
intrp /lib64/ld-linux-x86-64.so.2
bintype elf
class ELF64
lang c
arch x86
bits 64
machine AMD x86-64 architecture
os linux
minopsz 1
maxopsz 16
pcalign 0
subsys linux
endian little
stripped true
static false
linenum false
lsyms false
relocs false
rpath NONE
binsz 29504
[0x00401330 15% 87 /bin/true]> pd $r @ main
;-- section_end..plt:
;-- section..text:
/ (fcn) main 160
; DATA XREF from 0x004013ed (entry0)
0x00401330 83ff02 cmp edi, 2 ; [13] va=0x00401330 pa=0x00001330 sz=12601 vsz=
,=< 0x00401333 7403 je 0x401338 ;[1]
| 0x00401335 31c0 xor eax, eax
| 0x00401337 c3 ret
| ; JMP XREF from 0x00401333 (main)
`-> 0x00401338 53 push rbx
0x00401339 488b3e mov rdi, qword [rsi]
0x0040133c 4889f3 mov rbx, rsi
0x0040133f e84c050000 call sub.fwrite_890 ;[2]
0x00401344 be8d444000 mov esi, 0x40448d
0x00401349 bf06000000 mov edi, 6
|Usage: px[afoswqWqQ][f] # Print heXadecimal
| px show hexdump
| px/ same as x/ in gdb (help x)
| pxa show annotated hexdump
| pxA show op analysis color map
| pxb dump bits in hexdump form
| pxd[124] signed integer dump (1 byte, 2 and 4)
| pxe emoji hexdump! :)
| pxi HexII compact binary representation
| pxf show hexdump of current function
| pxh show hexadecimal half-words dump (16bit
)
| pxH same as above, but one per line
| pxl display N lines (rows) of hexdump
| pxo show octal dump
| pxq show hexadecimal quad-words dump (64bit
)
| pxQ same as above, but one per line
| pxr[j] show words with references to flags and
code
[0x004013d0 16% 320 /bin/true]> x @ entry0
- offset - 0 1 2 3 4 5 6 7 8 9 A B C D E
0x004013d0 31ed 4989 d15e 4889 e248 83e4 f050 544
0x004013e0 c7c0 3044 4000 48c7 c1c0 4340 0048 c7c
0x004013f0 3013 4000 ff15 f65b 2000 f40f 1f44 000
0x00401400 b81f 7260 0055 482d 1872 6000 4883 f80
0x00401410 4889 e576 1bb8 0000 0000 4885 c074 115
0x00401420 bf18 7260 00ff e066 0f1f 8400 0000 000
0x00401430 5dc3 0f1f 4000 662e 0f1f 8400 0000 000
0x00401440 be18 7260 0055 4881 ee18 7260 0048 c1f
0x00401450 0348 89e5 4889 f048 c1e8 3f48 01c6 48d
0x00401460 fe74 15b8 0000 0000 4885 c074 0b5d bf1
0x00401470 7260 00ff e00f 1f00 5dc3 660f 1f44 000
0x00401480 803d c15d 2000 0075 1155 4889 e5e8 6ef
0x00401490 ffff 5dc6 05ae 5d20 0001 f3c3 0f1f 400
0x004014a0 bf18 6e60 0048 833f 0075 05eb 930f 1f0
0x004014b0 b800 0000 0048 85c0 74f1 5548 89e5 ffd
0x004014c0 5de9 7aff ffff 662e 0f1f 8400 0000 000
0x004014d0 4154 55ba 0500 0000 53be 4845 4000 89f
0x004014e0 31ff 4883 c480 488b 2d73 5d20 0064 488
[0x004013d0 16% 320 /bin/true]> pxw @ entry0
0x004013d0 0x8949ed31 0x89485ed1 0xe48348e2 0x495
0x004013e0 0x4430c0c7 0xc7480040 0x4043c0c1 0xc7c
0x004013f0 0x00401330 0x5bf615ff 0x0ff40020 0x000
0x00401400 0x60721fb8 0x2d485500 0x00007218 0x0ef
0x00401410 0x76e58948 0x0000b81b 0x85480000 0x5d10
0x00401420 0x607218bf 0x66e0ff00 0x00841f0f 0x000
0x00401430 0x1f0fc35d 0x2e660040 0x00841f0f 0x000x
0x00401440 0x607218be 0x81485500 0x607218ee 0xfecp
0x00401450 0xe5894803 0x48f08948 0x483fe8c1 0xd14
0x00401460 0xb81574fe 0x00000000 0x74c08548 0x18b
0x00401470 0xff006072 0x001f0fe0 0x0f66c35d 0x000r
0x00401480 0x5dc13d80 0x75000020 0x89485511 0xff6
0x00401490 0xc65dffff 0x205dae05 0xc3f30100 0x004
0x004014a0 0x606e18bf 0x3f834800 0xeb057500 0x001.
0x004014b0 0x000000b8 0xc0854800 0x4855f174 0xd0f
0x004014c0 0xff7ae95d 0x2e66ffff 0x00841f0f 0x000
0x004014d0 0xba555441 0x00000005 0x4548be53 0xfb8
0x004014e0 0x8348ff31 0x8b4880c4 0x205d732d 0x8b4x
0x004014f0 0x00282504 0x89480000 0x31782444 0xfbfp
0x00405100 6 str.POSIX
0x00405106 6 str.ASCII
0x0040510c 9 str._usr_lib
0x00405115 16 str.CHARSETALIASDIR
0x00405125 10 str._50s__50s
0x00401330 256 main
0x004013d0 1 entry0
0x00000000 0 section.
0x00000000 0 section_end.
0x00400238 28 section..interp
0x00400254 0 section_end..interp
0x00400254 32 section..note.ABI_tag
0x00400274 0 section_end..note.ABI_tag
0x00400274 36 section..note.gnu.build_id
0x00400298 0 section_end..note.gnu.build_id
0x00400298 92 section..gnu.hash
0x004002f4 0 section_end..gnu.hash
0x004002f8 1320 section..dynsym
0x00400820 0 section_end..dynsym
0x00400820 587 section..dynstr
[0x004013d0 16% 256 /bin/true]> pd $r @ entry0
;-- entry0:
0x004013d0 31ed xor ebp, ebp
0x004013d2 4989d1 mov r9, rdx
0x004013d5 5e pop rsi
0x004013d6 4889e2 mov rdx, rsp
0x004013d9 4883e4f0 and rsp, 0xfffffffffffffff0
0x004013dd 50 push rax
0x004013de 54 push rsp
0x004013df 49c7c0304440. mov r8, 0x404430
0x004013e6 48c7c1c04340. mov rcx, 0x4043c0
0x004013ed 48c7c7301340. mov rdi, 0x401330 ; section..text
0x004013f4 ff15f65b2000 call qword [rip + 0x205bf6] ; [0x606ff0:8]=0 LEA reloc._libc_start_main_240
0x004013fa f4 hlt
0x004013fb 0f1f440000 nop dword [rax + rax]
0x00401400 b81f726000 mov eax, 0x60721f ; "6.1.1 20160802" @ 0x60721f
[0x004013d0 16% 2560 /bin/true]> pxA @ entry0
Kubernetes
Security

