

Autentizace a certifikáty

Datum zpracování: 21. 11. 2023

Zpracovali: Kevin Daněk, Jack Daniels Tennessee Apple 0.7L

Zadání

1. Zapojte lokální síť s DHCP.
2. Ve virtuálních strojích (nebo na vlastních počítačích) vytvořte každému členovi týmu uživatelský účet.
3. Zpřístupněte stroje pro přístup **ssh** z lokální sítě (konfigurace firewallu, ...).
4. Pomocí **ssh-keygen** vytvořte certifikáty a umístěte je na ostatní stroje.
5. Vytvořené certifikáty použijte i na router pro přístup bez hesla.
6. Proces dokumentujte screenshoty a kopiemi relevantních příkazů.

Úlohu zpracovávávejte ve 2 - 5 členných týmech.

Elaborát zpracujte do šablony v záhlaví kurzu, odevzdávejte ve formátu PDF.

Do abecedně seřazeného seznamu řešitelů na úvodním listu uveďte reálné složení týmu!

Příprava prostředí

Z důvodu technického stavu učebny A305, kde každou chvíli nějaká část vybavení nefunguje a ve které by cvičení měla správně probíhat, jsem se rozhodl cvičení realizovat mimo ní. Protože ovšem nemám sebou žádné síťové prvky, které bych mohl využít, natož další stroje, rozhodl jsem se využít Docker.

Docker slouží primárně k vytváření instancí obrazů, tzv. kontejnerů. Tyto obrazy ovšem mají vlastní souborový systém s nainstalovanými závislostmi. Obrazy jsou tak zpravidla založeny na linuxové distribuci, nejčastěji ubuntu nebo alpine. Přesně toho využiju a místo virtuálních strojů vytvořím instance linuxových distribucí, které se budou v rámci této úlohy chovat jako klienti.

Jak to je ale se sítí? Kontejnery jsou ve výchozím stavu absolutně izolované, ale docker dovoluje kontejnerům komunikovat mezi sebou nebo s hostitelským systémem pomocí definovaných sítí. Pro účely těchto cvičení tedy vytvořím síť s názvem PBE:

```
PS C:\Users\HP> docker network create pbe
```

Po vytvoření sítě už stačilo stáhnout nejnovější obraz ubuntu vytvořit jednotlivé kontejnery, resp. klienty. Každé spuštění potřebovalo několik parametrů:

- **dti** je kombinace několika příznaků, které spustí kontejner na pozadí, ale nechají otevřený STDIN a alokují virtuální TTY pro přístup emulující konzoli.
- **network** specifikuje název sítě, do které kontejner patří
- **privileged** umožní kontejneru přistupovat ke všem připojeným hardwarovým zařízením
- **entrypoint** specifikuje, co se má spustit při spuštění kontejneru
- **name** specifikuje název kontejneru
- **ubuntu** je název obrazu, ze kterého se má kontejner vytvořit

```
PS C:\Users\HP> docker pull ubuntu
PS C:\Users\HP> docker run -dti --network pbe --privileged --entrypoint /bin/sh --name user_1 ubuntu
PS C:\Users\HP> docker run -dti --network pbe --privileged --entrypoint /bin/sh --name user_2 ubuntu
```

Po spuštění obou kontejnerů můžeme zkontrolovat, zda-li opravdu běží pomocí příkazové řádky, nebo vizuálně v nějakém GUI rozhraní pro docker, např. Docker Desktop:

Container CPU usage ⓘ

0.00% / 400% (4 cores allocated)

Container memory usage ⓘ

174.88MB / 3.7GB









Show charts ▾

🔍 Search

⋮

☑️

Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	Last started	CPU (%)	Actions
<input type="checkbox"/>	<div><div></div><div><div>user_1</div><div>63d2104148d9 </div></div></div> <div>ubuntu</div>	Running	22:22 	54 minutes ago	0%	<div><div>■</div><div>⋮</div><div></div></div>	
<input type="checkbox"/>	<div><div></div><div><div>user_2</div><div>68da5fbaef08 </div></div></div> <div>ubuntu</div>	Running	23:22 	50 minutes ago	0%	<div><div>■</div><div>⋮</div><div></div></div>	

Příprava klientů

Ubuntu obrazy, které jsou přes docker hub distribuovány, obsahuje opravdu minimální kostru systému potřebnou pro spuštění. Abychom byly schopni s klienty nějak rozumně pracovat, je potřeba do nich nainstalovat závislosti.

