



5 AppArmor-Anwendung

In diesem Kapitel werde ich zunächst die erste Anwendung von AppArmor auf einer SUSE-Distribution vorstellen. Die Installation und Anwendung auf anderen Distributionen wird in Kapitel 7 besprochen. Weiterführende Informationen über die zur Verfügung stehenden Befehle, die Syntax und die manuelle Erzeugung von Profilen finden Sie ebenfalls in späteren Kapiteln.

5.1 Installation unter SUSE Linux

Zunächst sollten Sie sicherstellen, dass auf Ihrer SUSE-Installation alle benötigten Pakete vorhanden sind. Aktuell ist AppArmor verfügbar ab SUSE Linux 10.1 und dem *SUSE Linux Enterprise Server 9* mit Servicepack 3.

Die benötigten Pakete bei der SUSE Linux-Distribution lauten:

- `apparmor-utils`
- `apparmor-profiles`
- `apparmor-docs`
- `libapparmor`
- `apache2-mod-apparmor`
- `apparmor-parser`

Für die Konfiguration von AppArmor über *Yast2* ist zusätzlich noch das Paket `yast2-apparmor` erforderlich. Stellen Sie zunächst sicher, dass diese Pakete installiert sind. Am einfachsten erfolgt die Installation mit *Yast*, das dann gleichzeitig einige Abhängigkeiten berücksichtigen kann.

Wenn Sie den SUSE Linux Enterprise Server einsetzen, haben die entsprechenden Pakete andere Namen. Die Umbenennung in AppArmor ist hier noch nicht so weit fortgeschritten wie bei der SUSE Linux-Distribution 10.1. Bei dem SLES 9 wurden die Pakete mit einem späteren Update zur Verfügung gestellt. Sie sind nicht in der original ausgelieferten Version enthalten.

- `subdomain-utils`
- `subdomain-profiles`
- `subdomain-docs`
- `mod-change-hat`
- `subdomain-parser-common`
- `yast2-subdomain`

5.2 Starten von AppArmor

Sie können jederzeit AppArmor starten und stoppen. Hierzu hat die SUSE Linux-Distribution 10.1 ein Startscript `/etc/init.d/boot.apparmor`. Dieses Startscript kann auch mit `rcapparmor` aufgerufen werden. Hierbei handelt es sich lediglich um eine Verknüpfung in dem Verzeichnis `/sbin` auf das originale Script.



Achtung

Bei der Enterprise-Version heißt dieses Script `/etc/init.d/boot.subdomain`. Entsprechend kann dieses Script auch mit dem Namen `rcsubdomain` aufgerufen werden. Diese Verknüpfung existiert aus Gründen der Rückwärtskompatibilität auch bei der SUSE Linux-Distribution.

Dieses Startscript unterstützt die folgenden Funktionen:

```
# rcapparmor
Usage: /sbin/rcapparmor start|stop|restart|try-restart|reload|
      force-reload|status|kill
```

Mit `rcapparmor start` können Sie jederzeit AppArmor starten. Genauso können Sie mit `rcapparmor stop` jederzeit AppArmor anhalten.



Achtung

Dennoch müssen Sie beachten, dass nach dem Start von AppArmor zunächst kein Prozess durch AppArmor überwacht wird. Nur neu gestartete Prozesse können von AppArmor überwacht werden.

Nach einem `rcapparmor stop` ist das AppArmor-Kernel-Modul weiterhin geladen. Lediglich die Profile wurden gelöscht. Um auch das AppArmor-Kernel-Modul zu entladen, können Sie `rcapparmor kill` verwenden.

Den Zustand von AppArmor können Sie mit dem Argument `status` auslesen:

```
# rcapparmor status
apparmor module is loaded.
50 profiles are loaded.
50 profiles are in enforce mode.
0 profiles are in complain mode.
Out of 57 processes running:
```

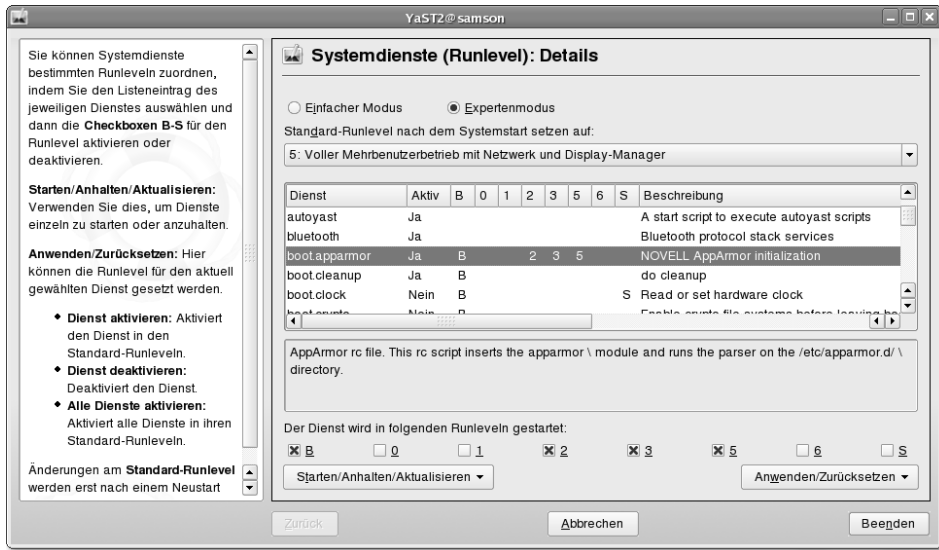


Abbildung 5.1: Mit dem Runlevel-Editor in Yast können Sie AppArmor zum Bootzeitpunkt aktivieren.

```
6 processes have profiles defined.
6 processes have profiles in enforce mode.
0 processes have profiles in complain mode.
```

Da AppArmor nur die Prozesse überwachen kann, für die es Profile besitzt und die nach dem Start von AppArmor geladen wurden, sollte AppArmor immer sehr früh zum Boot-Zeitpunkt gestartet werden. Dann kann AppArmor, entsprechende Profile vorausgesetzt, anschließend gestartete Prozesse überwachen. Um zu überprüfen, ob dies auf Ihrem System der Fall ist, können Sie in Yast den Runlevel-Editor starten oder das auf der Kommandozeile mit `chkconfig boot.apparmor` überprüfen (siehe Abbildung 5.1).

```
# chkconfig boot.apparmor
boot.apparmor on
```

Wird AppArmor so zum Bootzeitpunkt gestartet, werden die entsprechenden Applikationen überwacht.

