TLP: CLEAR

Pesquisa de Cibersegurança

Containers em Risco: Como Adversários podem Escapar do Escopo de Containers com a *CVE-2019-5736*

heimdall
security research

Acesse nossa comunidade no WhatsApp, clicando na imagem abaixo!

heimdall
security research
A DIVISION OF ISH

Acesse as análises produzidas pela ISH Tecnologia sobre Táticas, Técnicas e Procedimentos (TTPs) de Threat Actors, malwares emergentes, vulnerabilidades críticas e outros temas relevantes em cibersegurança. Clique na imagem abaixo para conferir nosso blog.

ISH

ALERTA HEIMDALL! HTTP2 RAPID RESET_IMPACTOS E DETECÇÃO DA CVE-2023-44487

Falhas de negação de serviço (DoS) não são apenas interrupções técnicas. Elas representam riscos reais à continuidade do negócio, à confiança dos clientes e à reputação da marca. Nisto temos a vulnerabilidade CVE-2023-44487, conhecida como HTTP/2 Rapid Reset.

BAIXAR

ISH

ALERTA HEIMDALL! BABUK2 EM 2025: RETORNO LEGÍTIMO OU COPYCAT ESTRATÉGICO

O cenário de ransomware manteve-se ativo em 2024 e se estendeu para este ano de 2025, com diversos grupos realizando ataques a uma ampla gama de organizações, setores e com surgimento de novos grupos. Nesse contexto, temos o ransomware Babuk2.

BAIXAR

ISH

ALERTA HEIMDALL! A ANATOMIA DO RANSOMWARE AKIRA E SUA EXPANSÃO MULTIPLATAFORMA

O cenário de ransomware manteve-se ativo em 2024 e se estendeu para este ano de 2025, com diversos grupos realizando ataques a uma ampla gama de organizações, setores e ganhando bastante popularidade. Nesse contexto, temos o ransomware Akira.

BAIXAR

SUMÁRIO

ESTRATÉGICO – CONHECENDO A AMEAÇA

Este relatório de segurança, desenvolvido pela equipe de inteligência Heimdall da ISH Tecnologia, tem como objetivo proporcionar uma compreensão aprofundada e um dimensionamento preciso das ameaças cibernéticas identificadas. O documento está estruturado em três níveis de abordagem: Estratégico, Tático e Operacional, garantindo uma visão completa e integrada das ameaças e ações recomendadas.

A adoção de tecnologias baseadas em contêineres, como *Docker* e *Kubernetes*, transformou a forma como sistemas operacionais, aplicações e serviços são implantados, escalados e gerenciados. Apesar dos ganhos em agilidade e portabilidade, essa abordagem também introduz novos vetores de ameaças. O isolamento fornecido pelos contêineres é, essencialmente, lógico.

O "container escape" ocorre quando um atacante consegue quebrar o perímetro do contêiner e obter acesso ao host subjacente. A partir disso, o adversário pode escalar privilégios, comprometer outros contêineres ou persistir no sistema comprometido.

Casos reais como a vulnerabilidade CVE-2019-5736 (runC overwrite), abusos do Docker socket e falhas em permissões de namespaces e cgroups mostram que essa é uma ameaça tangível, sendo explorada tanto por grupos APT quanto por ferramentas de Red Team em avaliações ofensivas.

Entre os vetores de escape mais comuns estão:

- Atribuição indevida de permissões com --privileged;
- Compartilhamento de diretórios críticos do host (/proc, /sys, /dev);
- Vulnerabilidades exploráveis em binários como o runC;
- Abuso de *capabilities* como CAP_SYS_ADMIN, que fornece quase poderes de root.

O uso indevido dessas configurações ocorre, muitas vezes, por conveniência em ambientes de teste ou falta de conhecimento técnico durante a orquestração de contêineres.

Essas falhas estão alinhadas às seguintes técnicas do MITRE ATT&CK:

- **T1611** – *Escape to Host*
- **T1068** – *Exploitation for Privilege Escalation*
- **T1203** – *Exploitation for Client Execution*

Mas claro que tal técnica, necessita de determinados contextos, abaixo segue os ambientes afetados, que incluem:

- Servidores Linux rodando *Docker* ou *Podman*;
- *Clusters Kubernetes* com permissões amplas de execução;
- Plataformas *CI/CD* (*GitLab Runners, Jenkins agents*);
- Provedores de nuvem pública que oferecem ambientes compartilhados (*multi-tenant*).

Abaixo podemos observar os produtos e tecnologias especialmente visados:

- *Docker Desktop* (ambientes de desenvolvimento);
- Imagens públicas não verificadas do *Docker Hub*;
- Orquestradores com *RBAC* mal configurado;
- Serviços em que usuários externos submetem código arbitrário (ex: funções *serverless*).

Estes ambientes tornam-se alvos ideais para testes de evasão e técnicas de movimento lateral dentro da infraestrutura.

TÁTICO: COMPREENDENDO OS DETALHES

Container Escape é a capacidade de um processo dentro de um contêiner acessar, manipular ou executar código diretamente no sistema host. Essa ação é viabilizada por configurações inseguras, vulnerabilidades no *runtime* do contêiner, ou vulnerabilidades no kernel do sistema operacional.

