

# Grundpraktikum Netz- und Datensicherheit

## *Thema:* **Linux Privilegien**

Lehrstuhl für Netz- und Datensicherheit  
Ruhr-Universität Bochum

Versuchdurchführung: Raum ID 2/168



Betreuung: Florian Feldmann  
Zusammengestellt von: Anton Kettling  
Stand: 19. März 2022  
Version: 1.0

## 1 Einleitung

Bisher haben Sie in diesem Praktikum nur auf Einzelnutzer-Systemen gearbeitet. In diesem Versuch lernen Sie weitere Linux-Tricks und ein System mit mehreren Nutzern kennen. In den anschließenden Fallbeispielen zu cron und shadow werden Sie Fehlkonfigurationen korrigieren müssen, nach denen ein realer Angreifer typischerweise als erstes schaut. Sie erlauben ihm eine sofortige Übernahme des ganzen Systems, deshalb möchten wir Sie für diese sensibilisieren.

Privilegien oder Berechtigungen definieren wie Nutzer mit einem System interagieren dürfen. Eine technische Kontrolle ist dabei immer nötig. Mitarbeiter haben zum Beispiel das Privileg jederzeit in die Uni zu dürfen. Deshalb können Sie bestimmte Türschlösser schließen, zu denen normale Studenten keinen Schlüssel haben.

Nach dem “Need-to-Have”-Prinzip erhält jeder Nutzer nur genau die Berechtigungen die er auch wirklich braucht. Der höhere Verwaltungsaufwand lohnt sich, denn:

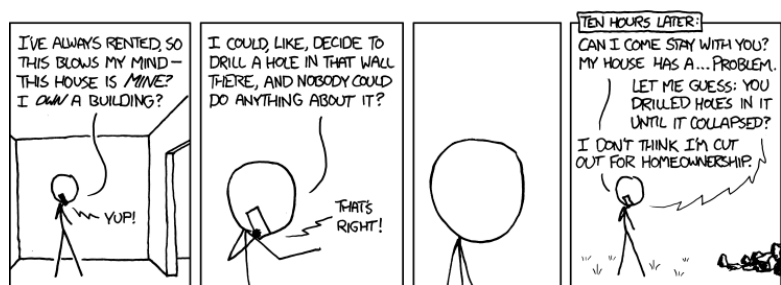
1. Damit schützen wir möglichst große Teile des Systems vor böswilligen Nutzern oder übernommenen Nutzerkonten.
2. Damit schützen wir Nutzer vor sich selbst. Als normaler Student kann ich nicht ausversehen in ein Labor laufen und mich verletzen und als normaler Nutzer kann ich nicht ausversehen wichtige Systemdateien löschen.
3. Damit geben wir dem System Struktur. Wenn ich nur Räume betreten und Dateien sehen kann, die auch Relevanz für mich haben, finde ich relevantes schneller.

Gute System-Administratoren wenden das Prinzip auch auf sich selbst an und arbeiten immer nur mit den Privilegien, die sie für die aktuelle Aufgabe brauchen. Es mag zuerst praktisch erscheinen als Administrator mit den höchsten Privilegien zu arbeiten. Aber der Befehl, alle Dateien unwiderruflich zu löschen, unterscheidet sich in nur einem Zeichen von dem Befehl um irgendeinen unwichtigen Ordner zu löschen.

Vertippen Sie sich dabei als root, dann löscht sich das Betriebssystem ohne Widerrede selbst inklusive aller Daten die auf dem System waren. Deshalb nutzen wir den root-Account nur wenn es wirklich nicht anders geht. Als verantwortungsvolle Administratoren arbeiten wir von einem normalen Nutzer aus und erhöhen nur bei Bedarf temporär unsere Rechte mit den in Sektion 8 dargestellten Techniken.

## 2 Nutzer verwalten

In modernen Betriebssystemen hat jeder Prozess und jede Datei einen Besitzer. Die meisten Prozesse und Dateien eines Systems wurden aber nicht von einer echten Person gestartet bzw. erstellt. Deshalb gibt es auf einem System immer auch System-Nutzer, die keinen echten Personen zuordenbar sind. Dies hat den Vorteil, dass man über Berechtigungen nicht nur die Interaktion von Personen, sondern auch von Prozessen mit Dateien steuern kann.



<https://xkcd.com/905/>

Jeder Nutzer hat eine Hauptgruppe und kann Mitglied mehrerer Nebengruppen sein. Informationen darüber gibt Ihnen der Befehl `id`.

```
nikki@ubuntu:~$ id
uid=1001(nikki) gid=1001(nikki) groups=1001(nikki),1002(it_services)
```

In der Ausgabe sehen Sie zuerst den Nutzernamen, dann den Namen der Hauptgruppe und abschließend eine Liste weiterer Gruppen, in denen der Nutzer Mitglied ist. In den meisten Fällen ist die Hauptgruppe des Nutzers eine gleichnamige Gruppe, in der nur der Nutzer selbst Mitglied ist. Jedem Nutzer bzw. jeder Gruppe ist außerdem eine einzigartige Nutzer-ID (`uid`) bzw. Gruppen-ID (`gid`) zugeordnet, diese steht in der Abbildung vor den zugehörigen Nutzer- und Gruppennamen.

Mit dem Befehl `usermod` können Sie unter anderem die Gruppenzugehörigkeit von Nutzern ändern. Um einen Nutzer Mitglied einer weiteren Nebengruppe zu machen, können Sie nachfolgenden Befehl nutzen.

1	<code>usermod -a -G &lt;groupname&gt; &lt;username&gt;</code>
---	---

Manche Nutzer haben sogenannte Home- oder Benutzer-Verzeichnisse. Auf dieses hat nur der Nutzer selbst Zugriff und kann dort beliebige Dateien ablegen. Außerdem liegen dort den Nutzer betreffende Dateien wie die Befehlshistorie und eine Autostart-Datei.

Davon abweichend liegt das Home-Verzeichniss des administrativen Nutzers “root”, auch “super-user” genannt, unter `/root/`. Der Nutzer root ist auf alles berechtigt und sollte, siehe Need-To-Have-Prinzip, unter keinen Umständen für Tätigkeiten genutzt werden die diese weitgehenden Rechte nicht brauchen.