5.2.1 Welche Prozesse werden überwacht?

Um nun schnell zu ermitteln, welche Prozesse bereits überwacht werden, gibt es eine Möglichkeit. Hierfür können Sie das Kommando `unconfined` benutzen. Dieses Kommando ruft den Befehl `netstat` auf und ermittelt so alle Netzwerkdienste. Wie Sie alle laufenden Prozesse auf ihren AppArmor-Status prüfen können, ist in Abschnitt 6.3.9 erläutert. Anschließend prüft `unconfined`, ob diese Netzwerkdienste von AppArmor überwacht werden. Sinnvoll ist es, dass alle über das Netzwerk er-

reichbaren Dienste als potenzielle Sicherheitslücken von AppArmor überwacht werden:

```
# unconfined
11314 /sbin/dhcpd not confined
11470 /usr/sbin/mdnsd confined by '/usr/sbin/mdnsd (enforce)'
11470 /usr/sbin/mdnsd confined by '/usr/sbin/mdnsd (enforce)'
11487 /usr/lib/zmd/zmd-bin not confined
11527 /sbin/portmap not confined
11527 /sbin/portmap not confined
11646 /usr/sbin/cupsd not confined
11646 /usr/sbin/cupsd not confined
11728 /usr/lib/postfix/master confined by '/usr/lib/postfix/master (enforce)'
11728 /usr/lib/postfix/master confined by '/usr/lib/postfix/master (enforce)'
11840 /usr/sbin/sshd not confined
```

Wenn Sie lediglich die geladenen Profile anzeigen möchten, genügt der folgende Befehl:

```
# cat /sys/kernel/security/apparmor/profiles
/usr/sbin/traceroute (enforce)
/usr/sbin/squid (enforce)
/usr/sbin/sendmail (enforce)
...
```

Diese Liste zeigt Ihnen, welche Prozesse potenziell von AppArmor überwacht werden, falls das entsprechende Programm gestartet wird.

5.3 Analyse der Protokolle

AppArmor protokolliert alle Verstöße gegen die geladenen Profile in einem Protokoll. Hierfür nutzt AppArmor den *Auditd*-Daemon, der in einem eigenen Kapitel besprochen wird (siehe Anhang A). Der *Auditd*-Daemon protokolliert seine Meldungen in der Datei `/var/log/audit/audit.log`. Falls der *Auditd*-Daemon nicht aktiv ist, werden die Protokolle über den *Syslogd*-Daemon in die Datei `/var/log/messages` geschrieben.

Hier möchte ich Ihnen kurz den Aufbau der AppArmor-Protokollmeldungen beschreiben. Diese Protokollmeldungen helfen Ihnen sowohl bei der Fehlersuche als auch bei der Anpassung vorhandener Profile an neue Anforderungen und beim Erstellen kompletter neuer Profile.

AppArmor kann vier verschiedene Protokollmeldungen schreiben, die im Folgenden einzeln betrachtet werden sollen.

- **PERMITTING:** Diese Meldungen werden erzeugt, wenn AppArmor sich in dem Lernmodus befindet. Jeder Zugriff der Applikation im Lernmodus wird dann protokolliert.

```

type=APPARMOR msg=audit(1155035013.332:16): PERMITTING access ◀
    to capability 'net_raw' (nmap(24874) profile /usr/bin/ ◀
    nmap active /usr/bin/nmap)
type=APPARMOR msg=audit(1155035013.448:17): PERMITTING r access ◀
    to /usr/share/nmap/nmap-mac-prefixes (nmap(24874) ◀
    profile /usr/bin/nmap active /usr/bin/nmap)

```

In der ersten Meldung wird der Zugriff auf eine Capability erlaubt. Der zugreifende Prozess ist nmap mit der Prozessnummer 24874. Das aktive Profil /usr/bin/nmap überwacht dabei den Prozess /usr/bin/nmap. In der zweiten Meldung wird der lesende (r) Zugriff auf die Datei /usr/share/nmap/nmap-mac-prefixes demselben Prozess gewährt.

- REJECTING: Diese Meldungen werden erzeugt, wenn AppArmor einen Prozess überwacht und dabei den Zugriff nicht erlaubt. Häufig führt das auch zu einer Fehlfunktion der Applikation. Das Profil muss erweitert werden, um den Zugriff zu erlauben. Möglicherweise handelt es sich aber auch um einen unerwünschten bössartigen Zugriff, der von AppArmor erfolgreich abgewehrt wurde. Im Folgenden sind zwei typische Ablehnungen aufgeführt:

```

type=APPARMOR msg=audit(1155037095.706:54): REJECTING r access ◀
    to /proc/swaps (httpd2-prefork(28354) profile /usr ◀
    /sbin /httpd2-prefork active /usr/sbin/httpd2-prefork)
type=APPARMOR msg=audit(1155037095.710:57): REJECTING x access ◀
    to /bin/mount (httpd2-prefork(28354) profile /usr/sbin ◀
    /httpd2-prefork active /usr/sbin/httpd2-prefork)

```

Hier wird für den Prozess /usr/sbin/httpd2-prefork der lesende Zugriff auf die Datei /proc/swaps und der ausführende Zugriff auf die Datei /bin/mount abgelehnt.

- LOGPROF-HINT: Diese Meldungen tauchen nur im Lernmodus auf und weisen den Befehl logprof auf besondere Umstände hin. Ruft ein Programm im Lernmodus einen weiteren Prozess auf, so schreibt AppArmor die folgende Meldung:

```

type=APPARMOR msg=audit(1155037256.176:287): LOGPROF-HINT ◀
    changing_profile pid=28383

```

Diese Meldungen werden nur von logprof ausgewertet.

- AUDITING: Diese Meldung erscheint nur im Audit-Modus. Im Audit-Modus wird jeder Zugriff einer Applikation protokolliert. Erlaubte Zugriffe werden mit dem Schlüsselwort AUDITING protokolliert und verbotene Zugriffe mit dem Schlüsselwort REJECTING.

```

type=APPARMOR msg=audit(1155037692.339:315): AUDITING r access ◀
    to /etc/localtime (nmap(28408) profile /usr/bin/nmap ◀
    active /usr/bin/nmap)

```

5.4 AppArmor-Benachrichtigungen und -Berichte

Novell AppArmor kann auch Benachrichtigungen und Berichte erzeugen. Die Konfiguration erfolgt über Yast2. Bei den Benachrichtigungen handelt es sich um regelmäßige E-Mails, die Sie über AppArmor-Aktivitäten informieren.

Sie erreichen die Konfiguration der Benachrichtigungen über Yast2 im Menü NOVELL APPARMOR unter dem Menüpunkt APPARMOR-KONTROLLEISTE (siehe Abbildung 5.2).



Abbildung 5.2: Über Yast2 können Sie die Benachrichtigung bei Sicherheitereignissen aktivieren.

Dort haben Sie die Möglichkeit, zwischen drei verschiedenen Benachrichtigungen zu wählen (siehe Abbildung 5.3).

