Linux Kernel Auditing

Postbank
Eine Bank fürs Leben.

Kernel Auditing
SLAC 2016

Benjamin Knust
Unix / Linux Engineer
8 Jahre Linux Consulting
3 Jahre Unix / Linux Engineer bei der Postbank Systems AG

Tauchsport in der Freizeit...
… habe leider wenig Freizeit... aber eine tolle Frau und zwei awesome kids ☺ die alle Zeit bekommen :-)

Produktfelder für alle finanziellen Bedürfnisse der Kunden

Die Markenarchitektur der Postbank

Vertriebsmarke für unsere **Privat-, Firmen- und Geschäftskunden**

Postbank

Produzentenmarken

Bausparen & Baufinanzierung	Immobilienfinanzierung & Konsumentenkredite
BHW Gemeinsam für Ihr Zuhause	DSL Bank Partnerbank der Finanzdienstleister

Einige Zahlen (Auszug)

Filialen

- 5.600 Standorte, davon 1.100 Postbank Finanzcenter
- > 1 Mio. Kundenkontakte pro Tag

Mobiler Vertrieb

- Über 3.000 kompetente Berater und Makler (HGB-Partner)
- Ca. 1,5 Mio. Kundenberatungen pro Jahr

Direktvertrieb

- 24/7 Call Center
- über 9 Mio. Kundenkonten zum Onlinebanking frei geschaltet
- ~ 3,500 Postbank Geldautomaten (inkl. Shell) + Cash Group
- ~ 1,750 Kontoauszugsdrucker / Service Terminals

Transaktions-Banking

- Verarbeitung von mehr als 7 Milliarden Zahlungsverkehrstransaktionen pro Jahr (Inklusive Drittkundengeschäft z.B. Deutsche Bank, HypoVereinsbank, HSH Nordbank)
- Verarbeitung von rund 2 Millionen Darlehn (Inklusive Drittkundengeschäft mit DEVK, SwissLife, KfW)

Stand: Q1 2014

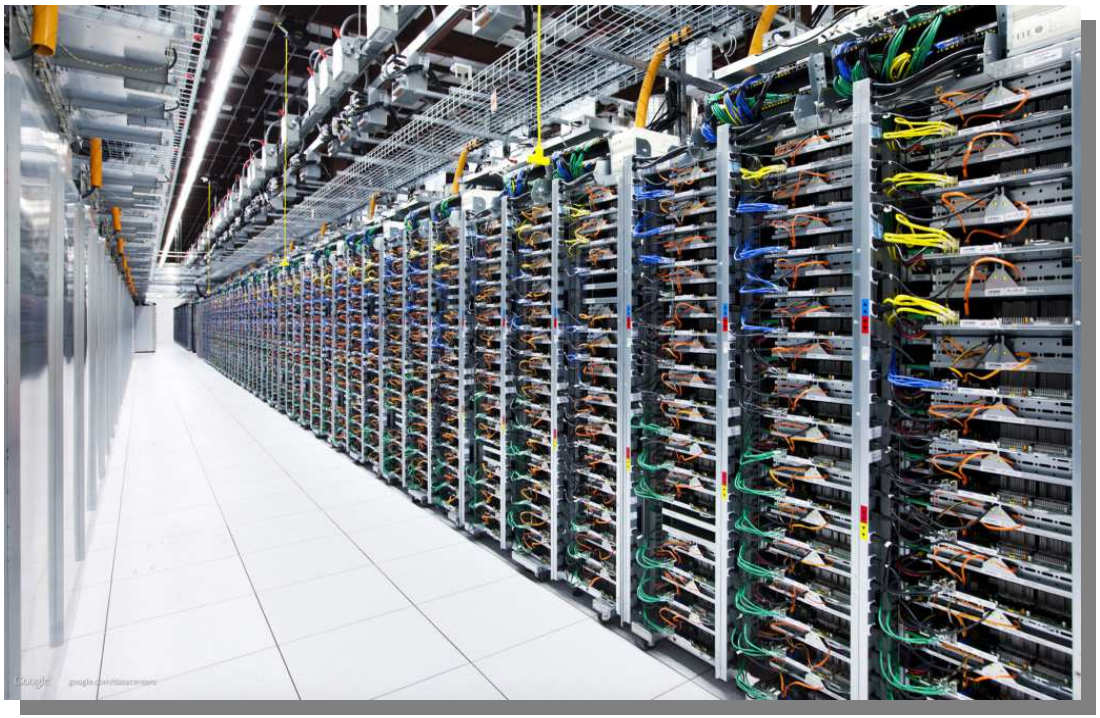
Verwendete „Commodity Hardware“ (SAP & eGrid 2.0)

DL 560 GEN 8 (jeder Cluster-Node)

- 4 * Xeon E5-4640 CPU (2,4 GHz/8-Core/20 MB/95 W) → 32 Core
- 48 DIMM-Sockets – bestückt mit bis zu 1,5 TB RAM
- Rackspace 2HE, Redundante Stromversorgung
- 6 * FibreChannel 8GBit (2 – 4 Disk / 2 Tape)
- 4 * 10GBit Ethernet (2 Interconnect, 2 LAN)
- 1 * ILO-Management

Was sagen uns die Zahlen ??

Viele Server auf denen wir uns austoben dürfen ?!?!? :-)



… Naja nicht so ganz ...

