Docker Security

Hou je containers veilig

Docker's introductie van een gestandaardiseerd image format, samen met de tooling die Docker biedt voor de orchestratie van containers, heeft gezorgd voor een explosie aan interesse voor containers binnen allerlei ondernemingen. Containers dragen bij aan het versimpelen van de distributie van applicaties en het delen van compute resources. Het gemakkelijker kunnen delen van compute resources heeft echter ook een keerzijde. Zodra er meer applicaties op een systeem draaien, wordt de kans ook groter dat één applicatie een vulnerability bevat. In dit artikel bespreek ik waar je qua security op moet letten wanneer je Docker wilt inzitten in productie. Natuurlijk is het altijd verstandig om dit alles ook in je andere omgevingen (test, acceptatie, etc.) toe te passen.

Docker is een open platform, waarmee je relatief eenvoudig gedistribueerde applicaties kunt worden bouwen, deployen en draaien. Docker maakt hiervoor gebruik van zogenaamde containers. Binnen organisaties wordt Docker al veelvuldig ingezet om het ontwikkelproces te versnellen en te versimpelen. Dit gebeurt in steeds meer gevallen ook in productie. Voor de introductie van Docker waren we al in staat om gedistribueerde applicaties op te bouwen uit lichtgewicht applicatie-containers die dynamisch kunnen veranderen, en zonder wijzigingen kunnen worden ingezet op ontwikkel-, test- en productieomgevingen. Echter, heeft Docker de orchestratie hiervan versimpeld, waardoor het gebruik van containers een enorme vlucht heeft genomen.

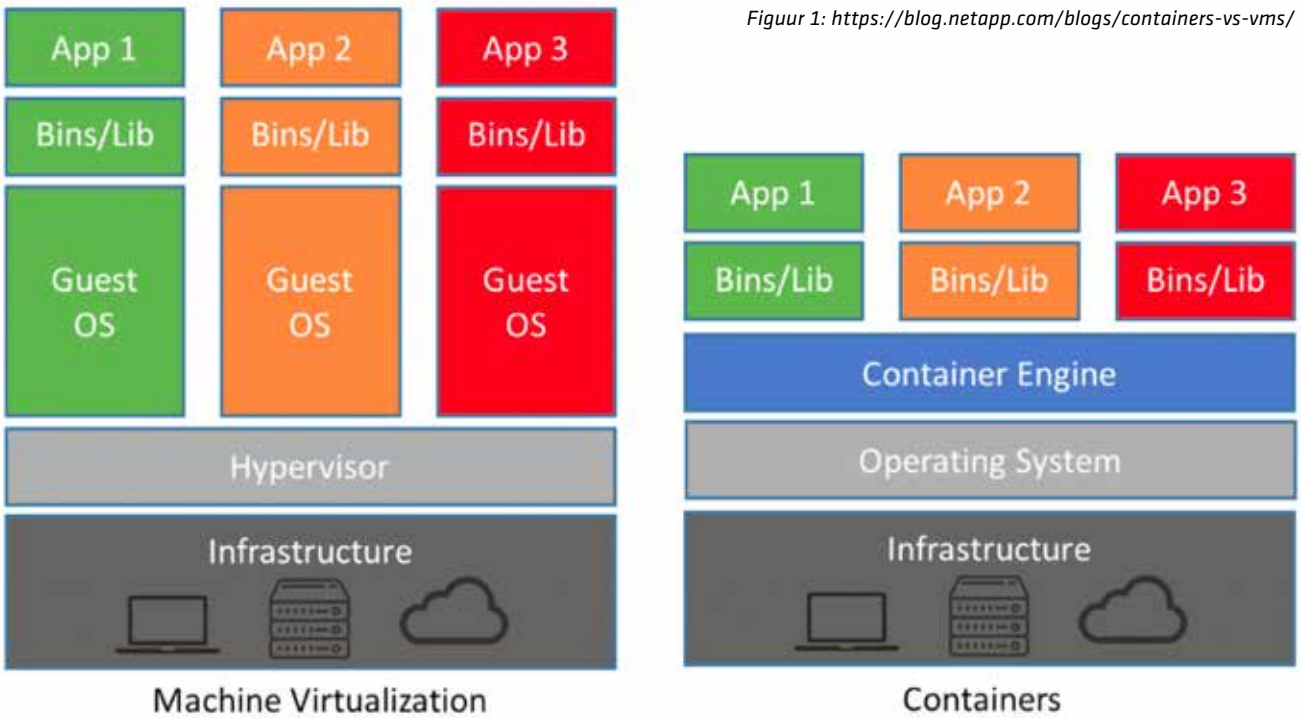
De inzet van Docker brengt ook implicaties met zich mee op het gebied van security. Desondanks brengt het gebruik van Docker ook voordelen met zich mee qua security:

