Цель работы

Изучение принципов работы виртуализации на уровне ОС на примере контейнеров Docker.

Задачи

В ходе выполнения лабораторной работы необходимо выполнить следующие действия:

1. Разработать Dockerfile для сервиса из варианта задания. Номер варианта – номер в списке группы по модулю количества студентов в группе.
2. Определить минимальный набор привилегий, необходимых для работы сервиса.
3. Настроить с помощью Docker и/или сторонних утилит привилегии для процесса и/или файлов в соответствии с пунктом 2.
4. Определить минимальный набор системных вызовов, необходимых для работы сервиса, и написать соответствующий профиль seccomp.

Ход работы

В ходе выполнения данной лабораторной работы было необходимо реализовать докер для сервиса samba. Samba — это программное обеспечение для организации обмена файлами и работы с общими ресурсами между компьютерами под управлением Linux/Unix и операционной системой Windows.

Инструкция EXPOSE показывает, какой порт пробрасывать из контейнера. Порты 137, 138, 139 для работы NetBIOS. 445 – для samba. В Windows SMB может работать напрямую через TCP/IP без необходимости использования NetBIOS через TCP/IP. Он будет использовать порт 445. В других системах найдутся службы и приложения, использующие порт 139. Это означает, что SMB работает с NetBIOS через TCP/IP. NetBIOS через TCP/IP имеет псевдоним NetBT (NBT). NetBIOS через TCP/IP представляет собой промежуточный уровень между NetBIOS и TCP/IP и создан для того, чтобы приложения на базе NetBIOS могли работать в сетях TCP/IP, то есть предназначен для отображения имен NetBIOS в IP-адреса и, наоборот.

На Рисунке 1 представлен Dokcerfile.

```
1 FROM ubuntu:22.04
2
3 LABEL maintainer="Artyom"
4 LABEL version="0.1"
5 LABEL description="Docker for Samba"
6 EXPOSE 137/udp 138/udp 139 445
7
8 RUN apt-get update && apt-get install -y samba && apt-get install -y vim && rm -rf /var/lib/apt/lists/*
9 RUN mkdir /dir_share_test
10
11 COPY smb.conf /etc/samba/smb.conf
12 CMD /usr/sbin/smbd -F
```

Рисунок 1 – Разработанный Dockerfile

RUN создаёт слой во время запуска. Docker фиксирует состояние образа после каждой инструкции RUN. Чаще всего используется для установки нужных пакетов внутрь контейнера. В нашем случае обновляются списки пакетов из репозиториев, устанавливается samba, устанавливается vim. Vim был установлен для возможности редактирования конфига при запущенном докере. rm -rf /var/lib/apt/lists/* позволяет очистить кэш apt и приводит к тому, что он не сохраняется в слое, сформированном командой RUN. С помощью RUN создается папка /dir_share_test.

COPY — копирует файлы и директории в контейнер. Копируем новый конфиг, в котором указаны параметры папки, которая будет раздаваться.

CMD — указывает команду и аргументы для выполнения внутри контейнера. Параметры могут быть переопределены. Использоваться может только одна инструкция CMD. В данном случае CMD запускает samba.

На рисунке ниже (Рисунок 2) представлена дописанная часть конфигурационного файла Samba.

```
243 [share]
244     path = /dir_share_test
245     available = yes
246     writable = yes
247     browseable = yes
248     public = yes
```

Рисунок 2 – Описание папки в smb.conf

Далее при помощи команды «sudo docker build -t my_samba .» был собран докер, представленный на рисунке 3.

```
art@art-virtual-machine:~/Desktop/lab3$ sudo docker build -t my_smaba .
[sudo] password for art:
[+] Building 1.5s (9/9) FINISHED
 => [internal] load .dockerignore                                                                    0.0s
 => => transferring context: 2B                                                                      0.0s
 => [internal] load build definition from Dockerfile                                                 0.0s
 => => transferring dockerfile: 360B                                                                 0.0s
 => [internal] load metadata for docker.io/library/ubuntu:22.04                                      1.4s
 => [1/4] FROM docker.io/library/ubuntu:22.04@sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b5  0.0s
 => [internal] load build context                                                                    0.0s
 => => transferring context: 30B                                                                     0.0s
 => CACHED [2/4] RUN apt-get update && apt-get install -y samba && apt-get install -y vim && rm -rf /var/lib/  0.0s
 => CACHED [3/4] RUN mkdir /dir_share_test                                                           0.0s
 => CACHED [4/4] COPY smb.conf /etc/samba/smb.conf                                                   0.0s
 => exporting to image                                                                               0.0s
 => => exporting layers                                                                              0.0s
 => => writing image sha256:78084c1a26decae6a877ae5aa210e3364591f2a6ccc908a2a68f23cd091527c2         0.0s
 => => naming to docker.io/library/my_smaba                                                          0.0s
```