Agenda

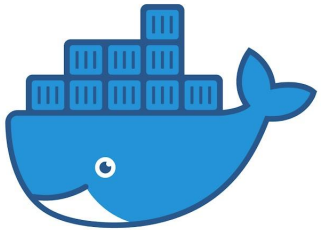
- Basic concepts
- Architecture
- Networking
- Direct access & security
- CVE-2018-1002105 explained

Why me

- I am guilty of setting-up a Kubernetes cluster on-premise in 2015/16
- (Un)fortunately it is still in-use today
- CoreOS + manually installed systemd services
- Bootstrapped through kubelet (migrated from fleet)
- Currently running k8s 1.14
- N+1 redundancy (one node at at time automatically reboots&upgrades with CLUO)
- **Don’t try this at ~~home~~ work** (or if you need to, follow Kelsey Hightower: Kubernetes The Hard Way)
- Simple way: **kops** (for cloud) and **kubeadm** (for on-prem)

Docker

- Layer around Linux namespaces and cgroups
- Virtualisation vs. containerization
- Docker image is a standard
- Pro-tip: **don’t just blindly run someone else’s code from DockerHub on your machine**
- Running an image with --privileged or mounting /dev
  = giving direct access to your host



DockerHub

Authored by: Kent Lamb

Unauthorized access to Docker Hub database

Article ID: KB000968

HUB SECURITY

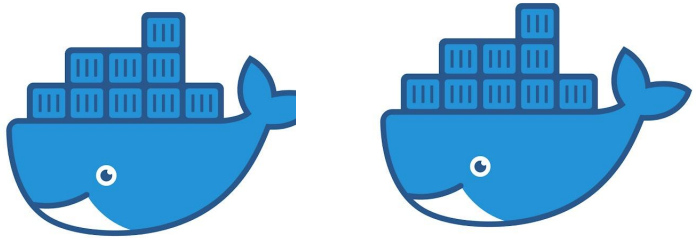
Issue

On Thursday, April 25th, 2019, we discovered unauthorized access to a single Docker Hub database storing a subset of non-financial user data. Upon discovery, we acted quickly to intervene and secure the site.

