

SMF und Sicherheit



Sicherheit

Webdienste, Webapplikationen, Datenbanken,
Emailsysteme, ...

Sicherheit erfordert tiefgehendes Wissen sowohl auf der Seite des Betriebssystems als auch auf der Seite der Anwendung.

Anwendungsbetreuer = root ?

Sicherheitsprobleme

Es gibt viele Facetten, einige davon sind

- Konfigurationsfehler
- Exploits
- Bufferoverflow
- Denial-of-Service
- Exhaustion
- ...

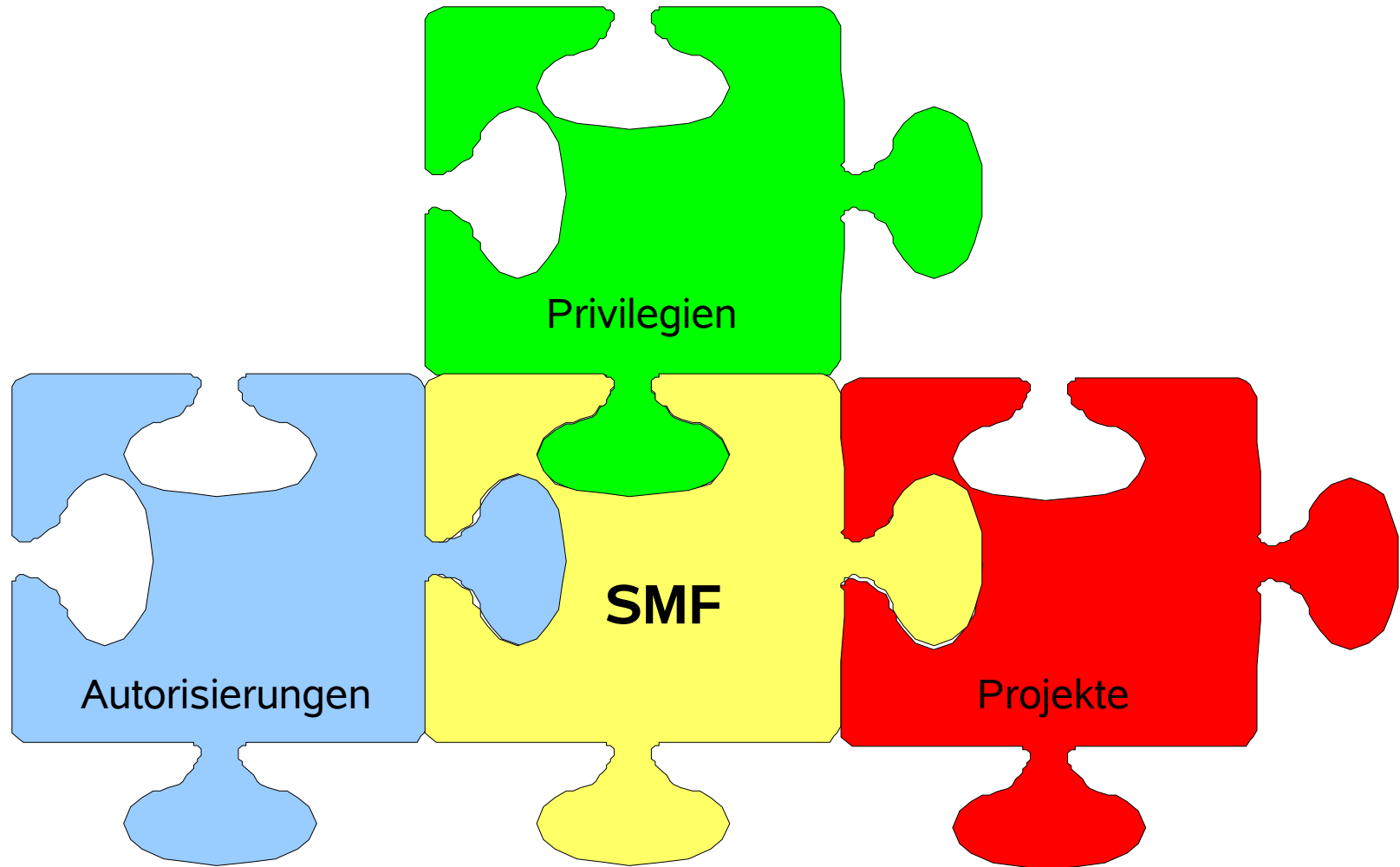
Kein 100% Schutz! => Schadensbegrenzung!

Solaris 10 Features

Die Auswirkung eines „Lecks“ kann durch Einsatz folgender Solaris 10 Technologien und Features minimiert werden.

