

1)

<https://www.cvedetails.com/cve/CVE-2024-34346/>

<https://nvd.nist.gov/vuln/detail/CVE-2024-34346>

O Deno Sandbox refere-se ao ambiente de execução seguro do Deno, um runtime para JavaScript e TypeScript criado por Ryan Dahl (o mesmo criador do Node.js). O Deno foi projetado para ser mais seguro do que o Node.js e, por padrão, executa código em um sandbox, restringindo o acesso a recursos do sistema.

Como funciona a sandbox do Deno?

Acesso restrito por padrão

- Diferente do Node.js, onde scripts podem acessar arquivos, redes e variáveis de ambiente sem restrições, no Deno **todas essas permissões são bloqueadas** por padrão.

Permissões explícitas

- Para conceder permissões a um script, é necessário fornecê-las manualmente ao executar o programa.

Isolamento e segurança

- Scripts maliciosos não podem modificar arquivos, acessar bancos de dados ou fazer requisições sem autorização.
- O ambiente de execução impede a execução de código não confiável sem permissões explícitas.

Uso de Web APIs seguras

- O Deno utiliza APIs modernas e seguras inspiradas no navegador, evitando dependências nativas inseguras.

A vulnerabilidade **CVE-2024-34346** afeta o runtime Deno, permitindo que operações de leitura e escrita em arquivos privilegiados possam enfraquecer a sandbox de segurança. Por exemplo, ler o arquivo `/proc/self/environ` pode fornecer acesso equivalente à permissão `--allow-env`, e escrever em `/proc/self/mem` pode conceder acesso equivalente a `--allow-all`. Usuários que concedem permissões de leitura e escrita ao sistema de arquivos inteiro podem não perceber que isso pode ter consequências adicionais e não intencionais.

Para mitigar essa vulnerabilidade, é recomendável atualizar para a versão 1.43 ou superior do Deno, que requer a permissão explícita --allow-all para ler ou escrever em arquivos ou diretórios privilegiados, fortalecendo a sandbox de segurança.

CVSS Vector string: CVSS:3.1/AV:A/AC:L/PR:H/UI:N/S:C/C:H/I:H/A:H

Attack Vector(AV) : Adjacent (o ataque só pode ocorrer em redes adjacentes)

Attack Complexity (AC) : Low (o ataque é simples de executar)

Privileges Required (PR): High (O atacante precisa de privilégios elevados no sistema para explorar a falha)

User Interaction (UI): None (Nenhuma ação do utilizador é necessária para que a vulnerabilidade seja explorada.)

Scope (S) : Changed (A exploração pode afetar outros componentes para além do alvo inicial)

Confidentiality Impact: High (O ataque pode expor informações sensíveis de forma significativa.)

Integrity Impact (I) : High (O ataque pode alterar dados críticos, comprometendo a integridade do sistema.)

Availability Impact (A) : High (Pode ficar com o sistema indisponível de forma significativa).

Base: 8.4

Impact: 6.0

Exploitability: 1.7

Cwe: CWE-863

<https://cwe.mitre.org/data/definitions/863.html>

O produto executa uma verificação de autorização quando um utilizador tenta acessar um recurso ou executar uma ação, mas não executa a verificação corretamente.

2)

```
R1-6615(config)#username admina615 secret segurancaaulas1
R1-6615(config)#username adminb615 secret segurancaaulas2
R1-6615(config)#username adminc615 secret segurancaaulas3
R1-6615(config)#
```

```
R1-6615(config)#do show running-config | include username
username admina615 secret 5 $1$8Vjg$0FNJP7x8mJJZto2CE43XkN.
username adminb615 secret 5 $1$jhg$8G8uQZmbH3X8iy0se/zYX.
username adminc615 secret 5 $1$DpBj$JbYH.hWNPA04H3nli/Z490
R1-6615(config)#
```

3)

```
A1 console is now available... Press RETURN to get started.
ip: RTNETLINK answers: Network is unreachable
root@A1:~# ls -la
total 24
drwx----- 2 root root 4096 Mar  3 17:58 .
drwxr-xr-x 1 root root 4096 Mar  6 16:13 ..
-rw----- 1 root root  440 Mar  3 17:58 .bash_history
-rw-r--r-- 1 root root  571 Mar  3 17:58 .bashrc
-rw-r--r-- 1 root root  113 Mar  3 19:07 .gns3_perms
-rw-r--r-- 1 root root  161 Mar  3 17:58 .profile
root@A1:~#
```

```
R1-6615(config)#ip domain name span.com
R1-6615(config)#crypto key generate rsa general-keys modulus 1024
The name for the keys will be: R1-6615.span.com

% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)

R1-6615(config)#
*Mar  6 17:25:25.277: %SSH-5-ENABLED: SSH 1.99 has been enabled
R1-6615(config)#line vty 0 4
R1-6615(config-line)#login local
R1-6615(config-line)#transport input ssh
R1-6615(config-line)#
```

