Fault Managment и поддержка GNU/Linux

- 1. Введение.
- 2. Включение аудита выполняемых на сервере команд (Security | Audit | Command).
  - 2.1 Настройка ядра Linux.
  - 2.2 Настройка системы.
- 3. Настройки защиты ядра Linux (Security | Attack | Crack Attemption Detected).
- 4. Настройки iptables, ... и прочих средств списков контроля доступа (Security | ACL).
- 5. Настройка lm_sensors (Environment)
- 6. Настройка syslog-ng
- 7. NOC

1. Введение.

Есть очень много способов реализовать журналирование событий в **GNU/Linux**. Рассмотрим один из лучших, с возможностью предварительной фильтрации журналируемых событий и отправкой их по сети на **syslog** сервер **NOC**.

Предварительная фильтрация журналируемых событий необходима для предотвращения перегрузки сетевого канала передаваемыми событиями и DDOS сервера NOC.

Чтобы реализовать аудит будем использовать средства ядра Linux, также дополнительные утилиты: *iptables*, ... для настройки правил отлова событий в сети. В качестве сервиса журналирования - *syslog-ng*.

2. Включение аудита выполняемых на сервере команд (Security | Audit | Command).

2.1 Настройка ядра Linux.

На исходные тексты ядра необходимо предварительно наложить патчи grsecurity.net после чего, как минимум, включить следующие опции:

```
CONFIG_GRKERNSEC_AUDIT_GROUP=y
CONFIG_GRKERNSEC_AUDIT_GID=65111
CONFIG_GRKERNSEC_EXECLOG=y
CONFIG_GRKERNSEC_AUDIT_CHDIR=y
```

2.2 Настройка системы.

Создаём группу для аудита:

```
# groupadd -g 65111 audit
```

И добавляем в неё необходимых пользователей:

```
# usermod -a -G audit noc
```

```
# usermod -a -G audit ...
```

Теперь ядро будет слать в системный журнал все выполняемые пользователями команды. Будьте осторожны, некоторые пользователи сервисов могут выполнять огромное количество команд. Добавляйте только необходимых пользователей!

3. Настройки защиты ядра Linux (Security | Attack | Crack Attemption Detected).

Включаем следующие опции настройки ядра:

```
CONFIG_PAX_NOEXEC=y
CONFIG_PAX_PAGEEXEC=y
CONFIG_PAX_MPROTECT=y
CONFIG_PAX_ASLR=y
CONFIG_PAX_RANDKSTACK=y
CONFIG_PAX_RANDUSTACK=y
CONFIG_PAX_RANDMMAP=y
CONFIG_PAX_MEMORY_SANITIZE=y
CONFIG_PAX_MEMORY_STACKLEAK=y
CONFIG_PAX_REFCOUNT=y
CONFIG_PAX_USERCOPY=y
CONFIG_GRKERNSEC_KMEM=y
CONFIG_GRKERNSEC_IO=y
CONFIG_GRKERNSEC_BPF_HARDEN=y
CONFIG_GRKERNSEC_PERF_HARDEN=y
CONFIG_GRKERNSEC_RAND_THREADSTACK=y
CONFIG_GRKERNSEC_PROC_MEMMAP=y
CONFIG_GRKERNSEC_KSTACKOVERFLOW=y
CONFIG_GRKERNSEC_BRUTE=y
CONFIG_GRKERNSEC_MODHARDEN=y
CONFIG_GRKERNSEC_HIDESYM=y
CONFIG_GRKERNSEC_RANDSTRUCT=y
CONFIG_GRKERNSEC_KERN_LOCKOUT=y
CONFIG_GRKERNSEC_BLACKHOLE=y
CONFIG_GRKERNSEC_FLOODTIME=60
CONFIG_GRKERNSEC_FLOODBURST=3
```

Бинарные файлы должны собираться с PIE/SSP (`CFLAGS="-fPIE -fstack-protector-all -D_FORTIFY_SOURCE=2" LDFLAGS="-Wl,-z,now -Wl,-z,relro"`). Теперь ядро способно определять некоторые виды популярных атак, предотвращать их распространение и информировать нас о произошедшей попытке взлома. Можно отметить в этом разделе больше опций или меньше... Некоторые опции сильно влияют на производительность, которая в свою очередь зависит от аппаратной платформы и поддерживаемых процессором инструкций.

4. Настройки iptables, ... и прочих средств списков контроля доступа (Security | ACL).