- **Docker containers zijn klein**. Er zijn hoogstens enkele processen per container (meestal slechts één).
- **Docker containers zijn taak specifiek**. Doordat er bekend is welke taak de container moet doen, is het al snel duidelijk als er afwijkingen zijn, die niet tot deze taak behoren. Dit in tegenstelling tot multipurpose systemen.
- **Docker containers zijn geïsoleerd,** zowel van elkaar als van het operating systeem waarop Docker draait. Het enige dat gedeeld wordt is de kernel, maar daarover later meer.
- **Docker containers zijn reproduceerbaar.** Doordat elke build elke keer dezelfde container oplevert, kan iedereen de container bouwen en begrijpen waarom dat gebeurt.

Er zijn vier grote gebieden, die in acht moeten worden genomen, wanneer we het hebben over Docker Security.

- De security van **de kernel** en de daardoor ondersteunde functies van *namespaces* en *cgroups*.
- De attack surface van de **Docker daemon** zelf.
- Loopholes in de **container configuratie policy**, zowel de standaard waardes, als de door de gebruikers opgegeven waardes;
- Het host OS.

Arjan Gelderblom is een Security Software Engineer bij First8. Hij specialiseert zich in alle aspecten van het beveiligen van applicaties, van ontwerp tot end-of-life.



Figuur 1: https://blog.netapp.com/blogs/containers-vs-vms/

De kernel

In tegenstelling tot Virtuele Machines (VM's) wordt de kernel gedeeld door alle containers en de host. Dit verschil zit hem in hoe beide technieken (VM's en containers) tot de isolatie van hun applicaties komen. Een VM emuleert hardware (door middel van de hypervisor) waaruit de virtuele machine bestaat. Hierop wordt dan voor elke VM een compleet Operating System geïnstalleerd (inclusief kernel) en daarop draaien dan de applicaties.

Containers daarentegen regelen de isolatie door middel van namespaces. Hierdoor kunnen processen van verschillende containers elkaar niet zien, laat staan beïnvloeden.

Kernel exploits

Mocht een container een kernel panic weten te veroorzaken, dan zullen ook de andere containers en de host onderuit gaan, omdat de kernel door iedereen wordt gedeeld.

Denial-of-service

Alle containers delen ook kernel resources. Als één container een resource kan monopoliseren (bijvoorbeeld memory of user ID's), dan kunnen anderen hier geen gebruik meer van maken. Hierdoor zullen de legitieme containers niet reageren en is het resultaat een denial-of-service.

Docker Daemon

Het draaien van containers door middel van Docker, impliceert het draaien van de Docker Engine als daemon. Deze daemon heeft rootrechten nodig voor het gebruik van *cgroups* en *namespaces*.

Docker kent enkele zeer krachtige features. Een voorbeeld daarvan is het delen van een directory tussen de Docker host en de container. Docker staat dit toe zonder imitaties op de access rights van de container. Dit betekent dat je in een container een `/host directory` kan laten mounten naar de

NAMESPACING VAN GEBRUIKERS ZORGT ERVOOR DAT ELKE CONTAINER EEN EIGEN UNIEKE RANGE ID'S KRIJGEN VOOR GEBRUIKERS EN GROEPEN

CGROUPS EN NAMESPACES

cgroups (afkorting van **control groups**) is een functie van de linux kernel welke resourcegebruik (CPU, geheugen, schijf-I / O, netwerk, etc.) van een groep processen beperkt, registreert en isoleert.