Рисунок 3 – Сборка докера my_samba

После докер был запущен при помощи команды, представленной на рисунке 4. В приведенной команде используется $PWD – позволяет узнать текущую папку. -v – сопоставляет каталог узла с каталогом контейнера docker.

```
art@art-virtual-machine:~/Desktop/lab3$ sudo docker run -it -p 139:139 -p 445:445 -v $PWD/dir_share_test:/dir_share_test -d my_samba
a7057347f0fe0fc2449dea80ab75194fe879e9823532f47bf45dfbd44b07eaa6
art@art-virtual-machine:~/Desktop/lab3$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED         STATUS         PORTS                                                                                                NAMES
a7057347f0fe   my_samba   "/bin/sh -c '/usr/sb…"   8 seconds ago   Up 7 seconds   0.0.0.0:139->139/tcp, :::139->139/tcp, 137-138/udp, 0.0.0.0:445->445/tcp, :::445->445/tcp   vigorous_margulis
```

Рисунок 4 – Запуск докера

При помощи команды, представленной на рисунке ниже, можно запустить оболочку bash для контейнера. Используя ее, можно посмотреть содержимое внутренней ФС докера, как показано на рисунке 5.

Рисунок 5 – Настройка ФС докера

Для проверки того, что разработанный докер корректно работает, были открыты видимые общие папки на хосте через проводник. В результате был виден созданный файл file.txt.

Далее был определен минимальный набор привилегий, необходимых для работы сервиса. По умолчанию Docker отбрасывает все возможности, кроме тех, которые необходимы, используя подход белого списка. Данные привилегии представлены на рисунке ниже.

Таблица 1 – Возможные привилегии с их описаним.

Привилегия	**Описание**
CHOWN	Позволяет вносить произвольные изменения в идентификаторы файлов и GID.
DAC_OVERRIDE	Позволяет обходить проверки разрешений ядра при операциях чтения, записи и выполнения файлов.
FSETID	Не будут очищены биты режима set-user-ID и set-group-ID даже при изменении файла.
FOWNER	Позволяет процессам изменять режим и списки доступа, мандатную метку, расширенные атрибуты (см. статью) и флаги любых файлов так, словно процесс выполняется от лица владельца файла.
MKNOD	Создание специальных файлов.

NET_RAW	Для создания «необработанных» raw и пакетных packet сокетов.
SETGID	Произвольные манипуляции с идентификаторами процессов и дополнительным списком GID; подделка GID при передаче учетных данных сокетов через доменные сокеты UNIX; запись сопоставления идентификаторов групп в пользовательском пространстве имен.
SETUID	Произвольные манипуляции с uid процессов; подделка UID при передаче учетных данных сокетов через доменные сокеты UNIX; запись сопоставления идентификаторов пользователей в пользовательском пространстве имен.
SETFCAP	Устанавливать «файловые» привилегии.
SETPCAP	Если файловые возможности не поддерживаются: предоставьте или удалите любую возможность в разрешенном наборе возможностей вызывающего абонента для любого другого процесса или из него.
NET_BIND_SERVICE	Привязать сокет к привилегированным портам домена Интернета (номера портов меньше 1024).
SYS_CHROOT	Используйте chroot и изменяйте пространства имен с помощью наборов.
KILL	Разрешает посылать сигналы процессам любых пользователей
AUDIT_WRITE	Запись записей в журнал аудита ядра.

При помощи –cap-drop сбрасывались поочередно приведенные выше привилегии, после чего производилась попытка запуска. В результате было определено, что необходимо 3 привилегии: setgid, net_bind_service и setgid.

Далее необходимо было запустить сервис с минимальным набором системных вызовов. Seccomp - один из механизмов безопасности ядра Linux, который обеспечивает возможность ограничивать набор доступных системных вызовов для приложений.

Docker использует seccomp в режиме фильтра и имеет свой собственный DSL на основе JSON, который позволяет определять профили, компилируемые в фильтры seccomp. При запуске контейнера он получает профиль seccomp по умолчанию или профиль, указанные с помощью флага --security-opt.

Всего разрешено 103 системных вызова, а действие по-умолчанию – SCMP_ACT_ERRNO. Структура файла profile.json показана в приложении 1.

