim várias visões antes de ser divulgado. Ademais, as respostas obtidas não demonstraram sinais de que houve um problema relevante de entendimento por parte dos participantes. Para minimizar os riscos à *Validade Interna* no *survey*, muitas perguntas foram configuradas para receberem respostas abertas. Já para reduzir esse problema nos estudos empíricos com os aplicativos e nas propostas de melhorias para a detecção, foram considerados os resultados obtidos na revisão da literatura e no *survey*.

Validade Externa: Em relação ao *survey*, o número de participantes envolvidos pode ser considerado limitado e pequeno considerando o número de usuários e desenvolvedores Android existentes. Contudo, os resultados obtidos no *survey* estão alinhados com as conclusões extraídas dos estudos empíricos realizados que analisaram um grande número de aplicativos em relação a outros estudos apresentados na Tabela 1. Já para as melhorias da detecção, houveram testes em diversos ambientes diferentes.

Validade de Confiança: A fim de ajudar na replicação do *survey*, todas as perguntas utilizadas e os principais pontos em relação a sua aplicação foram disponibilizados no GitHub. Também foram disponibilizados os dados obtidos nos estudos empíricos: os aplicativos utilizados, os procedimentos realizados e os ambientes de teste. Finalmente, para as melhorias propostas, os processos realizados e o código fonte completo do aplicativo prova de conceito também foram disponibilizados.

4. Resultados

A seguir serão apresentados os resultados obtidos por meio do *survey* e das análises realizadas bem como melhorias para fortalecer a detecção do *root*.

4.1. Survey

O *survey* obteve um total de 262 respostas que podem ser acessadas através do GitHub. Desse total, 116 dos participantes declararam já ter desenvolvido para Android, porém apenas 17 haviam tido experiência com a detecção e, portanto, se qualificaram para o grupo da detecção. Entre os participantes não classificados no grupo anterior, 106

tinham experiência escondendo o *root* e, dessa forma, se enquadraram no grupo da evasão.

Analisando as respostas do **grupo da detecção**, um indicativo da falta de maturidade e padronização da detecção apareceu quando 58% dos participantes, 10 dos 17, disseram que implementaram a detecção por conta própria, talvez por desconhecimento ou insuficiência das técnicas existentes, e cada um utilizando uma verificação diferente.

Ademais, quando questionados sobre a melhor forma para detectar o *root,* não houve um consenso, em contraste, quando o assunto foi formas de burlar as detecções, 4 dos 6 desenvolvedores que responderam essa pergunta mencionaram o Magisk.

Por outro lado, os dados do **grupo da evasão** indicam que a mesma dispõe de métodos mais populares e bem estabelecidos, visto que a grande maioria das respostas, 95 dos 106 usuários do grupo, informou que utilizou o Magisk para esconder o *root*.

Além disso, foi possível concluir que o foco daqueles que escondem o *root* são os aplicativos financeiros, talvez por também serem os que mais fazem uso da detecção, visto que o interesse em realizar a evasão nesses aplicativos, demonstrado por 35 usuários, foi mais do que o dobro do segundo lugar, o aplicativo da Netflix, indicado por 13 usuários.

Por fim, algumas perguntas do grupo da detecção, que não eram obrigatórias, foram respondidas por menos da metade dos participantes do grupo. Essa quantidade menor pode ter reduzido a compreensão sobre o contexto da detecção.

4.2. Análise de aplicativos de checagem de root

De um total de 136 aplicativos analisados, apenas um conseguiu detectar o acesso *root* do aparelho descrito na Seção 3.2, o aplicativo Magisk Detector (com.pat.detector) na versão 1.3. Ao descompilar o código do Magisk Detector (apresentado na Figura 1), foi observado que na verdade ele se utiliza de uma falha pública (https://darvincitech.wordpress.com/2019/11/04/detecting-magisk-hide/, acessado em 07/10/2020) que afeta o Magisk desde a versão 19 (https://medium.com/csg-govtech/diving-down-the-magisk-rabbit-hole-aaf88a8c2de0, acessado em 10/10/2020). Por causa dessa falha, o Magisk não consegue esconder, para serviços isolados do Android, o caminho de alguns diretórios que são criados por ele.[2]

É interessante notar que mesmo existindo uma falha pública no Magisk que permite identificá-lo nas suas versões mais recentes, apenas um aplicativo se utilizava dela. Tais resultados reforçam as conclusões obtidas no *survey* (Seção 4.1) de que, atualmente, a detecção ainda é muito menos madura do que a evasão.