Необходимо создать специальные правила журналирования пакетов в *iptables* со специальными метками. Например для журналирования отброшенных пакетов необходимо последними в цепочках *INPUT, FORWARD, OUTPUT* добавить соответственно следующие правила:

```
# iptables -t filter -A INPUT -p ALL -m limit --limit 5/h --limit-burst 3 -j LOG --log-prefix 'iptables INPUT '

# iptables -t filter -A FORWARD -p ALL -m limit --limit 5/h --limit-burst 3 -j LOG --log-prefix 'iptables FORWARD
'

# iptables -t filter -A OUTPUT -p ALL -m limit --limit 5/h --limit-burst 3 -j LOG --log-prefix 'iptables OUTPUT '
--log-uid
```

Количество сетевых событий также может быть огромным, фильтруйте только необходимые, например атака **ssh heartbeat**:

```
# iptables -t filter -A INPUT -p tcp --dport 22 -m u32 --u32 "52=0x18030000:0x1803FFFF" -j LOG --log-prefix
'iptables SSH HEARTBEAT '
```

5. Настройка lm_sensors (Environment)

Устанавливаем в систему пакет **lm_sensors** с поддержкой **sensord**. В опции запуска **sensord** можно добавить **"-i 0"**, так он будет журналировать только тревожные сообщения. Запускаем **sensord** и добавляем его в вашу систему инициализации.

6. Настройка syslog-ng

Настройка этого сервиса предельно понятна и удобна. Необходимо указать 4 главных параметра: источник событий (*source*), место назначения событий (*destination*), фильтр событий (*filter*) и дать команду журналировать (*log*). Фильтр событый должен выбирать только события классифицируемые в NOC! Приведём рабочий пример файла настройки */etc/syslog-ng/syslog-ng.conf*, локальные журналы с него удалены:

```
@version: 3.7
# $Id$
#
# Syslog-ng default configuration file for Gentoo Linux

# https://bugs.gentoo.org/show_bug.cgi?id=426814
@include "scl.conf"

options {
        threaded(yes);
        chain_hostnames(no);

        # The default action of syslog-ng is to log a STATS line
        # to the file every 10 minutes.  That's pretty ugly after a while.
        # Change it to every 12 hours so you get a nice daily update of
        # how many messages syslog-ng missed (0).
        stats_freq(43200);
        # The default action of syslog-ng is to log a MARK line
        # to the file every 20 minutes.  That's seems high for most
        # people so turn it down to once an hour.  Set it to zero
        # if you don't want the functionality at all.
        mark_freq(3600);
};

source src {
    system();
    internal();
};

destination NOC { udp("1.1.1.1" port(514)); };

filter Environment { facility(daemon) and program("sensord") and level(alert); };
filter Network_Link { facility(kern) and message("^.*: (link up|link down)$"); };
filter Network_LAG { facility(kern) and message("^.*: (Adding slave|Removing slave)"); };
filter Network_NTP { facility(daemon) and program("^ntpd$") and (
                message("(now valid|now invalid)$")
                or message("adjusting local clock by")
                or message("clock is now synced$")
                ); };
filter Security_ACL { facility(kern) and (
                message("^iptables .*")
                or message("^ebtables .*")
                ); };
filter Security_Attack_Hardened { facility(kern) and (
                message("^PAX: terminating task: ")
                or message("^grsec: banning user with uid ")
                or message("^grsec: bruteforce prevention initiated ")
                or message("^grsec: denied resource overstep ")
                or message("^grsec: denied RWX mmap ")
                or message("^grsec: denied untrusted exec ")
                or message("^grsec: From ")
                ); };
filter Security_Audit_Command { facility(kern) and (
                message("^grsec: chdir to") and not message("^.*uid/euid:0/0 gid/egid:0/0, parent /usr/sbin/(run-
crons|cron)")
                or message("^grsec: exec of") and not message("^.*uid/euid:0/0 gid/egid:0/0, parent /usr/sbin/
(run-crons|cron)")
                ); };
filter Security_Audit_Cron { facility(cron) and message("^.* CMD .*$") and not message("^.*/usr/sbin/run-crons.
*$"); };
filter Security_Authentication { facility(authpriv) and message("^(.*|)pam_unix.*$"); };
filter System { program("^init$") and message("^(Entering|Switching to) runlevel: "); };

filter NOC { filter(Environment)

                or filter(Network_Link)

                or filter(Network_LAG)
                or filter(Network_NTP)
                or filter(Security_ACL)

                or filter(Security_Attack_Hardened)

                or filter(Security_Audit_Command)

                or filter(Security_Audit_Cron)

                or filter(Security_Authentication)

                or filter(System)

                ; };

log { source(src); filter(NOC); destination(NOC); };
```

7. NOC

Теперь необходимо ваш GNU/Linux сервер добавить в NOC **Service Activation -> Managed Object**, при добавлении выбрать профиль **OS.Linux** и указать в **Trap Source IP** адрес сервера.

Если решили добавить новые события с сервера GNU/Linux, то напишите ещё фильтры, у файле настройки */etc/syslog-ng/syslog-ng.conf*, выбирающие только необходимые вам события и добавьте их через **or** в агрегирующий фильтр **NOC**. Выбранные вами события появятся в **Fault Management -> Events** как **Unknown | Syslog**. Далее вам необходимо написать правила классификации событий.