Pentesting Cloud Sandboxes in the wild

Matthias Luft & Jan Harrie

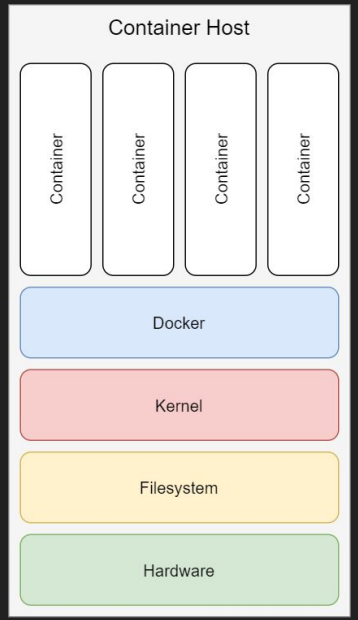
Who we are


Folgen
Matthias Luft
@uchi_mata
Principal Platform Security Engineer @Heroku @Salesforce. Opinions are my own. Pentest→Research→Leading→Security Engineering. Love Martial Arts, Outdoors, Dogs.
Seit Februar 2009 bei Twitter
341 Folge ich 771 Follower

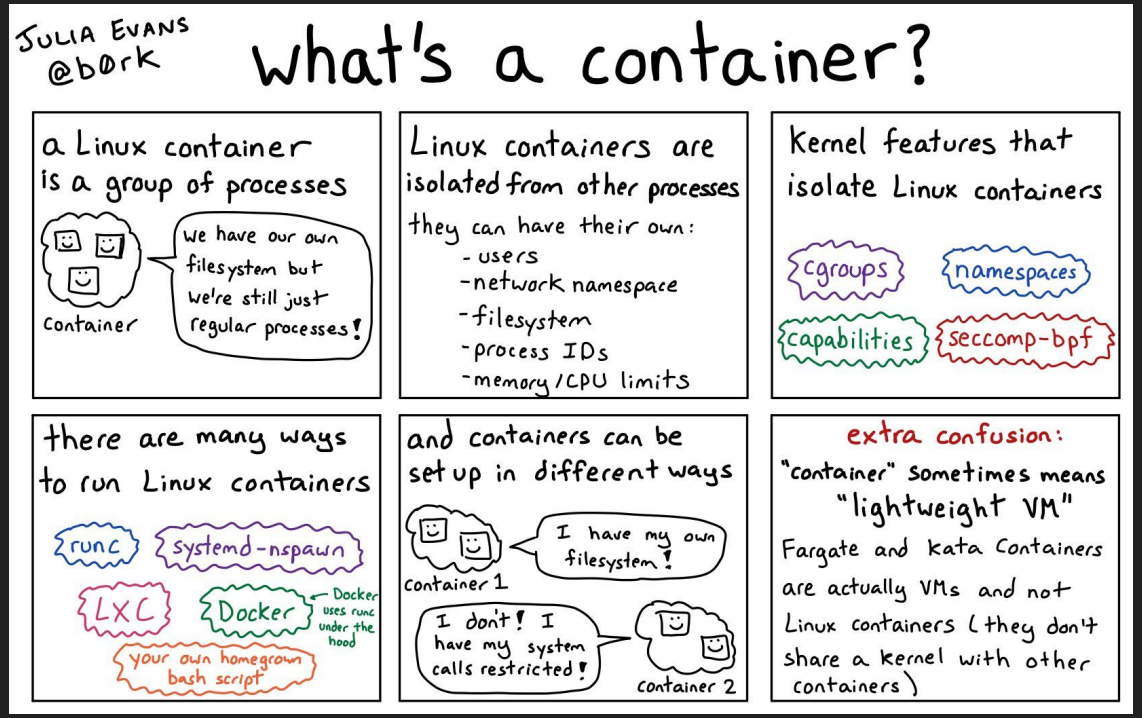

Folgen
jan
@NodyTweet
IT-Sec, Container and Orchestration, Freelancer
Milliways blog.nody.cc Seit Dezember 2011 bei Twitter
361 Folge ich 287 Follower

Short Container Re-Cap from last Year

Fucking Containers - how do they work? by Andreas Krebs, BSidesMuc 2019 [1]



Short Container Re-Cap from last Year



Source: https://twitter.com/b0rk/status/1225445956734390273

What We Will Cover

- Container Breakout Techniques & Attack Surface
- This is a very interesting topic, however:
  - Unless grossly misconfigured, not your most relevant risk.
  - Keep control plane, API, supporting services, and supply chain in mind!