[2]Disponível em <https://github.com/darvincisec/DetectMagiskHide/blob/master/app/src/main/java/com/darvin/security/IsolatedService.java>, acessado em 5 de outubro de 2020.

Figura 1: Código utilizado pelo Magisk Detector para procurar o Magisk. Fonte: jadx.[3]

Após a realização dessa pesquisa em 2020, constatou-se que o Magisk Detector não está mais disponível no Google Play para download. Acredita-se que o aplicativo apenas foi relançado com outro nome uma vez que a mesma conta de desenvolvedor do Google Play que havia lançado o Magisk Detector, pat.dev, agora tem o Root System Detector (https://play.google.com/store/apps/details?id=com.pat.rsd).

4.3. Análise da eficácia da detecção nos aplicativos mais baixados

Foi possível confirmar a existência da detecção de *root* em apenas 13 aplicativos, o que é aproximadamente 1.5% do total de 853 aplicativos analisados, sendo a categoria *Finance* aquela com mais aplicativos se utilizando da detecção. Esses resultados demonstram uma maior preocupação com segurança por parte desse nicho. Além disso, 374 aplicativos que aparentemente não procuravam pelo *root* tinham alguma funcionalidade que manipulava credenciais do usuário. A Tabela 2 detalha os aplicativos identificados, já a Figura 2 ilustra um exemplo de sinal de detecção de *root*.

Tabela 2. Aplicativos que realizavam detecção de root.

Categoria: Aplicativo	Quantidade
Finance: br.gov.caixa.tem, com.mobikwik_new, com.sbi.SBIFreedomPlus, com.sbi.lotusintouch	4
Auto and Vehicles: com.gigigo.ipirangaconectcar, **Beauty**: kr.co.company.hwahae, **Business** (us.zoom.videomeetings), **Food and Drink**: com.mcdonalds.mobileapp, **Game Adventure**: com.nianticlabs.pokemongo, **Health and Fitness**: nic.goi.aarogyasetu, **Parenting**: com.hp.pregnancy.lite, **Social**: com.snapchat.android, **Tools**: com.google.android.gms	1 de cada categoria

[3] Disponível em <https://github.com/skylot/jadx>, acessado em 8 de outubro de 2020.

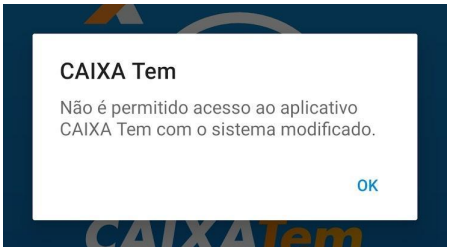

Figura 2: Exemplo de sinal de detecção de root.

Ademais, os 13 aplicativos também foram instalados no dispositivo de teste da Seção 3.2, adicionados na lista do MagiskHide e executados, para verificar se eles conseguiriam encontrar o *root*. Contudo, nenhum conseguiu identificá-lo em tal ambiente, sugerindo o quão eficaz o Magisk é em esconder o *root*.

Finalmente, vale ressaltar uma limitação do estudo empírico apresentado. É possível que alguns dos aplicativos analisados estivessem buscando o *root* de forma silenciosa (sem nenhum tipo de mensagem visual) ou apenas após uma autenticação, nesses casos, os aplicativos acabaram sendo classificados como sem detecção.

4.4. Fortalecendo a Detecção Contra o Magisk

O Magisk é eficaz contra os aplicativos atuais, mesmo já existindo uma falha nele que permite identificá-lo. Contudo, essa falha só afeta as versões de 19 em diante e, portanto, pode ser evitada pelos usuários. Dessa forma, a seguir serão propostas duas novas abordagens que conseguem identificar o Magisk, uma que aprimora uma técnica de detecção já existente e outra que pode ser considerada como uma nova forma de detecção.