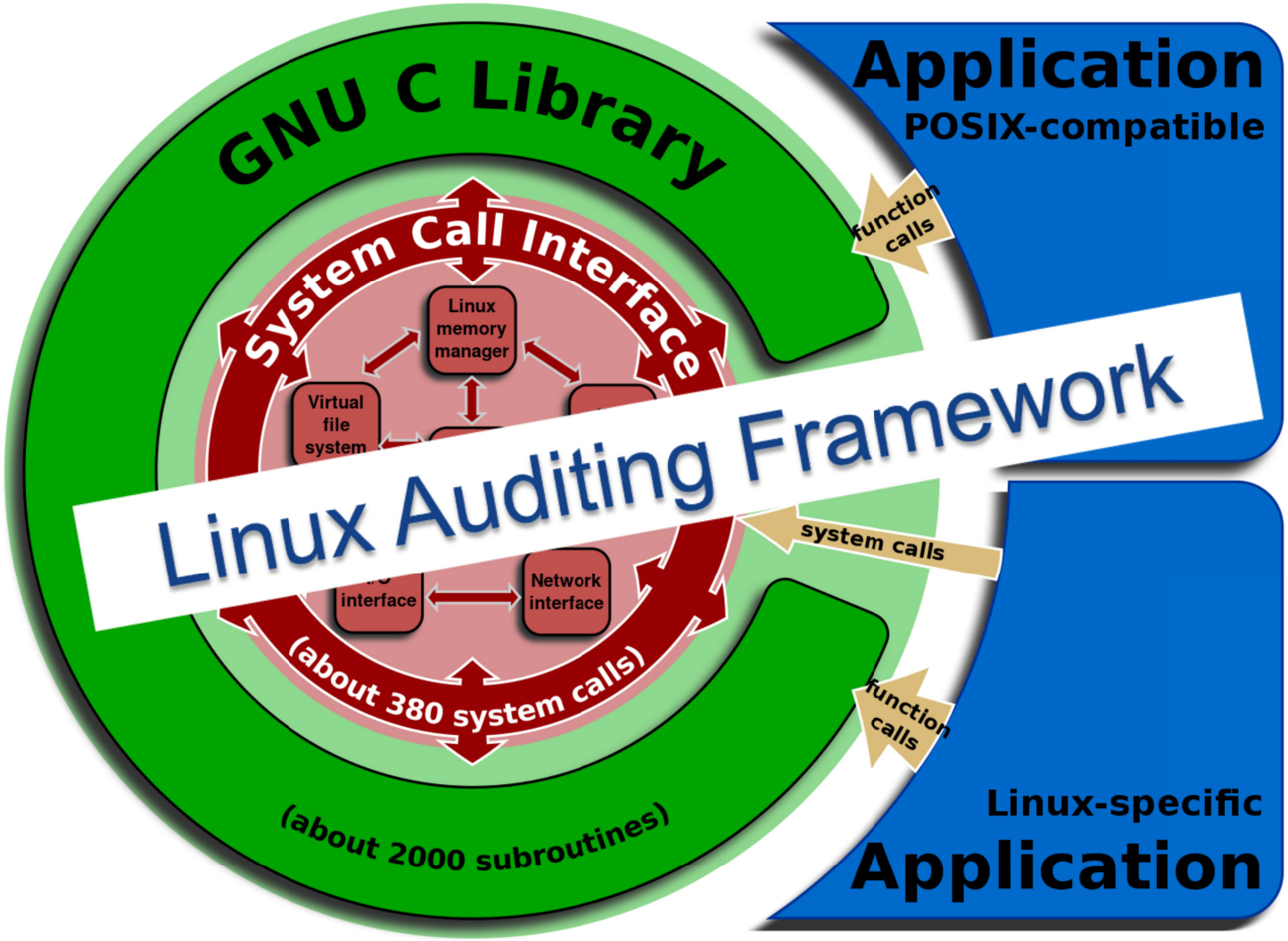
Information Security World: PCI Data Security Standard
https://www.flickr.com/photos/purpleslog/2906633775/
https://creativecommons.org/licenses/by/2.0/

Fire training of local fire department
https://www.flickr.com/photos/rauckhaus/15305196110/
https://creativecommons.org/licenses/by/2.0/

Einführung Kernel Auditing

Anforderung an eine „Echtzeit Alamierung“ bei “schreibendem Dateizugriff“...auf schutzbedürftige Objekte...

...Umzusetzen auf AIX, Solaris & Linux....



https://commons.wikimedia.org/wiki/File:Linux_kernel_System_Call_Interface_and_glibc.svg
This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license

Einführung Kernel Auditing

Light-weight Auditing Framework

From: **Rik Faith <faith@redhat.com>**

To: **linux-kernel@vger.kernel.org**

Subject: **[PATCH][RFC] Light-weight Auditing Framework**

Date: **Mon, 1 Mar 2004 11:28:45 -0500**

This note describes a patch against 2.6.4-rc1-bk2 that provides a low-overhead system-call auditing framework for Linux that is usable by LSM components (e.g., SELinux).

https://lkml.org/lkml/2004/3/1/125

Einführung Kernel Auditing

Hauptsächlich im Kernel Space

…/kernel/audit.c

…/kernel/auditsc.c

Rest im Userspace

/sbin/auditd

audit-audispd-plugins-1.8-0.34.27

audit-libs-1.8-0.34.26

audit-1.8-0.34.26

yast2-audit-laf-2.17.10-0.2.18

audit-libs-32bit-1.8-0.34.26

Einführung Kernel Auditing

Linux audit helps make your system more secure by providing you with a means to analyze what is happening on your system in great detail. ***It does not***, however, ***provide additional security itself***—**it does not protect your system** from code malfunctions or any kind of exploits. Instead, audit is useful for tracking these issues and helps you take additional security measures, like AppArmor or SELinux, to prevent them.

(SUSE Security Guide documentation > The Linux Audit Framework > Understanding Linux Audit, https://www.suse.com/documentation/sles-12/book_security/data/cha_audit_comp.html#)

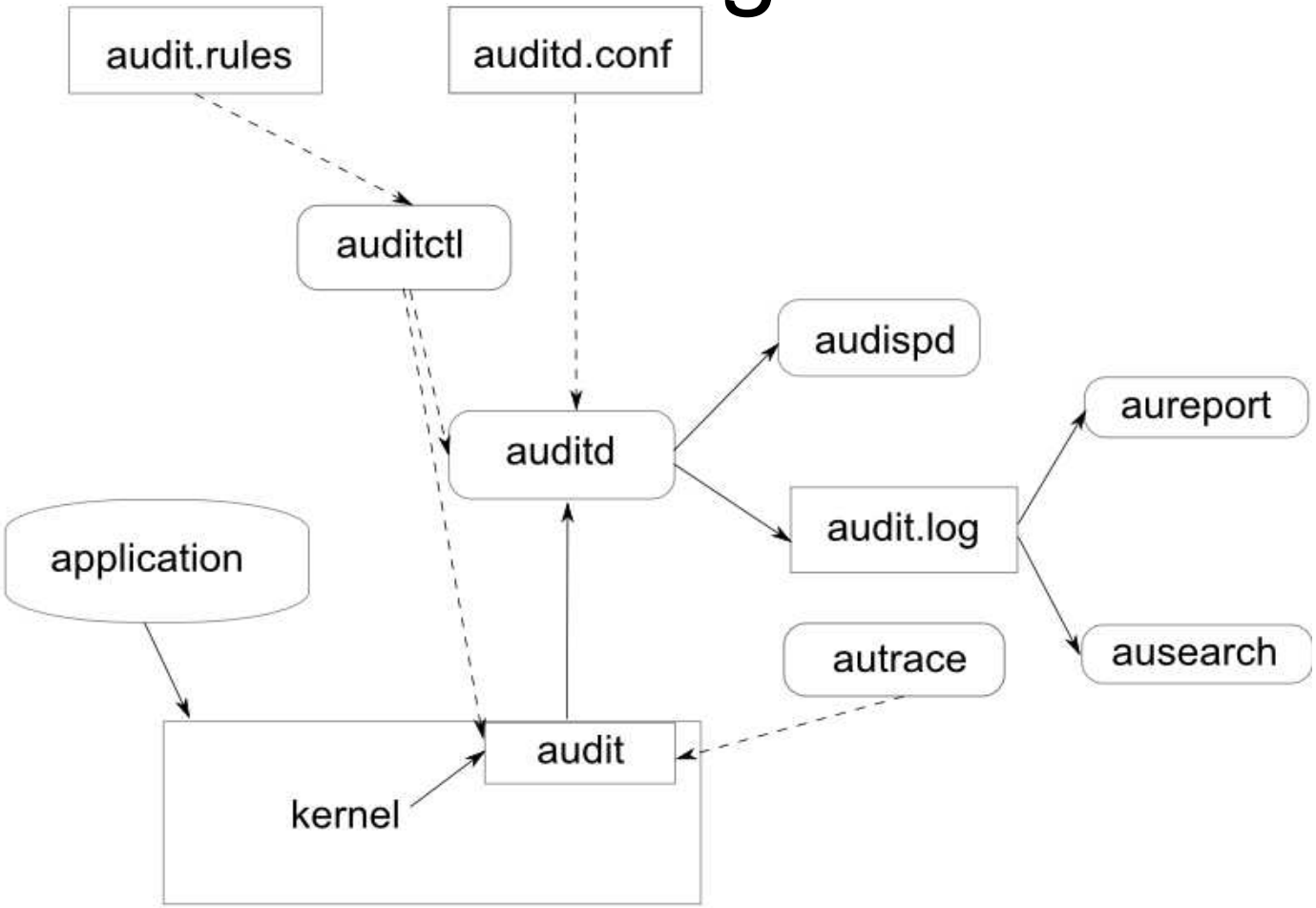
Einführung Kernel Auditing

Audit “mapped” Prozesse auf die User Ids durch die sie gestartet wurden

Audit enthält ein Werkzeugset um Reports zu erstellen und diese zu durchsuchen.

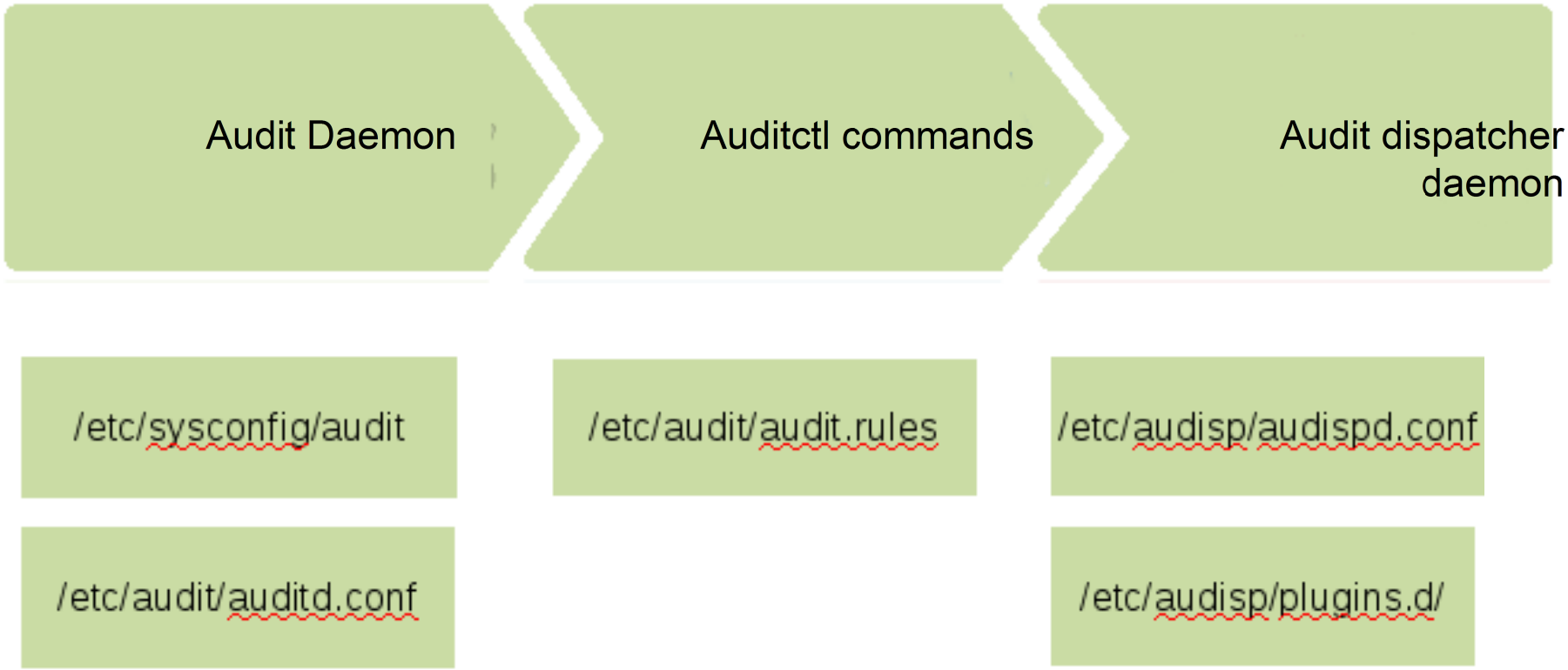
Audit ermöglicht es eigene Regeln zu erstellen und nur diese zu prüfen bzw. zu loggen.

Technisches Layout des Kernel Auditing



Durchgezogene Linien repräsentieren den Datentransfer zwischen Komponenten während gestrichelte Linien die Kontrolle zwischen Komponenten darstellen.

Konfigurationsablauf

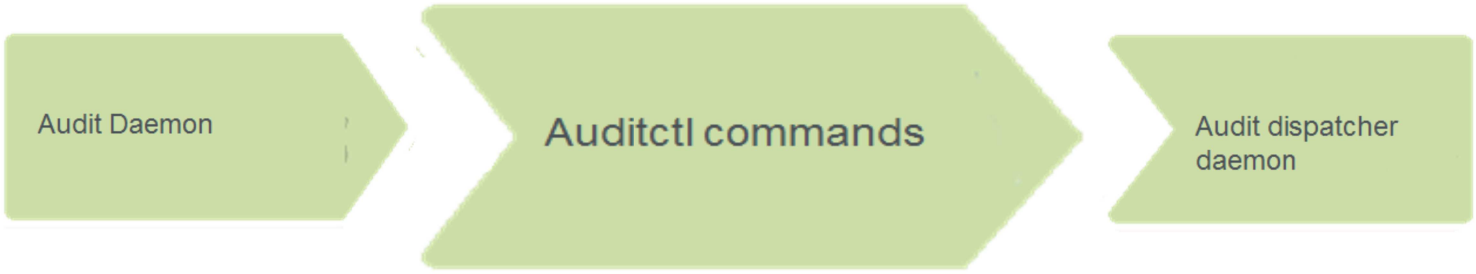


Beispiel: /etc/audit/auditd.conf

```
log_file = /var/log/audit/audit.log
log_format = RAW
log_group = root
priority_boost = 4
flush = INCREMENTAL
freq = 20
num_logs = 5
disp_qos = lossy
```

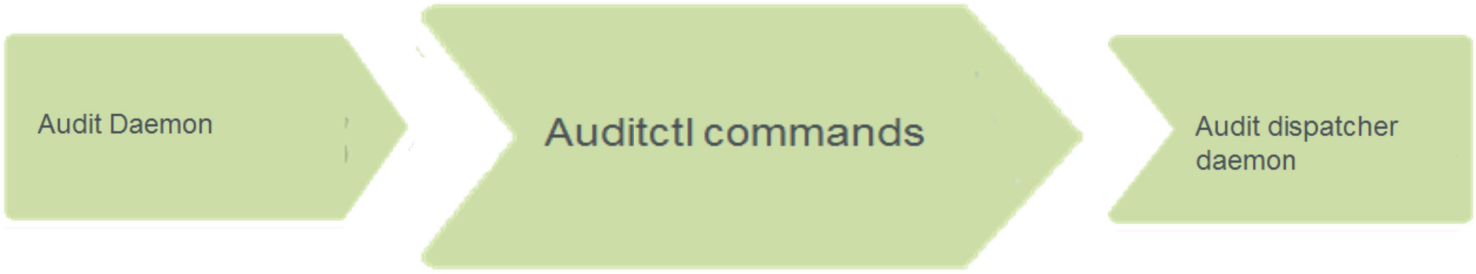

```
dispatcher = /sbin/audispd
name_format = NONE
##name = mydomain
max_log_file = 6
max_log_file_action = ROTATE
space_left = 75
space_left_action = SYSLOG
action_mail_acct = root
admin_space_left = 50
admin_space_left_action =
SUSPEND
disk_full_action = SUSPEND
disk_error_action = SUSPEND
```


```
##tcp_listen_port =
tcp_listen_queue = 5
tcp_max_per_addr = 1
##tcp_client_ports = 1024-
65535
tcp_client_max_idle = 0
```



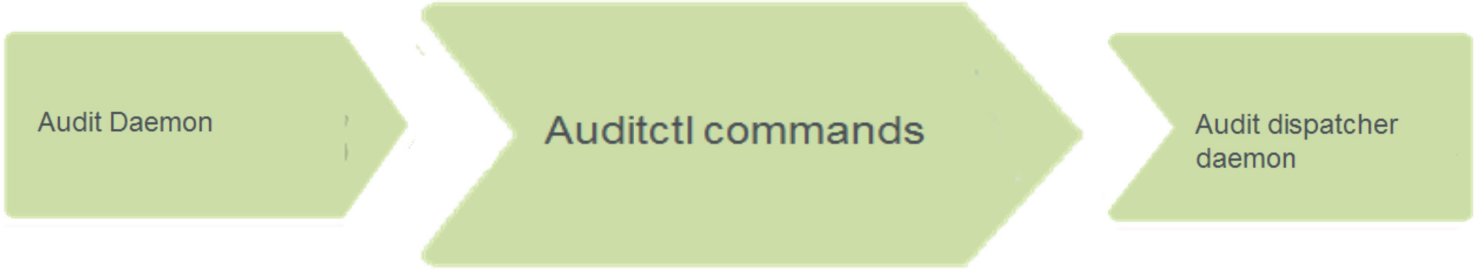
Auditctl Befehle innerhalb der Konfigurationsdatei bestehen aus zwei Abschnitten.

- Hauptbefehlssatz
- Regelsatz



Hauptbefehlssatz

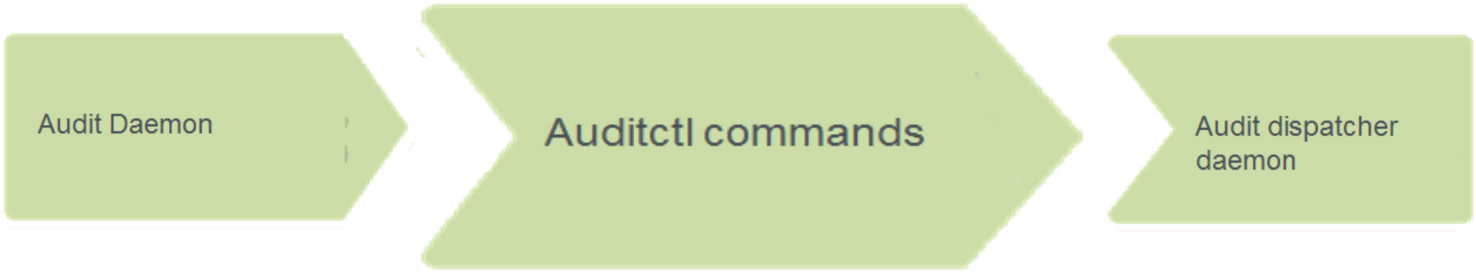
auditctl -e Aktivieren/Deaktivieren/Sperren
auditctl -f Setzt das failure flag
auditctl -r Setzt das rate limit im Kernel
auditctl -b Setzt das backlog Limit
auditctl -s Statusabfrage des Audit Daemons



Regelsatz

Filesystem Regeln – auch als “filewatches“ bezeichnet, erlauben das Überwachen von einzelnen Dateien oder Verzeichnissen auf Scheib-, Lesezugriff oder Attributsänderung

System call Regeln – erlauben das Überwachen von Syscall Aufrufen

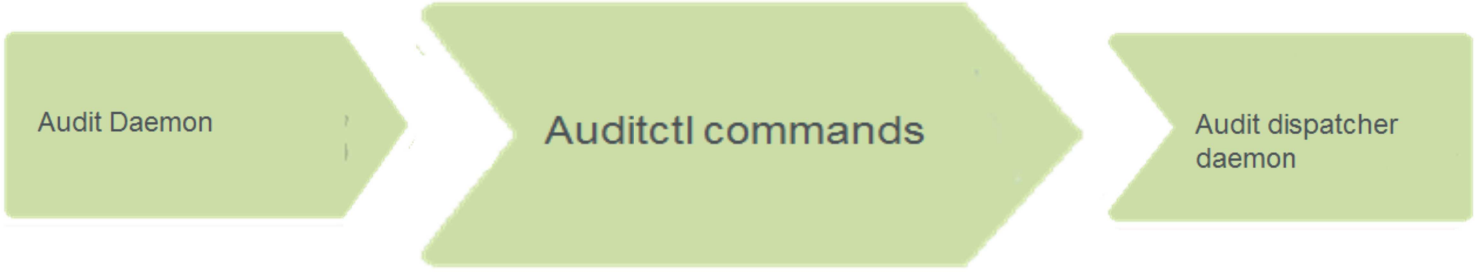


Filesystem Regeln

auditctl -w Pfad_zur_Datei -p Berechtigungen -k Key

Pfad_zur_Datei entspricht der Datei die überwacht werden soll.
Berechtigungen sind die zu überwachenden Berechtigungen

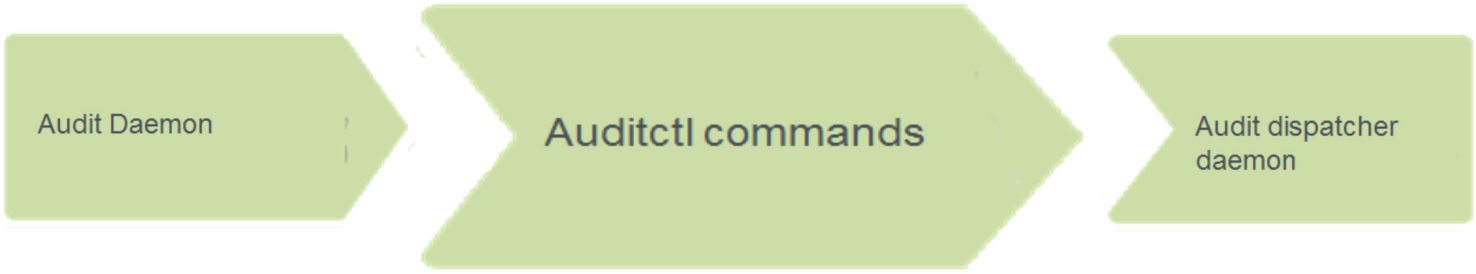
r — Lesezugriff
w — Schreibzugriff
x — Execute
a — Attributänderungen



-w /etc/shadow -p wa

-w /etc/passwd -p wa -k passwd_changes

-w /sbin/insmod -p x -k module_insertion

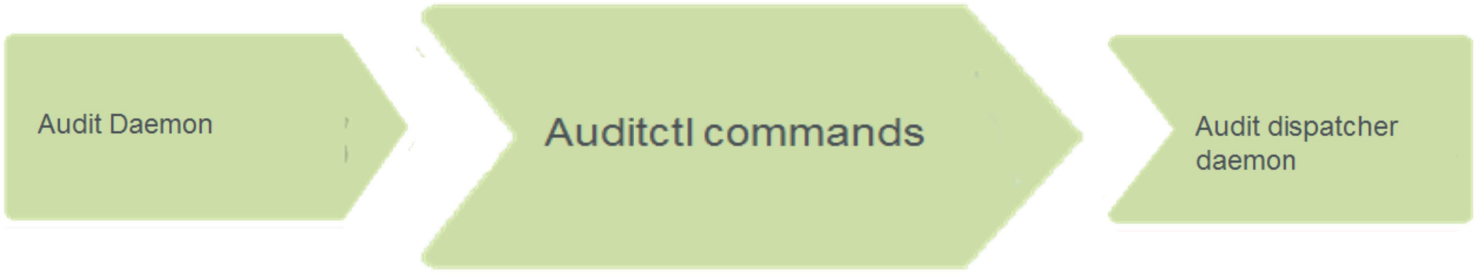


System call Regeln

-a action,list -S syscall -F field=value -k keyname

Action
Always or never

List
task, exit, user, und exclude



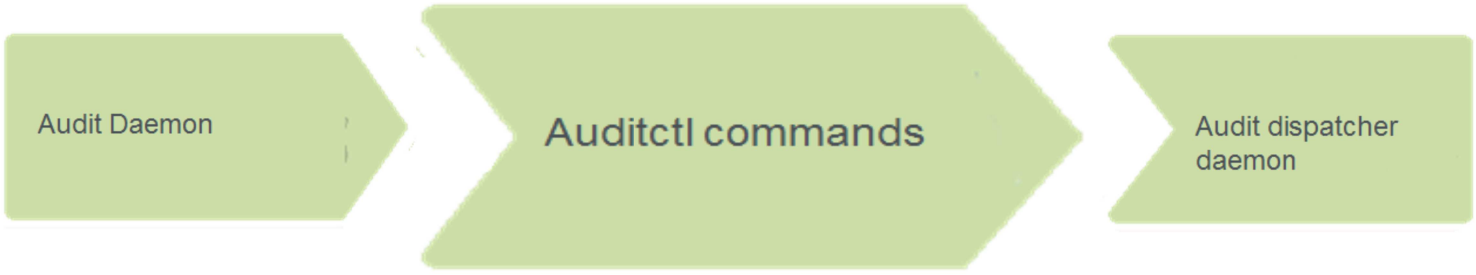
-S syscall

System_call gibt den Syscall mit Namen an. Eine Liste aller Systemcalls ist hier zu finden /usr/include/asm/unistd_64.h. Mehrere Systemcalls können in einer Regel gruppiert werden und hintereinander angegeben werden.

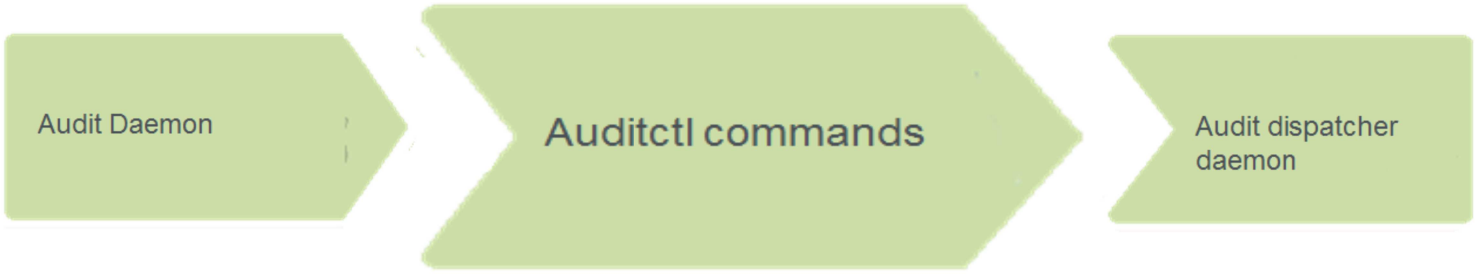
322 possible syscalls

E.g. #define __NR_read 0, #define __NR_write 1, #define __NR_open 2, #define __NR_close 3, #define __NR_stat 4, #define __NR_fstat 5, #define __NR_lstat 6, #define __NR_poll 7...