Namespaces zijn een functie van de linux kernel welke de resources van de kernel zodanig partitioneert dat de één groep van processen één set van resources ziet en een andere groep een andere set van resources.

`/ (root) directory` van zijn host. Bovendien kan de container hier zonder restricties dingen in aanpassen. Vandaar dat alleen vertrouwde gebruikers controle mogen hebben over de Docker daemon.

Om ervoor te zorgen dat ongewenste personen geen containers kunnen creëren en draaien op de host, is sinds Docker 0.5.2 de REST API endpoint als default komen vervallen en vervangen door een UNIX socket. Op de UNIX socket kunnen de standaard UNIX permissies worden toegepast. De REST API endpoint kan indien gewenst wel weer worden ingeschakeld. Zoals echter eerder gemeld, brengt dit extra security implicaties met zich mee.

De Docker Daemon is potentieel ook vulnerable voor input, zoals bijvoorbeeld `docker load` voor het laden vanaf disk of `docker pull` voor het laden van het netwerk.

Configuratie

Standaard wordt Docker geleverd met een configuratie, die in zoveel mogelijk omstandigheden werkt. Dit is ondanks de "secure by default" regel niet de meest veilige configuratie.

Zo is er standaard ongelimiteerd netwerkverkeer mogelijk tussen de containers op de standaard network bridge. Hierdoor heeft elke container de mogelijkheid om al het netwerkverkeer tussen de containers te lezen, die op dezelfde host draaien. Het is beter om dit (op enkele uitzonderingen na) te limiteren en enkel verkeer tussen specifieke containers toe te staan. Intercontainer communicatie kan worden uitgeschakeld door de docker daemon te starten met `dockerd --icc=false`.

Een container breakout kan ontstaan op het moment dat bijvoorbeeld iemand toegang krijgt op een container, die gebruik maakt van een root user. Met andere woorden: hij is root binnen de container. Er zijn twee zaken, die een container breakout kunnen helpen voorkomen.

De eerste mogelijkheid is om gebruik te maken van de mogelijkheid tot namespacing van gebruikers. Dit is een configuratie optie voor de Docker daemon. Namespacing van gebruikers zorgt ervoor dat elke container een eigen unieke range ID's krijgen voor gebruikers en groepen. Hierdoor zijn deze niet één op één te vertalen naar het host systeem. Standaard staat deze optie uit en zal bij een container breakout gezocht worden naar een gelijkende user op het host systeem. Zodra dus de root gebruiker op de container (altijd User & Group ID 0) een breakout forceert, zal dit dus altijd tot root leiden op de host. Door namespacing van gebruikers aan te zetten in de configuratie in `/etc/docker/daemon.json` of door de Docker daemon te starten met `dockerd --userns-remap="testuser:testuser"` kun je dit voorkomen.

De tweede mogelijkheid is om te voorkomen dat processen in een container als root draaien. Helaas kan dit niet door de Docker daemon worden afgedwongen, maar moet dit in de Docker file aangegeven worden door middel van een USER definitie. Zonder deze USER definitie in je Docker file zal alles standaard als root worden uitgevoerd. Tevens krijgen kwaadwillenden, die in een container komen, dan ook default root rechten in de betreffende container.

Ondanks dat niet vaak bekend wordt gemaakt dat een container breakout de oorzaak was van een breach, dien je er wel rekening mee te houden dat het mogelijk is.
CIS (Center for Internet Security) heeft een goede benchmark (zie referenties), die als startpunt gebruikt kan worden voor de configuratie van je Docker omgeving. Door middel van Docker Bench is dit (geautomatiseerd) te testen.

Host OS

Net zoals met elke andere vorm van virtualisatie of isolatie geldt, dat zodra men toegang tot het host systeem heeft, alle vormen van isolatie (opgelegd door Docker) geen enkel nut meer hebben. Dat maakt dat de security van de host ook een essentieel onderdeel vormt van Docker Security. Er moet dan ook op dat

CONTAINER BREAKOUT

Een **container breakout** betekent dat het voor een container mogelijk is om de isolatie functionaliteit te omzeilen, gevoelige informatie op de host te benaderen en/of hogere of extra privileges te verkrijgen.
Wil je zelf kijken of je een container breakout kunt uitvoeren? "Down by Docker" (**https://www.notsosecure.com/vulnerable-docker-vm/**) is een bewust onveilige Docker installatie, waarbij het mogelijk is om een container breakout uit te voeren.

