AFRIKA
TECNOLOGIA E NEGÓCIO

Análise de Exposição

VULNERABILIDADE EM ANÁLISE: MIASMA SUPPLY CHAIN ATTACK

POR RAFAEL FREITAS | 05/06/205

AFRIKA
TECNOLOGIA E NEGÓCIO

Sumário

1. Resumo executivo

Em 1º de junho de 2026, foi divulgado o comprometimento de **32 pacotes do namespace @redhat-cloud-services no npm (mais de 90 versões)** por um worm de roubo de credenciais batizado de **"Miasma"**. O código malicioso é executado automaticamente durante a instalação dos pacotes, antes mesmo de a aplicação ser usada, e coleta credenciais e segredos presentes em estações de desenvolvimento, máquinas de build e pipelines de CI/CD.

O ponto mais importante para qualquer organização é: **o risco não depende do uso da biblioteca em produção.** Basta o pacote afetado estar presente na árvore de dependências (direta ou transitivamente) e a instalação ter ocorrido em algum host.

A campanha foi rapidamente contida: as versões maliciosas foram removidas, a publicação no namespace foi restringida e os tokens com permissão de escrita foram invalidados. A Red Hat confirmou que nenhum produto oficial foi distribuído com as versões comprometidas. Ainda assim, todo ambiente que tenha instalado uma versão afetada deve tratar suas credenciais como potencialmente expostas e seguir as ações deste boletim.

Panorama rápido	
Vetor	Cadeia de suprimentos (pacotes npm)
Execução	Script preinstall, durante a instalação (npm install / npm ci)
Impacto	Roubo de credenciais e segredos; propagação automática tipo worm
Plataformas	Linux, macOS e Windows (runners de CI/CD Linux como alvo principal)
Severidade	Alta
Status	Contido — versões maliciosas removidas e publicação no namespace restringida

2. O que aconteceu

A campanha identificada pelos pesquisadores como "Miasma: The Spreading Blight", é uma variante da família de malware de cadeia de suprimentos conhecida como Mini Shai-Hulud. A origem foi o **comprometimento da conta GitHub de um colaborador da Red Hat**, que permitiu injetar workflows maliciosos do GitHub Actions em repositórios de origem e abusar do fluxo legítimo de publicação (via OIDC/trusted publisher). Como resultado, os pacotes maliciosos foram publicados com assinaturas de proveniência autênticas e atestações SLSA forjadas, o que os fez parecer legítimos.

Foram afetados 32 pacotes e mais de 90 versões, com média aproximada de 80 mil downloads semanais. A divulgação ocorreu poucas horas após a publicação e a resposta foi coordenada com o npm e a comunidade de segurança. Vale registrar que a **família segue ativa**: novas variantes do mesmo padrão foram observadas nos dias seguintes, o que reforça a necessidade de monitoramento contínuo das dependências.

3. Como o ataque funciona

O Miasma combina quatro características que o tornam perigoso: executa-se sozinho na instalação, disfarça-se de pacote legítimo, rouba um amplo conjunto de credenciais e se autopropaga. A seguir, a anatomia do ataque.

3.1. Origem e disfarce

O comprometimento começou na conta GitHub de um colaborador da Red Hat. Com esse acesso, o atacante injetou workflows maliciosos do GitHub Actions nos repositórios de origem dos pacotes e abusou do fluxo de publicação confiável (trusted publisher via OIDC). Na prática, os pacotes maliciosos foram publicados pela própria esteira oficial, herdando assinaturas de proveniência autênticas, e o malware ainda **forjou atestações SLSA** para parecer legítimo. É isso que torna o ataque especialmente traiçoeiro: não houve typosquatting nem pacote falso — foram os pacotes verdadeiros, publicados pelo canal correto, que passaram a carregar código malicioso.

3.2. Cadeia de execução na instalação

A infecção dispara automaticamente no npm install / npm ci, por um gancho **preinstall**, sem qualquer ação do usuário e antes de a aplicação rodar — valendo tanto para dependências diretas quanto transitivas. As principais fases:

Fase	O que acontece
Entrega e execução	Gancho preinstall executa o dropper (index.js de ~4,3 MB) automaticamente na instalação, sem ação do usuário.
Desofuscação e Bun	Várias camadas de ofuscação (ROT, AES-128-GCM, ofuscador de strings e cifra própria) são desfeitas; o malware baixa o runtime Bun e executa a carga fora do Node (cadeia node → shell → bun) para escapar de detecções focadas em Node.
Validação de ambiente	Verifica locale/região e pode restringir a execução a ambientes de CI/CD.