man 2 read




```
-F  [n=v | n!=v | n<v | n>v | n<=v | n>=v | n&v |
n&=v]

n = field
v = value

37 possible fields
E.g. auid, dir, exit, filetype, uid, success
```



CIS Beispiele

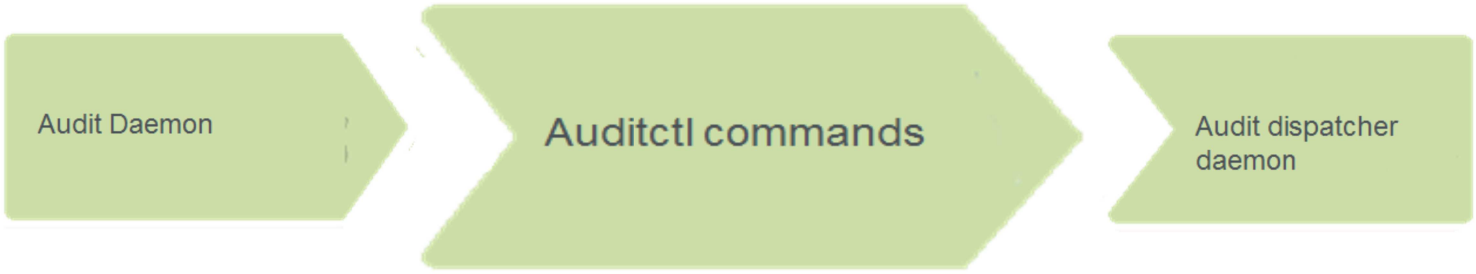
```
-a always,exit -F arch=b64 -S init_module -S
delete_module -k modules
```

```
-a always,exit -F arch=b64 -S creat -F exit=-EACCES -F
uid>=500
```

```
-a always,exit -S adjtimex -S settimeofday
```

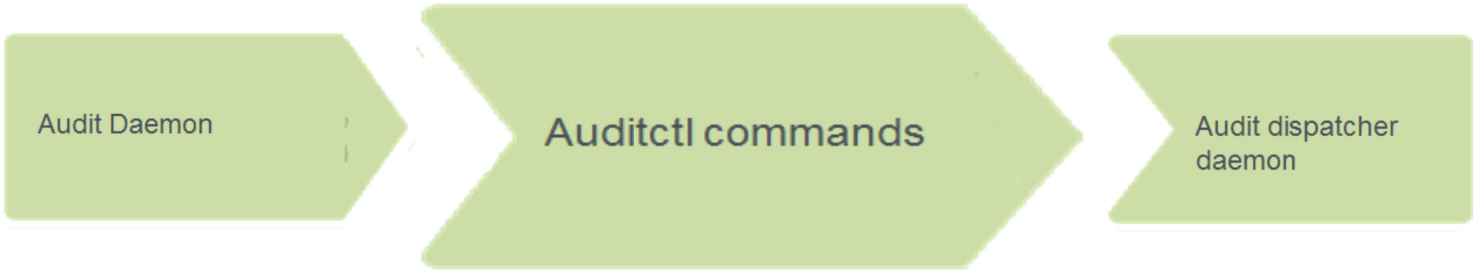
```
-a always,exit -S all -F pid=1005
```

```
-a always,exit -F dir=/transport -S all -F pid!=1005
-a always,exit -F dir=/transport -S all -F auid!=-1 |
4294967295
```



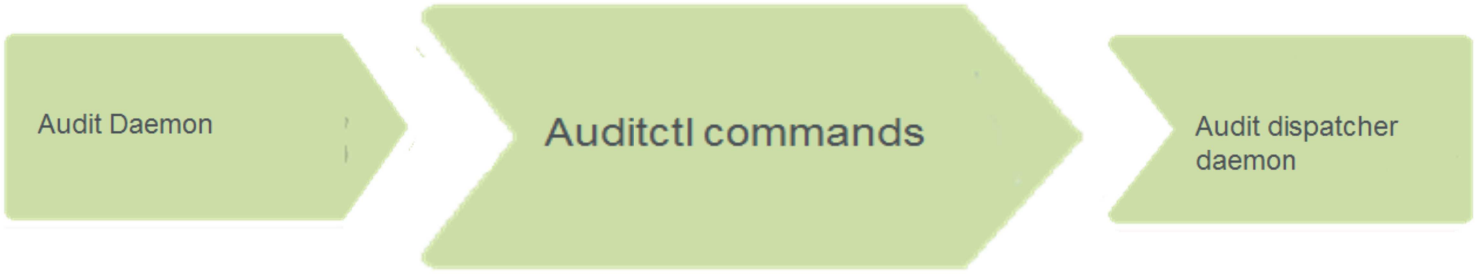
CIS Beispiele

```
# Collect Successful File System Mounts
-a always,exit -F arch=b64 -S mount -F auid>=500
   -F auid!=-1 -k mounts -k ids-sys-low

-a always,exit -F arch=b32 -S mount -F auid>=500
   -F auid!=-1 -k mounts -k ids-sys-low
```



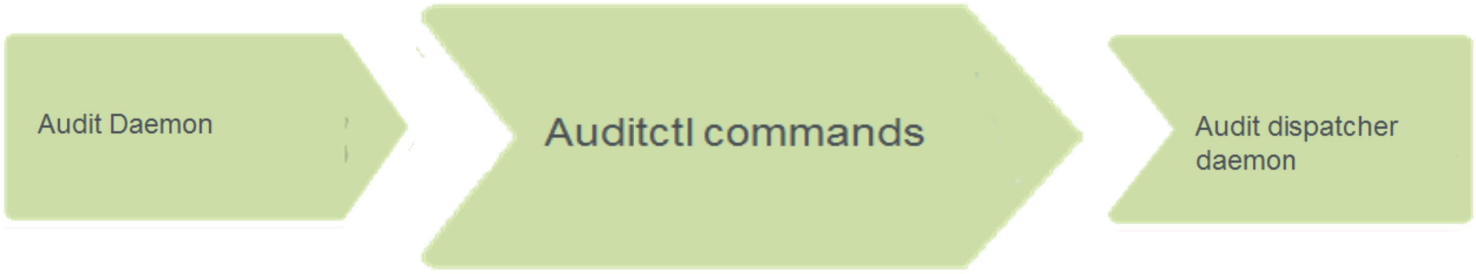
CIS Beispiele

```
### Collect Unsuccessful Unauthorized Access
    Attempts to Files
-a always,exit -F arch=b64 -S creat -S truncate
    -S ftruncate -F exit=-EACCES -F auid>=500 -F
    auid!=-1 -k access -k access -k ids-sys-hi

-a always,exit -F arch=b64 -S creat -S truncate
    -S ftruncate -F exit=-EPERM -F auid>=500 -F
    auid!=-1 -k access -k access -k ids-sys-hi
```



