MIRAK: Um Artefato para Robustecimento do Ambiente *Relying Party* RPKI

Yuri de Abreu de Melo[1], Frederico Sauer G. Oliveira[2], Hugo Batalha Moreno[2]
Anderson Fernandes Pereira dos Santos[1,4], Ronaldo Moreira Salles[1,3]

[1]Seção de Engenharia de Defesa - Instituto Militar de Engenharia (IME)

[2]Universidade do Estado do Rio de Janeiro (UERJ)

[3]CIICESI, ESTG, Instituto Politécnico do Porto (IPP)

[4]Venturus Centro de Inovação Tecnológica

{abreumelo,salles,anderson}@ime.eb.br, batalha.hugo@outlook.com

frederico.oliveira@uerj.br

Abstract. *Route validation through Routinator and the RPKI protocol has been discussed in the literature as the main approach to strengthening BGP routing. However, some studies have highlighted potential attack vectors targeting the route validator itself, justifying efforts to enhance the resilience of this solution. This article presents the MIRAK application, developed with proprietary techniques for low resource consumption and high speed, which automatically identifies vulnerabilities in Routinator, contributing to reduced attack risk. Initial results have been promising, encouraging further study to improve its efficiency and scope.*

Resumo. *A validação de rotas através do Routinator e o protocolo RPKI vem sendo discutidos na literatura como a principal combinação para o robustecimento do roteamento BGP. No entanto, alguns trabalhos evidenciaram oportunidades de ataques ao próprio sistema validador de rotas, justificando o esforço no aumento da resiliência dessa solução. Este artigo apresenta a aplicação MIRAK, desenvolvida com técnicas próprias para baixo consumo de recursos e rapidez, que realiza de forma automatizada a identificação de vulnerabilidades no Routinator, contribuindo para reduzir o risco de ataques. Os resultados iniciais se mostraram animadores, motivando o estudo para aprimorar a sua eficiência e amplitude.*

1. Introdução

Tipicamente, o processo de identificação de um alvo para ataque por uma ameaça envolve a evidenciação preliminar de vulnerabilidades. O que torna um ataque viável é o *gap* de tempo entre a descoberta da "brecha" e a disponibilização de uma correção, denominada *patch*. A isso, soma-se ainda o tempo até o usuário aplicar esta correção. Em caso de roteadores, essa atualização é raramente realizada, porque qualquer paralisação da rede é indesejada pelo negócio. Quando se trata de roteamento de WAN para borda de ASes *(Autonomous Systems)*, essa inércia comportamental fica mais evidente, já que a interrupção do serviço de roteamento suspenderia a conectividade de uma ou mais organizações. Dessa

forma, este trabalho representa um avanço no estado da arte, ao direcionar as soluções de defesa de forma proativa para o ambiente que hospeda o *software* validador RPKI (*Resource Public Key Infrastructure*), chamado de *Relying Party*, identificando vulnerabilidades passíveis de correção pelo operador de rede.

Até o momento da redação deste artigo, não foram encontrados trabalhos com o desenvolvimento de soluções específicas para o robustecimento do ambiente *Relying Party*. Nenhum dos vários *scanners* de rede testados indicaram as vulnerabilidades existentes no *software* validador. Este trabalho propõe o *scanner* MIRAK, desenvolvido com o uso de técnicas para baixo consumo de recursos. Com o MIRAK, inicialmente é feita uma identificação do ambiente validador, extraindo informações do Sistema Operacional e aplicações instaladas. Em seguida, ele filtra os dados coletados, correlacionando-os com informações públicas de *Common Vulnerabilities and Exposures* (CVE) existentes. Ao final, indica as vulnerabilidades existentes, bem como o nível de criticidade de cada uma.

2. Contextualização e Motivação

Para robustecer o roteamento global, uma das propostas mais discutidas na literatura é o RPKI, que provê autenticação de anúncios de rotas através de um mecanismo de certificação digital simplificado [Yang et al. 2024]. O Nic.br e a comunidade acadêmica global vêm promovendo sua adoção, mas o RPKI apresenta vulnerabilidades passíveis de exploração, sendo o *Relying Party* um alvo potencial de ataques [Rodday et al. 2024].

A infraestrutura RPKI se baseia na associação de chaves públicas a prefixos IP, e na vinculação desses prefixos a números de Sistemas Autônomos. Isso permite a geração dos *Route Origin Authorizations* (ROAs). Para acessar e validar tais informações, os ASes utilizam *softwares* validadores responsáveis por recuperar os ROAs em infraestruturas externas, e mantê-los em *cache* local. Os roteadores de borda BGP, por sua vez, comparam essas informações com os anúncios de atualização de rotas recebidos de outros ASes em um processo conhecido como ROV (*Route Origin Validation*). Quando há um ROA autenticando uma rota, o respectivo anúncio BGP é classificado como válido. Caso as informações fornecidas pelo ROA sejam incompatíveis com os dados recebidos, o anúncio é considerado inválido, sugerindo uma possível tentativa de sequestro de prefixos BGP [Fontugne et al. 2023]. A possibilidade de inserção maliciosa de rotas em um validador é um risco real já demonstrado, através de um recurso chamado SLURM (*Simplified Local Internet Number Resource Management*) [Melo et al. 2022].

3. Análise do Estado da Arte

À medida que a adoção RPKI aumenta, há preocupação em garantir a sua resiliência operacional. Três fatores contribuem para a ocorrência de vulnerabilidades nos *softwares* validadores [Mirdita et al. 2024]:

- Ausência de *patches* para correções de falhas;
- Utilização de versões desatualizadas pelos operadores de rede, mesmo quando há atualizações disponíveis; e
- Emprego de validadores descontinuados.

Tabela 1. Abordagens sobre a temática vulnerabilidades do RPKI (os autores).

Trabalhos	Vetor de ataque
[Mirdita et al. 2022]	Exploração de *zero-day*
[Hlavacek et al. 2022a]	Desabilitação ROV via DNS
[Hlavacek et al. 2022b]	Paralisação do *software Relying Party*
[Melo et al. 2022]	Introdução do SLURM no *Relying Party*
[Van Hove et al. 2023]	Vulnerabilidades no *software Relying Party*
[Fontugne et al. 2023]	Atraso no ecossistema RPKI
[Jacobsen et al. 2024]	Execução remota de código

De acordo com os pesquisadores, essas vulnerabilidades afetam 100% dos validadores Routinator e Fort, sendo o Routinator responsável por 70,5% do mercado global de validadores. A Tabela 1 faz uma síntese dos trabalhos relacionados.

Mirdita et al. (2022) analisa o código de duas versões populares do Routinator, 0.10.2 e 0.11.2, identificando vulnerabilidades *zero-day*, capazes de manipular a biblioteca rust-base64. Os autores destacam que essa vulnerabilidade afeta cerca de 84,9% das redes que implementam a validação RPKI.

Hlavacek et al. (2022a) indicam que os servidores DNS utilizados pelos validadores para localizar pontos de publicação frequentemente estão em redes cujos prefixos não possuem cobertura por ROAs, tornando-os vulneráveis a sequestros de prefixos.

Hlavacek et al. (2022b) evidenciaram um ataque capaz de desabilitar a validação RPKI em roteadores BGP através da paralisação do validador com caminhos de delegação longos e arbitrários. Caso o validador não consiga estabelecer conexão com o repositório, os ROAs em *cache*, ao expirar, deixam de ser aplicados, desativando a validação RPKI.

Van Hove et al. (2022) analisa oito implementações de validadores, elencando ataques possíveis, como o esgotamento de memória ao introduzir ROAs para cada um dos prefixos IPv6 existentes, ataques *Man-In-The-Middle*, além de vulnerabilidades decorrentes de omissões nas RFCs.