- Privilegien (Rechte auf Kernelebene)
- RBAC (Autorisierungen und Profile)
- Projekte (Ressource-Management)
- Virtualisierung (Zonen)

Integration der Komponenten



Inhaltsverzeichnis

1. SMF

- Architektur
- Vorteile der SMF
- Manifeste und Profile
- Überblick der SMF-Autorisierungen

2. RBAC

- Einführung
- ♦ Überblick der Konfiguration
- ♦ Integration Autorisierungen in der SMF

Inhaltsverzeichnis

3. Privilegien

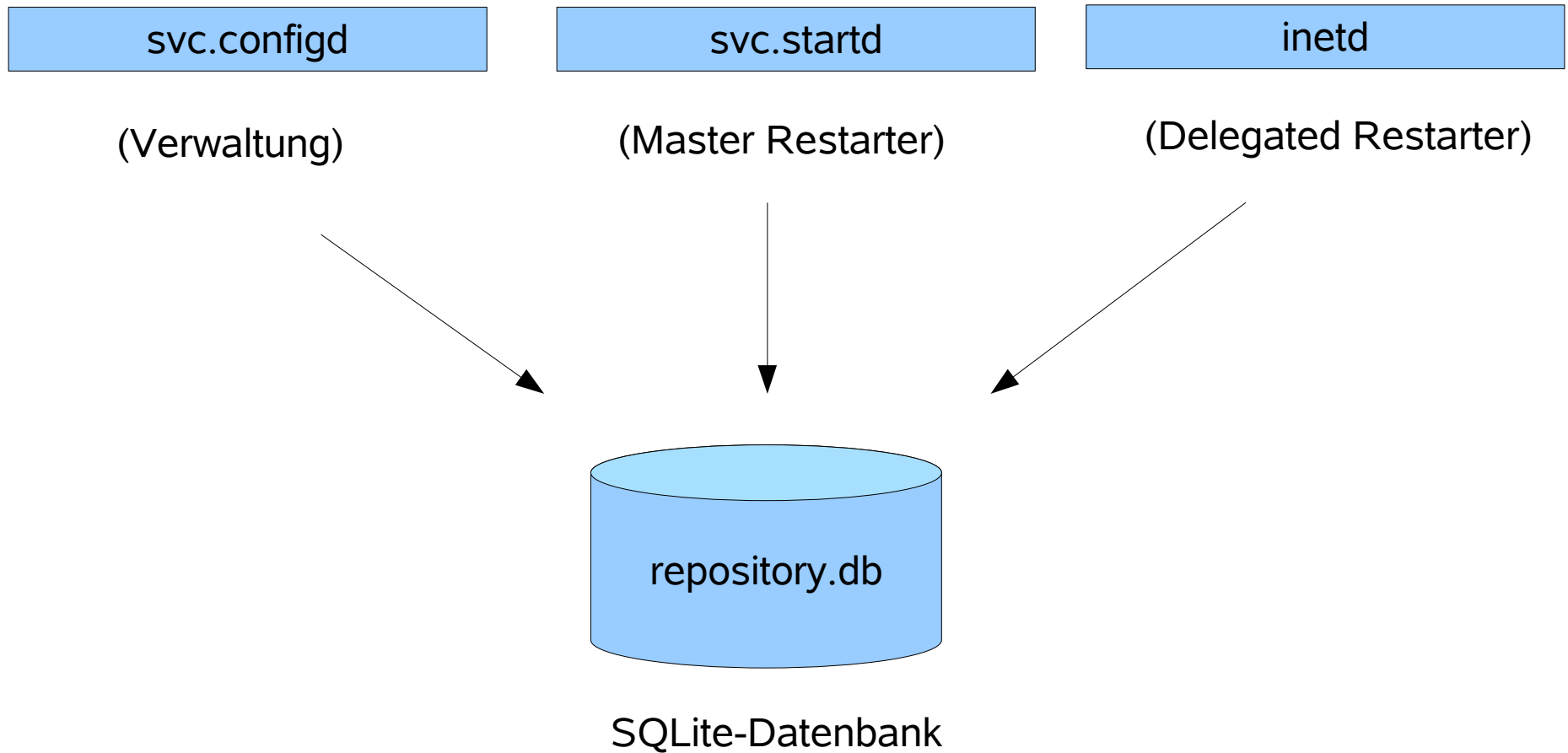
- Einführung
- Privilegien-Sets
- Zuweisung von Privilegien in der SMF

4. Projekte

- Einführung ins Ressourcenmanagement
- Projekte definieren
- SMF + Projekte

1. SMF

SMF - Architektur



Vorteile der SMF

Im Vergleich zu klassischen RC-Skripten bietet die SMF u.a. folgende Vorteile:

- Zentrale, vereinheitlichte Verwaltung
- Starten gemäß Abhängigkeiten
- Timeouts
- Statusüberwachung
- Automatischer Neustart bei Fehler
- Logdateien pro Instanz
- Paralleles Starten

Vorteile der SMF

Dienste werden vereinheitlicht beschrieben, was eine einfache Konfiguration erlaubt.

- ♦ Einfache Integration mit anderen Komponenten:
Projekte, Resourcepools, ...
- ♦ Direkte Angabe von Umgebungsparametern:
UID, GID, Privilegien, Arbeitsverzeichnis, ...
- ♦ Konfigurationssnapshots
Initial, Last-import, Started, Running, ...

Manifeste

Manifeste – Abstrakte Beschreibung eines Dienstes

- Start-/Stopmethoden
- Abhängigkeiten
- Timeouts
- Umgebungsparameter
- Statusüberwachung
- ...

Beispiel: Manifest utmp.xml

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM
"/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<!--
Copyright 2005 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms. -->

<service_bundle type='manifest' name='SUNWcsr:utmpd'>
  <service name='system/utmp' type='service' version='1'>
    <create_default_instance enabled='true' />
    <single_instance/>
```

... Manifest utmp.xml

```
<dependency  
  name='milestone'  
  grouping='require_all'  
  restart_on='none'  
  type='service'>  
  <service_fmri value='svc:/milestone/sysconfig' />  
</dependency>
```

```
<dependent  
  name='utmpd_multi-user'  
  grouping='optional_all'  
  restart_on='none'>  
  <service_fmri value='svc:/milestone/multi-user' />  
</dependent>
```

... Manifest utmp.xml

```
<exec_method type='method' name='start'  
  exec='/lib/svc/method/svc-utmpd' timeout_seconds='60' />
```

```
<exec_method type='method' name='stop' exec=':kill'  
  timeout_seconds='60' />
```

```
<stability value='Unstable' />
```

```
<template>  
  <common_name>  
    <loctext xml:lang='C'> utmpx monitoring </loctext>  
  </common_name>
```

... Manifest utmp.xml

```
<documentation>  
  <manpage title='utmpd' section='1M'  
    manpath='/usr/share/man' />  
  <manpage title='utmpx' section='4'  
    manpath='/usr/share/man' />  
</documentation>  
  
</template>  
  
</service>  
  
</service_bundle>
```

Profile

Profile – Definieren, ob Dienste aktiviert oder deaktiviert sein sollen.

- generic_open.xml
- generic_limited_net.xml
- ...

Werden zum Beispiel vom Befehl **netservices** verwendet.

SMF und Zugriffskontrolle

Die SMF ist RBAC-Compliant. Sie verwendet Autorisierungen, mit denen der Zugriff bezüglich der Verwaltung der Repository und deren Diensten geregelt werden kann.

Autorisierung:

Freischaltung von Berechtigungen, die auf Anwendungsebene überprüft werden, unabhängig von der UID.

Primäre SMF-Autorisierungen

- `solaris.smf.modify`

Erlaubt das Hinzufügen / Modifizieren / Löschen von Diensten oder deren Attributen. Typischerweise für root und Serviceadministratoren vorgesehen.

- `solaris.smf.manage`

Erlaubt es den Status eines Diensten zu ändern (restart, refresh, mark), aber keine Änderung der Konfiguration.

Untergeordnete Autorisierungen

Das Ändern von Eigenschaften kann auch nur für bestimmte Property-Gruppen beschränkt werden.

- `solaris.smf.modify.method`
- `solaris.smf.modify.dependency`
- `solaris.smf.modify.application`
- `solaris.smf.modify.framework`

Dienstspezifische Autorisierungen

- `action_authorization`

Kein Starten und Stoppen erlaubt. Ermöglicht nur restart, refresh, mark, oder clear. Keine Änderung von Attributen.

- `modify_authorization`

Hinzufügen / Ändern und Löschen von Attributen sind erlaubt.

Dienstspezifische Autorisierungen

- `value_authorization`

Nur bestehende Attribute dürfen innerhalb einer Property-Gruppe modifiziert werden.

- `read_authorization`

Erlaubt das Lesen von Attributen. (Nur für die Gruppe Application von Bedeutung)

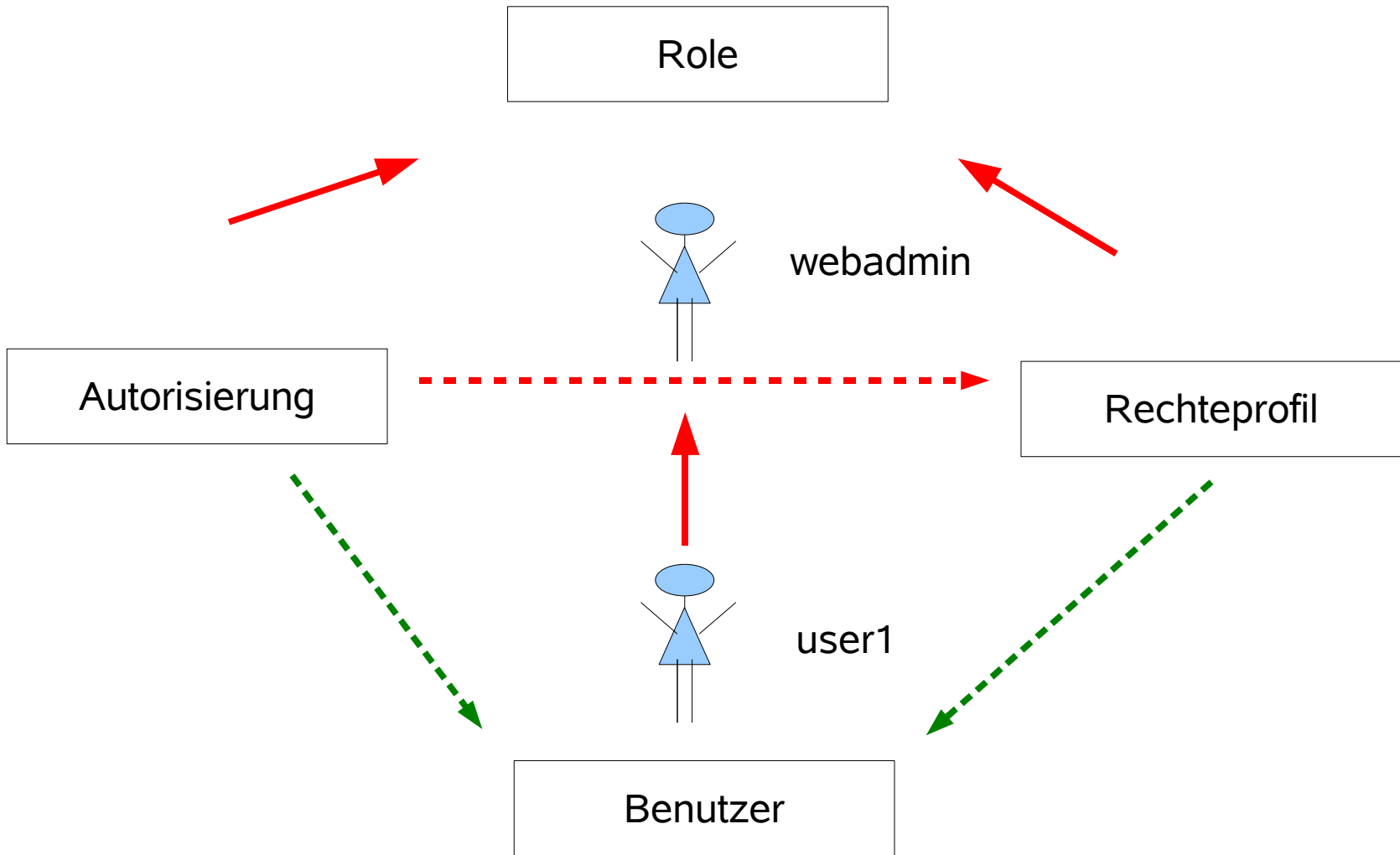
(Siehe auch Manualpage zu `smf_security(5)`.)

2. RBAC

Was ist RBAC?

- Rollenbasiertes Zugriffskontrolle
- 1992 beschrieben von Ferraiolo und Kuhn
- 2000 von NIST als vereinheitlichtes Modell veröffentlicht
- ANSI-Norm 359-2004 (2004)
- Unter Solaris seit Solaris 8 implementiert

RBAC-Modell



Rollen

- Sicherheitsbenutzer -> kein Login erlaubt
- Nur autorisierte Benutzer dürfen mittels „su“ wechseln.
- Darf nur Befehle ausführen, die in seinen Profilen gesetzt sind.

4-Augen Konzept

user1 darf wechseln, kennt das Kennwort nicht;
user2 darf nicht wechseln, kennt das Kennwort.

Autorisierungen

- Definiert in `/etc/security/auth_attr`
- Berechtigung auf Applikationsebene
- Sind hierarchisch aufgebaut, ähnlich DNS

Beispiel:

```
solaris.admin.usermgr.read  
solaris.admin.usermgr.write
```

(Rechte-)Profile

- Definiert in `/etc/security/prof_attr`
- Mehrere Dutzend vorgefertigte Profile
- Autorisierungen als Profil-Attribute definierbar
- Zu einem Profil sind in der Regel Befehle assoziiert, sowie deren Befehls-Attribute zur Ausführung (UID, EUID, GID, EGID, Privilegien)

(Vergleichbar mit sudo)

RBAC Konfigurationsdateien

- `/etc/user_attr`
Erweiterte Benutzerattribute
- `/etc/security/auth_attr`
Autorisierungen
- `/etc/security/prof_attr`
Profile
- `/etc/security/exec_attr`
Befehle zu den Profilen
- `/etc/security/policy.conf`
Defaultwerte: Autorisierungen, Profile

Beispiel Rollenuser

```
root@groo:~# roleadd -m webadmin
```

```
root@groo:~# passwd webadmin
```

```
...
```

```
root@groo:~# su - user1
```

```
user1@groo:/export/home/user1$ su - webadmin
```

```
Password:
```

```
Roles can only be assumed by authorized users
```

```
su: Sorry
```

```
root@groo:~# usermod -R webadmin user1
```

```
root@groo:~# roles user1
```

```
webadmin
```

Benutzerdefinierte Autorisierung

```
root@groo:/etc/security# tail -6 /etc/security/auth_attr
```

```
solaris.system.power.suspend.ram:::Suspend to
```

```
RAM:::help=SysPowerMgmtSuspendToRAM.html
```

```
solaris.system.shutdown:::Shutdown the System:::help=
```

```
SysShutdown.html
```

```
mycompany.*:::Benutzerdefinierte Autorisierungen::
```

```
mycompany.grant:::Benutzerdef. Autorisierungen vergeben::
```

```
mycompany.web.admin:::Admin Webserver::
```

```
mycompany.web.modify:::Webserver Konfiguration::
```

Zuweisung der Autorisierung

```
root@groo:~# grep root: /etc/user_attr
root:::type=role;auths=solaris.*,solaris.grant,mycompany.*,mycom
pany.grant;profiles=All;lock_after_retries=no;min_label=admin_low;
clearance=admin_high
```

```
root@groo:~# rolemod -A mycompany.web.admin webadmin
```

```
root@groo:~# grep webadmin /etc/user_attr
webadmin:::type=role;auths=mycompany.web.admin;profiles=All
user1:::type=normal;roles=webadmin
```

Service modifizieren

Beispiel: Webserver Apache 2.2

`svc:/network/http:apache22`

```
root@groo:~# svccfg -s apache22 setprop  
general/action_authorization = astring: mycompany.web.admin
```

Auszug Manifest

```
<property_group name='general' type='framework'>  
  <propval name='action_authorization' type='astring'  
value='mycompany.web.admin' />
```

Verwendung: action_authorization

```
root@groo:~# su – webadmin
```

```
$ svcs apache22
```

```
STATE      STIME    FMRI
online     21:29:29 svc:/network/http:apache22
```

```
$ /usr/sbin/svcadm restart apache22
```

```
$ svcs apache22
```

```
STATE      STIME    FMRI
online     21:30:03 svc:/network/http:apache22
```

```
$ /usr/sbin/svcadm disable apache22
```

```
svcadm: svc:/network/http:apache22: Permission denied.
```

Serviceparameter ändern

Konfigurationsparameter eines Dienstes können auch in der Repository stehen.

Beispiele:

Service	Parameter
syslog	config/log_from_remote
sendmail	config/local_only
smcwebserver	options/tcp_listen

=> Anwendungsbetreuer braucht SMF-Rechte

Parameter im Manifest

Das Setzen von Konfigurationsparametern eines Dienstes werden über `property_group` gesetzt.

Auszug vom Manifest: `system-log.xml`

```
<property_group name='config'  
                type='application'>  
  
<propval name='log_from_remote'  
          type='boolean' value='false' />  
</property_group>
```

Autorisierung: value_authorization

Autorisierung für die Property-Gruppe `start` und `httpd` setzen:

```
root@groo:~# svccfg -s apache22 setprop  
start/value_authorization = astring: mycompany.web.modify  
root@groo:~# svccfg -s apache22 setprop  
httpd/value_authorization = astring: mycompany.web.modify  
root@groo:~# svccfg -s apache22 setprop start/working_directory  
= astring: :default
```

```
root@groo:~# rolemod -A  
mycompany.web.admin,mycompany.web.modify webadmin
```

Verwendung: value_authorization

```
root@groo:~# su - webadmin
```

```
$ /usr/sbin/svccfg -s apache22 setprop start/working_directory =  
/var/tmp
```

```
$ /usr/sbin/svccfg -s apache22 setprop start/working_directory =  
:default
```

```
$ /usr/sbin/svccfg -s apache22 delprop start/working_directory  
svccfg: Permission denied.
```

3. Privilegien

Problemstellung

Apache-Konzept zur Schadensbegrenzung:
httpd-Master läuft unter root und startet Co-Workers, die die eigentliche Arbeit machen.

```
root@groo:~# ps -e -o user,pid,fname |grep httpd |sort +2n
  root 2008 httpd
webservd 2009 httpd
webservd 2010 httpd
webservd 2011 httpd
webservd 2012 httpd
webservd 2013 httpd
webservd 2014 httpd
```

Ziel => httpd-Master: kein root

Privilegien - Übersicht

Privilegien sind Rechte auf Kernelebene. Es existieren Dutzende verschiedener Privilegien. Auflisten mit Hilfe von `ppriv`.