## 3 Lokale Privilegien unter Linux

Wenn wir in der Informatik von Privilegien sprechen, reden wir meist von Zugriffsrechten. Damit wird festgelegt, ob ein Nutzer eine Datei lesen, in sie schreiben und/oder sie ausführen darf.

Viele Teile von Linux sind technisch als Datei umgesetzt. Auch I/O devices wie Tastaturen und Drucker, weshalb man über die gleichen Dateiberechtigungen auch den Zugriff auf diese Geräte steuern kann. Privilegien sind also ein essenzieller Teil des Linux Betriebssystems.

Privilegien kann man auf zwei Arten umsetzen. In Spezial-Betriebssystemen, die für Anwendungen mit höchstem Schutzbedarf konzipiert sind, bekommen Dateien eine Klassifizierung und Nutzer eine Sicherheitsfreigabe. Als Nutzer kann ich nur auf Dateien zugreifen, die meiner Sicherheitsfreigabe entsprechen, die Individualberechtigung ist hier also beim Nutzer gespeichert.

In Windows und Linux ist hingegen in den Metadaten jeder Datei gespeichert wer der Eigentümer dieser ist und welcher Nutzer wie mit der Datei interagieren kann.

Privilegien auf Windows und Linux unterscheiden sich dennoch erheblich.<sup>1</sup> Wir werden im Nachfolgenden deshalb nur auf Linux eingehen.

---

<sup>1</sup>Nicht Teil des Praktikums, nur für Interessierte:

<https://docs.microsoft.com/de-de/windows/win32/secauthz/mandatory-integrity-control>,  
<https://docs.microsoft.com/de-de/windows/win32/secauthz/access-control-lists>

## 4 Datei-Berechtigungen

In Linux gehört jede Datei einem Nutzer, dieser kann die Zugriffsrechte für die Datei festlegen. Umgehen kann die Zugriffsrechte nur root. Es gibt zwei Möglichkeiten, Zugriffsrechte darzustellen:

Üblich ist die Darstellung als Buchstaben-Tripel. Dabei wird für jedes gesetzte Recht der entsprechende Buchstabe und für jedes nicht-gesetzte Recht ein “-” notiert. Die Reihenfolge ist immer wie in Tabelle 1a dargestellt: Read, Write, Execute (rwx).

Die Darstellung als dezimale Summe ist kürzer und näher an der technischen Umsetzung, aber nicht so einfach zu lesen. Hier summiert man die Zahlen aller gesetzten Rechte auf. Lesen und Ausführen ist also  $4 + 1 = 5$ . Weitere Beispiele finden Sie in Tabelle 1b.

Pro Datei werden diese drei Berechtigungen für je drei Personenkreise vergeben: Für den Eigentümer der Datei (user), für eine assoziierte Nutzergruppe (group) und für alle Nutzer (other).

Jede Datei beinhaltet deshalb den Namen des Eigentümers (user), den Namen der assoziierte Gruppe (group) und dann drei mal drei Buchstaben für Berechtigungen, siehe Abbildung 1c.

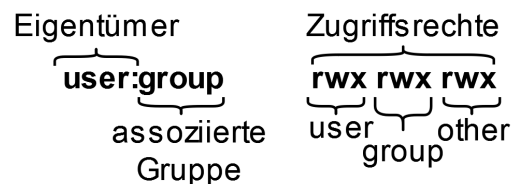
Dazu ein Beispiel.

Zugriffsrecht	Bustabe	Summand
Lesen (Read)	r	4
Schreiben (Write)	w	2
Ausführen (eXecute)	x	1
Keine Rechte	-	0

(a) Darstellung von Rechten

Zugriffsrecht	Bustaben	Zahl
Lesen und Schreiben	rw-	6
Nur Ausführen	--x	1
Alle Rechte	rwx	7
Keine Rechte	---	0

(b) Beispiele für Rechte



(c) Einfache Dateiberechtigung  
Abbildung 1:

### 4.1 Beispiel

Leo legt auf einem Familien-PC Dateien ab. Die Hauptgruppe von Leo ist Leo. Er ist außerdem in der Nutzergruppe Familie.

Als erstes speichert er einen Putzplan. Er gibt sich selber (user), Schreib- und Leseberechtigung (rw). Die anderen Familienmitglieder sind auch Teil der Nutzergruppe Familie (group) und sollen den Putzplan lesen, aber nicht verändern können. Er assoziiert die Gruppe Familie mit der Datei und setzt r als Gruppenberechtigung. Alle anderen (other) geht der Putzplan nichts an, deshalb bekommen diese keine Berechtigung.

⇒ Leo:Familie    rw-r----- oder 640 (dezimale Darstellung)

In den meisten Fällen reicht eine Differenzierung zwischen dem Eigentümer und allen anderen aus. In dem Fall wird als Gruppe die Hauptgruppe des Nutzers gewählt. Die meisten Dateien würden hier also Leo:Leo gehören.

Für private Dateien wählt Leo seine Hauptgruppe Leo als assoziierte Gruppe und gibt sich (user) alle Rechte (rwx). Da in Gruppe Leo nur der Nutzer Leo ist, kann er die Gruppenberechtigungen (group) frei wählen. Hier setzt er Sie wie für alle anderen (other) auf keine Rechte.

⇒ Leo:Leo    rwx----- oder 700 (dezimale Darstellung)

## 4.2 Sinnfreie Berechtigungen

Abschließend noch der Hinweis, dass nicht alle korrekten Berechtigungen auch sinnvoll sind. Man könnte meinen, dass zum Ausführen eines Skripts die Berechtigung `--x` ausreicht. Aber wie wollen Sie Skriptcode ausführen, wenn Sie ihn nicht lesen können? `--x` macht deshalb hier keinen Sinn. Nutzen Sie stattdessen `r-x`.

Gleichermaßen ergibt `-w-` selten Sinn, denn eine Datei in der Sie nur schreiben und nicht lesen können, können Sie auch nicht in einem Texteditor öffnen. Nutzen Sie stattdessen `rw-`.