CIS Beispiele

```
### Collect Use of Privileged Commands
-w /usr/sbin/useradd -p x -k privileged -k ids-
  exec-info
-w /usr/sbin/userdel -p x -k privileged -k ids-
  exec-info
-w /usr/sbin/usermod -p x -k privileged -k ids-
  exec-info
-w /usr/sbin/groupadd -p x -k privileged -k ids-
  exec-info
-w /usr/sbin/groupdel -p x -k privileged -k ids-
  exec-info
```




```
-a exit,never -F path=/dev/kvm -F perm=rw -F
   subj_type=qemu_t

-a exit,always -F
   path=/etc/pki/libvirt/private/serverkey.pem -F
   subj_type!=virtd_t -k virt_tls_privkey

-a exit,never -F arch=b64 -S open -S write  -F
   path=/opt/uc4g/executor/bin/ucxjlx6 -F auid=-1
```

Das Auditing Framework hat eine Schnittstelle zur Anbindung externer Log Systeme, SIEM oder Intrusion Detection Systeme.
Verwaltet wird die Schnittstelle über den Audispd bzw. die audispd.conf Konfigurationsdatei.

Audispd Konfigurationsdatei

```
q_depth = 150
overflow_action = SYSLOG
priority_boost = 4
max_restarts = 10
name_format = HOSTNAME
#name = mydomain
```

/etc/audisp/plugins.d/syslog.conf

```
active = no
direction = out
path = builtin_syslog
type = builtin
args = LOG_INFO
format = string
```

Rsyslog example

```
# Linux Kernel Auditing
# send messages to SIEM Gateway

if $programname contains 'audispd' then
 @SIEM_FQDN
&~
```

Audit log

Audit records werden standardmäßig nach
/var/log/audit/audit.log geschrieben

Hohe Anzahl an Informationen im Log eintrag

```
2015 Oct 13 14:00:17 hostname[local0.info]
  audispd: node=hostname type=SYSCALL
  msg=audit(1444737617.178:198679): arch=c000003e
  syscall=2 success=yes exit=7 a0=7fd5d61fa0b0
  a1=42 a2=120 a3=7ffecf96a0c0 items=1 ppid=8103
  pid=8108 auid=4294967295 uid=0 gid=0 euid=0
  suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts1
  ses=4294967295 comm="visudo"
  exe="/usr/sbin/visudo" key="SUDOERS"
```

Audit log

type=SYSCALL

Type beschtreibt um was für eine Art von Audit event es sich handelt. .

msg=audit(1364481363.243:24287):

Das msg field enthält einen Zeitstempel und eine unique ID. Mehrere Einträge können den selben Zeitstempel und die selbe ID enthalten, wenn diese vom selben Audit event ausgelöst wurden.

arch=c000003e

syscall=2

Das Syscall Feld enthält die numerische Ausgabe des aufgerufenen Syscalls.

success=no

exit=-13

Das Exit Feld enthält den Rückgabewert des eines Syscalls. Die Rückgabewerte können mit dem Aufruf ausearch - - interpret - -exit -13 lesbar gemacht werden.

items=1

ppid=2686

pid=3538

auid=500

Die AUID is die Audit User ID die jeden User beim Login zugeordnet wird und die er während der Session Laufzeit beibehält. Der AUDI Wert bleibt gleich beim Userwechsel oder sudo Aufrufen.

uid=500

gid=500

euid=500

suid=500

fsuid=500

egid=500

sgid=500

.

fsgid=500

tty=pts0

Das tty Feld gibt den Terminal wieder von dem der Prozress aufgerufen wurde.

ses=1

comm="cat“

exe="/bin/cat"

Exe gibt den Pfad und den Befehl wieder der den Event ausgelöst hat.

subj=unconfined_u:unconfined_r: unconfined_t:s0-s0:c0.c1023

Subj Feld gibt den SELinux context wieder den der Prozess als label gesetzt bekommen hat..

key="sshd_config"

Fragen bei der Einführung

- Node Angabe nicht zwingend brauchbar da Login Cluster
- RBAC System erschwert Auswertung
- Mögliche Log Flut muss lokal oder remote gefiltert werden
- Kompromisse notwendig (Log Korrelation)
- Welchen Regelsatz soll man nehmen (CIS)
- Patchen / Updates / Bugs
- Verhalten von Software (z.B. vim)

Audit log

- „human readable“ Reports

- aureport
- ausearch
- mkbar & mkgraph

- autrace
- ausyscall

aureport

- **Summary von syscalls am heutigen Tag**
- **aureport -ts today -i -s --summary**

```
Syscall Summary  Report
==========================
total  syscall
==========================
25909  open
312  openat
106  readlink
96  execve
20  setsockopt
10  finit_module
```

- **Summary von fehlerhaften logins**
- **aureport -l –failed**

```
Login Report
============================================
# date time auid host term exe success event
============================================
1. 1.10.2015 14:14:03 root ::1 ssh /usr/sbin/sshd no 21232
2. 1.10.2015 14:14:23 ben ::1 ssh /usr/sbin/sshd no 21564
```

aureport

- **Summary von ausgeführten Befehle der Nutzer am heutigen Tag**
- **aureport -ts today -u**

```
User ID Report
===================================
# date time auid term host exe event
===================================
1. 13.10.2015 00:00:02 unset ? ? /usr/lib/systemd/systemd 15031
2. 13.10.2015 00:00:02 unset ? ? /usr/lib/systemd/systemd 15032
3. 13.10.2015 00:00:01 unset (none) ? /bin/bash 14982
4. 13.10.2015 00:00:01 unset (none) ? /bin/bash 14983
5. 13.10.2015 00:00:01 unset (none) ? /bin/bash 14984
6. 13.10.2015 00:00:01 unset (none) ? /usr/bin/basename 14985
7. 13.10.2015 00:00:01 unset (none) ? /usr/bin/mktemp 14986
8. 13.10.2015 00:00:01 unset (none) ? /usr/bin/mkdir 14987
9. 13.10.2015 00:00:01 unset (none) ? /usr/bin/xargs 14988
10. 13.10.2015 00:00:01 unset (none) ? /usr/bin/find 14989
11. 13.10.2015 00:00:01 unset (none) ? /usr/bin/xargs 14990
12. 13.10.2015 00:00:01 unset (none) ? /usr/bin/find 14991
13. 13.10.2015 00:00:01 unset (none) ? /usr/bin/find 14992
14. 13.10.2015 00:00:01 unset (none) ? /usr/bin/find 14993
15. 13.10.2015 00:00:01 unset (none) ? /usr/bin/date 14994
...
101. 13.10.2015 00:08:05 ben (none) ? /usr/lib64/libexec/kcheckpass 15081
102. 13.10.2015 00:08:05 ben (none) ? /usr/lib64/libexec/kcheckpass 15082
103. 13.10.2015 00:08:05 ben (none) ? /usr/lib64/libexec/kcheckpass 15083
```