Ниже на рисунке 6 представлен запуск докера с новым seccomp профилем.

Рисунок 6 – Запуск докера с профилем seccomp и ограниченными привилегиями

Среди разрешенных системных вызовов можно отметить:

1. mmap – без него невозможна подгрузка динамических библиотек.

2. futex – без него контейнер падает с ошибкой и stack trace.

3. accept – без него контейнер запускается, но сервер не принимает соединения.

4. getcwd – без него контейнер запускается, но получить тестовый файл не удается.

5. setsid – аналогично, не отправляет тестовый файл.

6. sendmsg – без него не удается подключится к серверу.

7. epoll_create – без него сервер не принимает соединения.

8. dup2 – сервер принимает соединение, при попытке получения файла падает.

9. get_ppid – сервер не запускается.

10. rt_sig_proc_mask – сервер принимает соединение, но не удается скачать тестовый файл.

Контрольные вопросы

1. Какая стратегия по выбору и использованию привилегий для контейнера Docker является более правильной и почему?

- Так как Docker предоставляет контейнерам довольно широкий спектр привилегий, актуальна проблема безопасности работы приложений в этих контейнерах. Необходимо максимально ограничивать возможные привилегии процессов, а также используемые ими системные вызовы, так как это увеличит защищенность системы и поможет выявить вредоносное приложение, так как оно будет требовать привилегии или вызовы, не свойственные приложениям такого типа.

2. Какие привилегии разрешены в Docker по умолчанию?

- По умолчанию Docker сбрасывает все привилегии и подключает те, что находятся в так называемом белом списке. Без дополнительных настроек Docker’ом подключаются привилегии: CAP_CHOWN, CAP_DAC_OVERRIDE, CAP_FSEDIT, CAP_FOWNER, CAP_MKNOD, CAP_NET_RAW, CAP_SETGID, CAP_SERUID, CAP_SETPCAP, CAP_NET_BIND_SERVOCE, CAP_SYS_CHROOT, CAP_KILL, CAP_AUDIT_WRITE

3. Какие типы действий существуют для Docker seccomp профилей и за что они отвечают?

- SCMP_ACT_ALLOW – разрешение доступа к системному вызову; SCMP_ACT_ERRNO – зпрет доступа к системному вызову.

Вывод

В ходе выполнения данной лабораторной работы были изучены принципы работы виртуализации на уровне ОС на примере контейнеров Docker. Был разработан докер для сервиса samba. Для данного докера был определен минимальный набор привилегий и минимальный набор системных вызовов.