We want to update you on what we've learned from our ongoing investigation, including which Hub accounts are impacted, and what actions users should take.

Container manager

How to manage workload (multiple dockers on multiple server).
Diverse workload: batch processing, realtime



- **Kubernetes**
- Docker Swarm
- ...
- CoreOS fleet -> Kubernetes
- Rancher -> wrapper (Kubernetes)

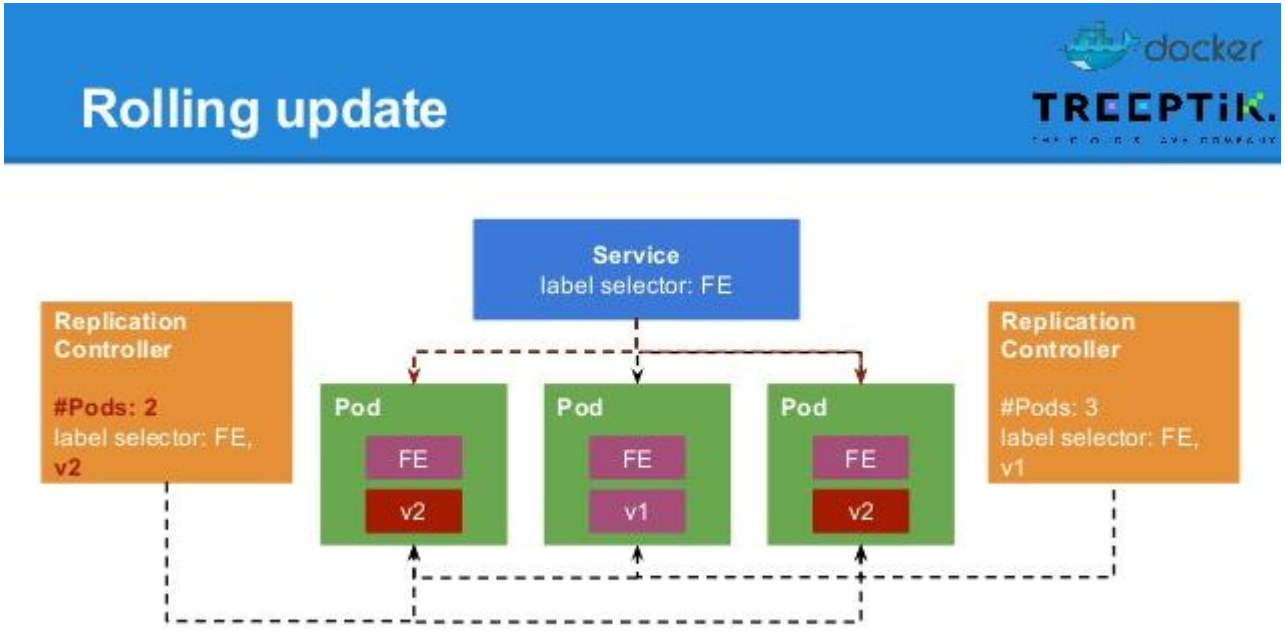
Kubernetes

- Means helmsman / pilot
- Open-sourced from Google 2014 (inspired by Borg), written in Go
- De-facto standard API for IaaS
- Helm is the package manager for k8s

Core concepts

- Label (everything can be tagged with a key/value pair)
- Pod = group of containers running on the same node (~ VM)
- Service = a way to expose network services from a pod
- Replication-controller = makes sure there are N pods
  - deprecated in favor of **ReplicaSets**
  - or declarative configuration - you usually have a **Deployment** object
  - **Deployment** manages **ReplicaSets** which in turn manage **pods**

Tying it all together



Architecture (1/2)

- Administrator
  - **kubectl** (available even as a Windows binary)
- Worker (minion)
  - **kubelet** (docker orchestrator, can be standalone / bootstrap everything else from static config files)
  - **kube-proxy** (enables simple service discovery, more about it in networking section)
- Master
  - **kube-apiserver** (stateless, receives HTTP REST requests)
  - **kube-scheduler** (schedules stuff around the cluster)
  - **kube-controller-manager** (implements the feedback loop - desired state, spec -> actual state, status)

Architecture (2/2)

- Database
  - well actually there is none :), you need **etcd** (distributed reliable key-value store)
  - Raft algorithm and quorum -> having an even amount of servers is stupid
- Only api server will write and read
  - Authentication
  - Authorization
  - Admission controller