Leider ist die Konfiguration der Benachrichtigung in der aktuellen Ausgabe defekt¹ und zeigt nach erfolgter Konfiguration immer noch den Status DIE BENACHRICHTIGUNG IST DEAKTIVIERT. Die Benachrichtigungen werden jedoch versandt. Je nach Wahl der Benachrichtigung ist die versandte E-Mail sehr knapp oder ausführlicher. Eine typische ausführliche E-Mail hat den folgenden Inhalt:

```
Subject: Verbose Security Report for Samson.
Date: Sat, 19 Aug 2006 15:31:17 +0200 (CEST)
From: root@samson.spenneberg.net (AppArmor Security Notification)
```

¹ https://bugzilla.novell.com/show_bug.cgi?id=177039

The following security events occurred since Thu Jan 1 01:00:00 1970:

```
type=APPARMOR msg=audit(1155994266.353:7635): REJECTING r access
to /var (ls(27036) profile /usr/sbin/sshd active ralf)
type=APPARMOR msg=audit(1155994267.361:7636): REJECTING x access
to /usr/bin/nail (bash(27037) profile /usr/sbin/sshd
active ralf)
type=APPARMOR msg=audit(1155994267.361:7637): REJECTING r access
to /usr/bin/nail (bash(27037) profile /usr/sbin/sshd
active ralf)
```

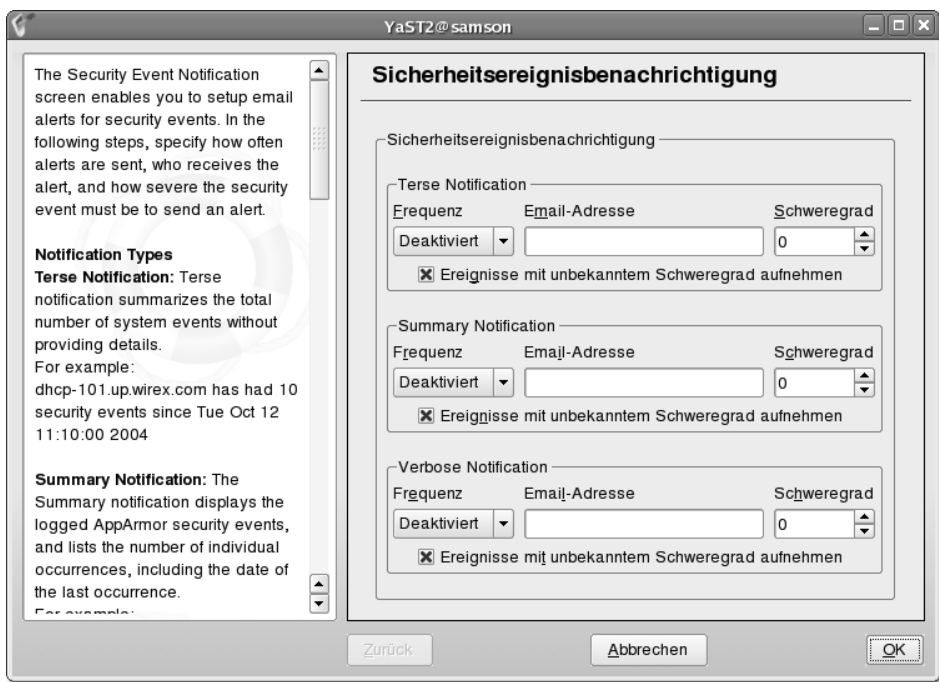


Abbildung 5.3: Über Yast2 können Sie drei verschiedene Benachrichtigungen aktivieren.

Mit der Berichterstellung können Sie drei verschiedene Berichte erzeugen lassen:

- Executive Summary Report
- Applications Audit
- Security Incident Report

Alle Berichte lassen sich automatisch zu einem bestimmten Zeitpunkt ausführen und per E-Mail versenden oder auch bei Bedarf sofort ausführen.

5.4.1 Executive Summary Report

Der *Executive Summary Report* fasst die Ereignisse, die im Security Incident Report ausführlich dargestellt werden, in wenigen Worten zusammen, sodass die verantwortliche Person sich schnell einen Überblick über die vorgefallenen Aktivitäten machen kann.

5.4.2 Applications Audit

Der *Applications Audit* prüft, welche der laufenden Anwendungen von AppArmor überwacht werden und welche nicht. Der Bericht entspricht dem Befehl `unconfined`.

5.4.3 Security Incident Report

Der *Security Incident Report* führt alle Verstöße gegen die AppArmor-Richtlinien auf (siehe Abbildung 5.4). Dabei besteht die Möglichkeit, die Verstöße vorher zu filtern. Sie können den Zeitraum und den Namen des zu überwachenden Programms angeben.

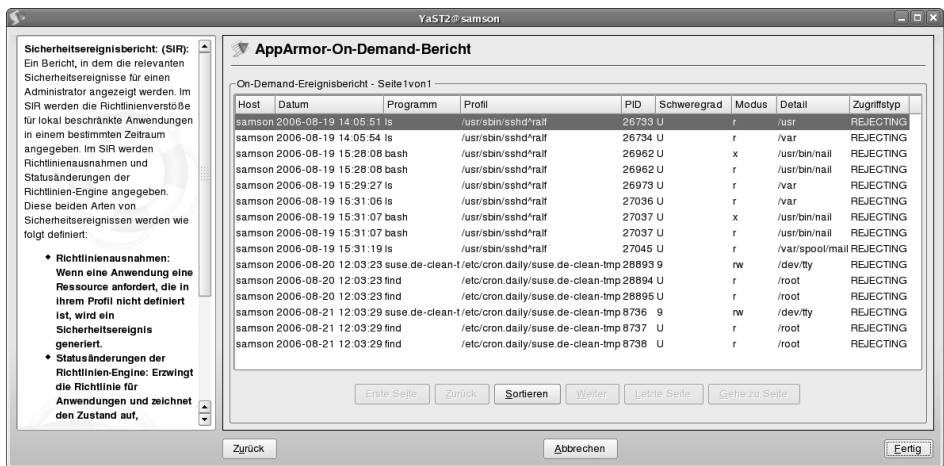


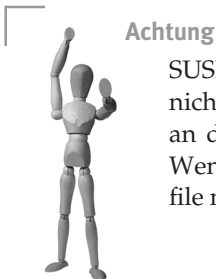
Abbildung 5.4: Mit Yastz können Sie einen Security Incident Report anfertigen.

5.5 Erzeugen eines Profils mit Yast

SUSE Linux bietet eine Vielzahl von vorbereiteten Profilen. So ist ein gewisser Mindestschutz gewährleistet, wenn Sie AppArmor aktivieren. Enthalten sind Profile sowohl für exponierte Netzwerkdienste als auch für grafische Applikationen und Kommandozeilenbefehle, die Daten aus nicht vertrauenswürdigen Quellen verarbeiten. Letztere sind zum Beispiel der Realplayer, der Firefox-Browser und der Acrobat-Reader. Aber auch die Befehle `netstat` und `ping` gehören dazu. Obwohl die

Profile für diese Applikationen enthalten sind, sind sie nicht unbedingt aktiv. Um eine Liste aller geladenen Profile zu erhalten, können Sie diese entweder aus dem Kernel auslesen, oder Sie betrachten die in dem Verzeichnis `/etc/apparmor.d` vorhandenen Dateien. Weitere inaktive Profile befinden sich in dem Verzeichnis `/etc/apparmor/profiles/extras`. Um diese Profile zu aktivieren, müssen Sie lediglich das entsprechende Profil aus diesem Verzeichnis in das Verzeichnis `/etc/apparmor.d` kopieren. Bei dem nächsten Neustart des Systems wird das Profil aktiviert. Möchten Sie das Profil sofort aktivieren, so können Sie das mit dem Befehl `enforce` erledigen. Geben Sie lediglich den kompletten Namen des zu überwachenden Programms inklusive Pfad an.

```
# enforce /usr/bin/ethereal
Setting /usr/bin/ethereal to enforce mode.
```



Achtung

SUSE bezeichnet die in diesem Verzeichnis vorhandenen Profile als nicht ausreichend ausgereift. Die Profile wurden in vielen Fällen nicht an die aktuelle Distribution angepasst. Pfade stimmen daher nicht. Wenn Sie diese Profile verwenden möchten, müssen Sie meist die Profile noch anpassen.

