Sem vložte zadání Vaší práce.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Kubernetes klastr pro lámání hesel

Tomá Klas

Katedra informaní bezpenosti (KIB)
Vedoucí práce: Ing. Jií Buek, Ph.D.

9. března 2020

Poděkování

Doplte, máte-li komu a za co dkovat. V opaném pípad úpln odstrate tento píkaz.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. března 2020 . . . . . .

Odkaz na tuto práci

Klas, Tomá. *Kubernetes klastr pro lámání hesel*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Hlavní náplní práce je nakonfigurování klastru pro lámání hesel. Tento klastr je ízen pomocí technologie Kubernetes. Program vyuívá ke své správné funkcionalit kontejnery. Tyto kontejnery jsou tzv. Docker kontejnery. Pouité technologie jsou v práci detailn popsány a rozebrány. Dále se práce zabývá reerí ukládání hesel v souasných systémech a tím jak hesla vypadají. Na závr na klastru bude proveden test rzných metod pro lámání hesel. Tyto metody budou popsány a bude analyzováno, jak je klastr efektní a výkonný pro daný typ lámání.

Klíčová slova Kubernetes, Ansible, klastr, Docker, distribuované lámání, hesla, hashcat, nasazení.

Abstract

The main goal of the thesis is to setup a cluster managed by kubernetes for password recovery. Next step is to describe used technologies as Docker, Ansible and Hashcat. Thesis contains description of how the passwords are stored and most known attacks to crack them. Successful deployment and password cracking leads to analyzing speed of the cluster and the particular cracking method.

Keywords Kubernetes, Ansible, cluster, Docker, distributed cracking, passwords, hashcat, deployment.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Kapitola 1

Cíl práce

Cílem práce je sestavit Kubernetes klastr, nasazen bude pomocí technologie Ansible a následn na nm bude sputn software na lámání hesel. Rychlost a výsledky lámání budou analyzovány s ohledem na délku hesla a pouití daného druhu útoku. Budou popsány bn pouívané útoky na hesla a zpsob jejich ukládání na rzných operaních systémech. Technologie, je budou pouity, budou popsány do hloubky nutné k porozumní daného eení a následné moné replikace pro jiná eení.

1.1 Kubernetes

Jméno Kubernetes pochází z ecka a znamená to kormidelník. Projekt zloili Joe Beda, Brendan Burns, a Craig McLuckie,ke kterým se rychle pipojili inenýi z Googlu, jako Brian Grant a Tim Hockin. Software byl vydán v roce 2014.

Kubernetes je opensource technologie pro vytvoení a správu klastru. Pomáhá na tomto klastru plánovat sputní kontejner na základ jeho stavu. ídi automatickou aktualizaci kontejner a jejich opravu. Kontejnery sdruuje do pod, co je základní jednotka pro Kubernetes. Tyto pody káluje na poadovaný stav. Kubernetes také vyvaují zatíení a v pípad pádu aplikace restartuje kontejner, aby znovu splnily poadavky.

1.1.1 Stavební kameny Kubernetes

Aby Kubernetes zajistili funknost klastru je zapotebí rozdlit práci do nkolika komponent, které se starají o správný chod. Níe je znázornno, z eho se skládají. Dále se komponenty popíí a vysvtlí se na nich kompletní funkcionalita.

1.1.1.1 Master

1.1.1.2 Pod

1.1.1.3 Plánova

1.1.1.4 Controller Manager

1.1.1.5 API server

1.1.1.6 Kubelet

1.1.1.7 Kube-Proxy

1.1.2 Kubernetes a tajemství

1.1.3 Aktualizace obraz

1.1.4 Sdílené datové prostory

1.1.5 Cloudový poskytovatelé

1.2 Ansible

Ansible je automatizaní nástroj pro konfiguraci systému, nasazení softwaru, aktualizac. Jeho nejsilnjí stránka je nulové výpadky systému pi aktualizaci balík, nebo automatické nastavovat dané zaízení.

Jeho hlavními cíly jsou jednoduchost a nenáronost. Kód by ml být itelný i pro lidi, kteí nejsou obeznámeni s programem. Je schopen pokrýt rzn velké prostedí od malých podnik a po velice obsáhlou infrastrukturu.

Ansible se pipojí na vzdálený poíta pomocí OpenSSH pomocí uivatele, který je souasn pihláen. Na spravovaném poítai není treba ádný agent. Je monost nakonfigurovat Ansible, aby pro pipojení nepouíval OpenSSH, ale i kerberos nebo LDAP.