ausearch

- **Suche nach “bad logins”:**

ausearch -m USER_AUTH,USER_ACCT --success no

- **Suche nach failed Logins für user xyz**

ausearch --message USER_LOGIN --success no –interpret –uid 1010

aureport | mkbar

https://people.redhat.com/sgrubb/audit/visualize/

Erstellt Zusammenfassung von verschiedenen Events

Erstellt eine Zusammenfassung von Events

aureport -e -i --summary | mkbar events

Erstellt eine Zusammenfassung von File Events

aureport -f -i --summary | mkbar files

Erstellt eine Zusammenfassung von Login Events

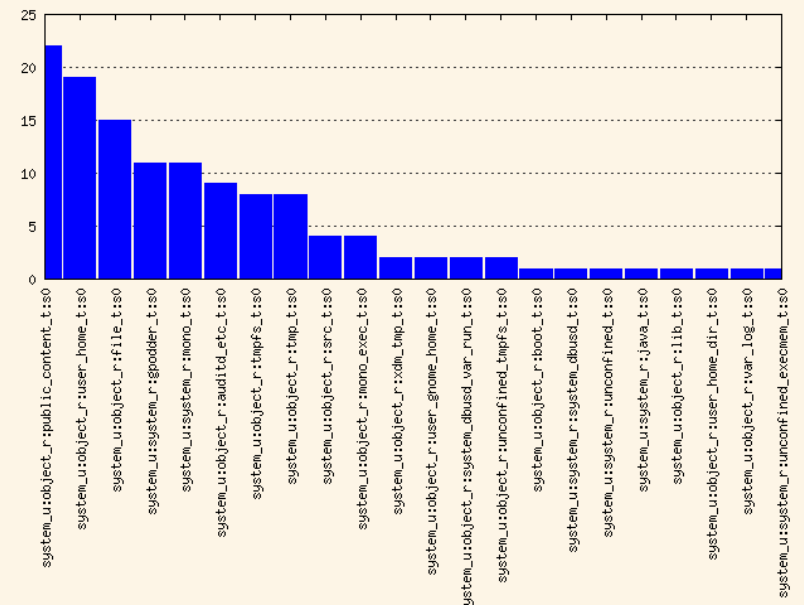
aureport -l -i --summary | mkbar login

Erstellt eine Zusammenfassung von User Events

aureport -u -i --summary | mkbar users

Erstellt eine Zusammenfassung von System Call Events

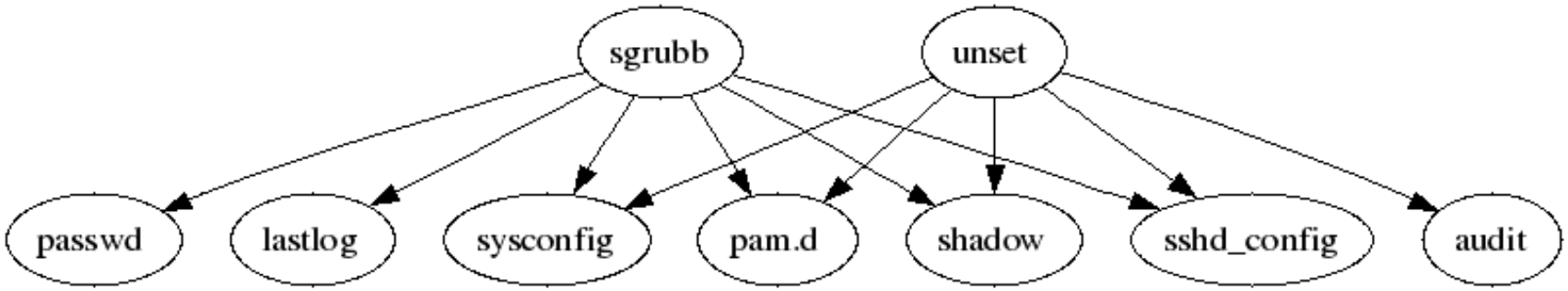
aureport -s -i --summary | mkbar syscalls



aureport | mkgraph

- **https://people.redhat.com/sgrubb/audit/visualize/**
- **Zeichnet die Beziehung von audit Objekten auf**

- **Anzeigen wer auf Dateien zugreift:**
  aureport -f -i | awk '/^[0-9]/ { printf "%s %s\n", $8, $4 }' | sort | uniq | ./mkgraph



autrace

Autrace

ist ein Programm das ähnlich strace einem abgesetzten Befehl analysiert. Der Befehl kan unter anderem zur Regelerstelleung bzw. Thread Analyse genutzt werden.

autrace /bin/cat /etc/shadow

...

Trace complete. You can locate the records with 'ausearch -i -p 18411‘

…

type=SYSCALL msg=audit(05/12/15 09:18:18.759:821849) : arch=x86_64 syscall=execve success=yes exit=0 a0=7f9205f20788 a1=7f92067b3068 a2=7fffe0340660 a3=7f92037fd4d0 items=2 ppid=25971 pid=18411 auid=unset uid=root gid=root euid=root suid=root fsuid=root egid=root sgid=root fsgid=root tty=(none) ses=unset comm=sar exe=/usr/bin/sar key=root-exe

…

ausyscall

Ausyscall ermöglicht die syscall Nummerierung zu Namen zu konvertieren.Ausyscall –dump gibt eine vollständige Liste der syscalls mit Nummerierung wieder.

Using x86_64 syscall table:

0 read

1 write

2 open

Appendix

http://www.commoncriteriaportal.org/

http://www.faqs.org/rfcs/rfc2196.html

https://people.redhat.com/sgrubb/audit/

https://www.youtube.com/watch?v=x2u_prS2HmM

https://www-01.ibm.com/support/knowledgecenter/linuxonibm/liaat/liaatkvmsecaudit.htm

http://www.ibm.com/developerworks/library/l-kvm-libvirt-audit/

https://www.kernel.org/doc/htmldocs/kernel-api/

http://lwn.net

https://lkml.org/lkml/2004/3/1/125


KEEP
CALM
AND
AUDIT
ON

Thank you !

Postbank
Eine Bank fürs Leben.