Operazione Rif. PA 2023-19410/RER approvata con DGR 1317/2023 del 31/07/2023 finanziata con risorse del Programma Fondo sociale europeo Plus 2021-2027 della Regione Emilia –Romagna.

Progetto n. 1 - Edizione n. 1

TECNICO PER LA PROGETTAZIONE E LO SVILUPPO DI APPLICAZIONI INFORMATICHE

MODULO: N. 5 Titolo: SICUREZZA DEI SISTEMI INFORMATICI E DISPIEGO DELLE APPLICAZIONI
DURATA: 21 ORE DOCENTE: MARCO PRANDINI

MONITORAGGIO

Monitoraggio

- Logging: lasciare una traccia dettagliata e persistente delle attività dei demoni
  - principi generali
  - syslog tradizionale
  - rsyslog
  - syslog-ng
- Diagnostica istantanea: comandi per sondare lo stato corrente di
  - CPU
  - memoria
  - disco

Log di sistema

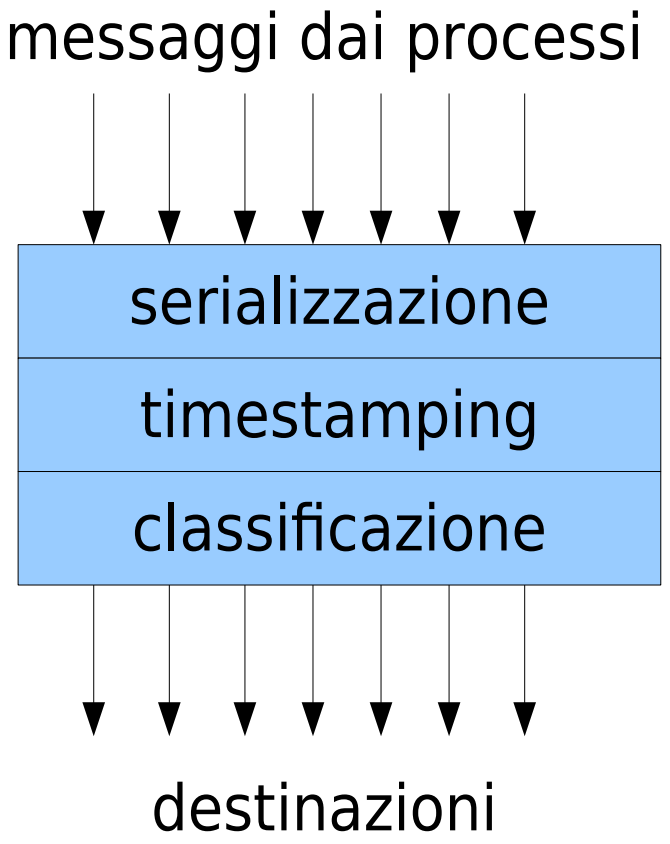
- I log (diari) tenuti dal sistema sono indispensabili per la diagnostica
  - anche per rilevare attività malevole o sospette
    - La loro stessa sicurezza va garantita!
      - Usare appropriatamente un integrity checker
      - Replicarli su macchine remote
- Logging su server remoto
  - Vantaggio aggiuntivo: centralizzazione
  - Implementazioni avanzate: shadow loggers
  - Problema: diventa un bersaglio appetibile
    - DoS

Linux logging

- Soluzioni comuni
  - Tipicamente producono file di testo
  - Nessuna garanzia di uniformità di formato a parte la marcatura temporale
  - BSD syslog (obsoleto)
    - klogd
  - Rsyslog
  - Syslog-ng
- In prospettiva integrato in **systemd**
  - *Journal*
  - Attivo dal boot, non dipende dall'avvio di altri servizi
  - Formato binario, visualizzabile con `journalctl`