Networking (1/2)

- Docker containers are usually connected to a bridge
  … and then you expose ports
- Of course there is also host networking
- But when you have lot of containers (on different hosts) this becomes messy (you want to abstract where a container is running)
- Pods (multiple containers with same IP) are in a flat (virtual) network
- So they can talk among themselves without problems
- However a random IP is assigned to each pod

Networking (2/2)

- You need to use services (which is just a destination NAT trick through kube-proxy for interpod communication - proxy, iptables rules, lvs)
- NodePort service is basically similar to docker & expose port
- In the cloud “service” (with type LoadBalancer) can automatically configure e.g., AWS ELB
- Poor-man’s solution - **MetalLB**

Networking - how does the illusion work?

- Flat net is created with an overlay network (**beware of hybrid cloud set-up and plaintext traffic)**
- We use Flannel (but you could have Calico, Weave, or sth else)
- Called CNI (container network interface)
- VXLAN (L2 encapsulated in L4 UDP port 4789) vs. VLAN (802.1q)

Flannel

Implement IPSEC mode #6

New issue

Open **eyakubovich** opened this issue on 13 Aug 2014 · 26 comments

eyakubovich commented on 13 Aug 2014 — Contributor

For users that require network level encryption, flannel should be able to use IPSEC as the encapsulation protocol.

👍 22 ❤️ 1

eyakubovich added the **enhancement** label on 13 Aug 2014

Assignees

No one assigned

Labels

kind/enhancement

Projects

None yet

Kubelet

- kube-apiserver talks to kubelet to make it start containers
- “By default kubelet allows anonymous authentication”

```
--anonymous-auth
    Enables anonymous requests to the Kubelet server. Requests that are not
    rejected by another authentication method are treated as anonymous
    requests. Anonymous requests have a username of system:anonymous, and a
    group name of system:unauthenticated. (default true)
```

- “The default authorization mode is AlwaysAllow, which allows all requests”
- When you see 10250/TCP open there is a big chance you can **execute code**

Kubelet - verification

```
$ curl -k https://localhost:10250/pods/ | jq . | more
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  145k    0  145k    0     0  5397k      0 --:--:-- --:--:-- --:--:-- 5397k
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {},
  "items": [
    {
      "metadata": {
        "name": "kube-proxy-lju1-mts-coreos4",
        "namespace": "kube-system",
        "selfLink": "/api/v1/namespaces/kube-system/pods/kube-proxy-lju1-mts-coreos4",
        "uid": "fbec2ae2a98e09a0c95bf2c1bcbbb1f7",
        "creationTimestamp": null,
        "annotations": {
          "kubernetes.io/config.hash": "fbec2ae2a98e09a0c95bf2c1bcbbb1f7",
          "kubernetes.io/config.seen": "2019-04-27T00:31:37.07508847Z",
          "kubernetes.io/config.source": "file"
        }
      },
      "spec": {
        "volumes": [
          {
            "name": "etc-kubernetes",
```

Exposed database (1/2)

- State is stored in etcd (scan for TCP ports 2379/2380 or legacy 4001/7001)
- Kubernetes heavily relies on JWT, so leaking a service token is (often) enough
- RCE -> update deployment and let controller-manager start rogue pods
- How to protect?
  - etcd3 supports roles, but easiest is just to authenticate with cert
  - encrypt data at rest - from k8s side (EncryptionConfig yaml)

Exposed database (2/2)

```
$ ETCDCTL_API=3 etcdctl get --endpoints=http://127.0.0.1:2379 /registry/secrets/fiction/default-token-n6tsp | ta
il -n +2 | jq . | head -15
{
  "kind": "Secret",
  "apiVersion": "v1",
  "metadata": {
    "name": "default-token-n6tsp",
    "namespace": "fiction",
    "uid": "10c32037-ba4a-11e7-ab84-aa0000eecfd2",
    "creationTimestamp": "2017-10-26T12:34:52Z",
    "annotations": {
      "kubernetes.io/service-account.name": "default",
      "kubernetes.io/service-account.uid": "10b2c4dc-ba4a-11e7-ab84-aa0000eecfd2"
    }
  },
  "data": {
    "ca.crt": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURNakNDQWhxZ0F3SUJBZ0lKQUs2VVhVc0FpQlpaTUEwR0NTcUdTSWIzR
FFFQkN3VUFNQlV4RXpBUkJnTlYKQkFNTUNtdDFZbVZ5Ym1WMFpYTXdIaGNOTVRZd01qRTFNVEV4TURNd1doY05Nall3TWpFeU1URXhNRE13V2pBV
gpNUk13RVFZRFZRUUREQXByZFdKbGNtNWxkR1Z6TUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FROEFNSUlCCkNnS0NBUUVBeTNjbVlaclBvc
3IyalNUZERsdldqV3BuWlBVNUd0dEJESlJGR0pmOXVZSWlpd3NEOWlucDRQR1AKZlBUZW9hc2xlMWsvQVIxOXRORHJXZnY5U0lGWE94dThwZHlBc
```