Fase	O que acontece
Roubo de credenciais	Coleta tokens e segredos de GitHub, npm, AWS, Azure, GCP, Vault e Kubernetes; em runners, lê a memória do processo para capturar segredos mascarados; também busca chaves SSH, dados de navegador e carteiras.
Escalada de privilégios	Cria regra de sudo sem senha em runners para obter privilégios elevados.
Exfiltração	Envia os dados por três canais, abusando da infraestrutura do GitHub e alternando entre dezenas de contas para dificultar o bloqueio.
Propagação (worm)	Com tokens npm roubados, republica outros pacotes do mantenedor e injeta código em repositórios da vítima, reacendendo a cadeia.
Armadilha destrutiva	Planta um "token isca" cuja manipulação pode disparar o apagamento do diretório do usuário.

3.3. Propagação tipo worm e persistência

O diferencial do Miasma é a autopropagação. Com os tokens npm roubados, ele republica — com proveniência forjada, outros pacotes mantidos pela vítima e injeta o próprio código em repositórios dela, de modo que o próximo build reacende a cadeia. A exfiltração não usa um único servidor fácil de bloquear: abusa do próprio GitHub (cria repositórios públicos com os dados roubados e comita em repositórios da vítima) e alterna entre dezenas de contas controladas pelo atacante. Para manter acesso, instala persistência e um monitor de tokens. Há ainda um **"token isca"**: sua revogação ou manipulação precipitada pode acionar uma rotina destrutiva no host (apagamento do diretório do usuário) — por isso a ordem das ações de resposta importa (Seção 8).

Abrangência e raio de alcance

O alcance da campanha vai além dos pacotes diretamente comprometidos:

- **Escala direta.** 32 pacotes e mais de 90 versões sob o namespace @redhat-cloud-services, com média de aproximadamente 80 mil downloads semanais por pacote.
- **Amplificação transitiva.** Como muitos desses pacotes são dependências de outros componentes, o alcance potencial se estende a milhares de projetos a jusante que sequer os utilizam diretamente.
- **Multiplataforma.** A carga funciona em Linux, macOS e Windows, baixando o runtime adequado a cada sistema; os runners de CI/CD em Linux foram o alvo principal.
- **Efeito worm.** A autopropagação amplia o alcance de forma dinâmica, contaminando outros pacotes do mantenedor e repositórios da vítima.

- **Parte de uma onda maior.** O Miasma é uma variante da família Mini Shai-Hulud, cujo ferramental foi disponibilizado publicamente; campanhas semelhantes têm atingido diversos ecossistemas e novas variantes surgiram poucos dias depois. É uma ameaça recorrente, não um evento isolado.
- **Quem está no escopo.** Qualquer organização com aplicações ou esteiras Node.js/npm que tenha instalado uma versão afetada — direta ou transitivamente — em estações de desenvolvimento, ambientes de build ou pipelines.

5. Impacto potencial

As consequências de uma exposição se distribuem em várias frentes:

- **Credenciais e segredos.** Tokens de GitHub e npm, segredos de pipeline (GitHub Actions), chaves de AWS, Azure e GCP, tokens de Vault e Kubernetes, chaves SSH, credenciais de registries e de ferramentas de desenvolvimento.
- **Acesso a nuvem e infraestrutura.** Com as credenciais obtidas (inclusive via endpoints de metadados/IMDS), o atacante pode acessar ambientes de nuvem, dados e serviços com os privilégios das identidades comprometidas.
- **Repositórios e cadeia de entrega.** Tokens de GitHub e npm permitem manipular repositórios e publicar novas versões maliciosas, transformando a própria vítima em vetor de propagação.
- **Confiança em proveniência.** A forja de atestações SLSA enfraquece um dos principais mecanismos de confiança da cadeia de suprimentos, exigindo verificação para além da proveniência.
- **Risco operacional.** A armadilha destrutiva pode levar à perda de dados em estações de desenvolvimento se a resposta for conduzida na ordem errada.
- **Exposição regulatória e reputacional.** O vazamento de credenciais e segredos pode acionar obrigações de resposta a incidentes e de comunicação, além de impacto reputacional.