## 5 Ordner-Berechtigungen

Analog zu Dateien lassen sich selbige Berechtigungen auch auf Verzeichnissen setzen, haben dort aber eine andere Bedeutung. Auf Ordnern gesetzt bedeuten sie, dass der entsprechende Nutzerkreis:

Bustabe	Zugriffsrecht auf Ordner
-	Keine Rechte hat
r	Die im Ordner liegenden Dateien sehen kann (z.B. mit <code>ls</code> )
w	Neuen Dateien im Ordner anlegen kann
x	Ins Verzeichnis wechseln kann (z.B. mit <code>cd</code> )

Tabelle 1: Darstellung von Rechten auf Ordnern

Dass Sie Berechtigungen für einen Ordner vor sich haben erkennen Sie an einem zusätzlichen “d” vor den Berechtigungen: `drwxrwxrwx`. Bei Dateien wird statt dem “d” ein “-” angezeigt. Es gibt noch weitere Buchstaben, die an dieser Stelle eine Spezial-Datei anzeigen. Bei manchen Verknüpfungen steht dort zum Beispiel ein “l”.

### 5.1 Sticky-Bit

Außerdem gibt es für kollaborativ genutzte Ordner eine besondere Berechtigung: Das Sticky-Bit. Wir nehmen einen Ordner mit Berechtigung `777` an. In diesem kann also jeder Dateien erstellen und somit auch die Dateien anderer Nutzer überschreiben. Ist das Sticky-Bit gesetzt, kann zwar immer noch jeder Dateien erstellen, aber nur die selbst erstellten Dateien ändern, löschen und überschreiben. Einzig der Besitzer des Ordners kann auch fremde Dateien löschen. Wir erkennen das Sticky-Bit wie folgt:

<code>drwxrwxrwx</code>	bzw.	<code>drwxrwxrw-</code>	(Ohne Sticky-Bit)
$\Rightarrow$ <code>drwxrwxrw<b>t</b></code>		$\Rightarrow$ <code>drwxrwxrw<b>T</b></code>	(Mit Sticky-Bit)

Ist das Sticky-Bit und das letzte “x” gesetzt, wird statt diesem ein kleines “t” angezeigt. Ist dort kein “x” gesetzt, wird das Sticky-Bit dort mit einem großen “T” dargestellt. In Dezimal-Notation schreiben Sie vor die drei Zahlen eine 1, also zum Beispiel `1777`.

## 6 Privilegien anzeigen

Im letzten Versuch haben Sie bereits den Befehl `ls` genutzt. Mithilfe vom Parameter `-l` lassen sich die Dateiberechtigungen mit angeben. Sollten Sie im Versuch stattdessen eine Dampfloke sehen, haben Sie sich verschrieben.

### 6.1 Alias

Da Sie diesen Befehl häufig benutzen macht es Sinn ein kürzeres *Alias* dafür anzulegen. Das geht mit `alias <neue Schreibweise>='<Befehl>'`. Ein typischer Alias für `ls -l` ist `ll`. Damit Sie diesen Alias nicht jedes mal neu anlegen müssen, schreiben Sie diesen Befehl in `~/.bashrc`, eine Textdatei in ihrem Homeverzeichnis die bei jedem Start einer Konsole ausgeführt wird. Einen persistenten Alias können Sie also zum Beispiel so anlegen:

```
1 echo alias ll=\"ls -l\" >> /home/nikki/.bashrc && source /home/nikki/.bashrc
```

**Achtung:** Mit “>>” hängen Sie den Alias-Befehl unten an die Datei an. Mit “>” löschen Sie die Datei und schreiben nur diesen Alias-Befehl hinein. Übernehmen Sie obigen Befehl deshalb genau.

Ebenfalls bereits kennengelernt haben Sie den `find`-Befehl. Statt nach Dateinamen, können Sie mit dem Parameter `-perm` nach bestimmten Berechtigungen suchen. Hier ist es am einfachsten die Dezimal-Schreibweise zu nutzen. Beispielsweise sucht `find folder -perm -1765` nach Dateien im Ordner `folder` mit der Berechtigung 1765 oder höher. Ohne das “-” vor der Berechtigung sucht es nur nach Dateien mit exakt dieser Berechtigung.

## 7 Privilegien anpassen

Um den Besitzer einer Datei oder eines Ordners zu ändern, verwenden Sie folgenden Befehl.

```
1 chown <Nutzer>[:<Gruppe>] <Datei/Ordner>
```

Der geklammerte Ausdruck ist nur notwendig, wenn Sie als assoziierte Gruppe nicht die Hauptgruppe des Eigentümers setzen wollen. Es ist vorgesehen eine Gruppe zu setzen, in der der Eigentümer selber auch Mitglied ist. Um sicherzustellen, dass Sie sich nicht ausversehen selber die Rechte wegnehmen, benötigt dieser Befehl manchmal administrative Privilegien.

Außerdem können Sie mit folgendem Befehl die Rechte eines Ordners oder einer Datei ändern.

```
1 chmod <Rechte> <Datei/Ordner>
```

Nutzen Sie die Dezimal-Notation, zum Beispiel: `chmod 777 folder`. Sie können damit auch das Sticky-Bit setzen: `chmod 1777 folder`.

Wollen Sie die bestehende Berechtigung nur modifizieren, können Sie auch additiv oder subtraktiv arbeiten. Schreiben Sie ein “+” oder “-” und das Bit, das Sie setzen wollen.

Zum Beispiel setzt `chmod +t folder` das Sticky-Bit und `chmod +r file` die Leseberechtigung für alle drei Nutzerkreise.

Wollen Sie die Berechtigung nur bei einem Nutzerkreis anpassen, schreiben Sie vor das +/- ein “u”(user), “g”(group) oder “o”(other). Zum Beispiel ergänzt `chmod g+w folder` die Leseberechtigung für Mitglieder der mit der Datei assoziierten Nutzergruppe (group) und `chmod u+r folder` die Leseberechtigung für den Eigentümer (user). Falls Ihnen diese Nutzerkreise nicht bekannt vorkommen, schauen Sie sich noch einmal Abbildung 1c an.

Wenn Sie Berechtigungen für viele Dateien ändern wollen kann es Sinn machen, `chmod --recursive <Rechte> <Ordner>` auf den Ordner anzuwenden, in dem alle Dateien liegen.