syslog

- Principi di base mantenuti anche dalle evoluzioni



syslog: selettori e destinazioni

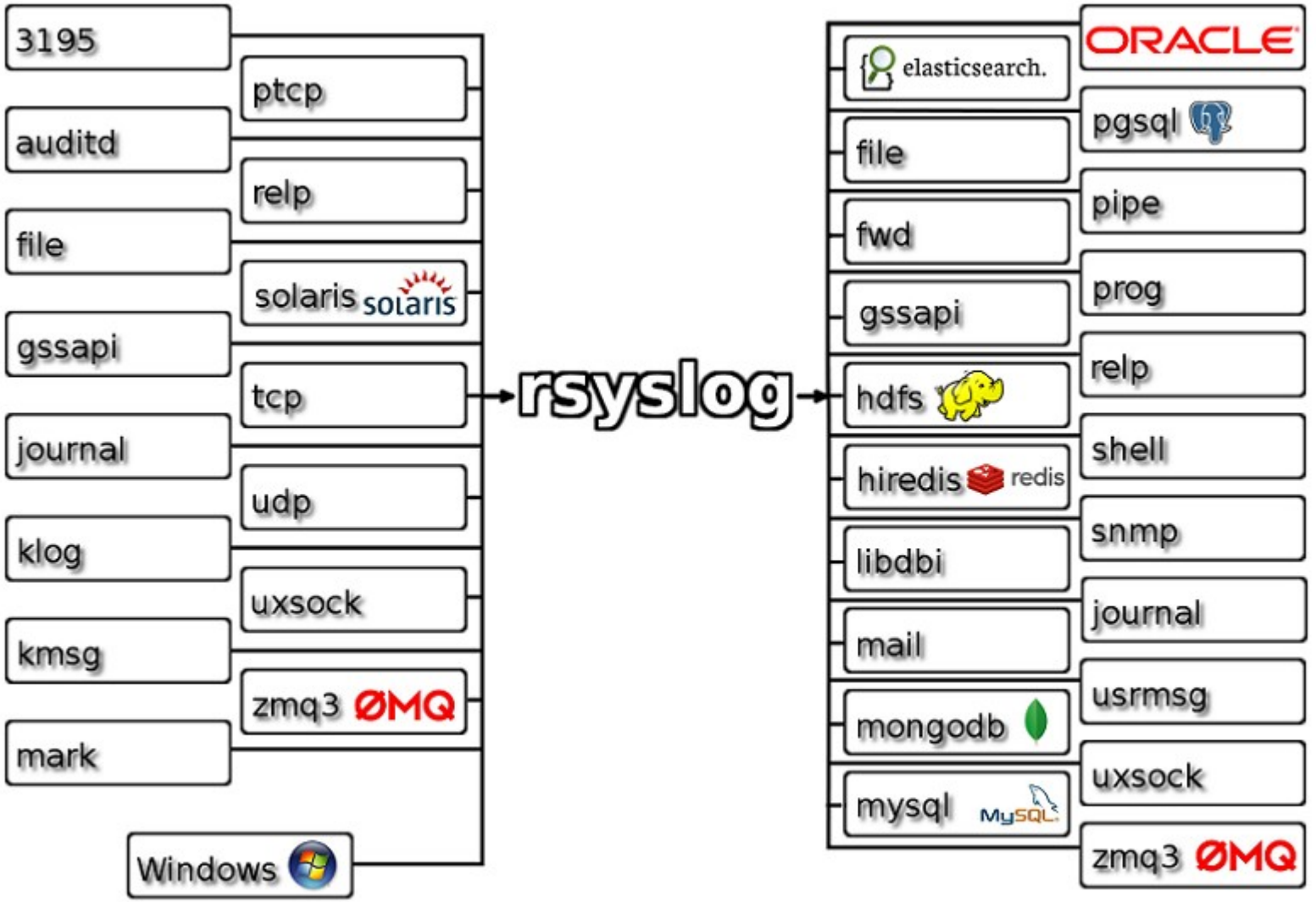
- Ogni messaggio è etichettato con una coppia *<facility>.<priority>*
  - Facility = argomento
    - `auth, authpriv, cron, daemon, ftp, kern, lpr, mail, news, syslog, user, uucp, local0..local7`
  - Priority = importanza in ordine decrescente:
    - `emerg, alert, crit, err, warning, notice, info, debug`
- Le destinazioni possibili sono
  - **File**: identificato da path assoluto
  - **STDIN** di un processo: identificato da una pipe verso il programma da lanciare
  - **Utenti** collegati: username, o * per tutti
  - **Server** syslog remoto: @indirizzo o @nome
    - La comunicazione avviene di default su UDP, porta 514

syslog: selettori

- `/etc/syslog.conf` contiene le direttive di configurazione generale e le regole minime di smistamento dei messaggi
- Ogni riga = una regola
  - <etichetta di interesse> <destinazione>
  - Vengono parsate tutte, quindi un messaggio può finire su più destinazioni
- Trattamento delle priority
  - Soglia: una regola che specifica una priority fa match con tutti i messaggi di tale priority e superiori a meno che non sia preceduta da "="
  - Priority speciale *none*: serve per ignorare i messaggi con la facility specificata prima del punto
- Es:

```
kern.*                  /dev/console
*.info;mail.none;       /var/log/messages
*.emerg                 *
kern.crit               "|/usr/bin/alerter"
*.=warning              @loghost
```

rsyslog


3195
auditd
file
gssapi
journal
klog
kmsg
mark
Windows
ptcp
relp
solaris
tcp
udp
uxsock
zmq3
rsyslog
elasticsearch.
file
fwd
gssapi
hdfs
hiredis
redis
libdbi
mail
mongodb
mysql
MySQL
ORACLE
pgsql
pipe
prog
relp
shell
snmp
journal
usrmsg
uxsock
zmq3

rsyslog

- Struttura modulare per caricare solo le funzioni necessarie
  - es. attivazione della ricezione di messaggi via rete (v8.4 / v8-16):

```
$ModLoad imudp          /      module(load="imudp")
$UDPServerRun 514       /      input(type="imudp"
port="514")
```

  - es. integrazione del kernel logging

```
$ModLoad imklog         /      module(load="imklog")
```

- File di configurazione modulare
  - Direttive globali in `/etc/rsyslog.conf`
  - Direttive specifiche in file separati sotto `/etc/rsyslog.d/`
    - stessa sintassi
- Scarto di messaggi (per evitare che vengano catturati da troppi selettori)
  - Basta mettere `~` come destinazione

rsyslog – modalità di output evolute

- Template per definire canali di output
  - Possono sostituire le destinazioni in modo più flessibile
    - Definizione:

```
$template apacheAccess,"/var/log/external/%fromhost%/apache/
%msg:R,ERE,1,ZERO:imp:([a-zA-Z0-9\-]+)\.--end%-access.log"
```

    - Utilizzo:

```
local6.notice ?apacheAccess
```

Segnaposto che verrà sostituito dal nome dell'host che origina il messaggio

Segnaposto che verrà sostituito per mezzo di una elaborazione del messaggio fatta via regex

- TCP logging
  - Per evitare perdita di messaggi (finché non ci sono crash!)

    http://blog.gerhards.net/2008/04/on-unreliability-of-plain-tcp-syslog.html

```
*.* @@indirizzo
```