“Quasi” SSRF

- The normal SSRF is still a vector (try https://first_ip_in_cluster_ip_range, https://kubernetes.cluster.local or sth)

- Additionally: each pod by default contains /var/run/secrets/kubernetes.io/serviceaccount/
- So rogue code just read the “token” file
- Contact kube-api with that JWT
- Depending on privileges (in kube-system pod -> game-over)
- You could at least see all other pods, if not execute custom code and/or DoS existing stuff in the cluster

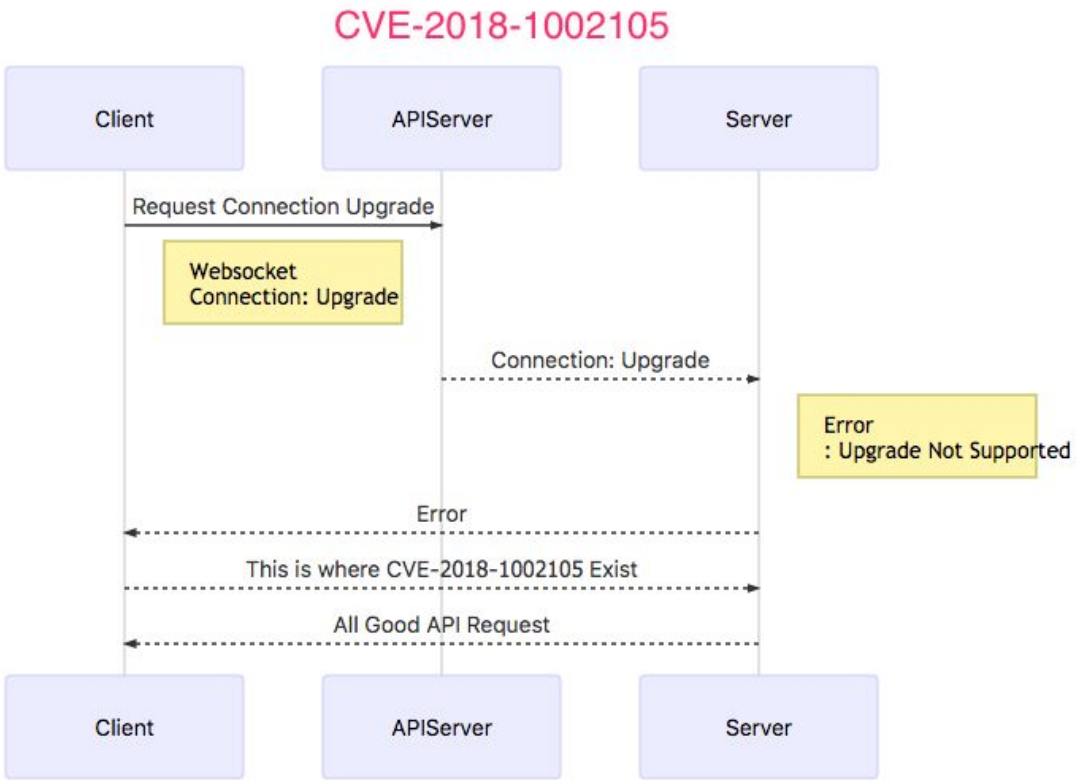
Critical security issue

“*In all Kubernetes versions prior to v1.10.11, v1.11.5, and v1.12.3, incorrect handling of error responses to proxied upgrade requests in the kube-apiserver allowed specially crafted requests to establish a connection through the Kubernetes API server to backend servers, then send arbitrary requests over the same connection directly to the backend, authenticated with the Kubernetes API server's TLS credentials used to establish the backend connection.*” --CVE-2018-1002105

Exec (or portforward) permission to cluster-admin

https://github.com/evict/poc_CVE-2018-1002105

Prettier picture



Checklist

- Enable RBAC
- Use TLS between all components (also etcd!) and use IPsec for hybrid set-ups
- Patch your servers (make sure you have redundancy to be able to do it at any time)
- Try to not execute privileged containers
- Beware of DockerHub
- For untrusted code (like CTF challenges) make sure to “overwrite” /run/secrets that gets mounted inside of pod

Questions

?