## 8 Privilegien anheben

Alle von Ihnen ausgeführten Programme arbeiten normalerweise “im Kontext” ihres Nutzers, verfügen also ausschließlich über Ihre Rechte. Sie haben auch gelernt, dass Sie so selten wie möglich mit dem administrativen Nutzer `root` arbeiten sollen. Wenn Sie ein Programm nun trotzdem mit administrativen Rechten starten müssen, können Sie eine der folgenden Techniken verwenden.

### 8.1 sudo

Der Befehl `sudo` wird einem Befehl vorangestellt, um diesen im Kontext und somit mit den Rechten eines anderen Nutzers auszuführen. Ohne weitere Parameter führt man

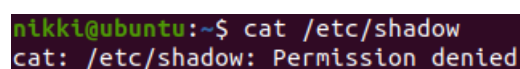
A terminal window with a dark background. The prompt is 'nikki@ubuntu:~\$'. The user enters 'cat /etc/shadow'. The output is 'cat: /etc/shadow: Permission denied'.

Abbildung 2:

damit Befehle im Kontext von `root` aus. Wenn Sie zum Beispiel als normaler Nutzer den Befehl aus Abbildung 2 ausführen bekommen Sie eine Fehlermeldung, denn die Datei ist nur durch `root` lesbar; Warum das so ist erfahren Sie später. Mit `sudo cat /etc/shadow` können Sie die Datei trotzdem lesen.

Wenn Sie `sudo` in einer Konsole das erste mal ausführen fragt es Sie nach Ihrem Passwort. Außerdem müssen Sie auf den `sudo`-Befehl berechtigt sein. Welcher Nutzer `sudo` nutzen darf, ist in der sogenannten `sudoers`-file festgelegt. In dieser steht normalerweise, dass jeder Nutzer der Nutzergruppe “`sudo`” den Befehl `sudo` nutzen darf. Um `sudo` nutzen zu können, müssen Sie Ihren Nutzer also zuerst der Nutzergruppe `sudo` hinzufügen.

`sudo` gilt nur für den ersten Befehl. Außerdem funktioniert es nicht nativ für Pipes.

Mit `sudo echo "X">>> file` können Sie z.B. nicht in eine geschützte Datei schreiben, denn das `sudo` führt nur `echo` mit erhöhten Rechten aus, nicht die Pipe. Wenn Sie mit `sudo` einen Schreibschutz umgehen wollen, nutzen Sie deshalb z.B. `sudo nano file`

### 8.2 su -

Wenn Ihr Nutzer nicht auf `sudo` berechtigt ist oder Sie viele Befehle im Kontext des Administrators ausführen müssen, können Sie auch eine Konsole im Kontext von `root` öffnen. Der Befehl dafür lautet “`su -`” und hier wird nicht nach Ihrem, sondern nach dem Passwort des Zielnutzers, hier `root`, gefragt!

Mit `su` erstellen Sie eine komplett neue Konsole in der bestehenden Konsole. Es öffnet sich also kein neues Fenster. Sie werden feststellen, dass Ihre außerhalb definierten Aliase in dieser Umgebung nicht mehr funktionieren und Ihre Eingaben nun auch in einer anderen Datei, der Historie von `root`, protokolliert werden. Deshalb können Sie den außerhalb zuletzt aufgerufenen Befehl nun auch nicht mehr mit den Pfeiltasten erneut aufrufen. Diese neue Umgebung verlassen Sie mit dem Befehl `exit`. Nun befinden Sie sich wieder in Ihrer ursprünglichen Umgebung. Führen Sie `exit` jetzt ein zweites mal aus, schließen Sie deshalb damit das Konsolenfenster.

Wenn Sie die neue Konsole nicht im Kontext von `root` sondern eines anderen Nutzers öffnen wollen, können Sie dessen Nutzernamen mit `su -l <username>` übergeben.

Merken Sie sich: `sudo` verlangt Ihr Passwort, `su` das des Zielnutzers, bei `su -` also das Passwort von `root`.

### 8.3 SUID- und GUID-Bit

Nun gibt es auch Programme, die jeder Nutzer im Kontext des Eigentümers des Programms ausführen können muss. Ein Beispiel für ein solches Programm haben wir bereits besprochen: `sudo`. Wie soll das `sudo`-Programm einen Befehl als Administrator ausführen, wenn Sie ihm nicht das Administrator-Passwort übergeben? Auf der Programmdatei `/usr/bin/sudo` ist das SUID-Bit gesetzt und sie gehört `root`. Dadurch wird `sudo` immer im Kontext von `root` gestartet. Die Entwickler von `sudo` haben entschieden, dass Sie bei der Ausführung des Programms Ihr eigenes Passwort angeben und in der `sudoers-file` stehen müssen. Das ist aber technisch nicht notwendig. Hätten die Entwickler diese Sicherheitsmaßnahmen nicht eingebaut, könnte jeder ohne Angabe von irgendwas Programme mit administrativen Berechtigungen ausführen.

Vielleicht erkennen Sie hier schon ein Problem. Wenn die Entwickler eines solchen Programms einen Fehler machen und jeder das Programm ohne derartige Prüfungen ausführen kann, dann kann jeder administrative Rechte erlangen. So gerade mit einem `sudo`-Derivat namens `pkexec` passiert<sup>2</sup>. Das ist auf Linux standardmäßig installiert, hat das SUID-Bit gesetzt und war fehlerhaft implementiert. Dadurch ist zum Zeitpunkt der Erstellung dieses Dokuments quasi jedes Linux-System angreifbar. Nehmen Sie deshalb bitte mit: Das SUID-Bit zu verwenden bedarf großer Vorsicht.

Das SUID-Bit ist eine Spezialberechtigung wie das Sticky-Bit. Statt an der Stelle des letzten wird es an der Stelle des ersten “x” dargestellt. Ist dort ein “x” gesetzt, wird das SUID-Bit als kleines “s”, sonst als großes “S” dargestellt:

<code>drwxrwxrwx</code>	bzw.	<code>drw-rwxrwx</code>	(Ohne SUID-Bit)
$\Rightarrow$ <code>drw<b>s</b>rwxrwx</code>		$\Rightarrow$ <code>drw<b>S</b>rwxrwx</code>	(Mit SUID-Bit)

In Dezimal-Notation schreiben Sie vor die drei Zahlen eine 4, also zum Beispiel 4777. Ein großes “S” ist zwar korrekt, aber sinnfrei. Denken Sie an Sektion 4.2. Denn wenn der Eigentümer der Datei diese nicht ausführen kann, macht es auch keinen Sinn diese im Kontext des Eigentümers auszuführen.

Auch an der Stelle des mittleren x kann eine Spezialberechtigung angezeigt werden: Das GUID-Bit. Ist es gesetzt, wird die Datei immer im Kontext, also auch mit den Rechten, der assoziierten Gruppe ausgeführt. Auch hier macht es keinen Sinn, das GUID-Bit ohne Ausführberechtigung der Gruppe zu setzen. Statt einem großem “S” wird diese Fehlkonfiguration allerdings durch ein kleines “l” an der Stelle des mittleren “x” angezeigt.

<code>drwxrwxrwx</code>	bzw.	<code>drwxrw-rwx</code>	(Ohne GUID-Bit)
$\Rightarrow$ <code>drwxrw<b>s</b>rwx</code>		$\Rightarrow$ <code>drwxrw<b>l</b>rwx</code>	(Mit GUID-Bit)

In Dezimal-Notation schreiben Sie vor die drei Zahlen eine 2, also zum Beispiel 2777.

Nun können Sie das SUID- bzw. GUID-Bit nicht nur als Berechtigung auf eine Datei, sondern auch auf einen Ordner setzen. In dem Fall wird der Eigentümer bzw. die assoziierte Gruppe aller neu in diesem Ordner erzeugten Dateien automatisch auf den Eigentümer bzw. die assoziierte Gruppe des Ordners gesetzt. Das hat Vorteile für kollaborativ genutzte Ordner. Wenn Sie die assoziierte Gruppe eines solchen Ordners auf eine gemeinsame Nutzergruppe und auch das GUID-Bit setzen, stellen Sie sicher, dass egal wer die Datei in diesem Ordner erstellt, immer alle Mitglieder dieser Gruppe Zugriff auf alle Dateien des Ordners haben.

---

<sup>2</sup>Nicht Teil des Versuchs: Siehe CVE-2021-4034, z.B. auf <https://cve.mitre.org/>



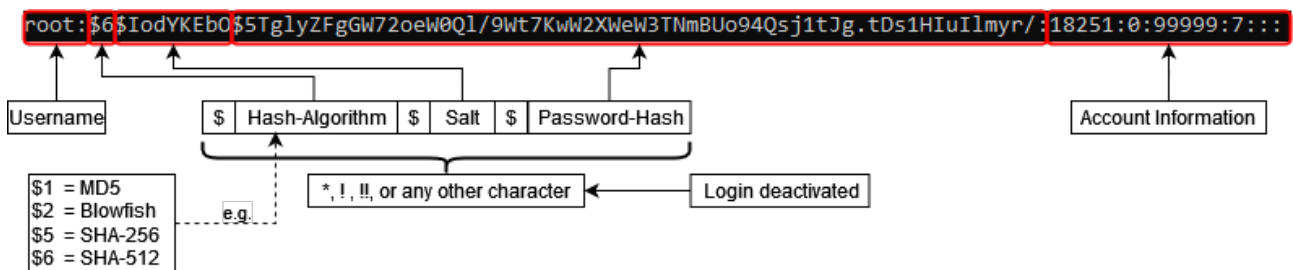
## 9 Weitere Techniken

Damit die Fallbeispiele für Sie nachher Sinn machen, führen wir in die darunter liegende Technik hier kurz ein.

### 9.1 Shadow

In der Datei `/etc/shadow` werden die Passwörter aller Nutzer des Systems gespeichert. Sie liegen dort zwar gehasht, also eigentlich unbrauchbar vor, aber gerade unsichere Passwörter können aus diesen Hashes wieder gewonnen werden. Auch auf Grund des Need-To-Have-Prinzips sollten normale Nutzer deshalb weder auf ihr lesen noch schreiben können (`-rw-r-----`).

Im Verlaufe des Versuchs werden Sie trotzdem in die Datei reinschauen. Damit Sie dann verstehen, was Sie dort alles sehen können, nachfolgend einer Übersicht über den Syntax. Die Abbildung 3 enthält eine Zeile dieser Datei.

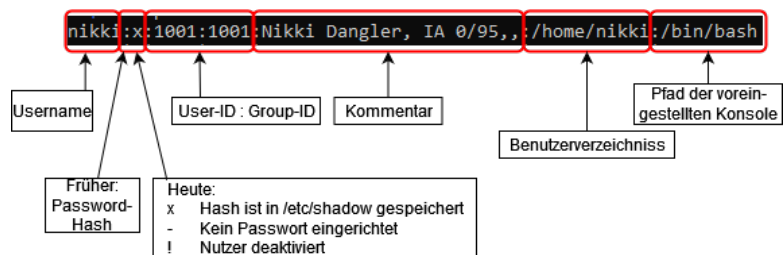


Sie sehen mehrere Felder, die durch einen Doppelpunkt voneinander getrennt sind. Nach dem Nutzernamen folgt der verwendete Hashalgorithmus. Danach folgt ein Salt, der die Verwendung des darauf folgenden Hashes noch sicherer macht. Das letzte Feld enthält weitere Account-Informationen.

### 9.2 Passwd

Einem Nutzerkonto haften mehr Information als nur das Passwort an. Da diese aber durchaus von normalen Nutzern einsehbar sein sollen, werden sie in einer Textdatei mit laxeren Leseberechtigungen gespeichert: `/etc/passwd` (`root:root -rw-r--r--`).

Die Felder sind durch Doppelpunkte getrennt. In Abbildung 4 sehen Sie einen Eintrag aus `/etc/passwd`.



**Um Verwirrungen vorzubeugen:** Neben dieser Textdatei gibt es den gleichnamigen Befehl `passwd`. Das zugehörige Programm liegt unter `/usr/bin/` und kann dafür genutzt werden das eigene Passwort zu ändern. Dafür modifiziert dieses Programm, `/usr/bin/passwd`, die Text-Dateien `/etc/shadow` und eventuell `/etc/passwd`.

```
nikki@ubuntu:~$ passwd
Changing password for nikki.
Current password:
New password:
Retype new password:
passwd: password updated successfully
```

Abbildung 5: Example usage of `/usr/bin/passwd`



## 10 Versuch

In nachfolgendem Szenario sind Sie Nikki Dangler, eine überaus fähige Systemadministratorin. Im Rahmen der “Die RUB ist vorbereitet”-Strategie wurden unter anderem Sie damit beauftragt, die Computersysteme der RUB zu hardenen, also sicherer zu machen. Heute betreuen Sie ein Linux Server-System mit personengebundenen Nutzerkonten für alice, bob und jonas.

**Hinweis:** In allen Aufgaben, in denen Sie `chmod` oder `chown` nutzen, müssen Sie das Ergebnis mit `ls -l` oder ähnlichem kontrollieren und das per Screenshot auch im Bericht dokumentieren.

### Wichtiger Hinweis zu Passwörtern:

Die Nutzernamen und Passwörter weichen in diesem Versuch von den sonst verwendeten ab. Sie werden in diesem Versuch häufig zwischen Benutzern wechseln und mit dieser klaren Zuordnung soll Verwirrung vermieden werden.

- Sie sind Nikki Dangler. Ihr Nutzernamen lautet **nikki**, Ihr Passwort lautet **12345**.
- Der administrative Account hat den Nutzernamen **root** und das Passwort **admin123**. Nutzen Sie ihn nur auf Anweisung.
- Weiterhin existieren die Nutzernamen alice, bob und jonas. Zu diesen kennen Sie das Passwort nicht.

### 10.1 Erkundung der Umgebung

Da Sie sich auf ein Ihnen fremdes System verbinden ist es ratsam, sich erst einmal mit der Umgebung vertraut zu machen. Nutzen Sie für die nachfolgenden Aufgaben ausschließlich die Konsole; Diese können Sie zum Beispiel über *Strg* + *Alt* + *T* starten.

1. Kontrollieren Sie zuerst ob die Informationen, die ich Ihnen gegeben habe, korrekt sind. Mit `uname -sn` können Sie sich die Art des Betriebssystems und den Namen des Computers ausgeben lassen. Mit `grep nikki /etc/passwd` können Sie sich die auf den Nutzer nikki beziehende Zeile aus `/etc/passwd` ausgeben lassen. Existieren die Nutzer alice, bob und jonas? Schauen Sie in `/etc/passwd` nach. Welche Konsolen haben diese Nutzer als Standard-Konsole hinterlegt?
2. Geben Sie den absoluten Pfad zum Home-Verzeichniss von nikki an. Welche Rechte sind auf diesem gesetzt? Nutzen Sie `ls -l /home/`
3. Legen Sie sich “ll” als persistenten Alias für `ls -l` an und nutzen Sie diesen für alle weiteren Aufgaben. Sie finden den Befehl dazu in Sektion 6. Erklären Sie den Befehl kurz. `source` sorgt dafür, dass die Änderungen schon in der aktuell geöffneten Konsole wirksam sind.
4. Geben Sie den absoluten Pfad zum Home-Verzeichniss von root an. Können Sie in dieses hineinschauen? Warum (nicht)? Nutzen Sie `ll /home/`
5. Wie lautet der Nutzer, unter dem Sie aktuell angemeldet sind? Welchen Gruppen gehört Ihr Nutzer an? Nutzen Sie den Befehl `id` und erklären Sie die Ausgabe.

## 10.2 su und sudo

In den nachfolgenden Aufgaben werden Sie immer wieder Programme als root ausführen müssen. Deshalb sollten Sie ihren Nutzer auf das Programm sudo berechtigen:

1. Führen Sie `touch /home/bob/test` aus. Wie lautet die Fehlermeldung?  
**Hinweis:** `touch` “berührt” Dateien, um den Zeitpunkt der letzten Änderung auf die aktuelle Zeit zu setzen. In der Praxis wird es häufig genutzt um leere Dateien anzulegen.
2. Versuchen Sie den Befehl erneut mit `sudo`. Geben Sie Ihr Passwort ein. Wie lautet die Fehlermeldung?
3. Starten Sie nun eine Konsole im Kontext des Nutzers root. Nutzen Sie “`su -`”. Welches Passwort geben Sie ein?
4. Welchen Nutzernamen gibt Ihnen `id` nun aus? Stimmt dieser mit dem Nutzer, der Ihnen am Anfang jeder Eingabezeile angezeigt wird, überein? In welchem Verzeichnis befinden Sie sich?
5. Fügen Sie den Nutzer nikki zur Nutzergruppe “sudo” hinzu. Den Befehl finden Sie ins Listing 2
6. Lassen Sie sich mit `history | tail -n 20` die 20 zuletzt ausgeführten Befehle anzeigen. Finden Sie dort auch die in 1. und 2. ausgeführten touch-Befehle? Warum (nicht)?
7. Kehren Sie mit `exit` zu ihrer unprivilegierten Konsole zurück.  
**Hinweis:** Wenn Sie `exit` nun ein zweites mal aufrufen, beenden Sie damit ihre eigentliche Shell und schließen das Fenster in dem diese läuft!
8. Normalerweise müssen Sie nach einer Änderung der Gruppenzugehörigkeit das System neustarten. In diesem Versuch behelfen wir uns mit dem folgenden Befehl

```
exec su -l nikki
```

Der Befehl fragt nach Ihrem Passwort. Führen Sie den Befehl in allen Konsolen aus, die Sie nutzen. Wenn Sie im Verlauf dieses Versuchs eine neue Konsole starten, müssen Sie den Befehl auch dort eingeben!

Prüfen Sie, z.B. mit `id`, ob Sie nun in der neuen Gruppe sind und geben Sie die Group-ID für Nutzergruppe sudo an.