Ich zeige Ihnen nun, wie Sie selbst ein Profil für ein Programm erzeugen können. Am einfachsten erfolgt das mit Yast. Ich werde in einem späteren Kapitel auch weitere Methoden vorstellen. Hier soll nun ein Profil für das Programm `nmap` erzeugt werden. Im Weiteren werde ich Ihnen die Schritte vorstellen, damit Sie diese auch auf Ihrem System direkt nachvollziehen können. Hierzu muss zunächst das Programm installiert werden. Dies erfolgt ganz einfach mit `yast -i nmap`.

Nun starten Sie `Yast2` und rufen den ASSISTENT ZUM HINZUFÜGEN VON PROFILEN auf (siehe Abb. 5.5).

In dem neuen Fenster geben Sie den Namen der zu überwachenden Anwendung mit ihrem kompletten Pfad an (siehe Abb. 5.6). Anschließend bestätigen Sie die Eingabe mit `CREATE`. Leider ist bei SUSE Linux 10.1 dieser Dialog noch nicht in die deutsche Sprache übersetzt worden. Nun müssen Sie den Befehl `nmap` benutzen. Rufen Sie zum Beispiel als `root` den Befehl folgendermaßen auf:

```
# nmap -sS localhost
# nmap -sS -0 -v some_other_host
```

Es ist sinnvoll, möglichst alle Funktionen der Applikation zu nutzen. `Yast2` wird anschließend ein Profil erzeugen, das genau diese Funktionen erlaubt. Natürlich sollten Sie sicherstellen, dass der Befehl gerade jetzt nicht für einen Angriff genutzt werden kann. Wenn Sie den Befehl ausreichend getestet haben, klicken Sie `SCAN SYSTEM LOG FOR APPARMOR EVENTS` (Abbildung 5.7). Nun präsentiert Ihnen `Yast2` auf den folgenden Dialogen die Dateien und Funktionen, die der getestete Befehl genutzt hat.

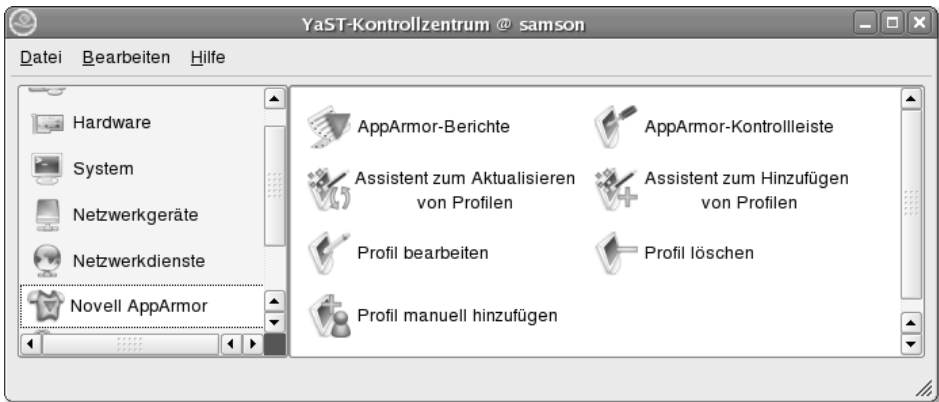


Abbildung 5.5: Über Yastz können Sie die wichtigsten AppArmor-Funktionen steuern.



Abbildung 5.6: Geben Sie bei der Anwendung den kompletten Pfad mit an.

Sie haben hier die Möglichkeit zu entscheiden, ob das Profil den Zugriff erlauben soll oder nicht. Sie müssen nun in jedem Dialog einzeln entweder ALLOW oder DENY auswählen. Damit die Anwendung später erfolgreich funktioniert, ist ein Erlauben erforderlich (Abb. 5.8). In vielen Fällen erkennt der Assistent, dass es bereits vorgefertigte Abstraktionen gibt. Wenn Sie diese auswählen, verringert sich die Anzahl der Regeln ungemein. Hier (Abb. 5.9) erkennt der Assistent an dem Zugriff auf das Gerät `/dev/tty`, dass der Prozess auf das Terminal zugreifen möchte. Anstatt nun

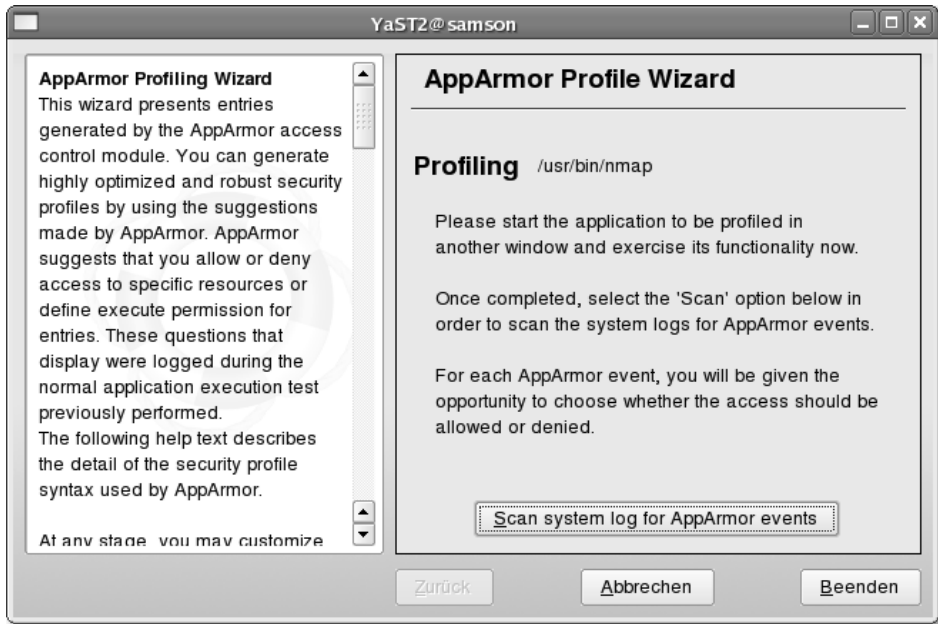


Abbildung 5.7: Der Wizard fordert Sie auf, die Applikation zu verwenden.

alle notwendigen Zugriffe einzeln zu erlauben, schlägt er die Abstraktion `abstractions/consol` vor. Diese befindet sich in dem Verzeichnis `/etc/apparmor.d`.

Allgemein bietet Ihnen der Dialog die folgenden Möglichkeiten (siehe auch Abschnitt 6.3.7):

- `ALLOW` erlaubt den angegebenen Zugriff auf die Ressource.
- `DENY` verweigert den angegebenen Zugriff auf die Ressource. Das Programm funktioniert möglicherweise nicht richtig.
- `GLOB`: Durch Anwählen dieses Buttons wird der letzte Pfadanteil der Ressource durch ein `*` ersetzt. Der Zugriff betrifft dann alle Dateien auf dieser Ebene. Ein weiteres Betätigen der Schaltfläche löscht eine weitere Ebene und ersetzt den `*` durch `**`. Hiermit wird der Zugriff auf alle Unterverzeichnisse und Dateien erlaubt.
- `GLOB w/EXT`: Dies entspricht dem `GLOB` mit dem Unterschied, dass die Dateien identische Endungen (Extensions) aufweisen müssen.
- `EDIT` erlaubt es Ihnen, die Pfadangabe der Ressource frei zu ändern.
- `INHERIT`: Wenn ein weiteres Programm aufgerufen wird, wird der entstehende Prozess ebenfalls von diesem Profil überwacht.
- `PROFILE`: Hiermit verlangen Sie, dass das aufgerufene Programm über ein eigenes Profil verfügt und dass dieses auch tatsächlich geladen ist. Verfügt das Programm über kein Profil, schlägt der Aufruf fehl!

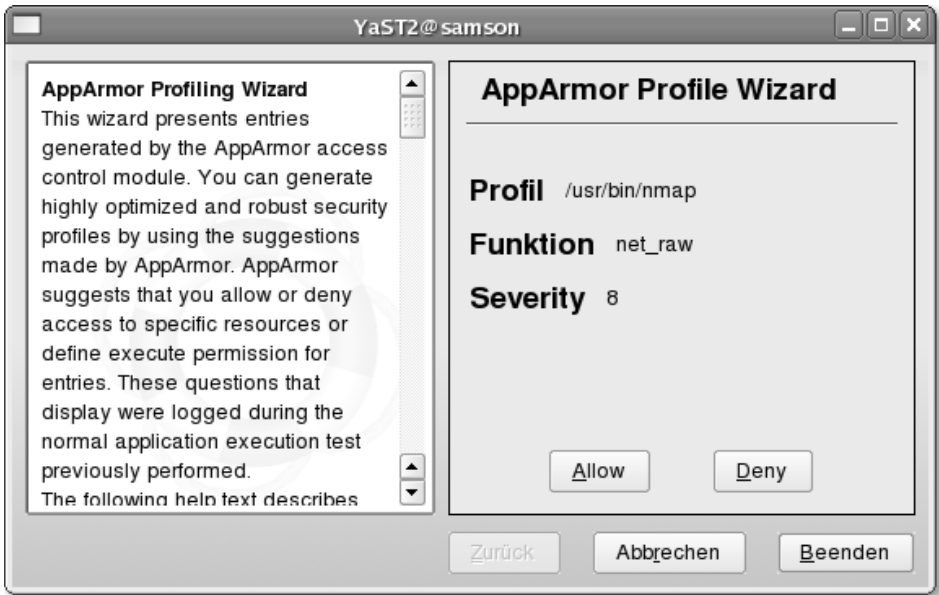


Abbildung 5.8: Nmap benötigt für die Erzeugung der Pakete für den Scan eine spezielle Capability: CAP_NET_RAW.

- UNCONFINED: Diese Funktion ist besonders gefährlich und startet den neuen Prozess ohne jede Überwachung.

Wenn Sie alle Dialoge beantwortet haben, beginnt der Assistent wieder von Neuem und bietet Ihnen an, die Protokolle zu analysieren. Sie können nun den Assistenten beenden. Der Assistent hat nun ein Profil für den Befehl Nmap erzeugt. Dieses Profil wurde in dem Verzeichnis `/etc/apparmor.d` unter dem Namen `usr.bin.nmap2` gespeichert und auch direkt geladen. Wenn Sie nun Nmap aufrufen, wird der Prozess von AppArmor überwacht. Im Folgenden sehen Sie das von dem Assistenten erzeugte Profil:

```

1 # vim:syntax=apparmor
2 # Last Modified: Mon Jun 26 12:10:10 2006
3 #include <tunables/global>
4
5 /usr/bin/nmap {
6 #include <abstractions/base>
7 #include <abstractions/containers>
8 #include <abstractions/daemon>
9
10 capability net_raw,
```

² Diese Namenskonvention ist gebräuchlich, aber nicht zwingend. Die Dateien dürfen jeden beliebigen Namen haben.

```

11
12 /usr/bin/nmap r,
13 /usr/share/nmap/nmap-mac-prefixes r,
14 /usr/share/nmap/nmap-os-fingerprints r,
15 /usr/share/nmap/nmap-services r,
16 }

```

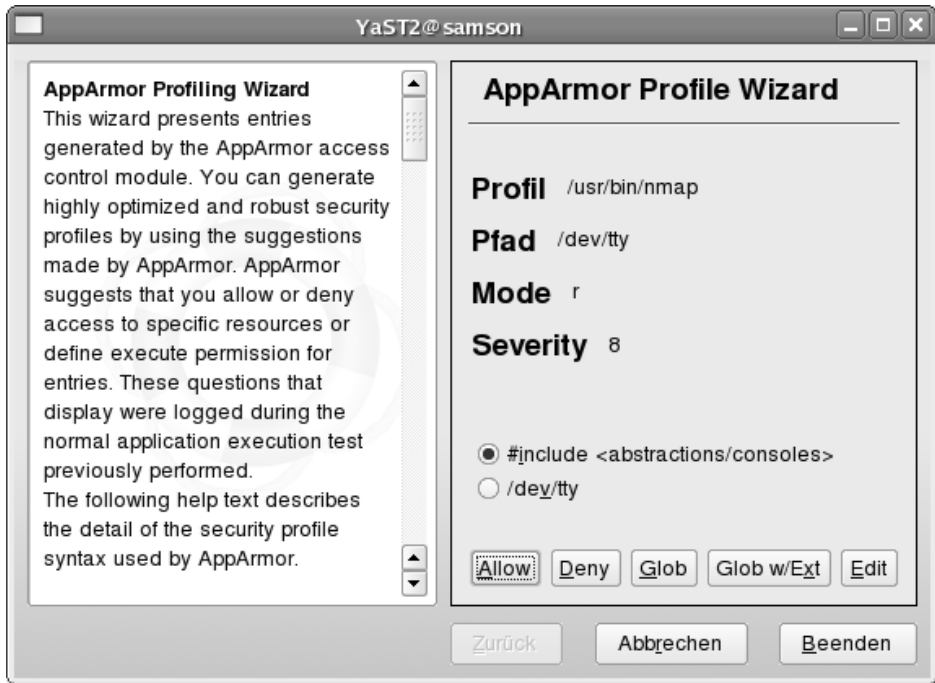


Abbildung 5.9: Stehen Abstraktionen zur Verfügung, bietet Ihnen der Assistent diese auch an.

Die Zeilennummern zu Beginn jeder Zeile wurden von mir der Lesbarkeit halber hinzugefügt. Die Zeilen 1-2 sind lediglich Kommentare und enthalten nur für den Vim einen Syntax-Highlighting-Hinweis. Leider ist in der Version SUSE Linux 10.1 keine Syntax-Beschreibung für AppArmor enthalten, obwohl die Manpage `apparmor.vim(5)` vorhanden ist. Wahrscheinlich wird ein Update dieses Problem beheben. Ab Zeile 3 beginnt das Profil. Hier wird über eine Include-Direktive der Inhalt der Datei `tunables/global` zu diesem Profil geladen. Alle Pfade in dieser Datei sind relativ zum Verzeichnis `/etc/apparmor.d`. Die geladene Datei ist daher `/etc/apparmor.d/tunables/global`. Anschließend beginnt das Profil für das Kommando `/usr/bin/nmap`. Zunächst werden hier einige Abstraktionen geladen. Wenn Sie wissen möchten, was sich in diesen Abstraktionen befindet, können Sie die entsprechenden Dateien in einem Editor öffnen.

Nun wird die Capability `CAP_NET_RAW` zur Verfügung gestellt. Insgesamt gibt es je nach Kernel bis zu 31 verschiedene Capabilities, die in der Manpage `capabilities(7)` erläutert werden. Die Zeilen 12–15 erlauben dem Nmap-Prozess, auf die verschiedenen Dateien zuzugreifen.

Wenn Sie nun den Nmap-Befehl benutzen, wird er immer von AppArmor überwacht. Falls eine bestimmte Funktion nicht zur Verfügung steht, sollten Sie die AppArmor-Protokolle analysieren. Falls Sie dort Meldungen finden, die auf einen Fehler im Profil hinweisen, können Sie den gesamten Vorgang wiederholen. Wenn Sie vorher das bereits erzeugte Profil nicht löschen, erweitert der Assistent das Profil nur um die fehlenden Punkte.

5.6 Erzeugen eines Subprofils (Hat) mit Yast

AppArmor verfügt über die sehr interessante Funktion des Subprofils. Ein Prozess wird von AppArmor entsprechend einem Profil überwacht. Für bestimmte Funktionen kann zudem ein Subprofil verwendet werden. Dieses wird als *Hat* bezeichnet. Dieser Hat kann mehr oder weniger Funktionen als das übergeordnete Profil aufweisen.



Hinweis

Im diesem Kapitel wollen wir diese Funktion für den *Apache* Webserver nutzen. Hierzu soll für eine *PHP*-Applikation ein *Hat* erzeugt werden. Dadurch soll diese *PHP*-Applikation nur die durch das Profil erlaubten Funktionen nutzen dürfen.

Um diese Funktion zu nutzen, muss eine Applikation speziell an AppArmor angepasst worden sein. Aktuell ist das nur für den *Apache* Webserver der Fall. Auf der AppArmor-Homepage gibt es aber auch schon ein `pam_apparmor`-PAM-Modul, das für alle PAM-fähigen Programme ein benutzerabhängiges Subprofil aktivieren kann. Dieses PAM-Modul wird in Abschnitt 8.2.2 besprochen. Der Webserver ist insbesondere interessant, da bei Webapplikationen aus Geschwindigkeitsgründen die Scripts häufig direkt von dem Webserver ausgeführt werden. Hierfür besitzt der Apache zum Beispiel die Module `mod_php` und `mod_perl`, die direkt *PHP*-Scripts und *Perl*-Scripts ausführen können. Wenn diese Scripts nun mehr oder weniger Privilegien benötigen als der Webserver selbst, ist das nur mit einem Subprofil möglich.

Der Apache kann ein Subprofil auswählen in Abhängigkeit von

- der URI,
- der Location,
- des Verzeichnisses (Directory) oder
- des VirtualHosts.

Die Funktionen des VirtualHosts werden in einem späteren Kapitel erläutert. Hier werden wir zunächst Subprofile in Abhängigkeit von der URI erzeugen.

Stellen Sie zunächst sicher, dass sowohl der Apache 2.x-Webserver als auch die Pakete `apache2_mod_php5` und `apache2_mod_apparmor` installiert wurden. Achten Sie darauf, dass diese Namen der SUSE Linux 10.1-Distribution entnommen wurden. Ältere und neuere Distributionen verwenden hier möglicherweise andere Namen, wie `apache2_mod_subdomain` oder `apache2_mod_changehat`.

Um nun für dieses Kapitel eine sinnvolle, aber auch übersichtliche Webapplikation einzusetzen, habe ich `phpSysInfo` ausgewählt. `phpSysInfo` ist eine Webapplikation, die Systeminformationen übersichtlich in einer Webseite darstellt. Sie können die Webapplikation von Sourceforge herunterladen (<http://phpsysinfo.sourceforge.net>). Wenn Sie keinen Internetzugang besitzen, können Sie auch das Paket von der CD aus dem Verzeichnis `/Software` nutzen.

Installieren Sie zunächst die Software, indem Sie das Quelltextpaket extrahieren und in die DocumentRoot des Apache 2.x-Webserver kopieren. Für unsere Zwecke genügt es für die Konfiguration, die vorbereitete Datei umzubenennen. Sie können diese Datei natürlich anpassen. Für die Demonstration in diesem Kapitel ist das jedoch nicht erforderlich.

```
# tar xzf phpsysinfo-2.5.2-rc3.tar.gz
# mv phpsysinfo /srv/www/htdocs/
# cd /srv/www/htdocs/phpsysinfo
# mv config.php.new config.php
```

Starten Sie nun den Apache 2.x-Webserver. Wenn Sie nun in einem Browser die URL `http://localhost/phpsysinfo` eingeben, sollte ein Bild wie in Abbildung 5.10 erscheinen. Falls Sie ein anderes Bild erhalten, kann dies drei Gründe haben:

1. Ihr Webserver läuft nicht. Prüfen Sie in den Protokollen, ob Sie tatsächlich den Webserver gestartet haben.
2. Ihr Browser verbindet sich nicht mit Ihrem Webserver. Prüfen Sie, ob der Browser einen Proxy benutzt, und deaktivieren Sie den Proxy für den Zielrechner `localhost`.
3. `AppArmor` überwacht bereits den Apache Webserver. Sie sehen ein Bild wie in Abbildung 5.11. Auf einer SUSE Linux 10.1-Distribution wird das Apache-AppArmor-Profil per Default nicht geladen. Lesen Sie in diesem Fall einfach weiter.

Nachdem wir nun wissen, wie `phpSysInfo` bei erfolgreicher Ausführung die Informationen anzeigt, werden wir nun die `AppArmor`-Überwachung für den Apache 2.x aktivieren.