Приложение А

profile.json


```
{
	"defaultAction": "SCMP_ACT_ERRNO",
	"defaultErrnoRet": 1,
	"archMap": [
		{
			"architecture": "SCMP_ARCH_X86_64",
			"subArchitectures": [
				"SCMP_ARCH_X86",
				"SCMP_ARCH_X32"
			]
		},
		{
			"architecture": "SCMP_ARCH_AARCH64",
			"subArchitectures": [
				"SCMP_ARCH_ARM"
			]
		},
		{
			"architecture": "SCMP_ARCH_MIPS64",
			"subArchitectures": [
				"SCMP_ARCH_MIPS",
				"SCMP_ARCH_MIPS64N32"
			]
		},
		{
			"architecture": "SCMP_ARCH_MIPS64N32",
			"subArchitectures": [
				"SCMP_ARCH_MIPS",
				"SCMP_ARCH_MIPS64"
			]
		},
		{
			"architecture": "SCMP_ARCH_MIPSEL64",
			"subArchitectures": [
				"SCMP_ARCH_MIPSEL",
				"SCMP_ARCH_MIPSEL64N32"
			]
		},
		{
			"architecture": "SCMP_ARCH_MIPSEL64N32",
			"subArchitectures": [
				"SCMP_ARCH_MIPSEL",
				"SCMP_ARCH_MIPSEL64"
```

```
                ]
        },
        {
                "architecture": "SCMP_ARCH_S390X",
                "subArchitectures": [
                        "SCMP_ARCH_S390"
                ]
        },
        {
                "architecture": "SCMP_ARCH_RISCV64",
                "subArchitectures": null
        }
],
"syscalls": [
        {
                "names": [
                        "accept",
                        "access",
                        "alarm",
                        "arch_prctl",
                        "bind",
                        "brk",
                        "capget",
                        "capset",
                        "chdir",
                        "chown",
                        "close",
                        "connect",
                        "dup2",
                        "epoll_create",
                        "epoll_ctl",
                        "epoll_pwait",
                        "epoll_wait",
                        "eventfd",
                        "eventfd2",
                        "execve",
                        "exit",
                        "exit_group",
                        "faccessat",
                        "fallocate",
                        "fchmodat",
                        "fcntl",
                        "fdatasync",
                        "fgetxattr",
                        "flock",
                        "fstat",
                        "fstatfs",
```

```
                "ftruncate",
                "futex",
                "getcwd",
                "getdents64",
                "getegid",
                "geteuid",
                "getgid",
                "getgroups",
                "getpeername",
                "getpgrp",
                "getpid",
                "getppid",
                "getrandom",
                "getresgid",
                "getresuid",
                "getsockname",
                "getsockopt",
                "gettid",
                "getuid",
                "getxattr",
                "ioctl",
                "listen",
                "listxattr",
                "lseek",
                "madvise",
                "mkdir",
                "mmap",
                "mprotect",
                "msync",
                "munmap",
                "nanosleep",
                "newfstatat",
                "openat",
                "pipe",
                "pipe2",
                "prctl",
                "pread64",
                "pselect6",
                "read",
                "readlink",
                "readv",
                "recvmsg",
                "rename",
                "rt_sigaction",
                "rt_sigprocmask",
                "rt_sigreturn",
                "sched_yield",
```

```
                "sendmsg",
                "sendto",
                "setgid",
                "setgroups",
                "setpgid",
                "setresgid",
                "setresuid",
                "set_robust_list",
                "setsid",
                "setsockopt",
                "set_tid_address",
                "setuid",
                "socket",
                "socketpair",
                "statfs",
                "sysinfo",
                "tgkill",
                "umask",
                "uname",
                "unlink",
                "utimensat",
                "vfork",
                "wait4",
                "write",
                "writev"

        ],
        "action": "SCMP_ACT_ALLOW"
},
{
        "names": [
                "process_vm_readv",
                "process_vm_writev",
                "ptrace"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "minKernel": "4.8"
        }
},
{
        "names": [
                "personality"
        ],
        "action": "SCMP_ACT_ALLOW",
        "args": [
                {
```

```
                        "index": 0,
                        "value": 0,
                        "op": "SCMP_CMP_EQ"
                }
        ]
},
{
        "names": [
                "personality"
        ],
        "action": "SCMP_ACT_ALLOW",
        "args": [
                {
                        "index": 0,
                        "value": 8,
                        "op": "SCMP_CMP_EQ"
                }
        ]
},
{
        "names": [
                "personality"
        ],
        "action": "SCMP_ACT_ALLOW",
        "args": [
                {
                        "index": 0,
                        "value": 131072,
                        "op": "SCMP_CMP_EQ"
                }
        ]
},
{
        "names": [
                "personality"
        ],
        "action": "SCMP_ACT_ALLOW",
        "args": [
                {
                        "index": 0,
                        "value": 131080,
                        "op": "SCMP_CMP_EQ"
                }
        ]
},
{
        "names": [
```

```
                "personality"
            ],
            "action": "SCMP_ACT_ALLOW",
            "args": [
                {
                    "index": 0,
                    "value": 4294967295,
                    "op": "SCMP_CMP_EQ"
                }
            ]
        },
        {
            "names": [
                "sync_file_range2",
                "swapcontext"
            ],
            "action": "SCMP_ACT_ALLOW",
            "includes": {
                "arches": [
                    "ppc64le"
                ]
            }
        },
        {
            "names": [
                "arm_fadvise64_64",
                "arm_sync_file_range",
                "sync_file_range2",
                "breakpoint",
                "cacheflush",
                "set_tls"
            ],
            "action": "SCMP_ACT_ALLOW",
            "includes": {
                "arches": [
                    "arm",
                    "arm64"
                ]
            }
        },
        {
            "names": [
                "arch_prctl"
            ],
            "action": "SCMP_ACT_ALLOW",
            "includes": {
                "arches": [
```

```
                        "amd64",
                        "x32"
                ]