1.2.1 Komponenty

1.2.1.1 Control node

Jakýkoliv poíta s nainstalovaným Ansible a pythonem, me spoutt píkazy nebo tzv. playbooky. Tento poíta se nazývá control node. Takových meme mít klidn více, ne vak poítae, které mají nainstalovaný operaní systém Windows.

1.2.1.2 Managed node

Je jakékoliv síové zaízení. Managed nodes meme také nayývat jako hosts. Tyto zaízení nemusejí mít nainstalovaný Ansible, ale musejí mít nainstalovaný python. Ansible me být nakofigurován, aby pouíval specifikovanou verzi pythonu, pokud není specifikována, spustí se na hostu jeho defaultní.

1.2.1.3 Iventory

Je seznam vech nastavovaných zaízení. asto se nazývá hostfile. v tomto souboru nastavujeme skupiny zaízení, jejich IP adresy a dalí specifikace, napíklad jaký python má daný host pouít.

1.2.1.4 Modules

Jsou to jednotlivé ásti kódu, které bude Ansible spoutt. Kadý modul má speciální pouití. Ve od správy uivatel (user) pes nastavení systému (systemd) a k instalovaní balík (apt, yum). Meme spustit jeden modul v tasku, nebo více v playbooku. Pro pehlednost neuvádím vechnz moné moduly, jeliko je jich pes ti tisíce.

1.2.1.5 Tasks

Jsou jednotky, které se museji provést. Nejasteji specifikované v deployment souboru.

1.2.1.6 Playbooks

Je seazený seznam task, které se musí vykonat. Niemu neukodí pokud se plazbook spustí znovu, protoe Asnible skontroluje stav daného tasku. Playbooky jsou psané podle konvenci YAMLu.

1.3 Docker

Docker je otevená platforma pro vývoj, dodání a spoutní aplikací. Umonuje oddlení aplikací od infrastruktury, tedy meme dodávat software rychleji a bez problém, které se váí k rznorodosti prostedí, ve kterém aplikace bí. Svou fylozofií jsou velice podobné virtualním poítam. Rozdíly mezi tmito rznými pohledy na vc budou rozebrány dále v textu.

Docker zprostedkovává platformu pro zabalení aplikace i se vemi jejímy závislostmi. Izoluje danou aplikaci od ostatních bících proces na daném poítai a zajiuje tak její bezpeí. Docker kontejner je velice nenároný na hardware, meme jich tedy na daném poítai spustit velice mnoho.

Fylosofie kontejner je taková, e kadý kontejner je odpovdný pouze za jednu danou ást aplikace. Pro píklad máme naí webovou aplikaci. Budeme tedy mít alespo ti docker kontejnery. Jeden na kterém pobí NGINX a bude zprostedkovávat naí aplikaci uivatelm. Dalí bude mít naí aplikaci a ve tetím pobí databáze.

Konterjnery fungují tedy jako malé poítae, mají izolované vekeré svoje systémové zdroje (pam, procesy, internetové rozhraní). Díky tomuto mohou být rychle a jednodue pidány, nebo odebrány.

1.3.1 Konterjner vs. virtuální poíta

Virtualizace je odpov na problém rznorodých prostedí mezi vývojái a zákazníky. Problém ,který virtualizace a kontejnerizace pedevím eí je rznorodost prostedí mezi zákazníkem a dodavatelem softwaru. Pi jeho pedávání dochází ke zmn prostedí, jsou nainstalované jiné verze závislostí a operaního systému a aplikace se me chovat neoekávan.

Podíváme se jak se tyto dv technologie lií a pro se svt ene práv smrem kontejnerizace, kdy zde ji je eení.

Virtuální poíta je regulérní stroj, který bí na daném hostovi. Tento stroj má svj kernel, svj operaní systém a ke zdrojm pistupuje pes tzv. hypervizor nap.: QEMU, nebo VirtualBox. Hypervizor zprostedkovává pístup virtuálního stroje k systémovým zdrojm.

Pro to, aby na hostovi, nebo-li na systému, který má nainstalovaný hypervizor mohlo bet více virtuálních stroj staí jedna jeho instance. Nevýhoda tohoto eení je taková, e se mnoho zdroj duplikuje. eknme, e na hostitelském systému pobí ti aplikace. Kadá taková aplikace bude izolovaná od ostatních pomocí virtuálního stroje. Dejme tomu, e to bude databáze, webový server a stroj pro vzdáleného uivatele. Níe uvidíme nákres tohoto eení.

To samé, jako je na obrázku výe se pokusíme realizovat pomocí Dockeru a kontejner. Kontejnery, jeliko vyuívají overlayFS jsou schopny poskytnout jádro operaního systému kontejneru bez zbytené kopie a vyuívají copy-on-write funkcionality. Níe uvidíme jak za pomoci jmenných prostor, kontrolních skupin a overlayFS je tento pístup úspornjí a rychlejí ne virtualizace.