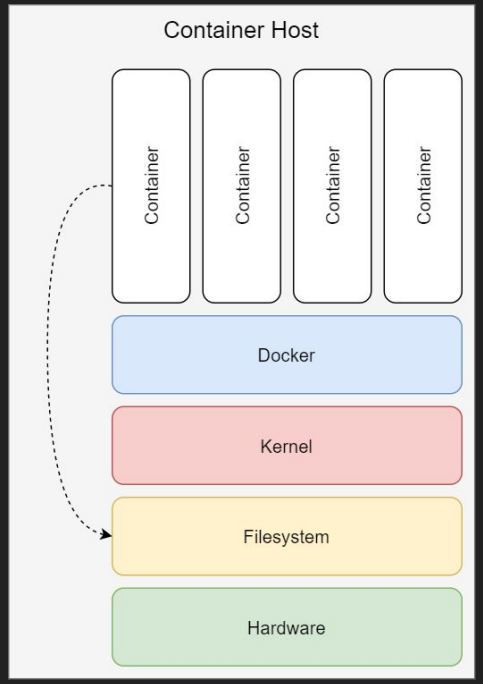
Attack Vectors – Host Filesystem

Container is started with access to host root directory

Leverage “normal” Linux privileges escalation techniques

Configure and (ab)use existing host services

⇒ Create user + SSH
⇒ Cronjob



Hands-on blog post “Container Breakouts – Part 1” [2]

Attack Vectors – Privileged Container

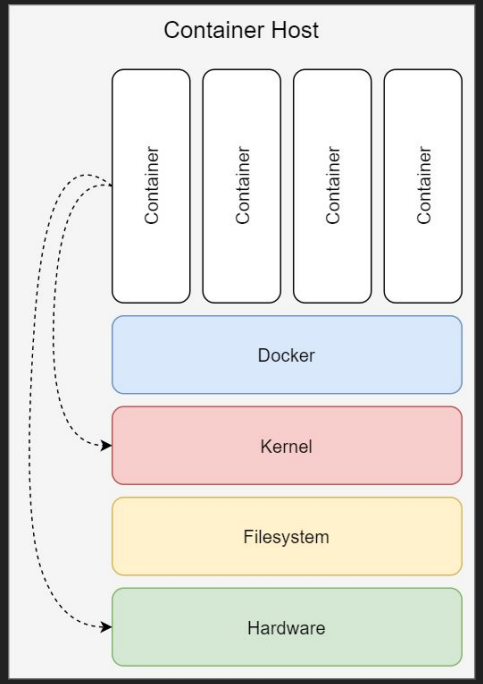
Container is started privileged

Full access to the kernel of the host
⇒ write-access to `/sys`

Leverage Capabilities [7]:
⇒ `CAP_SYS_ADMIN`
⇒ `CAP_SYS_MODULE`

Host devices
⇒ full access to `/dev`

Host keys [4]
⇒ full access to `/proc/keys`



Hands-on blog post “Container Breakouts – Part 2” [3]

Attack Vectors – Docker Socket

Container is started with mounted Docker Socket (rw)