- Shell execution
  - Passa il messaggio come parametro a un programma

```
*.* ^programma;template
```

rsyslog – selettori evoluti

- Rsyslog seleziona i messaggi in tre modi
  - i tradizionali facility.priority
  - Filtri basati su proprietà
  - Filtri basati su espressioni
- Filtri basati su proprietà
  - `:property, [!]compare-operation, "value"`
  - Es: `:msg, !contains, "error"  /var/log/good.log`
- Filtri basati su espressioni
  - Ancora in evoluzione (a marzo 2021)
    - la sintassi potrebbe cambiare in futuro
  - `if expr then destinazione`
  - Diventeranno un sistema completo di scripting, che consentirà di eseguire programmi arbitrari per determinare la destinazione

rsyslog – moduli

- Troppi per citarli esaustivamente

  http://www.rsyslog.com/doc/v8-stable/

- Tra i più interessanti:
  - RELP logging - per garanzia totale di consegna

    ```
    $ModLoad omrelp
    *.*  :omrelp:indirizzo
    ```

  - Output su tabelle di database

    ```
    $ModLoad ommysql
    ```

  - Acquisizione diretta dei messaggi del kernel (sostituisce klogd)

    ```
    $ModLoad imklog
    ```

syslog-ng

https://syslog-ng.org/

- Flessibile
  - Input compatibile con
    - Formati standard syslog (RFC3164, RFC5424)
    - JSON
    - Journald (systemd)
  - Output verso molteplici destinazioni
    - Tutti i DB SQL più diffusi
    - DB NOSQL (es. MongoDB)
    - Cloud databases (es. Redis)
  - Varietà di protocolli
    - Client-server
    - Message-based (AMQP, STOMP)
  - Capacità di elaborazione del contenuto dei messaggi
- E se non basta, estendibile con plugin
  - In C, Python, Java, Lua, o Perl

syslog-ng

- Configurazione:
  - Definizione di una *source*, che può unificare più ingressi fisici

```
source s_two {
 network(ip(10.1.2.3) port(1999));
 network(ip(10.1.2.3) port(1999) transport("udp"));
};
```

  - Definizione di una *destination*, con relative opzioni

```
destination d_file {
 file("/var/log/${YEAR}.${MONTH}.${DAY}/messages"
 template("${HOUR}:${MIN}:${SEC} ${TZ} ${HOST} [${LEVEL}] ${MSG}\n")
 template-escape(no));
};
```

  - Attivazione di un canale di log

```
log { source(s_two); destination(d_file); };
```

Monitoraggio dei parametri di sistema

- Comandi essenziali per il monitoraggio

(Utenti)	File	Processi	Spazio

`(w)`
`(last)`
	`fuser`	`ps`
		`top`
		`uptime`
			`df`
			`du`
			`free`

	`lsof`		`vmstat` `iostat`

- La maggior parte sono interfacce verso il filesystem **/proc**

https://www.kernel.org/doc/html/latest/filesystems/proc.html →
https://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html

proc filesystem

- Uno pseudo-filesystem che mostra come file i parametri di funzionamento del sistema
- Non risiede su disco, è una “apparizione” generata dal sistema operativo attraverso il meccanismo dei virtual filesystem
  - l’esplorazione della gerarchia viene gestita mostrando un’organizzazione di file e cartelle corrispondenti alle strutture dati del kernel
  - le chiamate in lettura a un file sono gestite mostrando i dati, presenti nelle strutture di memoria del kernel, corrispondenti al file utilizzato
  - sono presenti anche file scrivibili: inserirvi dati equivale a riconfigurare istantaneamente i corrispondenti parametri di funzionamento del kernel
- Parti principali
  - directory con nomi numerici corrispondenti ai PID dei processi attivi
  - directory che rappresentano alcuni macro-sistemi hardware
    - acpi, bus, driver, irq, tty
  - directory che rappresentano parametri o funzioni del sistema operativo
    - fs, sys, sysvipc
  - file con informazioni (principalmente statistiche di uso) globali del sistema

ps

- **ps (1)** supporta un numero strabiliante di opzioni, perché è compatibile con ben tre sintassi
  - Unix, singole lettere precedute da singolo trattino
  - BSD, singole lettere, senza trattino
  - estensioni GNU, parole precedute da doppio trattino
- Le opzioni delle tre famiglie possono essere mescolate nello stesso comando, a meno di non creare contraddizioni o ambiguità (…)
- Per gli usi più comuni, ci sono esempi collocati all'inizio della man page
- Suggerimenti – andiamo a leggere la man page
  - la sezione **PROCESS SELECTION BY LIST** mostra come ottenere una lista di processi secondo le loro proprietà (es. comando lanciato, pid, utente, ecc.)
    - è molto meglio usare queste opzioni che non "greppare" l'intera lista di processi!
  - la sezione **OUTPUT FORMAT CONTROL** illustra come formattare la lista prodotta
    - in particolare le opzioni (equivalenti) `-o`, `o`, `--format` seguite da una stringa di specificatori documentata nella sezione **STANDARD FORMAT SPECIFIERS** permettono un controllo completo sui campi che si vogliono far comparire nella lista
  - la sezione **PROCESS STATE CODES** spiega il significato della colonna STAT e dà un'indicazione fondamentale dello stato del processo

uptime

- uptime prende il nome dal primo elemento di output
- riporta anche il *carico* del sistema
  - ad ogni invocazione dello scheduler viene registrato il numero totale di processi in stato R (runnable) o D (uninterruptable sleep)
  - i campioni vengono accumulati e mediati su tre scale temporali diverse per ottenere un'indicazione del trend nel tempo
    - lo stato di "salute" va valutato in confronto al numero di processori disponibili