6. Por que isso importa para o seu ambiente

A exposição não se mede pelo uso da biblioteca em produção. Ela depende de dois fatores: **(1)** o pacote afetado estar presente na árvore de dependências, direta ou transitivamente; e **(2)** a instalação (npm install / npm ci) ter ocorrido em algum host desde 1º de junho de 2026.

Ou seja, mesmo quem não utiliza o pacote diretamente pode ter sido exposto por uma dependência transitiva. E os ambientes mais sensíveis são justamente os de bastidores — **estações de desenvolvimento, máquinas de build e pipelines de CI/CD** —, onde costumam existir credenciais com permissões elevadas.

7. Como identificar a exposição (com ou sem SCA)

A pergunta central é sempre a mesma: **os pacotes e versões afetados do @redhat-cloud-services (Apêndice A) aparecem na árvore de dependências, direta ou transitivamente? E houve execução de instalação em algum host desde 1º de junho?**

Com ferramenta de SCA (Veracode, Snyk e similares)

Rode o inventário de dependências (direto e transitivo) de cada aplicação e compare com a lista de pacotes e versões afetados do Apêndice A. É o caminho mais direto quando a ferramenta já está disponível.

Sem SCA — alternativas equivalentes, conforme o que já existir no ambiente

- **Lockfiles.** Rode scanners gratuitos (OSV-Scanner, Trivy ou Grype) sobre package-lock.json, yarn.lock e pnpm-lock.yaml. Por projeto, use npm ls @redhat-cloud-services, npm why <pacote> e npm audit. Em último caso, uma busca textual pelo namespace nos lockfiles, executada em todos os repositórios.
- **GitHub.** O dependency graph e o Dependabot sinalizam automaticamente; a busca de código por organização localiza o namespace em qualquer package.json ou lockfile.
- **Repositório de artefatos.** Em Artifactory, Nexus ou equivalente, os logs de download dos pacotes/versões afetados mostram exatamente quais builds e estações os baixaram.
- **Estações e agentes de build.** Varredura de node_modules e do cache do npm em busca do pacote e da versão afetada.
- **EDR e logs de rede.** Correlação dos indicadores do Apêndice B — execução do Bun a partir de diretórios temporários, acesso a endpoints de metadados de nuvem (169.254.169.254) por processos de instalação e criação de repositório público chamado “Miasma: The Spreading Blight”.

8. Recomendações

8.1. Se você identificar exposição

1. Identifique os hosts (estações, máquinas de build e runners de CI/CD) que instalaram ou construíram versões afetadas na janela de exposição.

> **Atenção! A ordem importa**
> Antes de revogar tokens, remova os mecanismos de persistência e encerre os processos do Bun relacionados ao incidente. A revogação precipitada do “token isca” plantado pelo malware pode disparar uma rotina destrutiva no host. Só então prossiga com a rotação de credenciais.

2. Remova qualquer versão afetada; fixe os lockfiles em uma versão sabidamente segura (anterior); limpe os caches do npm e reconstrua a partir de um estado limpo.
3. Rotacione todas as credenciais acessíveis ao ambiente afetado: tokens npm e GitHub (PATs e fine-grained), segredos do GitHub Actions, chaves AWS, GCP e Azure, tokens de Vault e Kubernetes, chaves SSH e credenciais de registries/Docker. Mesmo com os tokens de publicação já invalidados pelo npm, rotacione por precaução.

4. Audite as contas GitHub em busca de repositórios públicos chamados “Miasma: The Spreading Blight” ou criados na janela de exposição; revise os logs de CI/CD por conexões externas e execuções de script inesperadas; valide os lockfiles e a proveniência dos pacotes.

8.2. Prevenção e boas práticas contínuas

- **Desabilite a execução de scripts de ciclo de vida na instalação** (npm install --ignore-scripts) onde for viável, especialmente em CI/CD.
- **Evite atualizações automáticas de dependências;** fixe versões e revise antes de promover para os ambientes.
- **Governança de bibliotecas de terceiros:** mantenha uma lista de pacotes aprovados (allow-list) e um processo de revisão para inclusão e atualização.
- **Adote SCA e monitoramento contínuo** da árvore de dependências, com scans recorrentes.
- **Menor privilégio para segredos de CI/CD:** segregue credenciais, reduza escopos e prefira credenciais efêmeras (OIDC) com permissões mínimas.
- **Verifique proveniência e use lockfiles versionados** como parte do controle de qualidade da esteira.

