SUID-, SGID- und Sticky-Bit

Autor: Erik Zettel
Website: www.zettel-it.de

***SUID*:**

SUID (**S**et-**U**ser-**ID**-Bit) ist ein Zugriffsbit für Dateien oder Verzeichnisse in unixähnlichen Betriebssystemen und Unix-Derivaten. Grundsätzlich muss die Funktion dieses Bits auf Verzeichnisse und Dateien unterschieden werden.

Ist das Bit auf ein Verzeichnis gesetzt, so gehören alle Dateien, die in diesem Verzeichnis angelegt werden nicht dem Benutzer, der sie anlegt, sondern dem Besitzer des Verzeichnisses. Will man ein solches Bit auf ein Verzeichnis setzen, bietet sich die Ausführung von *chmod* mit Parametern in symbolischer Schreibweise an (*chmod u+s <verzeichnis>*). In der Praxis hat dies ein geringere Bedeutung im Vergleich zum *SGID*. So z.B. ist ein Verzeichnis mit den Zugriffsrechten *drwsrwxrwx* ein Verzeichnis, in das alle mit *cd* wechseln, den Inhalt mit *ls* anzeigen und selbst Dateien und Verzeichnisse anlegen können. Hierbei wird jede neu angelegte Datei unter der Benutzer-Kennung des Besitzers des Verzeichnisses gespeichert.

Viel wichtiger ist das gesetzte *SUID*-Bit, wenn es auf Dateien und insbesondere Programmen gesetzt ist (*chmod u+s <datei>*). In diesem Fall können Programme, die z.B. dem Systemadministrator *root* gehören auch von anderen Benutzern des Systems unter seiner Kennung ausgeführt werden. Dies ist beispielsweise beim Programm *passwd* der Fall. Sieht man sich mit *ls -l /bin/passwd* die Zugriffsrechte der Benutzer auf diesen Befehl an (-rws--x--x), so erkennt man dies. So ist es auch unprivilegierten Benutzern durch den Aufruf des Befehls *passwd* gestattet ihr Passwort in */etc/shadow* zu ändern, obwohl sie auf die Datei kein Schreib- oder Leserecht besitzen. Ähnlich verhält es sich mit weiteren Programmen im /bin-Verzeichnis. *chfn*, *su*, *sudo*, *mount* und *umount* nutzen ebenfalls das gesetzte *SUID*-Bit. Dies ist jedoch nur möglich, wenn das entsprechende *x*-Bit für die Gruppe oder Andere gesetzt ist (z.B. *-rws--x--x*).

Vorteil hierbei ist die Einfachheit, mit der privilegierte Prozesse auch von unprivilegierten Benutzern ausgeführt werden können. Ein klarer Nachteil von SUID-Programmen ist jedoch, dass das sie ein erhebliches Sicherheitsrisiko darstellen, falls diese Programme fehlerhaft sind. Lokale Angreifer könnten sie dazu benutzen, um das ganze System zu kompromittieren.

***SGID*:**

SGID (**S**et-**G**roup-**ID**-Bit) ist wie das SUID-Bit ein Zugriffsbit auf Dateien und Verzeichnisse in unixähnlichen Betriebssystemen und Unix-Derivaten. Wie beim SUID-Bit muss man die Funktion dieses Bits auf Dateien und Verzeichnisse unterscheiden.

Ist das Bit auf Verzeichnisse gesetzt, so gehören alle Dateien und Unterverzeichnisse des Verzeichnisse nicht der primären Gruppe des Benutzers an, der diese Dateien und Unterverzeichnisse anlegt, sondern der primären Gruppe des Besitzers des Verzeichnisse, auf dem das *SGID*-Bit gesetzt ist. Will man ein solches Bit auf ein Verzeichnis setzen, bietet sich wiederum der Befehl *chmod* an, der mit Parametern in symbolischer Schreibweise aufgerufen wird (*chmod g+s <verzeichnis>*). So ist es nun beispielsweise möglich, dass ein Projekt innerhalb ei-

ner Benutzergruppe bearbeitet werden kann. Die Benutzer der Gruppe, die der primären Gruppe der Gruppe des Verzeichnisses angehören, können gemeinsam an einem Projekt arbeiten, ohne dass fehlende Schreib- und Leserechte diesem im Wege stehen. Dazu müssen natürlich entsprechende Rechte für die Gruppe gesetzt sein. Mit dem Befehl *chgr* ist es außerdem möglich, dass Benutzer ihre primäre Gruppe ändern, um beispielsweise an einem Projekt mitzuarbeiten. Die Zugriffsrechte für ein solches Verzeichnis könnte z.B. durch *chmod 2770* (*drwxrws*---) gesetzt werden.

Ein auf Programme oder Dateien gesetztes *SGID*-Bit bewirkt, dass Programme mit den Rechten der Gruppe auch von nicht-privilegierten Benutzern ausgeführt werden können, wenn das *s[x]*-Bit für die Gruppe setzt ist, die die primäre Gruppe der jeweiligen Benutzer ist (z.B. *-rwx-rws*---). Auch Nicht-Gruppenmitglieder können so durch ein entsprechend gesetztes *x*-Bit für „Others“ (z.B. -rwxrws--x) dementsprechende Privilegien genießen. Hierbei entsteht jedoch in manchen Fällen ein gewisses Sicherheitsrisiko (ähnlich *SUID*), wenn ein Programm, auf welchem das *SGID*-Bit gesetzt sind, fehlerhaft programmiert ist und es so lokalen Angreifern ermöglicht das System zu kompromittieren.

***Sticky-Bit*:**

Sticky-Bit (auch t-Bit genannt) ist wie das SUID- und das SGID-Bit ein Zugriffsbit auf Dateien und Verzeichnisse unter unixähnlichen Betriebssystemen und Unix-Derivaten. Hierbei unterscheidet sich im Gegensatz zu den anderen Zugriffsbits jedoch seine Eigenschaften in den unterschiedlichen Unix-Betriebssystemen.

So hat ein gesetztes *Sticky-Bit* auf Dateien oder auch Programmen unter GNU/Linux keinerlei Funktion. Unter einigen Unix-Derivaten führt dies aber dazu, dass Programme, die mit einem gesetzten *t*-Bit ausgeführt werden nach deren Beendigung im Swap- oder Arbeitsspeicher verbleiben, um so durch den erneuten Aufruf schneller geladen werden können.

Unter GNU/Linux ist vielmehr ein gesetztes *Sticky-Bit* auf Verzeichnisse von Bedeutung (*chmod o+t <verzeichnis>*). Ein solches auf ein Verzeichnis gesetztes Bit ermöglicht das Erstellen eines gemeinsam genutzten Verzeichnisses (oktal: *777*), in dem alle Benutzer des jeweiligen Systems Schreib-, Lese- und Ausführungsrechte besitzen, aber nur der Besitzer einer Datei selbst, der Besitzer des Verzeichnisses (*/tmp* gehört *root*) oder der Systemadministrator dazu berechtigt sind, Dateien oder ggf. Unterverzeichnisse zu ändern oder gar zu löschen. Dies trifft auf das */tmp*-Verzeichnis (oktal: *1777*, *drwxrwxrwt*) zu. Dies kann in der Praxis von Benutzern eingesetzt werden, die Dateien auf einem GNU/Linux-System einer großen Benutzer-Schar zugänglich machen wollen, aber verhindern wollen, dass etwa die Dateien gelöscht oder manipuliert werden. Sie sollen einzig und allein gelesen oder ausgeführt werden. Dies könnte auch in einem Unternehmen oder von Softwareentwicklern genutzt werden, um Dateien allen Benutzern eines Systems zugänglich zu machen, ohne dass deren Löschung oder Änderung befürchtet werden muss.