4.4.1. Busca por Nome e Conteúdo de Arquivos

O propósito deste ponto é demonstrar um procedimento que pode ser realizado para encontrar arquivos a serem verificados na checagem de arquivos, descrita em Nguyen-Vu et al. (2017). Além disso, objetiva-se mostrar um arquivo que é criado pelo próprio Magisk e permite detectá-lo.

Como mencionado anteriormente, o Magisk utiliza uma funcionalidade, disponibilizada pelo kernel Linux do Android, para esconder determinados diretórios criados por ele no dispositivo (*namespaces*).[4] Contudo, essa solução não é perfeita, visto que a mesma funciona como uma *block list* e, assim, permite a procura por arquivos em diretórios não esperados pelo Magisk. Portanto, a fim de realizar uma busca por arquivos expostos, os passos abaixo foram reproduzidos no ambiente de teste detalhado na Seção 3.2:

1) Um aplicativo de terminal (sem a permissão de acesso ao armazenamento externo) foi instalado no dispositivo de teste, visto que assim seria possível procurar sinais do Magisk do ponto de vista de um aplicativo padrão;

[4] Disponível em <https://github.com/topjohnwu/Magisk/blob/f42a87b51ae64cf5ed4937e3cd17c7f865f33bc1/native/jni/magiskhide/hide_policy.cpp#L92>, acessado em 18 de junho de 2021.

2) O aplicativo de terminal foi inserido na lista do MagiskHide;

3) Usando as ferramentas find e grep, foi procurado por arquivos que tinham o termo “magisk” em seus nomes ou conteúdos, dentro de diretórios comuns do Android e acessíveis pelo aplicativo (como /system, /vendor, /mnt).

Então, foi possível identificar o arquivo “/system/addon.d/99-magisk.sh”, que é criado no momento da instalação do Magisk, caso o diretório “/system/addon.d” exista.[5] Assim, pode-se identificar aparelhos rooteados com o Magisk e realizar uma melhor implementação da detecção de arquivos. Entretanto, esse diretório não existe em todos os Androids e, além disso, outros arquivos podem aparecer de acordo com cada sistema. Por isso, essa metodologia foi repetida múltiplas vezes em algumas ROMs instaladas no aparelho da Seção 3.2. A Tabela 3 resume os arquivos encontrados.

Tabela 3: Arquivos que indicam a presença do Magisk.

ROM	Versão	Arquivos
MIUI	11.0.6.0	(nenhum arquivo identificado)
Pixel Experience	10.0-20200531-14-31	/system/addon.d/99-magisk.sh
Resurrection Remix	8.5.9-20200822	/system/addon.d/99-magisk.sh
Xiaomi EU	12.0.3.0	/system/media/theme/miui_mod_icons/com.topjohnwu.magisk.png

4.4.2. Análise de Ícones

A análise de ícones é uma forma de verificação proposta por este trabalho que não foi encontrada na revisão da literatura. Ela pode ser classificada como um subtipo da checagem de aplicativos instalados visto que o objetivo de ambas é verificar se um determinado aplicativo (que tem relação com o *root*) está instalado no aparelho. O diferencial desta análise é que os aplicativos instalados no aparelho são analisados através de seus ícones. Dessa forma, o ícone de cada um deles é comparado com o do Magisk Manager. Existem vários algoritmos que podem ser utilizados para comparar imagens, um deles é o pHash[6], que produz um hash para cada imagem analisada e, posteriormente, permite que diferentes hashes tenham suas distâncias calculadas com o objetivo de informar o grau de semelhança entre as imagens.

A fim de realizar uma prova de conceito, foi desenvolvido um aplicativo que implementa essa checagem e está disponível no GitHub.[7] A Figura 3 demonstra uma das funções dele, a *getMagiskIconDistance*, que utilizando o algoritmo pHash, calcula o

[5] Disponível em <https://github.com/topjohnwu/Magisk/blob/3e4caabecb324f6375a031505e5b8bbf5b859aa6/scripts/flash_script.sh#L83>, acessado em 6 de outubro de 2020.
[6] Disponível em <https://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html>, acessado em 7 de outubro de 2020.
[7] Disponível em <https://github.com/sbseg/magisk-icon-checker>, acessado em 18 de junho de 2021.

hash do ícone de um aplicativo e, então, o avalia contra alguns modelos de ícones do Magisk ilustrados na Figura 4.

Como pode ser observado na Tabela 4, o aplicativo implementado para a prova de conceito foi executado em diversos aparelhos e conseguiu identificar todos ambientes que utilizavam o Magisk. Para dispositivos rooteados de outra forma, as técnicas tradicionais ainda são necessárias e, portanto, recomenda-se utilizá-las em conjunto dessa proposta.

```
private int getMagiskIconDistance(Drawable icon) {
  int shorterDistance = Long.SIZE;
  long hashApp = phash.getHash(icon);
  /* comparando icone recebido com os do magisk */
  for (long magiskHash : magiskHashes) {
      int curDistance = phash.distance(hashApp, magiskHash);
      /* salvando a menor distancia, que equivale
         ao maior grau de semelhanca encontrado */
      shorterDistance = Math.min(shorterDistance, curDistance);
  }
  return shorterDistance;
}
```

Figura 3: Obtendo a semelhança de um ícone arbitrário com o do Magisk.

Figura 4: Ícones utilizados na função *getMagiskIconDistance*.

Tabela 4: Ambientes em que o aplicativo prova de conceito foi executado.

Aparelho	**ROM**	**Configuração**	**Resultado do Aplicativo**
Redmi K20	Pixel Experience (10.0-20200531-14-31)	Magisk (20.4) com o aplicativo prova de conceito na lista do MagiskHide	Root detectado
Redmi K20	Xiaomi EU (12.0.3.0) com tema de ícones	Magisk (20.4)	Root detectado
Samsung Galaxy Note 5	Resurrection Remix (6.2.1-20181007)	Magisk (21.0)	Root detectado
Redmi Note 4	Resurrection Remix (6.2.1-20181107)	Magisk (20.4) com Magisk Manager com *package name* aleatório	Root detectado
Moto G4 Plus	Resurrection Remix (6.2.0-20180916)	Aparelho rooteado sem o Magisk	Root não detectado

Finalmente, a Figura 5 apresenta a única tela do aplicativo desenvolvido como prova de conceito, que mostra as distâncias calculadas para cada ícone analisado e no final o resultado. Pode-se observar que o Magisk Manager foi encontrado mesmo estando com um *package name* aleatório.

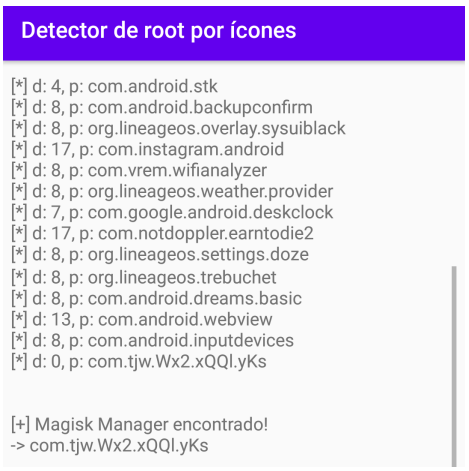

Figura 5: Aplicativo prova de conceito.

Observou-se após a realização dessa pesquisa que o Magisk começou a trocar também o ícone do aplicativo para um genérico do Android. Apesar disso, o novo ícone ainda pode ser detectado e analisado com o *package name* do aplicativo para indicar a presença Magisk. Principalmente se utilizado para gerar uma pontuação que, com outras técnicas, ajuda a chegar em um resultado final mais preciso da avaliação do dispositivo.

5. Conclusões e Trabalhos Futuros

Habilitar o acesso *root* em dispositivos móveis é um procedimento que gera consequências negativas do ponto de vista de segurança. Portanto, os desenvolvedores Android começaram a utilizar técnicas para detectar o *root*, e posteriormente, surgiram métodos para realizar a evasão dessas técnicas.