EEN CONTAINER BREAKOUT KAN ONTSTAAN OP HET MOMENT DAT BIJVOORBEELD IEMAND TOEGANG KRIJGT OP EEN CONTAINER, DIE GEBRUIK MAAKT VAN EEN ROOT USER

```
# ------------------------------------------------------------
# Docker Bench for Security v1.3.3
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Community Edition Benchmark v1.1.0.
# ------------------------------------------------------------

Initializing Fri Jul 14 09:18:42 UTC 2017

[INFO] 1 - Host Configuration
[WARN] 1.1  - Ensure a separate partition for containers has been created
[NOTE] 1.2  - Ensure the container host has been Hardened
[PASS] 1.3  - Ensure Docker is up to date
[INFO]      * Using 17.06.0 which is current
[INFO]      * Check with your operating system vendor for support and security maintenance for Docker
[INFO] 1.4  - Ensure only trusted users are allowed to control Docker daemon
[INFO]      * docker:x:992:vagrant
[WARN] 1.5  - Ensure auditing is configured for the Docker daemon
[WARN] 1.6  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.7  - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]      * File not found
[INFO] 1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO]      * File not found
[INFO] 1.11 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[WARN] 1.12 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-containerd
[WARN] 1.13 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-runc

[INFO] 2 - Docker daemon configuration
[WARN] 2.1  - Ensure network traffic is restricted between containers on the default bridge
[PASS] 2.2  - Ensure the logging level is set to 'info'
[PASS] 2.3  - Ensure Docker is allowed to make changes to iptables
[PASS] 2.4  - Ensure insecure registries are not used
```

Figuur 2: https://github.com/docker/docker-bench-security

niveau nagedacht worden hoe de risico's te beperken zijn.

Een grote stap kan gemaakt worden door het verkleinen van de attack surface. Zorg dat er geen onnodige services draaien op de Docker host en kies voor een OS, dat speciaal ontworpen is voor het draaien van containers, zoals CoreOS, Red Hat Atomic en RancherOS. Dit verkleint niet alleen de attack surface, maar brengt ook extra features, zoals het draaien van OS services in containers voor additionele isolatie van processen.

Welk OS er ook gekozen wordt, deze moet uiteraard up-to-date zijn en blijven. Net zoals voor de Docker daemon dient de configuratie van het OS op orde te zijn. Voor verschillende OS'en bestaan er ook handleidingen voor hardening van je systeem. Hardening is het beveiligen van een systeem door bijvoorbeeld het dichtzetten van bepaalde poorten en het installeren en instellen van een firewall. Ook zijn er patches beschikbaar voor Linux kernels voor hardening van de kernel, zoals grsecurity en PaX (zie referenties).

Ook al is het geen directe beveiliging, het is zeer aan te raden om auditing in te zetten. Hierdoor kun je in het geval dat het misgaat altijd nog zien wie en wat er gewijzigd is op je systeem. Voor Docker kun je bijvoorbeeld denken aan:

- `/usr/bin/docker`
- `/var/lib/docker`
- `/etc/docker/daemon.json`

Een complete lijst van bestanden staat in de CIS Benchmark voor Docker (zie referenties).

Conclusie

Zoals je hebt kunnen lezen, is het geen rocket science om de security van Docker op orde te krijgen. Met gezond verstand en wat onderzoek in de opties van Docker is het zeer goed mogelijk om de security van Docker op orde te krijgen om in productie te draaien. Het is van groot belang om van Docker, net zoals bij alles wat je draait in productie, te weten wat er standaard geboden wordt om de security voor jouw specifieke omgeving zo optimaal mogelijk te krijgen. ■

REFERENTIES

1: **https://www.cisecurity.org/benchmark/docker/**
2: **https://grsecurity.net**
3: **https://pax.grsecurity.net**