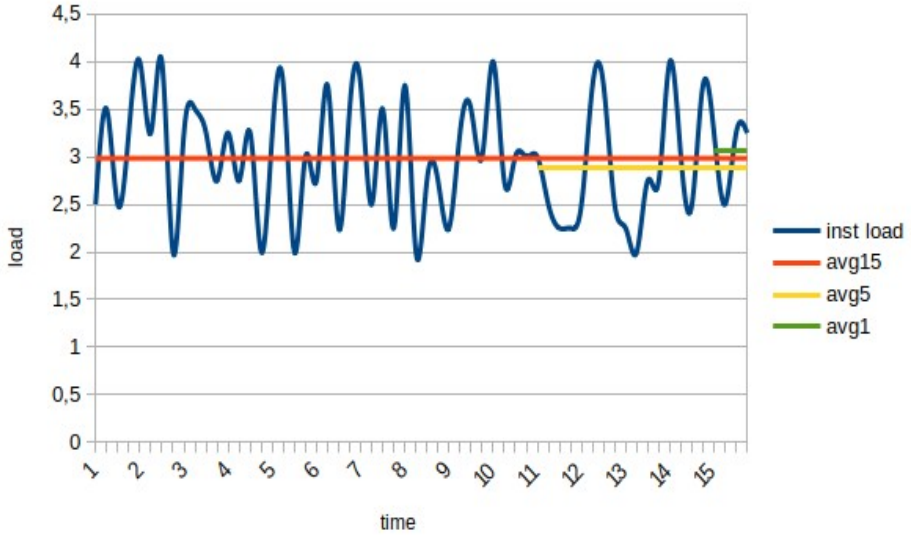
```
21:27:56 up  7:10,  2 users,  load average: 0.00, 0.00, 0.00
```

ora — n. utenti connessi — carico medio negli ultimi...

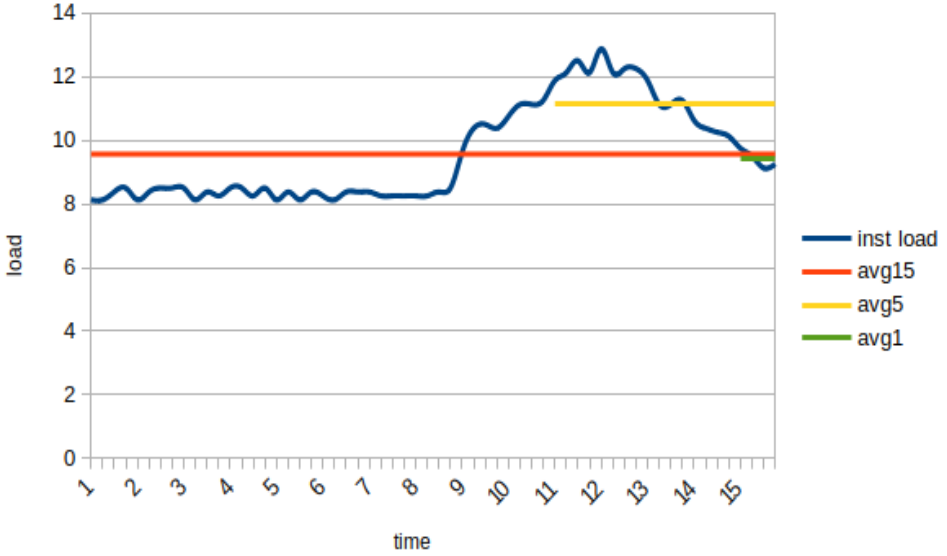
tempo trascorso dal boot — 1' 5' 15'

i tre carichi di uptime: esempi di interpretazione

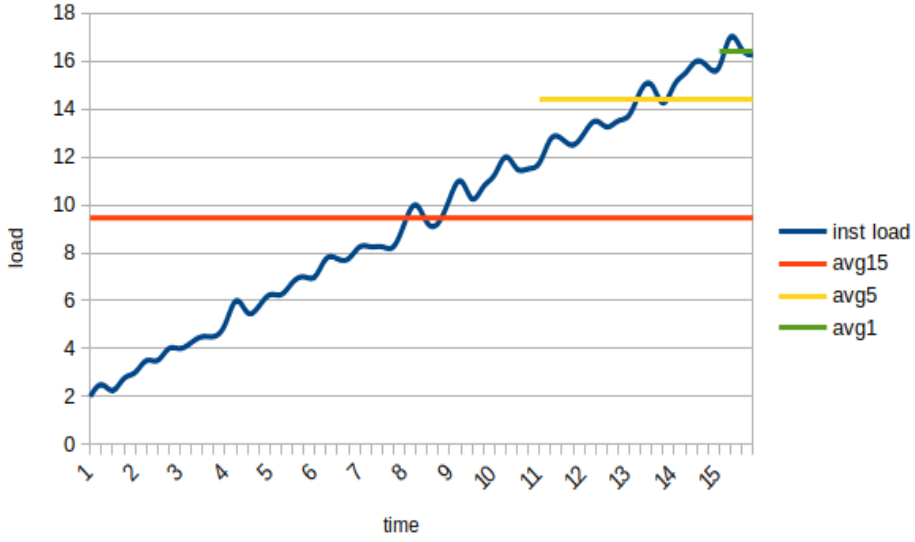
carico costante: avg1≈avg5≈avg15



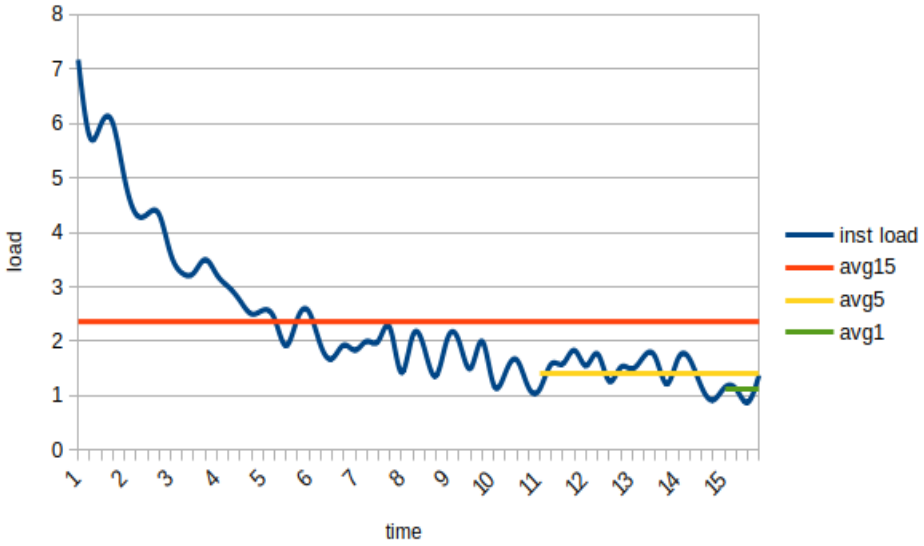
picco recente: avg5>(avg1≈avg15)