O objetivo geral deste trabalho foi investigar o estado da arte e da prática das técnicas para detecção de *root*, avaliar a eficácia delas contra os métodos de evasão e propor melhorias para o lado da detecção. Na revisão da literatura em busca de técnicas de evasão e detecção de *root*, foram encontradas 6 principais técnicas, além de um serviço do Google, para a detecção e o Magisk e algumas outras ferramentas para a evasão.

Os resultados do *survey,* realizado para identificar técnicas utilizadas por desenvolvedores na detecção e por usuários na evasão, indicaram que a detecção não está tão madura quanto à evasão, e também, que o Magisk é a ferramenta mais usada para esconder o *root*. Esses resultados são corroborados pelos dados dos estudos empíricos realizados para avaliar a atual eficácia da detecção contra métodos de evasão identificados, visto que apenas um aplicativo, dentre centenas analisados, conseguiu detectar o *root*. Finalmente, sobre características de aparelhos rooteados que poderiam ser utilizadas para melhorar a detecção, propõe-se utilizar duas particularidades que podem ser usadas para detectar o Magisk, que são um arquivo criado por ele e o ícone de seu aplicativo.

Foi possível observar que a evasão está em vantagem na disputa contra a detecção, com o Magisk conseguindo esconder o *root* contra quase todos aplicativos. Adicionalmente, as implementações das técnicas de detecção ainda parecem pouco maduras, visto que além de não serem eficazes, a preocupação com elas ocorre, em sua

maioria, no nicho de aplicativos financeiros. Finalmente, foram sugeridas duas abordagens para melhorar a detecção, que se mostraram eficazes contra o Magisk.

Em relação a trabalhos futuros, pode ser interessante realizar entrevistas com desenvolvedores que já trabalharam detectando *root*, em busca de extrair mais informações sobre suas experiências. Outra investigação importante é repetir o estudo empírico realizado com os aplicativos mais baixados, porém agora tentando utilizar credenciais e meios para verificar detecções de *root* silenciosas. Por fim, um trabalho futuro valioso é o de procurar heurísticas que permitam a busca por nome e conteúdo de arquivos de forma automatizada, sem a avaliação direta de um humano sobre falsos-positivos.

Referências

CASATI, Luca; VISCONTI, Andrea. The Dangers of Rooting: Data Leakage Detection in Android Applications. Mobile Information Systems, 2018.

EVANS, Nathan; BENAMEUR, Azzedine; SHEN, Yun. All your Root Checks are Belong to Us: The Sad State of Root Detection. Proceedings of the 13th ACM International Symposium on Mobility Management and Wireless Access, p. 81-88. 2015.

KASPERSKY. Rooting your Android: Advantages, disadvantages, and snags. Disponível em < https://www.kaspersky.com/blog/android-root-faq/17135/ >. Acessado em 29 de março de 2020.

KOTIPALLI, Srinivasa Rao; IMRAN, Mohammed. Hacking Android. Packt Publishing, 2016.

NGUYEN-VU, Long; CHAU, Ngoc-Tu; KANG, Seongeun; JUNG, Souhwan. Android rooting: An arms race between evasion and detection. Security and Communication Networks, 2017.

POLISCIUC, R.; ALBINI, L.; Grégio, A.; BONA, L. Análise de Aplicativos no Android utilizando Traços de Execução. Simpósio Brasileiro de Segurança da Informação, 2020.

STATISTA. Smartphone users worldwide 2020. Disponível em < https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/> . Acessado em 29 de março de 2020.

SUN, San-Tsai; CUADROS, Andrea; BEZNOSOV, Konstantin. Android rooting: Methods, detection, and evasion. Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices, p. 3-14. 2015.

TERMUX. Termux root packages. Disponível em < https://github.com/termux/termux-root-packages/tree/master/packages>. Acessado em 16 de maio de 2020.

WOHLIN, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. Experimentation in software engineering. Springer Science & Business Media, 2012.