ssh -oHostKeyAlgorithms=+ssh-rsa -oKexAlgorithms=+diffie-hellman-group14-sha1
admina615@192.168.100.250

```
root@A1:~# ls -l .ssh/known_hosts
-rw-r--r-- 1 root root 274 Mar  6 21:43 .ssh/known_hosts
root@A1:~# cat .ssh/known_hosts
cat: .ssh/known_hosts: No such file or directory
root@A1:~# cat .ssh/known_hosts
|1|475IbVbQvT3808aevMEA/dHb/MU=|vh2Q4ZsER/4FvrI0xukNKyYGMs4= ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQCHvJMUlFHQpFuxJBD7Na8U
tHl17LQWlr7471Zf1Ypfc5uCjb1eCIYkunLglxhvTVNzOH7UV3DbQBerjfbB10kcvqtF1BK40fPoJvIMvJ6DSQS0V+WQZTRelMxUEKCB03GwkCK/Uq+B06agxb
5bnc99FtOHNP4te/2BLIvSZAhHJvw==
root@A1:~#
```

Antes da comunicação ssh ser estabelecida, o terminal A1 realiza um ARP Request para descobrir o Mac do Router (pois ainda não está na cache ARP), depois o Router envia um ARP Reply com o MAC da interface e0/0. Após isso, inicia tentativa de conexão SSH com o processo three-way handshake que é usado pelo protocolo TCP, para estabelecer uma comunicação confiável entre cliente e servidor. O terminal A1 envia um pacote SYN para o router, para indicar que quer iniciar uma conexão ssh com o router. A seguir o Router responde com um pacote SYN-ACK para mostrar que recebeu a solicitação de

conexão. Por último o terminal A1 responde com um ACK, confirmando que recebeu a resposta do servidor (Router).

4)

Após a resposta “yes”, o terminal A1 e o router voltam a trocar pacotes:

- O router envia a sua chave pública SSH para A1;

- Seguidamente começam a negociar os algoritmos de criptografia, autenticação e integridade que serão usados na conexão;

- A seguir geram uma chave de sessão, para que a comunicação seja segura e confidencial;

- Por fim, a chave pública do router foi armazenada no ~/.ssh/known_hosts do terminal A1, garantindo que futuras conexões possam verificar a autenticidade do servidor.

5)

```
root@A1:~# ls -l .ssh/known_hosts
-rw-r--r-- 1 root root 274 Mar  6 21:43 .ssh/known_hosts
root@A1:~# cat .ssh/known_hosts
cat: .ssh/known_hosts: No such file or directory
root@A1:~# cat .ssh/known_hosts
|1|475IbVbQvT3808aevMEA/dHb/MU=|vh2Q4ZsER/4FvrI0xukNKyYGMs4= ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCHvJMULFHQpfuxJBD7Na8U
tH1l7LQwr7471Zf1YpfC5uCb1eCIYkunLg1xhvTVNz0H7UV3DbQBerjfbB10kcvqtF1BK40fPoJvIMvJ6DSQS0V+WQZTRe1MxUEKCB03GwkCK/Uq+B06agxb
5bnc99FtOHP4te/2BLIvSZAhHJvw==
root@A1:~#
```

Neste ficheiro pode-se verificar, uma hash que contém o nome do host e o “salt”, o “|1|”, mostra a versão da hash.

A seguir aparece “ssh-rsa”, que mostra que está armazenada a chave pública do servidor encriptada no caso em ssh-rsa.

6)

Após voltar à sessão de autenticação, voltam a existir os pacotes arp, para o terminal A1 verificar se sabe o mac do router, seguidamente volta a existir novamente o processo de three-way handshake, já explicado anteriormente. A seguir o cliente e servidor negociam os parâmetros criptográficos, onde o router envia a sua chave pública e por fim existe o tráfego onde é enviada a senha já completamente criptografada.

7)

Aparece tudo encriptado, usando a ferramenta “find”, não consegui encontrar a password do utilizador. Em comparação ao telnet, o ssh mostra-se muito mais seguro pois não aparecem as informações dos utilizadores em clear texto.