The screenshot shows the phpSysInfo web application running in a Mozilla Firefox browser. The page title is "System Information: (127.0.0.1)". The application displays several sections of system data:

System Vital

Canonical Hostname	
Listening IP	127.0.0.1
Kernel Version	2.6.16.13-4-default
Distro Name	SUSE LINUX 10.1 (i586) VERSION = 10.1
Uptime	5 hours 32 minutes
Current Users	1
Load Averages	0.04 0.01 0.00

Network Usage

Device	Received	Sent	Err/Drop
lo	70.59 MB	70.59 MB	0.0
eth0	27.70 MB	68.14 MB	0.0
sit0	0.00 KB	0.00 KB	0.0

Hardware Information

Processors: 1
 Model: Intel(R) Pentium(R) 4 CPU 3.00GHz
 CPU Speed: 2.99 GHz
 Cache Size: 1024.00 KB
 System: 5690.86
 Bogomips:
 PCI Devices:
 - Ethernet controller: Intel Corporation 82562EZ 10/100 Ethernet Controller
 - Host bridge: Intel Corporation 82853/PE/P DRAM Controller/Host-Hub Interface
 - IDE interface: Intel Corporation 82801EB
 - IDE interface: Intel Corporation 82801EB/ER
 - ISA bridge: Intel Corporation 82801EB/ER
 - Multimedia audio controller: Intel Corporation 82801EB/ER
 - PCI bridge: Intel Corporation 82801 PCI Bridge
 - SMBus: Intel Corporation 82801EB/ER
 - (4x) USB Controller: Intel Corporation 82801EB/ER
 - VGA compatible controller: Intel Corporation 82853G Integrated Graphics Controller
 IDE Devices:
 - hdc: SAMSUNG DVD-ROM SD-616E
 - hda: HDS722580VLAT20 (Capacity: 74.51 GB)
 SCSI Devices: none
 USB Devices: - Standard Microsystems Corp.

Memory Usage

Type	Percent Capacity	Free	Used	Size
Physical Memory	82%	88.19 MB	408.08 MB	496.27 MB
- Kernel + applications	19%		94.04 MB	
- Buffers	7%		36.35 MB	
- Cached	56%		277.69 MB	
Disk Swap	0%	745.12 MB	44.00 KB	745.16 MB

Mounted Filesystems

Mount	Type	Partition	Percent Capacity	Free	Used	Size
/	reiserfs	/dev/hda2	4%	70.47 GB	3.31 GB	73.77 GB
/dev	tmpfs	udev	0% (1%)	248.03 MB	108.00 KB	248.13 MB
Totals :			4%	70.71 GB	3.31 GB	74.02 GB

At the bottom of the page, there are dropdown menus for "Template: classic" and "Language: en", a "Submit" button, and a footer indicating the page was created by phpSysInfo-2.5.2_rc3on Jun 26, 2006 at 02:21 PM, with a load time of 0.5424 sec.

Abbildung 5.10: phpSysInfo zeigt Informationen über den Zustand des Systems an.

Kopieren Sie hierzu das Profil in das entsprechende Verzeichnis, aktivieren Sie es mit `enforce` und starten Sie den Apache erneut:

```
# cp /etc/apparmor/profiles/extras/usr.sbin.httpd2-prefork /etc/
    apparmor.d/
# enforce /usr/sbin/httpd2-prefork
```

```
Setting /usr/sbin/httpd2-prefork to enforce mode.
# rcapache2 restart
```

Wenn Sie nun erneut die phpSysInfo-Webseite aufrufen, sollten Sie sehr viele Fehlermeldungen sehen (Abb. 5.11). Um nun ein Subprofil für diese Webapplikation zu erzeugen, das phpSysInfo mit den nötigen Berechtigungen für die Ausführung versorgt, gehen wir zunächst genauso vor wie bereits in dem letzten Kapitel. Rufen Sie *Yast2* auf, und wählen Sie dort im Menü NOVELL APPARMOR den ASSISTENT ZUM HINZUFÜGEN VON PROFILEN aus. Wir wählen bewusst nicht den Assistenten zum Aktualisieren von Profilen aus. Auch der jetzt gewählte Assistent kann existierende

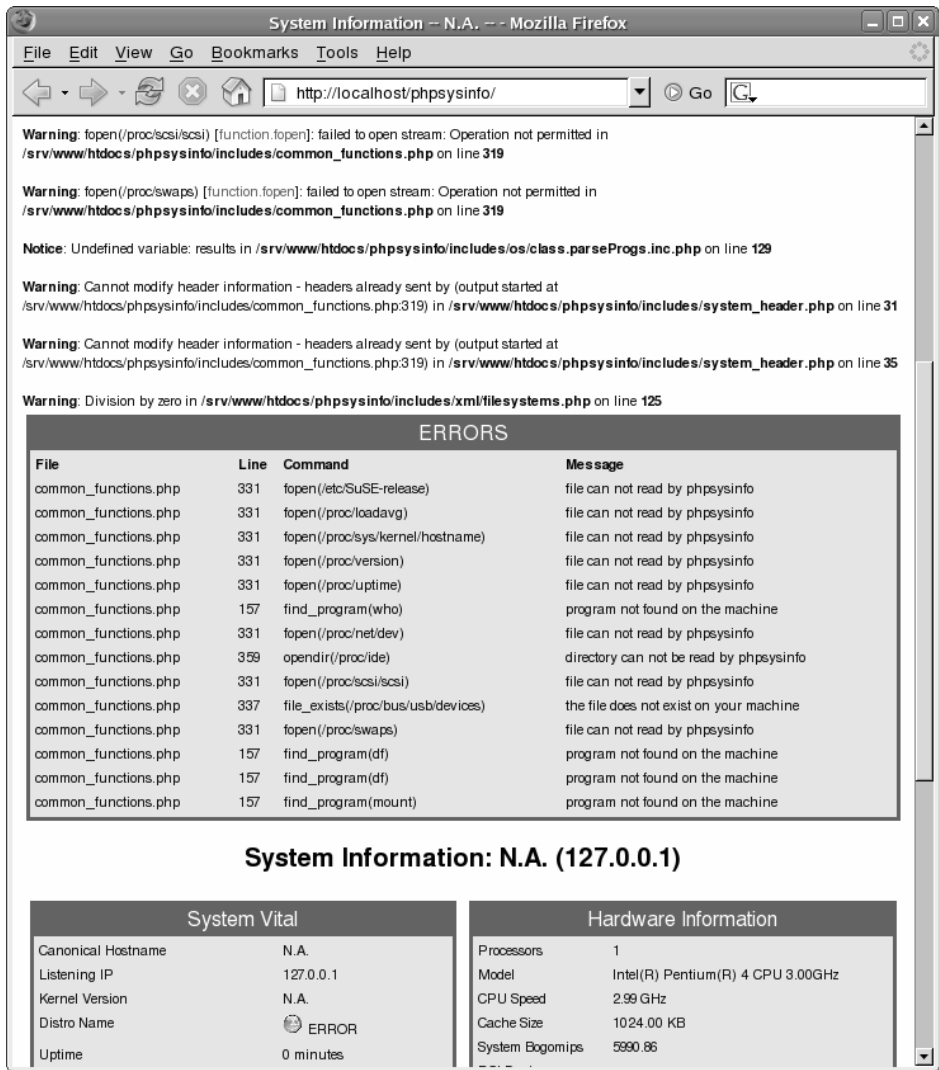


Abbildung 5.11: Wenn AppArmor aktiv ist, werden viele Zugriffe von phpSysInfo verhindert.