carico crescente: avg1>avg5>avg15



carico calante: avg1<avg5<avg15



free

```
las@client:~$ free
               total        used        free      shared  buff/cache   available
Mem:          237040      121248        9596        1464      106196       98720
Swap:        1045500       29572     1015928
```

- la maggior parte della memoria usata per cache può essere liberata per usi prioritari, da cui `available ≈ free + buff/cache`
  - l'impatto sulle prestazioni della rinuncia alle cache non è nullo
- `used swap > 0` significa solo che in qualche momento è servita

ps – uptime – free → top

- Comandi che scattano un'istantanea del sistema
  - ps: stato dei processi
  - uptime: carico del sistema
  - free: occupazione memoria
- Comandi di monitoraggio interattivi
  - top riassume ps, uptime, free + uso dettagliato cpu
  - aggiornato regolarmente
  - permette di interagire coi processi
  - utile per stima intuitiva dello stato di salute

top

```
9:31am  up 50 min,  2 users,  load average: 0.02, 0.02, 0.04
71 processes: 70 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  4.3% user,  5.2% system,  0.1% nice, 90.2% idle
Mem:   384480K av,  380688K used,    3792K free,    1312K shrd,   51312K buff
Swap:  128516K av,       0K used,  128516K free                  139136K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 1179 root      13   0  3092 3092  2592 S     2.8  0.8   0:50 magicdev
 9299 root      16   0  1044 1040   832 R     2.8  0.2   0:00 top
    1 root       8   0   520  520   452 S     0.0  0.1   0:03 init
    2 root       9   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root       9   0     0    0     0 SW    0.0  0.0   0:00 kapm-idled
    4 root      19  19     0    0     0 SWN   0.0  0.0   0:00 ksoftirqd_CPU0
    5 root       9   0     0    0     0 SW    0.0  0.0   0:00 kswapd
    6 root       9   0     0    0     0 SW    0.0  0.0   0:00 kreclaimd
    7 root       9   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    8 root       9   0     0    0     0 SW    0.0  0.0   0:00 kupdated
    9 root      -1 -20     0    0     0 SW<   0.0  0.0   0:00 mdrecoveryd
   71 root       9   0     0    0     0 SW    0.0  0.0   0:00 khubd
  465 root       9   0     0    0     0 SW    0.0  0.0   0:00 eth0
  546 root       9   0   592  592   496 S     0.0  0.1   0:00 syslogd
  551 root       9   0  1124 1124   448 S     0.0  0.2   0:00 klogd
  569 rpc        9   0   592  592   504 S     0.0  0.1   0:00 portmap
  597 rpcuser    9   0   788  788   688 S     0.0  0.2   0:00 rpc.statd
```

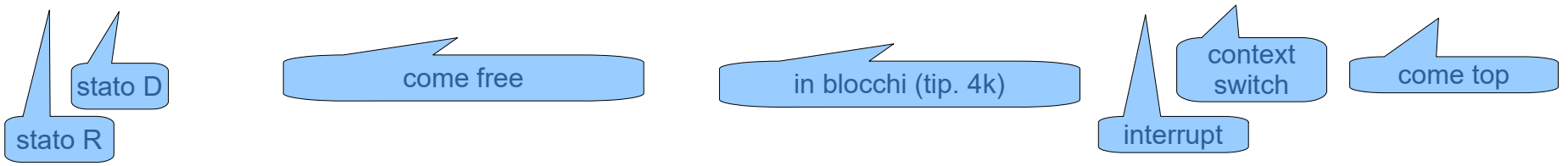
top – esempi di interpretazione del carico

- Un carico elevato può essere dovuto a molte cause diverse
- Esaminare l'uso di memoria e CPU può dare un'indicazione
- Es.
  - CPU sostanzialmente scarica
    - molti processi D? → un dispositivo non risponde?
  - CPU usata principalmente in userspace
    - sistema CPU-bound
  - CPU usata principalmente in iowait
    - sistema I/O bound
    - possono essere periferiche lente ma anche sovraccaricate per altri motivi
      - swap molto usata? → disco bombardato di swapout-swapin
      - sistema memory-bound
- indagini più approfondite si possono svolgere con **vmstat** e **iostat**
  - idealmente disponendo di una **baseline**, cioè dei valori tipici misurati durante la condizione di uso ottimale del sistema

Evoluzione delle risorse

- vmstat – uso di memoria, paging, I/O, trap
  - utile invocarlo col periodo (in secondi) per monitorare

```
root@Client:~# vmstat 1
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0  53404  39008  89588    0    0    12     1   12   31  0  0 100  0  0
 0  0      0  53344  39008  89588    0    0     0     0   19   27  0  1 99   0  0
 0  0      0  53344  39008  89588    0    0     0     0   14   19  0  0 100  0  0
```



- iostat - statistiche su uso CPU e I/O
  - soprattutto per valutare l'uso dei dispositivi di I/O

```
root@Client:~# iostat  /dev/sda1
Linux 3.16.0-4-amd64 (Client)  03/21/18   _x86_64_    (1 CPU)
...
Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda1              0.65        11.62         0.81     123899       8668
```