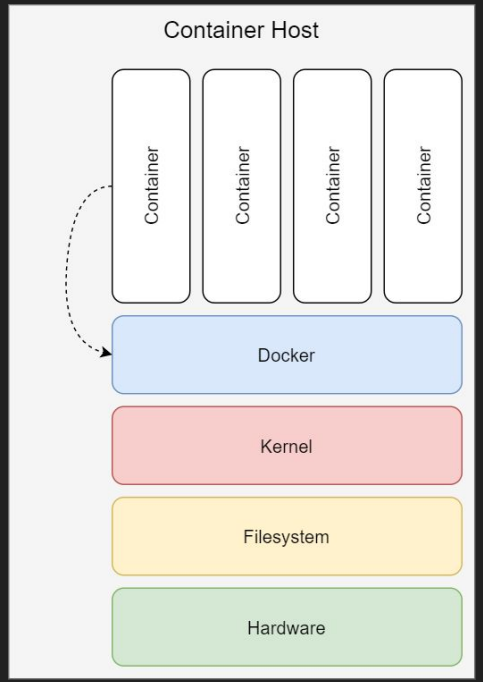
Full access to container control plain

Access to all running containers

Start “super-power” container
⇒ privileged
⇒ host network
⇒ host pid/ipc space
⇒ host hostname

⇒ finally enter host filesystem namespace:

```
nsenter -t 1 -m
```



Hands-on blog post “Container Breakouts – Part 3” [5]

Attack Vectors – Cloud Interfaces

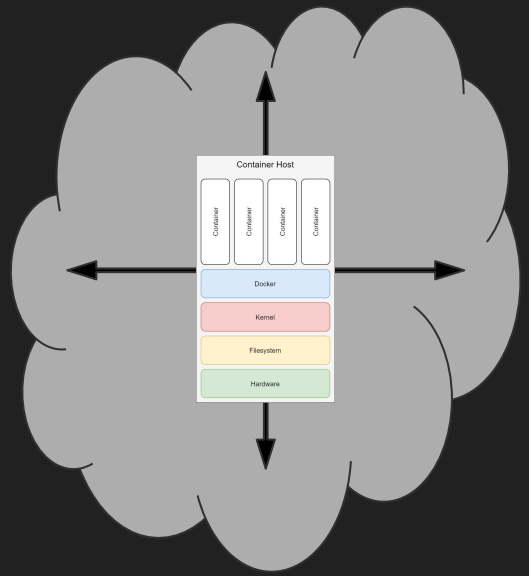
Metadata API available from inside the container

Default configuration generally less impact, **BUT**

… one is going into the cloud to use cloud features.

Over permissive configuration leads to.

- Control over workload
- Access to cloud storage
- Cloud account takeover



Cloud Platform Comparison

Service	PID Namesace	User Namespace	AppArmor Profile/ SELinux	Available Capabilities	Filtered Seccomp	Metadata-Service	Remark
AWS ECS	Yes	No	unconfined	14	0	http://169.254.169.254/	Custom Cluster
AWS Fargate	Yes	No	unconfined	14	65	None	-/-
Azure Container Instances	Yes	No	unconfined	14	65	None	-/-
Docker local	Yes	No	docker-default	13	62	None	Docker Engine
fly.io	No	No	kernel	38	disabled	None	Firecracker
Google Cloud Run	No	No	unconfined	38	disabled	http://169.254.169.254/ http://metadata.google.internal/	gVisor
Heroku	Yes	No	lxc-container-default	23	47	None	LXC

Details can be found: github.com/NodyHub/bsidesmuc2020

Summary Attack Vectors

- Host Mounts
- Privileges
- Sockets
- Control Plane

Attack Vectors Automation

- Break-out-the-box, Botb with a silent T by the amazing Chris Le Roy
  - https://github.com/brompwnie/botb

- The quite literal `amicontained` by the amazing Jezz Frazzelle
  - https://github.com/genuinetools/amicontained/