8)

```
root@A1:~# date
Thu Mar  6 22:04:39 UTC 2025
root@A1:~# ssh-keygen -F 192.168.100.250
# Host 192.168.100.250 found: line 1
|1|475IbVbQvT3808aevMEA/dHb/MU=|vh2Q4ZsER/4FvrI0xuklNKyYGMs4= ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQChvJMUlFHQpFuxJBD7Na8Uth117LQW7471Zf1YpfC5uCjb1eCIYkunLg1xhvTVNzOH7UV3DbQBerjfbB10
kcvtF1BK40fPoJvIMvJ6DSQS0V+WQZTRElMxUEKCB03GwkCK/Uq+806agxb5bnc99FtOHNP4te/2BLIvSZAhHJvw==
root@A1:~# cat .ssh/known_hosts
|1|475IbVbQvT3808aevMEA/dHb/MU=|vh2Q4ZsER/4FvrI0xuklNKyYGMs4= ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQChvJMUlFHQpFuxJBD7Na8Uth117LQW7471Zf1YpfC5uCjb1eCIYkunLg1xhvTVNzOH7UV3DbQBerjfbB10
tcF1BK40fPoJvIMvJ6DSQS0V+WQZTRElMxUEKCB03GwkCK/Uq+806agxb5bnc99FtOHNP4te/2BLIvSZAhHJvw==
root@A1:~#

root@B1:~# cat .ssh/known_hosts
|1|vly338j2oE/yF7i2ecEqJmaKg=|5I/M0DkFAD/Vd3oD0YBZV05mMu0= ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQChvJMUlFHQpFuxJBD7Na8Uth117LQW7471Zf1YpfC5uCjb1eCIYkunLg1xhvTVNzOH7UV3DbQBerjfbB10kcvcq
tcF1BK40fPoJvIMvJ6DSQS0V+WQZTRElMxUEKCB03GwkCK/Uq+806agxb5bnc99FtOHNP4te/2BLIvSZAhHJvw==
root@B1:~# ssh-keygen -F 192.168.200.251
# Host 192.168.200.251 found: line 1
|1|vly338j2oE/yF7i2ecEqJmaKg=|5I/M0DkFAD/Vd3oD0YBZV05mMu0= ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQChvJMUlFHQpFuxJBD7Na8Uth117LQW7471Zf1YpfC5uCjb1eCIYkunLg1xhvTVNzOH7UV3DbQBerjfbB10kcvcq
tcF1BK40fPoJvIMvJ6DSQS0V+WQZTRElMxUEKCB03GwkCK/Uq+806agxb5bnc99FtOHNP4te/2BLIvSZAhHJvw==
root@B1:~#
```

O comando `ssh-keygen -F`, serve para verificar se uma determinada chave de um host está presente no ficheiro `known_hosts`. Neste caso, mostra a chave pública do router. Como foi dito no ponto 5), a chave é criptografia em `ssh-rsa`.

Após iniciar a conexão a partir do terminal B1, o ficheiro `.ssh/known_hosts` não foi atualizado quando verifiquei no terminal A1, no entanto no terminal B1, aparecia essa nova chave do host, mas apenas essa.

Com o comando `ssh-keygen -F IP(R1.e0/1)`, verificou-se a chave do host do router com o ip da segunda rede.

9)

```
root@A1:~# ssh -oHostKeyAlgorithms+=ssh-rsa -oKexAlgorithms+=diffie-hellman-group14-sha1 adminb615@192.168.100.250

Welcome

(adminb615@192.168.100.250) Password: █
```

Não apareceu a pergunta, pois no ficheiro `known_hosts`, já contem a chave pública enviada pelo router. Ou seja, mesmo autenticando com um utilizador diferente, se a máquina já tiver no ficheiro `known_host` a chave pública de um determinado servidor, já não precisa de perguntar por ela.

10)

```

R1-6615(config)#aaa new-model
R1-6615(config)#do wr
Building configuration...
[OK]
R1-6615(config)#parser view DiogoAntunes-VIEWA
R1-6615(config-view)#secret segurancaaulas
R1-6615(config-view)#commands exec include configure terminal
R1-6615(config-view)#commands configure include interface E0/1
R1-6615(config-view)#commands configure include ip address
R1-6615(config-view)#commands interface include description
R1-6615(config-view)#commands interface include shutdown
R1-6615(config-view)#commands interface include no shutdown
R1-6615(config-view)#end
R1-6615#enable
*Mar  6 23:08:10.225: %SYS-5-CONFIG_I: Configured from console by console
R1-6615#enable view DiogoAntunes-VIEWA
Password:
R1-6615#sh parser view
R1-6615(config)#username admina615 view DiogoAntunes-VIEWA secret segurancaaul$
R1-6615(config)#aaa au
R1-6615(config)#aaa authenti
R1-6615(config)#aaa authentication login default local
R1-6615(config)#aaa authorization exec default local
R1-6615(config)#

```

```

R1-6615#sh run | begin parser
parser view DiogoAntunes-VIEWA
secret 5 $1$R6A6$xsrSts05a5t7ibjQ41M2E1
commands interface include shutdown
commands interface include description
commands configure include interface
commands configure include ip
commands exec include configure terminal
commands exec include configure
commands configure include interface Ethernet0/1

```

11)

```

(admina615@192.168.100.250) Password:

      Bem Vindo!!!! R1-6615 Ligado!!!!

      Bem Vindo ao modo EXEC!!!!
R1-6615>en
Password:
R1-6615#enable view DiogoAntunes-VIEWA
Password:
R1-6615#sh parser view
Current view is 'DiogoAntunes-VIEWA'
R1-6615#sh privilege
^
% Invalid input detected at '^' marker.

```

12)

```
(adminb615@192.168.100.250) Password:
(adminb615@192.168.100.250) Password:

      Bem Vindo!!!! R1-6615 Ligado!!!!

      Bem Vindo ao modo EXEC!!!!
R1-6615>en
Password:
R1-6615#enable view DiogoAntunes-VIEWA
Password:
R1-6615#
```

O utilizador B continua a conseguir aceder à view criada.