Spazio disco

- *df* mostra l'utilizzo dello spazio disco:

```
$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev              101812       0    101812   0% /dev
tmpfs              23704    3236     20468  14% /run
/dev/sda1       19277044 1654936  16619820  10% /
tmpfs             118520       0    118520   0% /dev/shm
tmpfs               5120       0      5120   0% /run/lock
tmpfs             118520       0    118520   0% /sys/fs/cgroup
tmpfs              23704       0     23704   0% /run/user/1001
```

- *du* permette di calcolare lo spazio occupato dai file (in una directory). Senza opzioni particolari du riporta l'occupazione totale delle dir passate come argomento ed anche di tutte le subdir in esse presenti. Es:

```
# du /tmp
1 /tmp/.font-unix
1 /tmp/.X11-unix
1 /tmp/.ICE-unix
5 /tmp/orbit-root
72   /tmp
```

- du **–s** riporta invece il *summary*, senza dettagli sulle subdir.

Uso dei file

- Quali file sta usando un processo:

```
# ls -l /proc/2208/fd/
total 0
lrwx------    1 root     root    64 Apr 26 10:11 0 -> /dev/pts/0
lrwx------    1 root     root    64 Apr 26 10:11 1 -> /dev/pts/0
lrwx------    1 root     root    64 Apr 26 10:11 2 -> /dev/pts/0
lr-x------    1 root     root    64 Apr 26 10:11 3 -> /etc/man.config
```

- Quali processi stanno usando un file:

```
# fuser /etc/man.config
/etc/man.config:    2208    2212    2213    2219
```

  - o un intero filesystem:

```
# fuser -m /var
/var:                  546   597c    714   714c   879c    898    916   916c
  964   1013   1020   1021   1318   6493   9244   9244m   9249   9249m   9275

c       current directory.
e       executable being run.
f       open  file.  f is omitted in default display mode.
r       root directory.
m       mmap'ed file or shared library.
```

Uso globale dei file

- lsof – list open files
  - elenca tutti i file impegnati da tutti i processi
- opera su tutti i namespace riconducibili al concetto astratto di file
  - regular file
  - directory
  - block special file,
  - character special file
  - executing text reference
  - library
  - stream o network file (socket internet o UNIX domain, NFS file)
- Osservazione: un file cancellato (unlink)dopo l'apertura sarà irreperibile sul filesystem, ma referenziato dal processo e quindi visibile a lsof

lsof

```
COMMAND    PID TID         USER   FD      TYPE        DEVICE SIZE/OFF      NODE NAME

cat       3210              las  cwd       DIR         8,1     4096    920949 /home/las
cat       3210              las  rtd       DIR         8,1     4096         2 /
cat       3210              las  txt       REG         8,1    43744    655603 /usr/bin/cat
cat       3210              las  mem       REG         8,1  3031632    654527 /usr/lib/locale/locale-archive
cat       3210              las    0u      CHR       136,1      0t0         4 /dev/pts/1
cat       3210              las    1w      REG         8,1        0    922420 /home/las/ghostfile (deleted)
cat       3210              las    2u      CHR       136,1      0t0         4 /dev/pts/1
lsof      3220              las  cwd       DIR         8,1     4096    920949 /home/las
lsof      3220              las  rtd       DIR         8,1     4096         2 /
lsof      3220              las  txt       REG         8,1   175584    666005 /usr/bin/lsof
lsof      3220              las  mem       REG         8,1  3031632    654527 /usr/lib/locale/locale-archive
...
```

Esecuzioni pianificate
(utili tra l'altro per automatizzare il monitoraggio)

- L'esecuzione periodica di programmi è compito di *crond*
  - ogni utente ha la propria *cron table* (*crontab*),
    - guardate in `/var/spool/cron` per trovarle
  - i task di sistema sono spesso raccolti in `/etc/crontab`
    - tipicamente preconfigurato per l'esecuzione di script a periodicità di uso comune
    - `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly`, `/etc/cron.monthly`
  - `/etc/crontab` si può editare direttamente, per le tabelle utente meglio usare

    ```
    crontab -e [-u username]
    ```

- L'esecuzione singola in un istante preciso è compito di *atd*
  - `atq` per elencare i job in attesa
  - `atrm` per rimuoverli

Esecuzione posticipata - at

- ***atd*** **è un demone che gestisce code di compiti da svolgere in momenti prefissati. L'interfaccia ad *atd* consiste di 4 comandi:**

- `at [-V] [-q queue] [-f file] [-mldbv] TIME`
  pianifica un comando al tempo TIME
- `atq [-V] [-q queue] [-v]`
  elenca i comandi in coda
- `atrm [-V] job [job...]`
  rimuove comandi dalla coda
- `batch [-V] [-q queue] [-f file] [-mv] [TIME]`
  esecuzione condizionata al carico

Esecuzione posticipata - at

- Se non viene specificato un file comandi per at o batch, verrà usato lo standard input.
- La specifica dell'ora è flessibile e complessa. Per una definizione completa si veda la documentazione in `/usr/share/doc/at/timespec`
- Alcuni esempi:

```
echo 'wall "sveglia"' | at 08:00
echo "$HOME/bin/pulisci" | at now + 2 weeks
echo "$HOME/bin/auguri" | at midnight 31.12.2021
```

Esecuzione periodica - cron

- ***crond*** **è un demone che esamina una serie di file di configurazione ogni minuto, e determina quali compiti specificati nei file debbano essere eseguiti.**
- **I file di configurazione (crontab) sono distinti in due insiemi:**
  - **Uno per utente (`/var/spool/cron/crontabs/<utente>`)**
    - **si visualizza / edita / sostituisce con**

```
crontab -l / crontab -e / crontab <nuova_tab>
```

  - **System-wide (`/etc/crontab`)**
    - **Solitamente quest'ultimo non fa altro che richiamare l'esecuzione di tutto ciò che trova in alcune directory:**

```
/etc/cron.hourly/
/etc/cron.daily/
/etc/cron.weekly/
/etc/cron.monthly/
```

ha un campo in più rispetto
ai file personali per indicare
a nome di che utente eseguire
ogni task configurato

Esecuzione periodica - cron

- Ogni crontab contiene un elenco di direttive nella forma

MINUTO ORA G.MESE MESE G.SETTIMANA <comando>

Es.

```
*     *       27      *     *     $HOME/bin/paga
30    8-18/2  *       *     1-5   $HOME/bin/lavora
00    00      1       1     *     /usr/sbin/auguri
30    4       1,15    *     6     /bin/backup
```

- L'azione è eseguita quando l'ora corrente corrisponde a tutti i selettori di una riga (campi in AND logico)
- **ECCEZIONE**: se sono specificati (diversi da *) entrambi i giorni (settimana e mese), i due campi sono considerati in OR logico
  - nell'esempio, il backup viene eseguito ogni mese il giorno **1** + il giorno **15** + ogni **sabato**, alle 4:30

Monitoraggio della sicurezza

- Le fasi di un attacco possono lasciare tracce.
- Individuarle con accuratezza e tempestività è di fondamentale importanza per evitare o limitare danni.

I termini del monitoraggio

- IDS = Intrusion Detection System
  - è genericamente un sistema in grado di rilevare tentativi di attacco
    - *signature based* (riconosce attacchi noti)
    - *anomaly detection* (riconosce deviazioni dall'uso standard)
- IPS = Intrusion Prevention System
  - semplificando: un IDS in grado di interagire con sistemi di controllo dell'accesso per bloccare il traffico malevolo
- SIEM = Security Information and Event Management
  - piattaforma che integra strumenti, politiche e procedure per la gestione integrata delle fonti di informazione e degli incidenti
- Parametri di qualità del rilevamento degli eventi
  - **Falso positivo (FP)**: segnalazione di attacco errata da evento innocuo
  - **Falso negativo (FN)**: attacco reale che non genera una segnalazione

NIDS, HIDS, EDR

- Due strategie di rilevazione
  - Basate sulla rete
    - NIDS / Network-based IDS
  - Basate sull'endpoint
    - HIDS / Host-based IDS
    - EDR / Endpoint Detection and Response
- NIDS usa i dati intercettati sui canali di comunicazione
  - sistema dedicato sul perimetro o con sonde, per il traffico di tutti i sistemi
- HIDS è un processo userland sul sistema da proteggere
  - vede il traffico diretto a un singolo sistema
  - monitora il filesystem, i processi, le attività utente
  - esamina in tempo reale i log file
  - verifica periodicamente contenuti e metadati dei file
- EDR può essere definito come un HIDS fortemente integrato col sistema operativo

NIDS

- Vantaggi
  - Visibilità di tutto il traffico, entrante e uscente
  - Richiede un solo punto di installazione
    - vero solo se rete semplice (dispositivi mobili che lasciano la rete??)
    - con più sonde: possibilità di ragionare su flussi
  - Un malfunzionamento non incide sugli endpoint
- Svantaggi
  - Maggior tasso di FP
    - processi legittimi possono generare occasionalmente traffico anomalo
  - Più soggetto a sovraccarico o evasione
    - es. pacchetti frammentati
  - Non può esaminare il traffico cifrato
  - Un punto di analisi per un'intera rete → richieste hardware
    - < 20-30 Mbps: raspberry PI
    - < 200-300 Mbps: pc desktop di un paio d'anni
    - < 1 Gbps: server almeno 8 core / 16 thread
    - > 10 Gbps: 40+ core e probabilmente serve accelerazione hardware

HIDS

- Vantaggi:
  - Minor tasso di FP
    - Pacchetti di rete sospetti possono essere correttamente classificati solo esaminando l'interazione con l'obiettivo finale
  - Economico
    - Sfrutta per definizione i sistemi già esistenti
    - Non molto impegnativo computazionalmente (distribuito)
- Svantaggi
  - Punti ciechi
    - Se un evento/pacchetti non lascia tracce sul filesystem è invisibile
    - Non valuta il traffico uscente (egress) – solo entrante (ingress)
    - Non individua scansioni che non toccano servizi attivi
  - Richiede l'installazione di un agente sulla macchina
  - Se la macchina è compromessa può essere neutralizzato

EDR

- Vantaggi su HIDS
  - in grado di raccogliere eventi dai device driver di filesystem e di rete e comunicazioni interprocesso
  - in grado di analizzare eseguibili e librerie al caricamento e a run time (system call, fork)
  - capacità anti-tampering
  - maggiori possibilità di risposta (isolamento di comunicazioni e di processi)
- Svantaggi
  - richiede interfacciamento stretto con OS (non così ovvio)
    - hooking
    - minifilters
  - difficile trovare soluzioni open e non molto costose

Host IDS – integrity check

- La rilevazione di intrusioni sull'host è tipicamente svolta per mezzo di un *integrity checker*
- Principio:
  - Si memorizza in un database lo stato del filesystem quando è certamente "pulito"
  - Si confronta periodicamente il filesystem col database
- Tra i più diffusi:
  - Tripwire (commerciale)
  - AIDE (fork FOSS di Tripwire)
  - AFICK

  https://www.sans.org/reading-room/whitepapers/detection/ids-file-integrity-checking-35327

Integrity checking – homemade

- I tool crittografici di base permettono di costruire a mano elenchi con hash dei contenuti dei file essenziali
- E i metadati? E i file speciali?
- In alcuni casi già la distribuzione del sistema aiuta
  - es. Linux RPM-based
    - file Size
    - Mode (include permessi e file type)
    - MD5 sum
    - Device major/minor number
    - readlink(2) path
    - User/Group ownership
    - mTime
- Dove si mette il database? È protetto da manipolazioni?