Attack Vectors Automation

- Host Mounts
- Privileges
- **Sockets**
- Control Plane

```
root@7e8f50936949:/# botb -h
[+] Break Out The Box
Usage of botb:
  -aggr string
        Attempt to exploit RuncPWN (default "nil")
  -always-succeed
        Always set BOtB's Exit code to Zero
  -autopwn
        Attempt to autopwn exposed sockets
  -cicd
        Attempt to autopwn but don't drop to TTY,return exit code 1 if successful else 0
  -config string
        Load config from provided yaml file (default "nil")
  -endpoints string
        Provide a textfile with endpoints to use for test (default "nil")
  -find-docker
        Attempt to find Dockerd
  -find-http
        Hunt for Available UNIX Domain Sockets with HTTP
  -find-sockets
        Hunt for Available UNIX Domain Sockets
  -hijack string
        Attempt to hijack binaries on host (default "nil")
  -k8secrets
        Identify and Verify K8's Secrets
  -keyMax int
         Maximum key id range (default 100000000) and max system value is 999999999 (default 100000000)
  -keyMin int
         Minimum key id range (default 1) (default 1)
```

Attack Vectors Automation

- Host Mounts
- Privileges
- Sockets
- **Control Plane**

```
root@7e8f50936949:/# botb -h
[+] Break Out The Box
Usage of botb:
  -aggr string
        Attempt to exploit RuncPWN (default "nil")
  -always-succeed
        Always set BOtB's Exit code to Zero
  -autopwn
        Attempt to autopwn exposed sockets
  -cicd
        Attempt to autopwn but don't drop to TTY,return exit code 1 if successful else 0
  -config string
        Load config from provided yaml file (default "nil")
  -endpoints string
        Provide a textfile with endpoints to use for test (default "nil")
  -find-docker
        Attempt to find Dockerd
  -find-http
        Hunt for Available UNIX Domain Sockets with HTTP
  -find-sockets
        Hunt for Available UNIX Domain Sockets
  -hijack string
        Attempt to hijack binaries on host (default "nil")
  -k8secrets
        Identify and Verify K8's Secrets
  -keyMax int
         Maximum key id range (default 100000000) and max system value is 999999999 (default 100000000)
  -keyMin int
         Minimum key id range (default 1) (default 1)
```

Attack Vectors Automation

- Host Mounts
- Privileges
- Sockets
- **Control Plane**

```
 -metadata
       Attempt to find metadata services
 -path string
       Path to Start Scanning for UNIX Domain Sockets (default "/")
 -pwn-privileged string
       Provide a command payload to try exploit --privilege CGROUP release_agent's (default "nil")
 -pwnKeyctl
       Abuse keyctl syscalls and extract data from Linux Kernel keyrings
 -recon
       Perform Recon of the Container ENV
 -region string
       Provide a AWS Region e.g eu-west-2 (default "nil")
 -rev-dns string
       Perform reverse DNS lookups on a subnet. Parameter must be in CIDR notation, e.g., -rev-dns 192.168.0.0/24
(default "nil")
 -s3bucket string
       Provide a bucket name for S3 Push (default "nil")
 -s3push string
       Push a file to S3 e.g Full command to push to https://YOURBUCKET.s3.eu-west-2.amazonaws.com/FILENAME would
be: -region eu-west-2 -s3bucket YOURBUCKET -s3push FILENAME (default "nil")
 -scrape-gcp
       Attempt to scrape the GCP metadata service
 -verbose
       Verbose output
 -wordlist string
       Provide a wordlist (default "nil")
```