9. Führen Sie die Befehle `touch /home/bob/test` und `sudo touch /home/bob/test` erneut aus. Welcher funktioniert? Warum?
10. Führen Sie `history | tail -n 20` erneut aus. Finden Sie dort den `usermod`-Befehl? Warum (nicht)? Sehen Sie dort die `sudo`-Befehle?
11. Sie wollen testen ob bob auf sein eigenes Home-Verzeichnis zugreifen kann. Welches Passwort würde `su -l bob` als Eingabe verlangen? Kennen Sie dies? Wenn nicht, führen Sie den Befehl mit einem `sudo` davor aus. Welches Passwort wird jetzt verlangt. Sollten Sie nicht nach einem Passwort gefragt werden: Welches würde verlangt werden, wäre es nicht schon in Ihrer Konsole gecached? Warum sieht die für bob geöffnete Konsole anders aus, als die von Ihnen verwendete `/bin/bash`?  
**Hinweis:** Die Antwort auf die letzte Frage finden Sie auch in `/etc/passwd`
12. Kontrollieren Sie mit `id`, dass Sie als “bob” angemeldet sind. Können Sie mit `touch` jetzt Dateien in `/home/bob/` und in `/home/jonas` erstellen? Warum(nicht)? Verlassen Sie diesen Kontext mit `exit`. Kontrollieren Sie, dass Sie nun wieder als “nikki” angemeldet sind.

### 10.3 Setzen einfacher Berechtigungen

Schon an ihrem ersten Arbeitstag finden Sie in Ihren E-Mails drei Beschwerden:

1. alice beschwert sich, dass bob Dateien in ihrem Homefolder angelegt hat. Ihnen fällt auf, dass Alices Home-Verzeichniss die Berechtigung `drwxr-xrwx` hat. Korrigieren Sie dies mit `chmod` zu `drwxr-xr-x` und geben Sie diese neue Berechtigung in Dezimal-Notation an.  
**Hinweis:** Da Ihnen die Datei nicht gehört müssen Sie hier `sudo` verwenden. **Hinweis:** Vergessen Sie nicht, dass Sie das Ergebnis von `chmod` und `chown` mit `ll` überprüfen und einen Screenshot davon in den Bericht tun müssen.
2. jonas beschwert sich, dass er in das Verzeichnis unter `/root/` nicht reinschauen kann. Er behauptet, dass das System “wahrscheinlich wie immer kaputt” sei. Erklären Sie an Hand der Zugriffsberechtigung des Ordners `drwx-----` warum das System ihm den Zugriff verweigert.
3. bob weist Sie darauf hin, dass das System täglich um 23 Uhr eine Fehlermeldung zeigt:

```
/usr/local/bin/backItUp.sh: Permission denied on execution
```

Sie verstehen zwar noch nicht warum diese Fehlermeldung jeden Tag erscheint, aber Sie stellen fest, dass das execute-bit auf `backItUp.sh` nicht gesetzt ist. Korrigieren Sie dies. Muss auch das read-bit gesetzt sein, damit das Skript ausgeführt werden kann?

4. Auf Ihren Wunsch hat ein befreundeter Administrator außerdem einer Liste der meist genutzten Passwörter unter `/home/nikki/bad_passwords_2017/` kopiert. Die gewählte Form ist nicht ideal und Sie können leider auch keine dieser Dateien lesend öffnen.  
Setzen Sie auf allen Dateien und Ordnern in diesem Verzeichnis `u+rx`. Nutzen Sie dafür die `--recursive` oder `-R` flag. Welchen Hinweis hat der Kollege Ihnen in allen Dateien hinterlassen?

## 10.4 Gruppenprivilegien

alice und bob sind Teil eines Forschungsprojekts das klären soll ob sich LED-Weihnachtsbeleuchtung im Januar auf den Klimawandel auswirkt. Aus Angst vor Repressalien durch die LED-Industrie wollen sie ihre Ergebnisse vorerst geheimhalten, auch vor jonas.

1. Fügen Sie alice und bob der Nutzergruppe ledx hinzu.
2. Ändern Sie den Besitzer des Projekt-Ordners /srv/ledx-projekt/ auf nikki:ledx. Nutzen Sie dafür chown
3. Erstellen Sie jetzt mit nachfolgendem Befehl eine Test-Datei. Welchen Besitzer hat die Datei? Welche Berechtigungen sind auf der Datei gesetzt?

```
echo "ja, kann er" > /srv/ledx-projekt/testrechnung.txt
```

4. Löschen Sie die erzeugte Datei mit `rm /srv/ledx-projekt/testrechnung.txt`. Setzen Sie nun das GUID-Bit auf den Ordner. Prüfen Sie mit `ll`, ob das Bit gesetzt wurde. Führen Sie dann den Befehl aus Unteraufgabe 3 erneut aus. Worin unterscheiden sich Besitzer und Berechtigungen der Datei nun zu Unteraufgabe 3? Warum?
5. Setzen Sie die Berechtigungen auf den Ordner so, dass weder Sie noch jonas, aber Mitglieder der Gruppe ledx auf den Ordner voll zugreifen können. Sie können Besitzer bleiben.  
**Hinweis:** Arbeiten Sie additiv und subtraktiv, z.B. mit `g-rwx`, damit Sie das GUID-Bit nicht wieder entfernen.
6. Überprüfen Sie Ihre Arbeit indem Sie als Nikki, alice, bob und jonas mit `ls -l` auf das Verzeichnis zuzugreifen versuchen. Tipp: `su -l`
7. jonas hat zwar keine Berechtigung auf das Verzeichnis, aber auf die Datei. Kann er die Rechnungen mit `cat /srv/ledx-projekt/testrechnung.txt` lesen? Warum (nicht)?

## 10.5 Fallbeispiel cron

Wie in Section 9.3 beschrieben können falsche Dateiberechtigungen in Verbindung mit cronjob einem Angreifer Vorschub leisten.