Profile aktualisieren. Dabei ist es aber möglich, die Aktualisierung auf ein bestimmtes Profil zu beschränken. Der andere Assistent aktualisiert alle Profile!

Bei dem Assistenten geben Sie nun als Applikation `/usr/sbin/httpd2-prefork` an. Benutzen Sie nun wieder die Applikation. Die Webapplikation sollte nun im Wesentlichen funktionieren, da jetzt wieder alle Zugriffe erlaubt, aber auch protokolliert werden. Nachdem Sie die Webapplikation genutzt haben, wählen Sie wieder im Assistenten `SCAN SYSTEM LOG`. Nun wird der Assistent Ihnen die erfolgten Zugriffe präsentieren, und Sie müssen entscheiden, ob der Zugriff erlaubt ist. Hierbei sollte der Assistent Ihnen sehr früh anbieten, einen zusätzlichen Hat hinzuzufügen. Bestätigen Sie das mit `ADD REQUESTED HAT`. Hiermit wird nun ein neues Profil für diese URI eröffnet (Abbildung 5.12). Alle weiteren Zugriffe werden nun, nach einer Bestätigung durch Sie, diesem Profil hinzugefügt. Achten Sie darauf, dass der Name des bearbeiteten Profils sich geändert hat. Er lautet nun `/usr/sbin/httpd2-prefork^/phpsysinfo/` (Abbildung 5.13).

Damit die von `phpSysInfo` aufgerufenen externen Programme unter der Kontrolle von AppArmor ausgeführt werden, wählen Sie für alle diese Zugriffe auf externe Kommandos `INHERIT` aus. Falls eine Datei gelesen werden muss, wählen Sie `ALLOW`. Eine Ausnahme ist der Befehl `mount`. Dieser Befehl muss `unconfined` ausgeführt werden. Wenn Sie dies nicht möchten, wählen Sie `DENY`. Sie müssen dann auf diese Funktion in `phpSysInfo` verzichten. Wenn Sie `UNCONFINED` auswählen, erhalten Sie eine Warnmeldung, die Sie auf die möglichen Gefahren hinweist.


Wenn Sie alle Fragen beantwortet haben und den Assistenten beenden, sollte die Webapplikation wieder so funktionieren wie vor der Aktivierung von AppArmor.



Abbildung 5.12: Yast bietet an, einen neuen Hat für die PHP-Applikation zu erzeugen.

Der einzige Unterschied ist die Tatsache, dass die Webapplikation nun von AppArmor überwacht wird und keine zusätzlichen Funktionen ausüben kann, die von einem Angreifer für einen Einbruch missbraucht werden könnten.

Hinweis



Wenn die Applikation noch nicht funktioniert und scheinbar allgemeine PHP-Fehler auftreten, dann müssen Sie möglicherweise noch das Apache-Profil selbst anpassen. Rufen Sie erneut den Assistenten auf, und geben Sie wieder den kompletten Pfad `/usr/sbin/httpd2-prefork` an. Starten Sie nun den Webserver neu, und klicken Sie direkt anschließend den Button `SCAN SYSTEM LOGS FOR APPARMOR EVENTS`. Es fehlten wahrscheinlich in dem Profil noch die Zugriffe auf die PHP-Konfigurationsdateien. Diese fügen Sie hiermit hinzu. Jetzt sollte die Webapplikation fehlerfrei laufen.

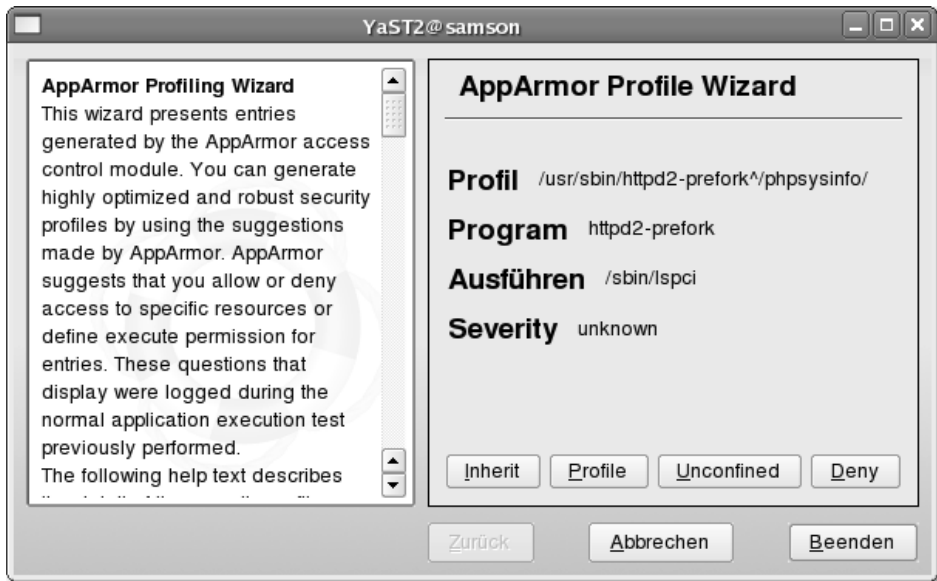


Abbildung 5.13: AppArmor bittet um die Bestätigung, in dem Hat Befehle (`lspci`) ausführen zu dürfen.

Das erzeugte Subprofil befindet sich ebenfalls in der Datei `/etc/apparmor.d/usr.sbin.httpd2-prefork`. Sie finden es am unteren Ende der Datei:

...

```
^/phpsysinfo/ {
    #include <abstractions/base>
    #include <abstractions/bash>
```

```
#include <abstractions/consoles>

/bin/bash ixr,
/bin/df ixr,
/bin/mount ux,
/etc/SuSE-release r,
/lib/ld-2.4.so ixr,
/proc/** r,
/sbin/lspci ixr,
/srv/www/htdocs/phpsysinfo/** r,
/sys/** r,
/usr/bin/lsscsi ixr,
/usr/bin/who ixr,
/usr/sbin/lsub ixr,
/var/log/apache2/access_log w,
/var/log/apache2/error_log w,
/var/run/nscd/socket w,
}

...
```

Die Namen von Subprofilen beginnen mit dem `^`. Falls Ihr Subprofil einen anderen Aufbau hat, wundern Sie sich nicht. Der Aufbau hängt stark von dem Einsatz des Assistenten und der Wahl der File-Globbing³ ab. Wahrscheinlich ist zu Beginn Ihr Profil wesentlich größer. Wenn Sie mehr Übung im Umgang mit dem Assistenten bekommen, werden Sie auch die Dialoge so beantworten, dass die entstehenden Profile kompakter werden. Natürlich können Sie auch anschließend das Profil editieren und mit `ncapparmor reload` die Profile neu laden.

Das Profil wird nun geladen, sobald als URI `/phpsysinfo/` im Browser ausgewählt wird.

Eine andere und möglicherweise einfachere Variante der Administration der Subprofile in Apache wird in Abschnitt 8.2.1 beschrieben.

³ Als *Globbing* bezeichnet man die Verwendung von *Wildcards* bei der Angabe von Dateinamen. AppArmor erlaubt die Verwendung von `?`, `*` und `**`.