Místo, které jsme na hostitelském systému uetili vak není jediná výhoda. Na tomto píkladu se vak rozdíly mezi tmito technologiemi vysvtlují nejlépe. Dalími výhodami je rychlost sputní kontejneru a virtuálního stroje. Pi sputní se pouze pipojí obraz OS, vytvoí se izolované procesy a popípad se omezí i zdroje, které má kontejner vyuívat. Nehled na to, e pokud kontejner nemá omezení nebo limit vyuitých zdroj alokuje si je dynamicky oproti virtuálnímu poítai, který si pro sebe naalokuje danou pam pi sputní.

1.3.2 Stavební kameny Dockeru

Jak je ji uvedeno v pedelé kapitole, Docker vyuívá vychytávky linuxového kernelu pro svojí funkcionalitu. Díky tomuto perfektn funguje na poítaích, kde bí OS zaloený na Linuxu. V následujících sekcích budou tyto technologie blíe popsány a bude vysvtlena jejich dleitost.

1.3.2.1 Jmenné prostory

Jmenné prostory zasteují vekeré zdroje systému tak, e kadý proces sputn v daném prostoru me pouívat pouze prostedky, které se váí k tomuto prostoru. Kadému procesu se to jeví tak, e má svoje vlastní globální prostedky, které

mohou vidt i ostatní procesy z jmenného prostoru, ale ne z jiného. V tabulce 1.1 je moné videt, jaké jmenné prostory lze v Linuxu nalézt.

Tabulka 1.1: Linuxové jmenné prostory

Jméno	**Popis**
Cgroup	Cgroup root adresá
IPC	Systém pro komunikaci proces, POSIX fronty
Network	Síové rozhraní, protocoly, porty, etc
Mount	Pipojená zaízení
PID	ID proces
User	Uivatelská ID a ID skupin
UTS	Hostname a NIS doménu

Pi sputní kontejneru dojde k vytvoení procesu na hostitelském systému. Procesy dostanou od systému njaké PID a chovají se jako normální procesy. Pokud se vak pihlásíme do kontejneru (command: docker exec -it name bash) a podíváme se na procesy bící v daném kontejneru uvidíme, e procesy mají jiná PID a urit mají i PID=1. Toto nám umoují jmenné prostory.

Kadý kontejner me mít svj vlastní souborový systém a svoje síové rozhraní. Ve co meme oddlit mezi hostitelem a kontejnery je uvedeno v tabulce výe.

1.3.2.2 Kontrolní skupina

Je to vlastnost Linuxového kernelu. Jejich hlavní funkcí je limitovat zdroje. V Dockeru se pouívají protoe dovolují sdílet prostedky mezi hostitelským systémem a dalímy kontejnery.

asto dochází k zámn pojm mezi kontrolními skupinami a jmennými prostory. Znovu to tedy shrme. Kontrolní skupinz, nebo-li cgroups omezují co meme pouít a jemenné prostorz nebo-li namespaces omezují co jsme schopni vidt v systému.

1.3.2.3 Docker daemon

Docker daemon nebo-li dockerd poslouchá dotazy na docker API a spravuje objekty jakou jsou docker obrazy, kontejnery, sí a úloit. Komunikuje ale i s dalími daemony, aby byl schopen ídit slubu Docker.

1.3.2.4 Docker klient

Je to primární cesta, jak komunikovat s Dockerem. Kdy pouijeme píkazy, jako jsou "docker run", klient odele píkazy daemono zmínného výe.

1.3.2.5 Docker registr

Docker registr je úloit pro nae Docker obrazy. Bez pedchozího nastavení hledá dockerd obrazy, které chceme spustit ve veejném Docker registru. Obrazy vak mohou být dostupné i lokáln, nebo na njaké jiné slub, nap.: gitlab container registry.

Do styku s registrem pícházíme hlavn ve chvílích, kdy provádíme píkazý docker pull, docker push a docker run. Tyto píkazy vdy potebují znát obraz, který bude spoutn jako základní vrstva pro nový kontejner, nebo bude stáhnut na lokání poíta, i nasdílen do registru.

1.3.2.6 Obrazy

Meme si to pedstavit jako ablonu, na které je sputn kontejner. Obraz me být sloen z vícero obraz, nebo z nich vycházet.

Pro vytvoení obrazu je teba soubor Dockerfile. Tento soubor obsahuje jednoduché kroky, které je teba vykonat pro vytvoení konkrétního obrazu. Nap.: jaké pouijeme a zveejníme porty, jaké balíky chceme ve vytvoeném obrazu mít atd.