```
root@groo:/# ppriv -l
contract_event
contract_identity
contract_observer
cpc_cpu
dtrace_kernel
dtrace_proc
dtrace_user
file_chown
file_chown_self
...
```

Basis-Privilegien

Standardmäßig erhält jeder Benutzer die Basis-Privilegien:

```
user1@groo:/export/home/user1$ ppriv $$
```

```
762: -sh
```

```
flags = <none>
```

```
  E: basic
```

```
  I: basic
```

```
  P: basic
```

```
  L: all
```

Über `/etc/security/policy.conf` können Defaultprivilegien gesetzt werden:

```
#PRIV_DEFAULT=basic
```

Detailbeschreibung von Privilegien

```
user1@groo:/export/home/user1$ ppriv -v -l basic
```

file_link_any

Allows a process to create hardlinks to files owned by a uid different from the process' effective uid.

proc_exec

Allows a process to call `execve()`.

proc_fork

Allows a process to call `fork1()/forkall()/vfork()`

proc_info

Allows a process to examine the status of processes other than those it can send signals to. Processes which cannot be examined cannot be seen in `/proc` and appear not to exist.

proc_session

Allows a process to send signals or trace processes outside its session.

Ermitteln notwendiger Privilegien

Über den Debug-Modus von ppriv lässt sich meistens sehr einfach ermitteln, welche Privilegien benötigt werden.

```
user1@groo:/export/home/user1$ ppriv -D -e cat /etc/shadow
cat[786]: missing privilege "file_dac_read" (euid = 102, syscall =
225) needed at zfs_zaccess+0x1dc
cat: /etc/shadow: Permission denied
```

```
user1@groo:/export/home/user1$ ppriv -v -l file_dac_read
file_dac_read
```

Allows a process to read a file or directory whose permission bits or ACL do not allow the process read permission.

Ermitteln notwendiger Privilegien

Alternativ lässt sich auch **truss** einsetzen, um die notwendigen Privilegien zu ermitteln.

```
user1@groo:/export/home/user1$ truss cat /etc/shadow 2>&1 |  
grep „Err.*\[,
```

```
open64("/etc/shadow", O_RDONLY)          Err#13 EACCES  
[file_dac_read]
```

Privilegien-Sets

Privilegien werden über Sets zugeordnet. Es existieren 4 Sets:

- **Effective Set**

Definiert die Privilegien, die ein Prozess zur Laufzeit besitzt.

- **Inheritable Set**

Definiert, was ein Prozess vom Elternprozess erben kann. Die tatsächliche Form hängt von der Art des Aufrufes statt.

...Privilegien-Sets

fork() - erbt alle Privilegien, neue können hinzugefügt werden.

exec() - erbet alle Privilegien, es können keine neuen Privilegien hinzugefügt werden.

- **Permitted Set**

Definiert die Privilegien, die verfügbar sind.

Entweder durch Erben oder durch Zuweisung.

- **Limited Set**

Die äußere Begrenzung der verfügbaren Privilegien. Standardmäßig: All

Setzen von Privilegien

Beispiel zum Setzen von Privilegien in den Sets:

```
root@groo:/# ppriv -s EIP=basic,file_dac_read,!file_link_any 841
```

```
user1@groo:/export/home/user1$ ppriv $$
```

```
841: -sh
```

```
flags = <none>
```

```
E: basic,file_dac_read,!file_link_any
```

```
I: basic,file_dac_read,!file_link_any
```

```
P: basic,file_dac_read,!file_link_any
```

```
L: all
```

```
user1@groo:/export/home/user1$ cat /etc/shadow
```

```
... <Ausgabe> ...
```

SMF-Service mit Privilegien

Ein Beispiel für einen Service, der mit Privilegien arbeitet, ist sar. Auszug Manifest:

```
<exec_method type='method' name='start'  
  exec='/lib/svc/method/svc-sar %m'  
  timeout_seconds='60'>  
  <method_context>  
    <method_credential user='sys' group='sys'  
      privileges='basic,file_dac_write' />  
  </method_context>  
</exec_method>
```

Service apache22 modifizieren

Nun wird der Service modifiziert, dass beim Starten entsprechend als Benutzer webservd verwendet wird.

```
root@groo:/# svccfg -s apache22 setprop start/user = astring:  
webservd
```

```
root@groo:/# svccfg -s apache22 setprop start/group = astring:  
webservd
```

```
root@groo:/# svccfg -s apache22 setprop start/privileges = astring:  
basic,\!proc_session,\!proc_info,\!file_link_any,net_privaddr
```

```
root@groo:/# svccfg -s apache22 setprop start/limit_privileges =  
astring: :default
```

...Service apache22 modifizieren

Wenn man schon dabei ist, kann man gleich auch noch andere Parameter setzen, die wir später verwenden wollen:

```
root@groo:/# svccfg -s apache22 setprop start/use_profile =  
boolean: false
```

```
root@groo:/# svccfg -s apache22 setprop start/supp_groups =  
astring: :default
```

```
root@groo:/# svccfg -s apache22 setprop start/working_directory =  
astring: :default
```

```
root@groo:/# svcadm disable apache22
```

...Service apache22 modifizieren

Darauf achten, dass die notwendigen Dateien / Verzeichnisse auch dem entsprechenden Benutzer gehören:

```
root@groo:/# chown -R webadmin:webserverd /etc/apache2
root@groo:/# chown -R webserverd:webserverd /var/apache2/2.2/logs
root@groo:/# svcadm enable apache22
```

Hinweis: Unter Umständen müssen noch in der `httpd.conf` die Einstellungen für **Lockfile** und **PidFile** angepasst werden.

Überprüfung

```
root@groo:/# ps -e -o user,pid,fname |grep httpd | sort +2n
```

```
webservd 1055 httpd
```

```
webservd 1056 httpd
```

```
webservd 1057 httpd
```

```
webservd 1058 httpd
```

```
webservd 1059 httpd
```

```
webservd 1060 httpd
```

```
webservd 1061 httpd
```

```
root@groo:/# ppriv 1061
```

```
1061: /usr/apache2/2.2/bin/httpd -k start
```

```
flags = <none>
```

```
E: basic,!file_link_any,net_privaddr,!proc_info,!proc_session
```

```
I: basic,!file_link_any,net_privaddr,!proc_info,!proc_session
```

```
P: basic,!file_link_any,net_privaddr,!proc_info,!proc_session
```

```
L: all
```

4. Projekte

Was sind Projekte?

Sicherheit betrifft das ganze System. Wird ein Service kompromittiert, sind dessen Auswirkungen auf das Betriebssystem und andere Prozesse zu limitieren.

Dies geschieht über die Kontrolle der genutzten Ressourcen (Memory, CPU, ...).

Projekte sind die Objekte zum Umsetzen des Resource-Managements.

Techniken des Resource- Managements

Es gibt drei unterschiedliche Ansätze, die im Normalfall auch kombiniert werden:

- **Partitionierung** (Aufteilen einer Ressource)
- **Limitierung** (Beschränkung nach oben)
- **Gewichtung** (Setzen von Verhältnismäßigkeit)

Partitionierung mit CPU-Pools

Als Beispiel für die 1. Technik wäre die Verwendung eines CPU-Pools.

Mit einem Pool wird ein Prozessorset assoziiert. Parameter und Anzahl der CPU's können dynamisch geändert werden.

Pool_Facility einrichten

Standardmäßig ist die Pool-Facility deaktiviert.

```
root@groo:~# svcs pools
STATE      STIME    FMRI
disabled   10:43:44 svc:/system/pools:default
root@groo:~# pooladm -e                (pool-facility aktivieren)
root@groo:~# poolcfg -dc 'create pset pset_webzone (uint
pset.max = 2)'                          (erstelle Prozessorset)
root@groo:~# poolcfg -dc 'create pool pool_webzone (string
pool.scheduler="FSS")'                  (erstelle Pool)
root@groo:~# pooladm -s                  (Konfiguration abspeichern)
```

Hinweis: Das Zuweisen einer CPU zum Pset und das assoziieren mit dem Pool entfällt in dem Beispiel, da nur 1 CPU verfügbar ist.

Pool-Konfiguration

```
root@groo:~# poolcfg -dc info
system default
  string system.comment
  int system.version 1
  boolean system.bind-default true
  string system.pool.objectives wt-load
...
pool pool_webzone
  int pool.sys_id 2
  boolean pool.active true
  boolean pool.default false
  string pool.scheduler FSS
  int pool.importance 1
  string pool.comment
  pset pset_default
```

...Pool-Konfiguration

```
pset pset_default
  int pset.sys_id -1
  boolean pset.default true
  uint pset.min 1
  uint pset.max 65536
  string pset.units population
  uint pset.load 52
  uint pset.size 1
  string pset.comment

cpu
  int cpu.sys_id 0
  string cpu.comment
  string cpu.status on-line
```

Pools und Projekte

Ein Pool muss nicht direkt dem Service als Attribut zugeordnet werden.

Alternativ kann der Pool auch implizit als Attribut eines Projekts definiert werden.

```
project.pool=pool_webzone
```

Limitierung und Gewichtung

Die anderen 2 Techniken des Resource-Management werden unter Solaris mit Hilfe von Resourcecontrols umgesetzt.

Ein Liste der verfügbaren Resourcecontrols erhält man mit Hilfe des Befehls:

```
#rctladm -l
```

Siehe auch: `man resource_controls`

Kontrollparameter

Kontrollparameter können auf verschiedenen Ebenen eingreifen.

- Prozessebene

Prozessspezifisch

- Taskebene

Gültig für den Prozess und dessen Kindprozesse

- Projektebene

Fasst verschiedene Prozesse/Tasks zusammen

Beispiele für Resource-Controls

- Prozess:
`process.max-address-space`
- Task:
`task.max-lwps`
- Projekt:
`project.max-shm-memory`

Resource-Controls, die im Namen ein max oder cap führen, gehören zu den limitierenden Parametern, diejenigen, die share im Namen führen, gehören zu den Gewichtungen.

Werte

Jedem Resource-Control Parameter können eine oder mehrere Werte zugewiesen werden. Eine Wertzuweisung enthält 3 Angaben:

(priv,200,deny)

Privileisierungsgrad Grenzwert Aktion

Beispiel:

`process.max-address-space=(priv,200mb,deny)`

Projekte verwalten

Projekte werden über die Datei /etc/project verwaltet. Es können Benutzerspezifische, Gruppenspezifische oder allgemeine Projekte definiert werden. Hier der Überblick über die standardmäßig existierenden Projekte:

```
root@groo:/# cat /etc/project
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
```

Projektdetails

Die Details über ein Projekt lassen sich mit dem Befehl `projects` anzeigen:

```
root@groo:/# projects -l group.staff
group.staff
  projid : 10
  comment: ""
  users  : (none)
  groups : (none)
  attribs:
```

Neues Projekt

```
root@groo:/# projadd -c 'Web Server' -K 'process.max-address-  
space=(priv,200mb,deny)' \  
> -K 'project.max-shm-memory=(priv,100mb,deny)' \  
> -K 'project.pool=pool_webzone' \  
> -K 'project.max-lwps=(priv,40,deny)' web  
root@groo:/# projects -l web  
web  
  projid : 100  
  comment: "Web Server"  
  users  : (none)  
  groups : (none)  
  attribs: process.max-address-space=(priv,209715200,deny)  
           project.max-lwps=(priv,40,deny)  
           project.max-shm-memory=(priv,104857600,deny)  
           project.pool=pool_webzone
```

SMF-Service + Pool

Als Startattribut dem Service den Pool zuweisen:

```
root@groo:/# svccfg -s apache22 setprop start/resource_pool =  
astring: pool_webzone
```

Service neu starten oder poolbind verwenden.

Definition im Manifest: Siehe auch

[/usr/share/lib/xml/dtd/service_bundle.dtd.1:](#)

```
<method_context resource_pool='pool_webzone' ...>
```

SMF-Service + Projekt

Analog zur Zuweisung eines Pools erfolgt die Zuweisung eines Projektes über den Methodenkontext. Eine direkte Änderung in der Repository erfolgt über svccfg:

```
root@groo:/# svccfg -s apache22 setprop start/project = astring:  
web
```

Manifesteintrag

```
<method_context project='web' >  
<method_credential user='webservd' ... />  
</method_context>
```

Service neu starten

Hinweis: Anscheinend existiert noch in der SMF ein Bug im Zusammenhang mit Projekten. Falls nach einem refresh des Dienstes dieser sich nicht unter dem angegebenen Projekt startet, sollte man dem Dienstkonto das Projekt als Defaultprojekt zuweisen:

```
root@groo:/# usermod -K project=web webservd  
root@groo:/# svcadm disable apache22  
root@groo:/# svcadm enable apache22
```

Kontrolle

```
root@groo:/# ps -e -o user,project,fname,pid |egrep 'PID|httpd'
```

USER	PROJECT	COMMAND	PID
webservd	web	httpd	1567
webservd	web	httpd	1564
webservd	web	httpd	1563
webservd	web	httpd	1566
webservd	web	httpd	1568
webservd	web	httpd	1565
webservd	web	httpd	1562

```
root@groo:/# poolbind -q 1567
```

```
1567 pool_webzone
```

Projektauslastung

```
root@groo:/# prstat -J
```

```
....
```

PROJID	NPROC	SWAP	RSS	MEMORY	TIME	CPU	PROJECT
10	39	266M	350M	43%	0:25:47	14%	group.staff
0	43	94M	92M	11%	0:14:57	10%	system
1	2	2392K	6528K	0,8%	0:00:08	3,0%	user.root
100	7	77M	30M	3,7%	0:00:02	0,1%	web
3	1	248K	2932K	0,4%	0:00:00	0,0%	default

```
root@groo:/# prctl -n project.max-shm-memory -i project web
```

```
project: 100: web
```

NAME	PRIVILEGE	VALUE	FLAG	ACTION	RECIPIENT
project.max-shm-memory	privileged	100MB	-	deny	-