Fontugne et al. (2023) apontou uma grande disparidade temporal entre a criação ou exclusão de um ROA e seu uso. Essa disparidade pode causar conflitos de roteamento, uma vez que dados obsoletos podem ser utilizados para decisões de roteamento.

Jacobsen et al. (2024) apresentam uma vulnerabilidade de *buffer overflow* nos validadores Fort, capaz de permitir a execução remota de código através da introdução de um *payload* malicioso.

Melo et al. (2022) evidenciaram e testaram a introdução maliciosa de um arquivo SLURM como método para ataque ao *software* validador RPKI. Mais recentemente, foi publicado que regras arbitrárias do SLURM podem tornar os ASes vulneráveis a sequestros de prefixos [Schulmann et al. 2024].

4. O Artefato MIRAK

Buscar a resiliência de sistemas que implementam o RPKI não é trivial, dado o conjunto de interdependências com outros componentes [Mirdita et al. 2024]. Neste trabalho, é apresentado o *software* MIRAK, que na versão atual é capaz de analisar o ambiente que

hospeda o validador RPKI Routinator, identificando vulnerabilidades locais passíveis de exploração.

O artefato foi organizado em duas aplicações associadas. A primeira, Mirak-extractor, abrange os processos voltados à obtenção de informações sobre o Sistema Operacional em que o validador RPKI está sendo executado. Sua saída fornece um arquivo estruturado em formato JSON denominado MIRAK. Através desse arquivo, a segunda aplicação, Mirak-app, faz a correlação das informações CVE obtidas com pesquisas automatizadas na *National Vulnerability Database* (NVD) com as características do ambiente que hospedará o validador de rotas, incluindo o Sistema Operacional, o validador e as aplicações eventualmente instaladas.

4.1. Mirak-extractor

O Mirak-extractor é uma ferramenta que diagnostica as características do ambiente hospedeiro *Relying Party* RPKI. É um sistema automatizado, identificando as aplicações instaladas e detalhes operacionais do ambiente em pouco tempo, gerando o arquivo MIRAK, que permite análises por outras aplicações, como o Mirak-app. Foi desenvolvido em Python com o uso do padrão *Singleton*, com baixo impacto em requisitos para instalação e execução. O arquivo MIRAK contém as características do ambiente necessárias para a pesquisa de CVEs correspondentes. Em sua versão atual, oferece suporte aos principais Sistemas Operacionais utilizados pelo Routinator, como Ubuntu 16.04, Debian 10 e Red Hat Enterprise Linux 9.5, ou superiores.

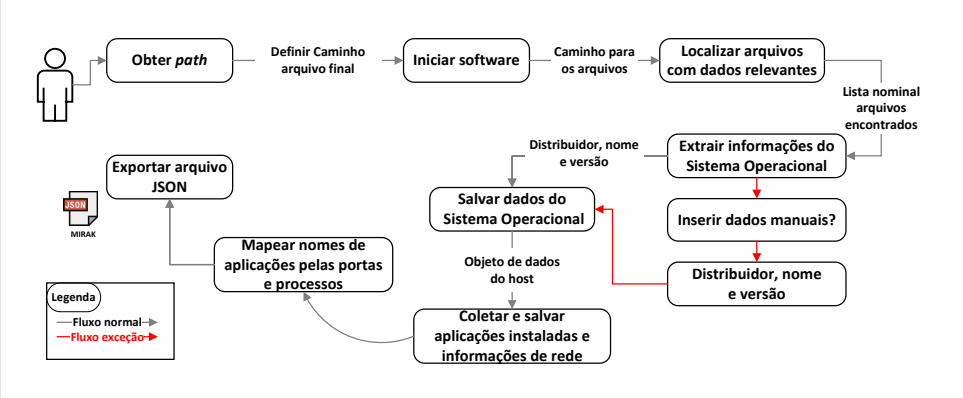

Figura 1. Mirak-extractor (os autores).