Kadý píkaz v Dockerfilu vytvoí na locálním poítai tzv. vrstvu, kterou pi úprav Dockerfilu mní nebo pedlává pouze pokud byla zmnna.

1.3.3 Systémová kontejnery

Docker kontejnery nejsou vak jediné, které se v produkním prostedí pouívají. Patí do tzv. aplikaních kontejner. Jejich úel je zpravidla spoutt pouze jeden proces. K takovýmto kontejnerm meme jet pidat kontejnerz Rocket. Hlavním rozdílem je to, e rtk nemá na systému sputného daemona jako má nap. Docker. Pi sputní se tedy pod bícím spustí dalí.

Dále tady máme systémové kontejnery. Ty jsou pouívané jako klasické OS. Na jednom silném stroji me bet nkolik takových kontejner a ty mohou uivateli poskytovat oddlené prostedí od celého serveru a nabídnout mu izolovaný prostor od ostattních pomocí výe zmínných technologií. Tuto monost zasteuje projekt LXC pozdji LXD.

1.3.4 Pro pouít Hashcat?

Kapitola 2

Hesla

Hesla meme vidt vude a ne jen v informatice. Pokud se podíváme zpt do historie nap. do doby velkého Caesara a jeho ifry, ke které je teba znát íslo, o které se posouvají znaky ve zpráv. Jak tedy meme vidt, hesla nesloui pouze k naí autentizaci vi njaké slub i serveru. Me je také pouít k podepsání citlivých dokument jako je teba píloha e-mailu. Následn pak nememe popít jeho poslání. Tomuto se íká elektronický podpis.

Hesla vak mají nejednu nevýhodu. Útoník me s naím nebo i bez naeho vdní odhalit nae heslo a tím nám naruit nae soukromý. Hesla mohou také být v systémech, které pouíváme uloena nepatiným zpsobem, jako je napíklad istý text bez pouití ádných ochranných prostedk.

Hesla té mohou ze systému uniknout. V tomto pípad, pokud byla hesla uloena neptiným zpsobem nemusí se potencionální útoník njak pemáhat, aby uivatele kompromitoval. Proto se zamíme na to jak mohou a jak skuten jsou uloena v nejpouívanjích systémech.

2.1 Haovací funkce

Jsou to takové funkce f: X ő Y, pro n je snadné z jakékoli hodnoty x Î X vypoítat y = f(x), ale pro náhodn vybraný obraz y Î f(X) nelze v relevatním ase najít její vzor x Î X tak, aby y = f(x).

Pitom víme, e takový vzor existuje nebo jich existuje dokonce velmi mnoho. To kolik jich existuje se odvíjí jakou haovací funkci pouijeme.

2.1.1 Vlastnosti haovací funkce

Abychom mohli funkci povaovat za haovací, musí mít následující vlastnosti:

- jakékoliv mnostv vstupních dat poskytuje stejn dlouhý výstup (otisk),
- malou zmnou vstupních dat dosáhneme velké zmny na výstupu,

- z hashe je prakticky nemoné rekonstruovat pvodní text zprávy,
- v praxi je vysoce nepravdpodobné, e dvma rzným zprávám odpovídá stejný hash, jinými slovy pomocí hashe lze v praxi identifikovat práv jednu zprávu (ovit její správnost).

2.1.2 Naorzeninový paradox

2.1.3 Windows

Windows se chovají jinak v domén a jinak mimo ní. Pokud je poíta v domén je preferován autentizaní protokol kerberos. V souasných Windows Server edicích je implementován Kerberos verze 5. Kerberos v základní nastavenim operuje na portu 88 a k ifrování pouívá symetrickou ifru. Pokud poíta není nastaven aby se autentikoval pomocí protokolu Kerberos pouívají Windows ifrování NTLM.

2.1.4 Linux

Hesla v linuxových systémech se skládají ze dvou konkretních soubor.

/etc/shadow - obsah a strukturu toho souboru meme vidt na následujícím obrázku.

/etc/passwd - obsah a strukturu tohoto souboru meme vidt na následujícím obrázku.

V /etc/shadow jsou hesla uloená pomocí hashe.

2.1.5 MacOS

2.2 Útoky na hesla

2.2.1 Hrubou silou

2.2.2 Pomocí masky

2.2.3 Se slovníkem

2.3 Entropie hesla

2.4 Ochrana ped rznými útoky

Kapitola 3

Závr

[1]1

Literatura

[1] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

Příloha **A**

Seznam pouitých zkratek

GUI Graphical user interface

XML Extensible markup language

Příloha **B**

Obsah piloeného CD

- `readme.txt` ..... struný popis obsahu CD
- `exe` ..... adresá se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS