Studiengang Informatik
apsi 2021

Fachhochschule Nordwestschweiz
Arno Wagner, Michael Schläpfer, Rolf Wagner

Übungsblatt 3 — 4. Oktober 2021

Aufgabe 3-1: Privilege Drop in C

In der Vorlesung haben Sie gesehen, wie ein Privilege Drop mit C durchgeführt werden kann. Schauen Sie sich den Source Code von `priv_drop.c`. Lesen Sie die Kommentare und machen Sie sich vertraut mit Funktionsweise des Programms.

Passen Sie anschliessend den Code an Ihre Umgebung an. Setzen Sie hierfür `TARGET_USER` entsprechend.

```
1 ...
2 // the user to drop privileges to
3 #define TARGET_USER "vagrant"
4 ...
```

Mittels der Hilfsfunktion `print_current_userinfo()` können Informationen zum Benutzerkontext ausgegeben werden. Verwenden Sie diese Hilfsfunktion um folgende Fragen zu beantworten.

Kompilieren Sie das Programm und führen Sie es als `root` aus (`sudo`).

(a) Mittels welcher Schritte wird der Privilege Drop durchgeführt? Weshalb wird er im Beispielcode zuerst in einem separaten Schritt vorbereitet?

(b) In welchem Benutzerkontext läuft das Programm vor dem Privilege Drop und in welchem danach?

(c) An welcher Stelle würden Sie nun `chroot` platzieren?

Aufgabe 3-2: Chroot in C

Wir bauen nun auf der ersten Übung auf und führen einen `chroot` aus, um den weniger privilegierten Benutzer einzuschränken.

Erweitern Sie hierfür den Programmcode um den Parameter `TARGET_DIR`, welcher es erlaubt, den Pfad anzugeben, innerhalb dessen das Programm "gefangen" werden soll. Passen Sie diesen Pfad an Ihre Umgebung an.

```
...
// the user to drop privileges to
#define TARGET_USER "vagrant"
// the directory to chroot to, must exist and must be writable and
    readable
// for the user we drop privileges to
#define TARGET_DIR "/home/vagrant/FHNW/v04/chroot_c_demo/rootdir/"
...
```

Nun verwenden erweitern wir das Programm um die `chroot` Funktionalität. Fügen Sie den nachfolgenden Programmcode hinzu.

```
...
int do_chroot() {
  // Does the chroot operation.
  // For simplicity, we hardcode the targed dir here.
  // The target dir must exist and be readable and writable for
  // the user we do the privilege drop to.
  // Return value is 0 on success, does not return on error
  int res;

  // call chroot()
  res = chroot(TARGET_DIR);
  on_error(res, "chroot() failed");

  // Call chdir() so the current work directory is not outside
  // of the chroot target anymore if it was.
  res = chdir("/");
  on_error(res, "chdir() failed");

  return 0;
}
...
```

Rufen Sie nun in die `do_chroot()` Funktion in Ihrem Programm auf. Wo genau, haben Sie sich ja anlässlich der vorangegangenen Aufgabe überlegt.

Um die korrekte Funktionsweise zu prüfen, erweitern wir den Programmcode um eine weitere Hilfsfunktion `print_rootdir()`. Diese listet den Inhalt des Root-Verzeichnisses auf.

```
...
void print_rootdir() {
  // Pointer for directory entry
  struct dirent *de;

  // opendir() returns a pointer of DIR type. Here we set it to the root
     directory.
  DIR *dr = opendir("/");

  // opendir returns NULL if it couldn't open the directory
  if (dr == NULL) {
    printf("Could not open current directory" );
  }

  // Refer http://pubs.opengroup.org/onlinepubs/7990989775/xsh/readdir.
    html
  // for readdir()
  while ((de = readdir(dr)) != NULL)
    printf("%s\n", de->d_name);

  closedir(dr);
}
...
```

Verwenden Sie diese Hilfsfunktion um die korrekte Funktionsweise Ihrer `chroot` Umsetzung zu überprüfen.

Aufgabe 3-3: Privilege Separated Service

In dieser Aufgabe bauen wir auf den vorangehenden auf und entwickeln einen Service, welcher Kleinbuchstaben entgegen nimmt, diese in Grossbuchstaben umwandelt und anschliessend wieder zurückgibt. Wie in der letzten Vorlesung gezeigt, werden die verschiedenen Aufgaben nun aber als getrennte Programme laufen und entsprechend Ihrer Aufgaben eingeschränkt.

Sie finden in der Aufgabenstellung die beiden Programme `tcp_listener.c` und `worker.c`. Machen Sie sich mit der grundsätzlichen Funktionsweise dieser beiden Programme etwas vertraut.

Kompilieren Sie den Code und führen Sie beide Programme jeweils mit `root` Rechten aus (`sudo ./tcp_listener` und `sudo ./worker`). Nun, da beide Programme laufen, können Sie sich mittels `telnet` verbinden und entsprechende Nachrichten absetzen.

Der Service nimmt dabei 3 Kleinbuchstaben entgegen, übersetzt sie in Grossbuchstaben und gibt sie wieder zurück.

```
vagrant@docker-host:~/FHNW/v04/exercises$ telnet localhost 8500
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
abc
ABC
Connection closed by foreign host.
```

Während die Programme nun bereits funktionsfähig und aufgeteilt in eine Frontend- und eine Backendkomponente sind, sind allerdings die Sicherheitsmassnahmen noch nicht ausreichend umgesetzt. Dies ist nun Ihre Aufgabe. Verwenden Sie das bisher gelernte und setzen Sie folgende Punkte um.

(a) Validieren Sie den Input.

(b) Müssen Sie den Input normalisieren? Wenn ja, setzen Sie dies um.

(c) Schränken Sie die beiden Programme mittels `chroot` ein und setzten Sie die benötigten minimalen Berechtigungen.

(d) Verwenden Sie `setuid` und `setgid` um die Privilegien einzuschränken.

(e) Setzten Sie die Ordnerberechtigungen (*Owner,Group,Other*) so, dass jedes der Programme nur die Berechtigungen hat, die auch wirklich benötigt werden.

(f) (Optional) Fixen Sie alle weiteren Schwachstellen, welche Sie sehen.