O fluxo operacional do Mirak-extractor, ilustrado na Figura 1, se inicia com a definição pelo usuário do local para o arquivo de saída (MIRAK). Em seguida, é conduzida uma varredura dos programas instalados e dos processos em execução no Sistema Operacional, com a identificação dos respectivos CPEs (*Common Platform Enumeration*).

Na etapa seguinte, são coletados dados referentes às portas de rede, estabelecendo uma correlação com os processos ou serviços que fazem uso delas. Por fim, ocorre a estruturação das informações e a exportação dos dados em formato JSON.

A varredura realizada pelo MIRAK vai além da análise do Sistema Operacional e aplicações, destacando-se pelo cuidado com o validador RPKI. O artefato atual identifica o Routinator, configurações, bibliotecas e serviços de rede, além de processos não

essenciais, como telnet ou http em estado *listening*, que podem representar riscos à função dedicada de validação — algo que as ferramentas generalistas não contemplam.

4.2. Mirak-app

O Mirak-app é uma aplicação de processamento e análise que utiliza para a busca de CVEs o arquivo MIRAK, já com os identificadores CPE estruturados, permitindo assim uma busca direcionada por CVEs na NVD. A Figura 2 ilustra o fluxo operacional do Mirak-app.

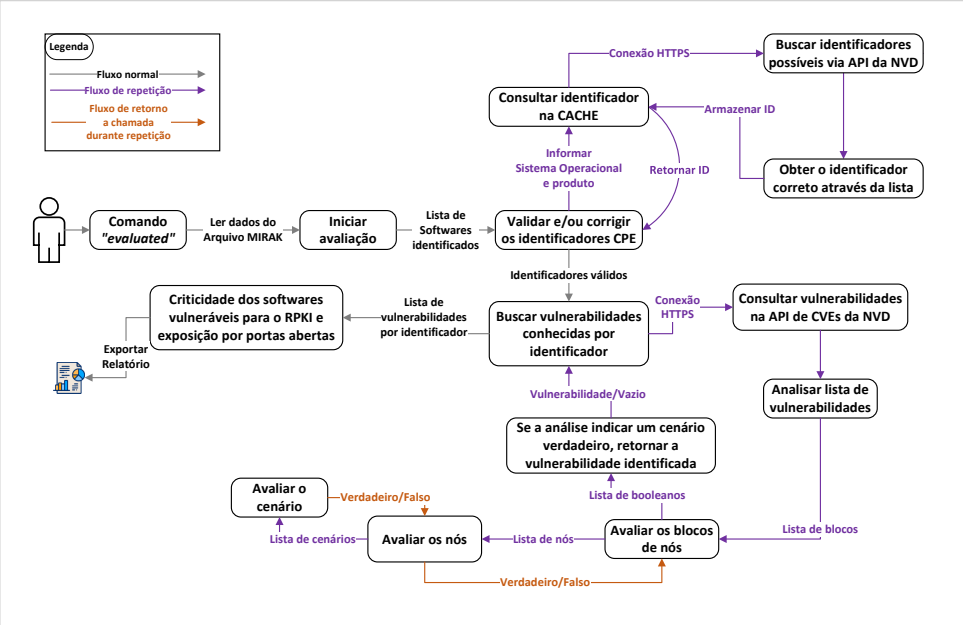

Figura 2. Mirak-app (os autores).

Inicialmente, o algoritmo verifica na *cache* se os identificadores CPEs extraídos apresentam dados faltantes, corrigindo-os se necessário. Em seguida, a lista de identificadores é submetida como *input* para a busca na NVD, obtendo-se assim as vulnerabilidades conhecidas para cada entrada CPE presente no arquivo MIRAK.

Estabelecer correlações entre informações CPE obtidas e identificar a presença de vulnerabilidades com base nos dados fornecidos pela NVD não é uma tarefa simples, porque as entradas possuem variações arbitrárias de formato, e este é o maior desafio para o desenvolvimento de um *scanner*. Para aferir a possibilidade de existência de uma vulnerabilidade, foram desenvolvidas comparações sucessivas baseadas em operadores lógicos.

