**Questions:**

1. Which capabilities API (seccomp-bpf, AppArmor, or SELinux) did you choose? Why did you make that choice?

**Answer**: We used seccomp-bpf. SELinux and AppArmour are technologies that are applied to existing unmodified applications. And as we were modifying the existing applications according to the required system calls and adding these system calls, we used seccomp-bpf.

1. What was the process you used to ascertain the list of system calls required by each program?

**Answer:** We usedstrace.

1. What system calls are needed by each?

**Answer:**

In Server.c

// Setup basic whitelist

|  |
| --- |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(rt\_sigreturn), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(exit), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(read), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(write), 0); |
|  |
| // Add system calls to seccomp-bpf |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(socket), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(setsockopt), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(bind), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(listen), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(accept), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(clone), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(wait4), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(exit\_group), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(open), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(execve), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(brk), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(access), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(fstat), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(mmap), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(close), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(mprotect), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(arch\_prctl), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(munmap), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(chdir), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(chroot), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(dup), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(fcntl), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(geteuid), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(getuid), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(setuid), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(sendto), 0);   |  | | --- | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |  | |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

In Client.c

// Setup basic whitelist

|  |
| --- |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(rt\_sigreturn), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(exit), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(read), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(write), 0); |
|  |
| // Add System calls to seccomp-bpf |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(socket), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(connect), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(fstat), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(sendto), 0); |
| seccomp\_rule\_add(ctx, SCMP\_ACT\_ALLOW, SCMP\_SYS(exit\_group), 0); |

1. What happens when your application calls the prohibited system call? What is the application behavior that results from the call?

**Answer:** When the application calls a prohibited system call, the program execution ends abruptly with a bad system call message. The application behavior is that it stops and does not process after that.