Attack Vectors Automation

- **Host Mounts**
- **Privileges**
- **Sockets**
- Control Plane

```
root@7e8f50936949:/# amicontained
Container Runtime: docker
Has Namespaces:
        pid: true
        user: false
AppArmor Profile: unconfined
Capabilities:
        BOUNDING -> chown dac_override fowner fsetid kill setgid setuid setpca
p net_bind_service net_raw sys_chroot mknod audit_write setfcap
Seccomp: filtering
Blocked Syscalls (64):
        MSGRCV SYSLOG SETSID USELIB USTAT SYSFS VHANGUP PIVOT_ROOT _SYSCTL ACC
T SETTIMEOFDAY MOUNT UMOUNT2 SWAPON SWAPOFF REBOOT SETHOSTNAME SETDOMAINNAME I
OPL IOPERM CREATE_MODULE INIT_MODULE DELETE_MODULE GET_KERNEL_SYMS QUERY_MODUL
E QUOTACTL NFSSERVCTL GETPMSG PUTPMSG AFS_SYSCALL TUXCALL SECURITY LOOKUP_DCOO
KIE CLOCK_SETTIME VSERVER MBIND SET_MEMPOLICY GET_MEMPOLICY KEXEC_LOAD ADD_KEY
 REQUEST_KEY KEYCTL MIGRATE_PAGES UNSHARE MOVE_PAGES PERF_EVENT_OPEN FANOTIFY_
INIT NAME_TO_HANDLE_AT OPEN_BY_HANDLE_AT CLOCK_ADJTIME SETNS PROCESS_VM_READV
PROCESS_VM_WRITEV KCMP FINIT_MODULE KEXEC_FILE_LOAD BPF USERFAULTFD MEMBARRIER
 PKEY_MPROTECT PKEY_ALLOC PKEY_FREE IO_PGETEVENTS RSEQ
Looking for Docker.sock
```

Mitigation

- Depends heavily on your runtime
  - Most public PaaS runtime environments won’t allow insecure configurations
- If you insist on hosting yourself:
  - Do the same and harden your runtime ;-)
    - Build abstraction layer for deploying workloads with restricted set of options or
    - Lint deployment artefacts

Mitigation

- `Dockerfile`:
  - https://github.com/goodwithtech/dockle
  - https://github.com/hadolint/hadolint
- `docker-compose.yml`:
  - No tools in our lists, does not seem to be too common anymore.
- Kubernetes YML:
  - https://github.com/darkbitio/mkit
  - https://www.styra.com
  - https://github.com/cruise-automation/k-rail
  - https://github.com/zegl/kube-score
  - Roll your own PSP/OPA/Admission Controllers
    - Good input: https://aws.github.io/aws-eks-best-practices/pods/

Conclusio

- Containers in the cloud can be operated securely
  - … and are a lot of fun to use from an engineering perspective!
- The challenge is NOT to mis-configure the attack surface
  - *Control plane* also covers provider-specific APIs (see e.g. [6])
- Selection of the cloud platform depends on your flavour
  - ...or on your company policy/golf court ;)

Further Work

- Cloud Metadata Scraper
- Container privilege footprinting
- Extend botb to scan for binaries with assigned privileges
- Extend amicontained documentation to explain the performed checks

Q&A

Slides: gurke.io/bsidesmuc2020
Content: github.com/NodyHub/bsidesmuc2020

References

[1] Fucking Containers - how do they work?, Andreas Krebs, BSidesMuc 2019
https://2019.bsidesmunich.org/talks/01-03_Fucking-Containers/
https://raw.githubusercontent.com/BSidesMUC/BsidesMunich2019/master/files/01-03_Fucking-Containers.pdf

[2] Container Breakouts – Part 1, Jan Harrie
https://blog.nody.cc/posts/container-breakouts-part1/

[3] Container Breakouts – Part 2, Jan Harrie
https://blog.nody.cc/posts/container-breakouts-part2/

[4] Keyctl-unmask: "Going Florida" on The State Of Containerizing Linux Keyrings, Mark Manning
https://www.antitree.com/2020/07/keyctl-unmask-going-florida-on-the-state-of-containerizing-linux-keyrings/
https://github.com/antitree/keyctl-unmask

[5] Container Breakouts – Part 3, Jan Harrie
https://blog.nody.cc/posts/container-breakouts-part3/

[6] Azure Control Plane Vulnerabilities
https://research.checkpoint.com/2020/remote-cloud-execution-critical-vulnerabilities-in-azure-cloud-infrastructure-part-i/

[7] Linux Capabilities - False Boundaries and Arbitrary Code Execution
https://forums.grsecurity.net/viewtopic.php?t=2522

References

- https://github.com/brompwnie/botb/
- https://github.com/genuinetools/amicontained/
- https://github.com/uchi-mata/system-analysis