Para lidar com essa imprecisão, o artefato usa a rotina *evaluate*, que utiliza a estrutura da CPE para buscar entradas onde haja correspondência com as características do ambiente obtido pelas informações contidas no arquivo MIRAK, com atenção especial às particularidades do *software* validador Routinator. Finalmente, é apresentado o resultado em tela conforme Figura 3.

Caso sejam encontradas vulnerabilidades, um relatório como o ilustrado na Figura 4 é exportado contendo informações para identificar o *software*, a vulnerabilidade, o nível de criticidade no padrão CVSS (*Common Vulnerability Scoring System*), a relevância do

```
⚠️ Warning: The host environment with IP 172.18.0.2 contains vulnerabilities that may affect correct operation.
A total of 104 vulnerabilities were discovered


-> Number of RPKI-related vulnerabilities found: 4
-> Number of other vulnerabilities found: 100
-> Number of ports not used by the RPKI solution discovered: 3
```

Figura 3. Interface de linha de comando do Mirak-app (os autores).

software dentro do contexto de execução para o RPKI e a relação das portas de rede com o RPKI.

Tabela_1

product	vendor	type	version	cve_id	description	base_score	base_severity	software_required	related_port	port_required
routinator	nlnetlabs	Aplication	0.11.1rc1-1focal	CVE-2022-3029	In NLnet Labs Ro	07/05/2025	HIGH	yes	yes	yes
routinator	nlnetlabs	Aplication	0.11.1rc1-1focal	CVE-2023-39915	NLnet Labs' Routi	07/05/2025	HIGH	yes	yes	yes
routinator	nlnetlabs	Aplication	0.11.1rc1-1focal	CVE-2023-39916	NLnet Labs' Rout	09/03/2025	CRITICAL	yes	yes	yes
routinator	nlnetlabs	Aplication	0.11.1rc1-1focal	CVE-2024-1622	Due to a mistake	07/05/2025	HIGH	yes	yes	yes
rsync	samba	Aplication	3.1.3-8ubuntu0.9	CVE-2022-29154	An issue was disc	07/04/2025	HIGH	yes	no	no

Figura 4. Relatório de saída MIRAK (os autores).

5. Metodologia de Validação

As ferramentas de enumeração de vulnerabilidades avaliadas, como o Vuls, Trivy e Grype não verificaram os validadores RPKI, mas apenas as aplicações amplamente conhecidas. A finalidade genérica dessas soluções, bem como o custo eventual e indisponibilidade de código-fonte ou APIs encorajou o investimento no desenvolvimento do MIRAK.

Para testar o MIRAK, foram criadas imagens em *Docker* simulando diferentes versões do Routinator em um ambiente Ubuntu 20.04 típico, e foram executadas as aplicações MIRAK, Vuls, Trivy e Grype. Conforme descrito na Tabela 2, apenas o MIRAK foi capaz de detectar na NVD CVEs conhecidas associadas ao Routinator, identificando todas elas. Nos testes iniciais, não foram identificados falsos positivos ou negativos com o uso do MIRAK.

Tabela 2. Comparativo entre MIRAK e ferramentas convencionais (os autores).

Versão Routinator	MIRAK	Vuls	Trivy	Grype
0.8.3	5/5	0/5	0/5	0/5
0.9.3	8/8	0/8	0/8	0/8
0.10.0	7/7	0/7	0/7	0/7
0.11.1	4/4	0/4	0/4	0/4
0.12.1	3/3	0/3	0/3	0/3
0.13.1	1/1	0/1	0/1	0/1

6. Conclusão e trabalhos futuros

Este artigo descreveu um artefato que foi desenvolvido com o objetivo de reduzir as oportunidades de ataques ao BGP através do uso de vulnerabilidades no validador de rotas

RPKI ou do sistema hospedeiro. De forma automatizada, o MIRAK identifica o *Relying Party* e seus componentes, busca as CVE pertinentes e identifica os módulos que precisam ser atualizados ou retirados. Os resultados do teste de validação foram animadores, justificando ajustes no algoritmo visando aumentar a sua eficiência e amplitude. Uma nova versão implementará técnicas ajustadas de correlação entre os identificadores CPE e os CVE da NVD, bem como o suporte a outras versões de Sistema Operacional e *softwares* validadores.