A arquitetura de contêineres é baseada em mecanismos do kernel Linux:

- **Namespaces**: isolam recursos como *PID*, *mount points*, rede e usuários;
- **Cgroups**: controlam consumo de *CPU*, memória e I/O;
- **UnionFS**/**OverlayFS**: montagens de sistema de arquivos em camadas.

Entretanto, essa arquitetura permite que contêineres tenham acesso ao kernel do host. Quando combinada com permissões indevidas ou exploits específicos, a barreira de isolamento é quebrada. Técnicas como bind-mount, acesso a dispositivos (/dev) e abuso de chamadas ptrace demonstram como contêineres podem se tornar vetores de ataque contra o host.

A seguir podemos observar alguns exemplos práticos de condições favoráveis ao escape:

- Contêineres iniciados com a *flag --privileged*, o que remove a maioria das restrições de segurança;
- Acesso ao socket */var/run/docker.sock*, permitindo execução de novos contêineres com parâmetros arbitrários;
- *Kernel* do *host* com vulnerabilidades conhecidas não corrigidas;
- *Capabilities* extras ativas (ex: CAP_NET_ADMIN, CAP_SYS_MODULE);
- Ausência de perfis AppArmor ou SELinux.

A conjunção de duas ou mais dessas condições é comum em ambientes de desenvolvimento e representa alto risco se replicada em produção.

OPERACIONAL: COMO EMULAR O ESCAPE

Uma das vulnerabilidades mais exploradas é a **CVE-2019-5736**. Ela permite que um processo dentro de um contêiner sobrescreva o binário do *runC* no host, levando à execução de código arbitrário com privilégios elevados.

A seguir, um exemplo de exploração da vulnerabilidade **CVE-2019-5736** usando um contêiner Docker vulnerável com *runC* desatualizado:

```
#!/bin/bash

echo -e '#!/bin/bash\necho "PWNED" > /tmp/pwned' > /tmp/overwrite

chmod +x /tmp/overwrite

docker run -v /tmp/overwrite:/overwrite alpine sh -c 'cp /overwrite /proc/self/exe'
```

Esse *exploit* sobrescreve o binário do *runC* do host com código malicioso, permitindo execução remota persistente em qualquer novo contêiner criado.

Outro exemplo clássico é o uso do *Docker socket*:

```
docker run -v /var/run/docker.sock:/var/run/docker.sock -it docker sh
```

Esse comando permite que um contêiner controle o *daemon Docker* do *host*, criando outros contêineres privilegiados, montando volumes arbitrários e executando código com efetivo controle da máquina.

Uma maneira de detectar depende da análise combinada de logs, comportamento de processos e uso de ferramentas específicas. Abaixo segue os exemplos:

- Monitoramento de *capabilities* elevadas por meio do pscap ou auditd;
- Uso de ferramentas para identificar comportamentos anômalos no runtime do contêiner;
- Inspeção do uso de docker.sock, alterações suspeitas em /proc, ou tentativas de acesso a /dev/kmsg e /sys/module;
- Alertas em tempo real de criação de contêineres com --privileged ou flags inseguras;
- Utilização de regras *YARA* para detectar *payloads* típicos de escape.

RECOMENDAÇÕES PARA MITIGAÇÃO

Mitigar *container escape* exige tanto *hardening* técnico quanto cultura de segurança, para isso recomendamos as seguintes medidas.

- Correção contínua de *runtimes* como *runC*, *containerd*, *CRI-O*;
- Proibir uso de contêineres privilegiados;
- Aplicar perfis *AppArmor*/*SELinux* obrigatórios em todos os pods ou contêineres;
- Isolar o *socket Docker* (se necessário, via *proxy* restritivo como *socat*);
- Monitoramento contínuo e alertas de *runtime*;
- Auditoria de imagens com *Snyk*, *Trivy* e *Dockle*;
- Aplicar políticas de *PodSecurityAdmission* e *PSPs* em clusters *Kubernetes*;
- Executar contêineres como usuários não-*root*.

Essas medidas não apenas reduzem o risco de escape, mas fortalecem a postura geral de segurança em ambientes *containerizados*.

Abaixo, podemos observar um snipet de código de exemplo para *hardening* com Docker:

```
docker run --cap-drop=ALL --cap-add=NET_BIND_SERVICE \
 --read-only --pids-limit=100 --memory=512m purple
```

Conclusão

A técnica de container escape representa uma ameaça aos ambientes de nuvem, pois compromete o isolamento que os contêineres devem garantir. Proteger esses ambientes exige atenção em todo o ciclo de vida, desde a construção da imagem até a execução em produção.

Práticas como atualização contínua, *hardening* de configurações e monitoramento em tempo real são fundamentais. Para uma defesa eficaz, é essencial que as equipes de desenvolvimento, segurança e operações atuem de forma coordenada, promovendo uma cultura de segurança voltada à proteção de *workloads containerizados*.

REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia;**
- **CTI Purple Team *by* ISH Tecnologia;**
- **MITRE ATT&CK;**

AUTORES

- **Cleriston de Freitas – Security Researcher**

heimdall
security research
A DIVISION OF ISH