        }
},
{
        "names": [
                "modify_ldt"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "arches": [
                        "amd64",
                        "x32",
                        "x86"
                ]
        }
},
{
        "names": [
                "s390_pci_mmio_read",
                "s390_pci_mmio_write",
                "s390_runtime_instr"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "arches": [
                        "s390",
                        "s390x"
                ]
        }
},
{
        "names": [
                "riscv_flush_icache"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "arches": [
                        "riscv64"
                ]
        }
},
{
        "names": [
                "open_by_handle_at"
        ],
```

```
                "action": "SCMP_ACT_ALLOW",
                "includes": {
                        "caps": [
                                "CAP_DAC_READ_SEARCH"
                        ]
                }
        },
        {
                "names": [
                        "bpf",
                        "clone",
                        "clone3",
                        "fanotify_init",
                        "fsconfig",
                        "fsmount",
                        "fsopen",
                        "fspick",
                        "lookup_dcookie",
                        "mount",
                        "mount_setattr",
                        "move_mount",
                        "name_to_handle_at",
                        "open_tree",
                        "perf_event_open",
                        "quotactl",
                        "quotactl_fd",
                        "setdomainname",
                        "sethostname",
                        "setns",
                        "syslog",
                        "umount",
                        "umount2",
                        "unshare"
                ],
                "action": "SCMP_ACT_ALLOW",
                "includes": {
                        "caps": [
                                "CAP_SYS_ADMIN"
                        ]
                }
        },
        {
                "names": [
                        "clone"
                ],
                "action": "SCMP_ACT_ALLOW",
                "args": [
```

```
                {
                        "index": 0,
                        "value": 2114060288,
                        "op": "SCMP_CMP_MASKED_EQ"
                }
        ],
        "excludes": {
                "caps": [
                        "CAP_SYS_ADMIN"
                ],
                "arches": [
                        "s390",
                        "s390x"
                ]
        }
},
{
        "names": [
                "clone"
        ],
        "action": "SCMP_ACT_ALLOW",
        "args": [
                {
                        "index": 1,
                        "value": 2114060288,
                        "op": "SCMP_CMP_MASKED_EQ"
                }
        ],
        "comment": "s390 parameter ordering for clone is different",
        "includes": {
                "arches": [
                        "s390",
                        "s390x"
                ]
        },
        "excludes": {
                "caps": [
                        "CAP_SYS_ADMIN"
                ]
        }
},
{
        "names": [
                "clone3"
        ],
        "action": "SCMP_ACT_ERRNO",
        "errnoRet": 38,
```

```
                "excludes": {
                        "caps": [
                                "CAP_SYS_ADMIN"
                        ]
                }
        },
        {
                "names": [
                        "reboot"
                ],
                "action": "SCMP_ACT_ALLOW",
                "includes": {
                        "caps": [
                                "CAP_SYS_BOOT"
                        ]
                }
        },
        {
                "names": [
                        "chroot"
                ],
                "action": "SCMP_ACT_ALLOW",
                "includes": {
                        "caps": [
                                "CAP_SYS_CHROOT"
                        ]
                }
        },
        {
                "names": [
                        "delete_module",
                        "init_module",
                        "finit_module"
                ],
                "action": "SCMP_ACT_ALLOW",
                "includes": {
                        "caps": [
                                "CAP_SYS_MODULE"
                        ]
                }
        },
        {
                "names": [
                        "acct"
                ],
                "action": "SCMP_ACT_ALLOW",
                "includes": {
```

```
                "caps": [
                        "CAP_SYS_PACCT"
                ]
        }
},
{
        "names": [
                "kcmp",
                "pidfd_getfd",
                "process_madvise",
                "process_vm_readv",
                "process_vm_writev",
                "ptrace"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "caps": [
                        "CAP_SYS_PTRACE"
                ]
        }
},
{
        "names": [
                "iopl",
                "ioperm"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "caps": [
                        "CAP_SYS_RAWIO"
                ]
        }
},
{
        "names": [
                "settimeofday",
                "stime",
                "clock_settime",
                "clock_settime64"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "caps": [
                        "CAP_SYS_TIME"
                ]
        }
},
```

```
{
        "names": [
                "vhangup"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "caps": [
                        "CAP_SYS_TTY_CONFIG"
                ]
        }
},
{
        "names": [
                "get_mempolicy",
                "mbind",
                "set_mempolicy"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "caps": [
                        "CAP_SYS_NICE"
                ]
        }
},
{
        "names": [
                "syslog"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "caps": [
                        "CAP_SYSLOG"
                ]
        }
},
{
        "names": [
                "bpf"
        ],
        "action": "SCMP_ACT_ALLOW",
        "includes": {
                "caps": [
                        "CAP_BPF"
                ]
        }
},
{
```

```
                        "names": [
                                "perf_event_open"
                        ],
                        "action": "SCMP_ACT_ALLOW",
                        "includes": {
                                "caps": [
                                        "CAP_PERFMON"
                                ]
                        }
                }
        ]
}
```