9. Como a Afrika pode ajudar

A Afrika acompanha esta campanha e atua em todo o ciclo — da verificação à prevenção —, **inclusive em ambientes que não utilizam ferramenta de SCA**. Nossas frentes de apoio e as tecnologias que as sustentam:

- **Verificação de exposição e AppSec.** Levantamento da árvore de dependências (direta e transitiva) em aplicações, repositórios e pipelines, com ou sem SCA, e identificação de pacotes maliciosos - com apoio de plataformas de segurança de aplicações como o Aikido.
- **SCA e governança de dependências com Veracode**. Como parceira Veracode, a Afrika pode apoiar na utilização da plataforma para identificação de componentes open source, análise de dependências diretas e transitivas, priorização de vulnerabilidades, geração de inventário/SBOM e acompanhamento das ações de remediação. No contexto da campanha Miasma, essa frente permite cruzar os resultados de SCA com a lista de pacotes e versões afetadas, apoiando uma avaliação mais objetiva da exposição das aplicações.
- **Proteção e rotação de credenciais.** Gestão de acesso privilegiado e de segredos, reduzindo a exposição e acelerando a rotação após um incidente - com o Segura (PAM).
- **Detecção de comportamento anômalo.** Identificação de exfiltração, acessos atípicos e processos suspeitos em endpoints e rede - com o Darktrace.
- **Validação de controles.** Simulação de ataques para confirmar se as defesas detectam e bloqueiam cenários como este - com o Cymulate.
- **Resposta a incidente.** Condução da contenção na ordem correta, remoção das versões afetadas, limpeza e reconstrução segura, e plano de rotação de credenciais.
- **Conscientização e capacitação.** Alertas e treinamento dos times de desenvolvimento sobre os riscos da cadeia de suprimentos.

A Afrika atua ainda em outras frentes de segurança de aplicações e ofensiva, como segurança de APIs (**Salt Security**), Pentests e programas de bug bounty e teste contínuo (**Bugcrowd**).

Para acionar nosso time ou ampliar a análise para o seu ambiente, fale com o seu contato na Afrika.

Apêndice A — Pacotes e versões afetados

Lista de referência das versões maliciosas identificadas sob o namespace @redhat-cloud-services. Por se tratar de uma campanha em evolução, consulte sempre os avisos oficiais (Referências) para a relação mais atualizada.

Pacote @redhat-cloud-services/	Versões	Pacote @redhat-cloud-services/	Versões
types	3.6.1, 3.6.2, 3.6.4	frontend-components-utilities	7.4.1, 7.4.2, 7.4.4
frontend-components	7.7.2, 7.7.3, 7.7.5	rbac-client	9.0.3, 9.0.4, 9.0.6
javascript-clients-shared	2.0.8, 2.0.9, 2.0.11	frontend-components-config-utilities	4.11.2, 4.11.3, 4.11.5
frontend-components-notifications	6.9.2, 6.9.3, 6.9.5	tsc-transform-imports	1.2.2, 1.2.4, 1.2.6
frontend-components-config	6.11.3, 6.11.4, 6.11.6	eslint-config-redhat-cloud-services	3.2.1, 3.2.2, 3.2.4
host-inventory-client	5.0.3, 5.0.4, 5.0.6	rule-components	4.7.2, 4.7.3, 4.7.5
frontend-components-remediations	4.9.2, 4.9.3, 4.9.5	frontend-components-translations	4.4.1, 4.4.2, 4.4.4
vulnerabilities-client	2.1.9, 2.1.11	frontend-components-advisor-components	3.8.2, 3.8.4, 3.8.6
entitlements-client	4.0.11, 4.0.12, 4.0.14	chrome	2.3.1, 2.3.2, 2.3.4
notifications-client	6.1.4, 6.1.5, 6.1.7	compliance-client	4.0.3, 4.0.4, 4.0.6
sources-client	3.0.10, 3.0.11, 3.0.13	integrations-client	6.0.4, 6.0.5, 6.0.7
frontend-components-testing	1.2.1, 1.2.2, 1.2.4	remediations-client	4.0.4, 4.0.5, 4.0.7
insights-client	4.0.4, 4.0.5, 4.0.7	topological-inventory-client	3.0.10, 3.0.11, 3.0.13