Pro přístup do kontejneru existuje příkaz `docker exec`, který v daném kontejneru spustí nějaký proces. V tomto případě potřebuju spustit `bash`, takže pro každého uživatele si spustím `/bin/bash`

```
PS C:\Users\HP> docker exec -it user_1 /bin/bash
```

V obou kontejnerech je `bash` spuštěn jako `root` uživatel, ale každý má jiný `hostname`. `Hostname` je v tomto případě nastaven na zkrácený tvar identifikačního hashe kontejneru:

	ID kontejneru	hostname
user_1	63d2104148d9a168c590983cb885e80670caa68ef128be64fc299f1bd7180a80	63d2104148d9
user_2	68da5fbeaf08033ec9f1c0a9f0fbed259e55e735fabf429851af2ec225dc0468	68da5fbeaf08

Instalace závislostí

Vzhledem k tomu, že tyto kontejnery neobsahují z počátku ani `ping` utilitu, bylo mi jasné, že je potřeba nainstalovat vše potřebné. Před samotnou instalací jsem ovšem potřeboval aktualizovat seznamy dostupných balíčků:

```
root@63d2104148d9:/# apt update
root@68da5fbeaf08:/# apt update
```

a poté se mohlo začít instalovat. Rozhodl jsem se nainstalovat:

- **net-tools**, které obsahují nástroj `ifconfig`
- **iputils-ping**, které obsahují nástroj `ping`
- **keychain**, které obsahují nástroj `ssh-keygen`
- **openssh-server** pro zprovoznění `ssh` serveru
- a **nano**, protože ho mám rád.

```
root@63d2104148d9:/# apt install -y net-tools iputils-ping keychain openssh-server nano
root@68da5fbeaf08:/# apt install -y net-tools iputils-ping keychain openssh-server nano
```

Uživatelé v síti

Abych mohl dále pokračovat ve cvičení, chtěl jsem se předem ujistit, že komunikace mezi klienty probíhá bez problémů. Docker network slouží jako virtuální síť pro kontejnery, a tudíž je každému kontejneru v rámci sítě přiřazena IP adresa.

```
root@63d2104148d9:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.20.0.2 netmask 255.255.0.0 broadcast 172.20.255.255
    ether 02:42:ac:14:00:02 txqueuelen 0 (Ethernet)
    RX packets 21781 bytes 30424632 (30.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 9168 bytes 609909 (609.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@68da5fbeaf08:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.20.0.3 netmask 255.255.0.0 broadcast 172.20.255.255
    ether 02:42:ac:14:00:03 txqueuelen 0 (Ethernet)
    RX packets 21695 bytes 30418868 (30.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8927 bytes 594003 (594.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Na obrázcích výše je vidět, že opravdu kontejnery mají přiřazenou IP adresu. Jejich adresy se pohybují v rozsahu 172.20.X.X. Další nesmírnou výhodou užití Docker network je to, že automaticky v rámci sítě přiřazuje kontejnerům doménová jména, která odpovídají názvu kontejneru. Adresa 172.20.0.1 je gateway pro síť PBE.

Kontejner	IP adresa	Doménové jméno
user_1	172.20.0.2	user_1
user_2	172.20.0.3	user_2

Poté je tedy jedno, jestli z kontejneru user_1 pingu adresu 172.20.0.3, nebo user_2, páč obojí mě navede na to samé místo:

```
root@63d2104148d9:~# ping user_2
PING user_2 (172.20.0.3) 56(84) bytes of data.
64 bytes from 68da5fbeaf08 (172.20.0.3): icmp_seq=1 ttl=64 time=30.5 ms
64 bytes from 68da5fbeaf08 (172.20.0.3): icmp_seq=2 ttl=64 time=0.086 ms
64 bytes from 68da5fbeaf08 (172.20.0.3): icmp_seq=3 ttl=64 time=0.062 ms
```



Konfigurace SSH

Síť a komunikace mezi klienty funguje na jedničku - čas dělat SSH. Pro každého klienta jsem spustil příkaz `ssh-keygen`, který do adresáře `/ . ssh` vygeneroval veřejný a privátní klíč.