1. Schreiben Sie in `/etc/crontab` folgendes rein: `#Hier koennte Schadcode stehen` . Warum können Sie dort rein schreiben, obwohl nur der Eigentümer Schreibberechtigung hat?
2. Korrigieren Sie dies mit `chown`. Können Sie noch immer reinschreiben?  
**Hinweis:** Die korrekten Berechtigungen/Eigentümer finden Sie in Sektion 9.3.
3. Schreiben Sie mit `sudo nano /etc/crontab` trotzdem `#Hier koennte weiterer Schadcode stehen` rein. Sollten normale Nutzer die Berechtigung auf den Befehl `sudo` haben?
4. In welchem Intervall und zu welcher Zeit wird das in `/etc/crontab` angegebene Skript ausgeführt? Was könnte der Zweck des Skripts sein? Mit diesem Wissen sollten Sie das Verhalten aus Aufgabe 10.3.3 erklären können.
5. Angenommen Sie können in den Konfigurationsdateien von cronjob, z.B. `/etc/crontab`, nichts verändern. Wie können Sie trotzdem täglich um 23 Uhr Schadcode von `crontab` ausführen lassen? Welche Datei müssen Sie dafür bearbeiten? Haben Sie Schreibrechte auf diese Datei? Wenn ja, entfernen Sie diese Berechtigung.

## 10.6 Fallbeispiel shadow

Die Textdatei `/etc/shadow` ist ebenfalls sehr schützenswert. Deshalb können auch hier laxer Zugriffsberechtigungen ein reales Sicherheitsrisiko sein.

1. Lassen Sie sich den Inhalt der Datei mit `cat` ausgeben. Warum können Sie in der Datei lesen? Sie sind doch weder `root` noch Teil der Gruppe `shadow`.
2. Wie lautet das gehashte Passwort Ihres Nutzers und wie der verwendete Salt?
3. `o+w` ist ebenfalls gesetzt. Warum ist das ein Problem? Wie könnte ein Angreifer das ausnutzen?
4. Korrigieren Sie diese fehlerhafte Konfiguration zu `-rw-r-----`. Können Sie immer noch die Passwort-Hashes lesen?

## 10.7 Fallbeispiel passwd

Sie suchen nun ein Programm mit dem jeder Nutzer sein eigenes Passwort ändern kann. Aus der vorherigen Aufgabe wissen Sie, dass ein solches Programm aber in `/etc/shadow` schreiben können muss und das kann nur root. Damit normale Nutzer das Programm “im Kontext” des root-Benutzers aufrufen können, muss deshalb für das Programm das SUID-Bit gesetzt werden.

1. Versuchen Sie Ihr Passwort mit `passwd` von 12345 zu 23456 zu ändern. Wie lautet die Fehlermeldung? Nutzen Sie nicht `sudo`!  
**Hinweis:** Es ist normal, dass Sie Ihre Tastatureingaben hinter “Current”- und “New Password” nicht sehen. Tippen Sie einfach und bestätigen Sie zeilenweise mit Enter. Das Programm empfängt Ihre Tastatureingaben, stellt sie nur nicht dar. Einen erfolgreichen Passwortwechsel sehen Sie in Abbildung 5.
2. Setzen Sie das SUID-Bit für das Programm `/usr/bin/passwd`.
3. Versuchen Sie das Passwort erneut zu ändern. Wie lautet nun die Meldung?
4. Bei der Verwendung des SUID-Bits ist Vorsicht geboten. Angenommen das SUID-Bit ist für das Programm zum Verschieben von Dateien, `mv`, gesetzt. Überlegen Sie wie ein Angreifer dies nutzen könnte um Passwörter neu zu setzen. Auch das ist eine Fehlkonfiguration nach dem ein typischer Angreifer suchen würde.
5. Deshalb hinterlegt `ll` den Namen von Dateien mit gesetztem SUID-Bit in einer prägnanten Farbe. Welcher?
6. Hat das Programm `/usr/bin/mv` das SUID-Bit gesetzt? Wenn ja, entfernen sie es. Welche Farbe hat der Dateiname nun mit `ll`?
7. Nach so vielen Fehlkonfigurationen sollten Sie besser nachschauen, ob auf dem System eine Datei existiert, die zwar das SUID-Bit, aber nicht `u+x` gesetzt hat. Durchsuchen Sie das System mit `find` nach einer solchen Berechtigung. Prüfen Sie das Ergebnis mit `ll`.
8. Korrigieren Sie diesen Fehler, in dem Sie zusätzlich `u+x` setzen.

## 10.8 Fallbeispiel tmp

Im Sammelverzeichnis `/tmp/` legen Programme und Nutzer Dateien an, die nur bis maximal zum nächsten Neustart benötigt werden.

1. Wie wird erreicht, dass jeder Nutzer und jedes Programm in `/tmp/` Dateien erstellen kann?
2. Nun kann es passieren, dass verschiedene Programme eine Datei mit gleichem Namen in `/tmp/` anlegen wollen. Wie kann verhindert werden, dass dabei Dateien überschrieben werden?
3. Setzen Sie das dafür benötigte Bit auf den Ordner `/tmp/`.
4. Führen Sie `free -mt | grep -i total` aus. Wie viel RAM ist gerade frei (“free”)?
5. Legen Sie mit `dd if=/dev/zero of=/tmp/128MB.txt bs=128M count=1` eine Textdatei der Größe 128MB an.
6. Ist jetzt weniger RAM frei? Wie viel? Stellen Sie eine Vermutung auf, wie das System den Ordner `/tmp/` anscheinend benutzt.



## 11 Kontrollfragen

1. Wie lautet die Berechtigung `rw---x---` in dezimaler Schreibweise?
2. Welche Interaktion mit einem Ordner erlaubt das “w”-Bit gesetzt?
3. Welche Berechtigungen braucht man auf eine Datei um diese ausführen zu können?
4. Was bedeutet das “d” am Anfang des Berechtigungsstrings? Können dort auch andere Zeichen stehen?
5. Welche Bits müssen Sie in `-----` setzen, damit alle die Datei lesen, aber nur der Eigentümer diese bearbeiten kann?
6. Was ist der Unterschied zwischen `chown` und `chmod`?
7. Mit welchem Nutzerkonto arbeitet ein guter Systemadministrator\*in normalerweise nicht? Nennen Sie einen Grund dafür.
8. Was bewirkt das GUID-Bit auf einem Ordner und was auf einer ausführbaren Datei?
9. Was ist der Unterschied zwischen `/etc/passwd`, `/usr/bin/passwd` und `/etc/shadow`?
10. Welches Passwort werden Sie in diesem Versuch für den administrativen Nutzer und welches für Ihr Nutzerkonto nutzen?