No aspecto experimental, até o momento, os testes foram conduzidos em ambientes controlados de laboratório. Em fases futuras, está prevista a realização de testes em cenários com topologias e cargas mais próximas das observadas em ambientes de produção reais. Essa etapa incluirá análises estatísticas mais robustas, com o objetivo de avaliar com maior precisão tanto o impacto operacional quanto a taxa de identificação de vulnerabilidades. Também está contemplada a implementação de ações automatizadas de mitigação, como *scripts* de atualização, além do desenvolvimento de uma interface gráfica destinada a facilitar a adoção por operadores de rede, aumentando a usabilidade da solução.

O código-fonte, documentação, licença e implementação estão disponíveis no GitHub, no link: https://github.com/hugo-bm/SBSeg25ArtigoMIRAK.

Referências

Fontugne, R., Phokeer, A., Pelsser, C., Vermeulen, K., and Bush, R. (2023). RPKI Time-of-Flight: Tracking Delays in the Management, Control, and Data Planes. In *Passive and Active Measurement: 24th International Conference, PAM 2023, Virtual Event, March 21–23, 2023, Proceedings*, page 429–457, Berlin, Heidelberg. Springer-Verlag.

Hlavacek, T., Jeitner, P., Mirdita, D., Shulman, H., and Waidner, M. (2022a). Behind the Scenes of RPKI. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 1413–1426, New York, NY, USA. Association for Computing Machinery.

Hlavacek, T., Jeitner, P., Mirdita, D., Shulman, H., and Waidner, M. (2022b). Stalloris: RPKI Downgrade Attack. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4455–4471, Boston, MA. USENIX Association.

Jacobsen, O., Schulmann, H., Vogel, N., and Waidner, M. (2024). Poster: From Fort to Foe: The Threat of RCE in RPKI. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 5015–5017, New York, NY, USA. Association for Computing Machinery.

Melo, Y., Salles, R., and Oliveira, F. (2022). Validação da solução RPKI para segurança do BGP. In *Anais Estendidos do XXII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 41–48, Porto Alegre, RS, Brasil. SBC.

Mirdita, D., Schulmann, H., and Waidner, M. (2024). SoK: An Introspective Analysis of RPKI Security. Technical report. Disponível em: `https://arxiv.org/abs/2408.12359`.

Mirdita, D., Shulman, H., and Waidner, M. (2022). Poster: RPKI Kill Switch. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications*

Security, CCS '22, page 3423–3425, New York, NY, USA. Association for Computing Machinery.

Rodday, N., Cunha, I., Bush, R., Katz-Bassett, E., Rodosek, G. D., Schmidt, T. C., and Wählisch, M. (2024). The Resource Public Key Infrastructure (RPKI): A Survey on Measurements and Future Prospects. *IEEE Transactions on Network and Service Management*, 21(2):2353–2373.

Schulmann, H., Vogel, N., and Waidner, M. (2024). RPKI: Not Perfect But Good Enough. Technical report. Disponível em: `https://arxiv.org/abs/2409.14518`.

Van Hove, K., van der Ham-de Vos, J., and van Rijswijk-Deij, R. (2023). rpkiller: Threat Analysis of the BGP Resource Public Key Infrastructure. volume 4, New York, NY, USA. Association for Computing Machinery.

Yang, Q., Ma, L., Tu, S., Ullah, S., Waqas, M., and Alasmary, H. (2024). Towards Blockchain-Based Secure BGP Routing, Challenges and Future Research Directions. *Computers, Materials & Continua*, 79(2):2035–2062.