```
root@63d2104148d9:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:rZ3CCgbNJloBf+nyWC4/SRZajsKIrNNP2SoFeFSLZ8 root@63d2104148d9
The key's randomart image is:
+---[RSA 3072]-----+
|  . ...      |
|  = .        |
| o + . .     |
|..=+. E .    |
|..+=+o .S .  |
| . *=0 +. o . |
|+=+.0o=  + o |
|=.+*. . .    |
|..o.oo.      |
+----[SHA256]-----+
```

```
root@68da5fbeaf08:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:2KWXrwV03Zw05tuDlLS70yeR9bMo+rRlEn6zGLIAfhc root@68da5fbeaf08
The key's randomart image is:
+---[RSA 3072]-----+
|                |
|      . o.      |
|      o .o.o+   |
|      o + oo +oo|
|      .. SE+. oo+|
|      . . .ooo *.|=|
|      . o o *o@.*o|
|      . o =o@ o.+|
|      oo+ .     |
+----[SHA256]-----+
```





Po vygenerování klíčů jsem následně spustil SSH server v druhém kontejneru:

```
root@68da5fbeaf08:~# service ssh start
* Starting OpenBSD Secure Shell server sshd
```

[OK]

A nadešla chvíle pravdy. Bude připojení do druhé stanice přes SSH fungovat?

```
root@63d2104148d9:/# ssh 172.20.0.3
The authenticity of host '172.20.0.3 (172.20.0.3)' can't be established.
ED25519 key fingerprint is SHA256:xlAw1YNa9IkjLSTQDjniSlKcttmtQr0J/QA3d9y8Hzo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.20.0.3' (ED25519) to the list of known hosts.
root@172.20.0.3's password:
```

Ano, ssh nám funguje. Je ovšem ještě co dopilovat. Když se totiž podíváme na obrázek výše, tak je vidět, že po nás chce druhá stanice heslo. To my ale nechceme zadávat pokaždý. Co s tím? Použijeme vygenerovaný veřejný klíč z první stanice a umístíme ho do seznamu povolených klíčů na druhé stanici.

Seznam povolených klíčů se nachází ve složce ~/.ssh v souboru s názvem authorized_keys. Protože ještě tento soubor na druhé stanici neexistuje, vytvořil jsem ho pomocí editoru Nano a rovnou do něj vykopíroval veřejný klíč z první stanice. Pro jistotu jsem ještě změnil práva k složce .ssh, aby k ní měl root všechna práva, a k souboru authorized_keys, aby mohl číst a zapisovat.

```
root@68da5fbeaf08:~/.ssh# nano authorized_keys
root@68da5fbeaf08:~/.ssh# cd ..
root@68da5fbeaf08:~# chmod 700 .ssh
root@68da5fbeaf08:~# chmod 600 .ssh/authorized_keys
```

Pokud jsme vše nastavili správně, tak by nyní po nás SSH nemělo chtít heslo a mělo by nás rovnou poslat do bashe na druhé stanici:

```
root@63d2104148d9:/# ssh 172.20.0.3
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@68da5fbeaf08:~#
```





a protože mají jednotlivé kontejnery i doménová jména, můžeme do druhé stanice přistoupit pomocí názvu `user_2`.

```
root@63d2104148d9:/# ssh user_2
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.90.1-microsoft-standard-WSL2 x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Nov 21 16:36:42 2023 from 172.20.0.2
root@68da5fbeaf08:~#
```



Závěr

Tohle cvičení mě velmi bavilo. Ani tak né z hlediska samotného nastavení SSH, se kterým jsem byl již seznámen kvůli nastavení pro správu repozitářů na githubu, ale kvůli prostředí, ve kterém jsme úlohu zpracovával. Emulovat stanice pomocí Docker kontejnerů a následně si s nimi hrát bylo docela zábavné a úloha mi tak hezky uplynula.