Pacote @redhat-cloud-services/	Versões	Pacote @redhat-cloud-services/	Versões
config-manager-client	`5.0.4`, `5.0.5`, `5.0.7`	hcc-pf-mcp	`0.6.1`, `0.6.2`, `0.6.4`
quickstarts-client	`4.0.11`, `4.0.12`, `4.0.14`	patch-client	`4.0.4`, `4.0.5`, `4.0.7`
hcc-feo-mcp	`0.3.1`, `0.3.2`, `0.3.4`	hcc-kessel-mcp	`0.3.1`, `0.3.2`, `0.3.4`

Apêndice B — Indicadores de comprometimento (IoCs)

Indicador	Tipo	Descrição
`@redhat-cloud-services`	Escopo npm	Namespace cujos pacotes foram comprometidos na campanha.
`index.js (~4,3 MB)`	Arquivo	Dropper malicioso que substitui o index.js na raiz do pacote.
`/tmp/b-<aleatório>/bun`	Artefato (Linux)	Runtime Bun baixado em diretório temporário; removido após execução, mas pode permanecer em caso de falha.
`/var/folders/.../b-<aleatório>/bun`	Artefato (macOS)	Mesmo comportamento do item acima em estações macOS.
`/tmp/p<base36>.js`	Arquivo	Carga secundária gravada em diretório temporário.
Repositório “Miasma: The Spreading Blight”	GitHub	Repositório público criado na conta da vítima para exfiltração (nomes aleatórios do tipo adjetivo-criatura-N).
`169.254.169.254` / `169.254.170.2`	Rede	Acesso a endpoints de metadados de nuvem (IMDS) por processos node/bun durante a instalação — atípico.

Exemplos de hash (SHA-256) do index.js malicioso:

- `396cac9e457ec54ff6d3f6311cb5cc1da8054d019ce3ffa1de5741506c7a4ea4` — remediations-client
- `d8d170af3de17bb9b217c52aaaffdf9395f35ef015a57ef676e406c121e5e223` — frontend-components-advisor-components 3.8.2
- `25e121e3b7d300c0d0075b33e5eca39a3e6a659fb9cfee52b70ef71686628f1b` — chrome 2.3.4

Observação: cada infecção gera uma carga única, portanto os indicadores baseados em hash valem apenas para versões específicas. Priorize a verificação por inventário de dependências e por comportamento.

Referências

Avisos oficiais e análises de fornecedores de segurança consultados (junho de 2026):

- Red Hat — RHSB-2026-006 (Supply chain compromise of @redhat-cloud-services npm packages) — https://access.redhat.com/security/vulnerabilities/RHSB-2026-006
- Microsoft Security Blog — Preinstall to persistence: Inside the Red Hat npm Miasma campaign — https://www.microsoft.com/en-us/security/blog/2026/06/02/preinstall-persistence-inside-red-hat-npm-miasma-credential-stealing-campaign/
- Wiz — Miasma: Supply Chain Attack Targeting RedHat npm Packages — https://www.wiz.io/blog/miasma-supply-chain-attack-targeting-redhat-npm-packages
- Orca Security — Red Hat npm Packages Compromised in Supply-Chain Attack — https://orca.security/resources/blog/red-hat-npm-supply-chain-attack/
- Aikido — Red Hat npm Packages Compromised to Spread a Credential-Stealing Worm — https://www.aikido.dev/blog/red-hat-npm-packages-compromised-credential-stealing-worm
- Mend.io — Advisory MSC-2026-6085 (RedHat Cloud Services packages) — https://www.mend.io/blog/redhat-cloud-services-packages-drop-multi-cloud-credential-stealer/
- Upwind — Miasma: A Worming npm Supply Chain Attack on Red Hat Cloud Services — https://www.upwind.io/feed/miasma-npm-supply-chain-worm-redhat-credential-harvest
- Rescana — Miasma Supply Chain Attack: Incident Analysis and Mitigation — https://www.rescana.com/post/miasma-supply-chain-attack-compromises-red-hat-redhat-cloud-services-npm-packages-with-credential-stealing-worm-cybersec
- The Hacker News — Miasma Supply Chain Attack Compromises Red Hat npm Packages — https://thehackernews.com/2026/06/miasma-supply-chain-attack-compromises.html
- StepSecurity — Variante via binding.gyp do worm Miasma (campanha relacionada) — https://www.stepsecurity.io/blog/binding-gyp-npm-supply-chain-attack-spreads-like-worm