Integrity checkers

- Caratteristiche da valutare:
  - Algoritmi usati per calcolare le impronte dei file
  - Performance e dimensioni del DB
  - Capacità di proteggere i propri stessi binari
  - Capacità di proteggere il database
  - Portabilità
  - Complessità degli aggiornamenti
    - Del sw
    - Del database

AIDE

- AIDE è un controllore di integrità configurabile
- Il file `/etc/aide.conf` definisce i tipi di controlli da applicare a file e directory
- Processo
  - Un database di riferimento deve essere costruito su di un sistema *pulito*
  - Una scansione periodica confronta i file di sistema con il database in base alla configurazione e riporta le differenze rilevanti

https://aide.github.io/

http://doc.opensuse.org/products/draft/SLES/SLES-security_sd_draft/cha.aide.html

AIDE – tipi di controllo implementati

DIRECTIVE	DESCRIPTION
p	permissions
i	inode
n	number of links
u	user
g	group
s	size
b	block count
m	Mtime
a	Atime
c	Ctime
S	check for growing size
md5	`md5` checksum
sha1	`sha1` checksum
rmd160	`rmd160` checksum
tiger	`tiger` checksum
R	p+i+n+u+g+s+m+c+md5
L	p+i+n+u+g
E	Empty group
>	Growing logfile `p+u+g+i+n+S`

AIDE – qualche esempio

- La seguente riga di selezione esaminerà tutto nella directory /etc, esaminando in particolare il numero di collegamenti, l'utente che possiede un dato file, il gruppo che possiede un dato file e la dimensione del file:

```
/etc n+u+g+s
```

- Gli oggetti possono essere ignorati o saltati utilizzando un punto esclamativo (!), come nell'esempio seguente, che fa sì che AIDE ignori tutto in /var/log:

```
!/var/log/.*
```

- I pattern sono sottostringhe ancorate alla radice: attenzione alle esclusioni apparentemente specifiche!

```
!/var/log/maillog
```

- quello che si voleva dire, forse era:

```
!/var/log/maillog$
```

AIDE – qualche esempio

- Un esempio di configurazione

```
MyRule = p+i+n+u+g+s+b+m+c+md5+sha1
/etc p+i+u+g     # check only permissions, inode, user and group for etc
/bin MyRule      # apply the custom rule to the files in bin
/sbin MyRule     # apply the same custom rule to the files in sbin
!/var/log/.*     # ignore the log dir it changes too often
```

- Caratteristiche
  - Compilato, molto veloce
  - Integrabile con permessi estesi (acl, selinux)

AIDE – quick start

- Configurare
  - le regole di controllo
  - il nome del database di riferimento (usato per i controlli)
    - `database=file:/usr/local/aide/aide.db`
  - il nome del nuovo database prodotto ad ogni aggiornamento
    - `database_out=file:/usr/local/aide/aide.db.new`
- Inizializzare il database
  - `aide --init`
- Rinominarlo per usarlo come riferimento per i controlli futuri:
  - `mv /usr/local/aide/aide.db.new /usr/local/aide/aide.db`
- Lanciare un integrity check:
  - `aide --check`

AIDE – uso appropriato

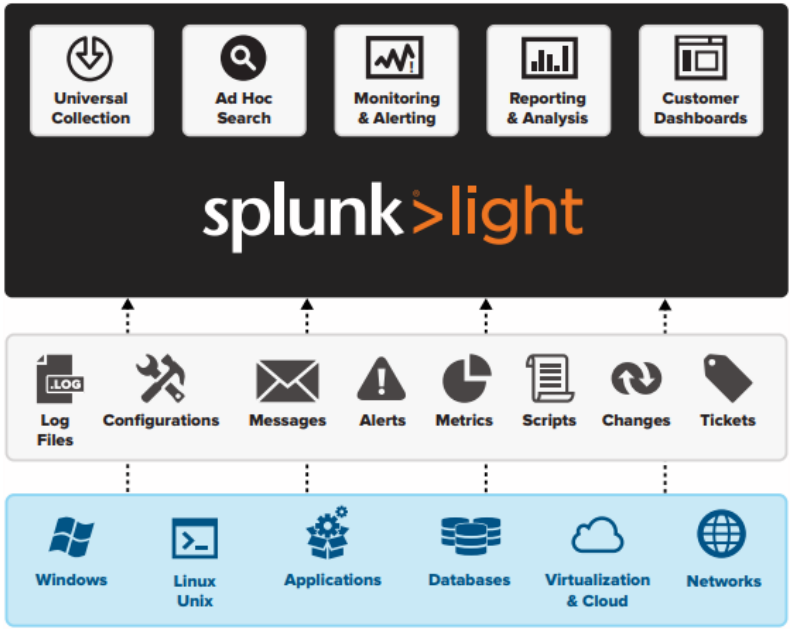
- A seconda del caso, AIDE può essere
  - eseguito come strumento forense, solo se si sospetta / si verifica un'irruzione
  - programmato per segnalare regolarmente qualsiasi cambiamento interessante
    - Due problemi:
      - reimpostare periodicamente il database per interrompere la segnalazione di modifiche non dannose ai file
      - **difendere l'integrità del binario e del database di AIDE!**
- Per ridurre il rischio di eseguire un AIDE compromesso / su un DB compromesso:
  - utilizzare le firme HMAC per il file di configurazione e il DB
  - masterizzare il DB su di un supporto non riscrivibile
  - eseguire il software da un sistema diverso e affidabile
  - oppure, se non se ne dispone, da un supporto di ripristino
    - **fermo macchina!**

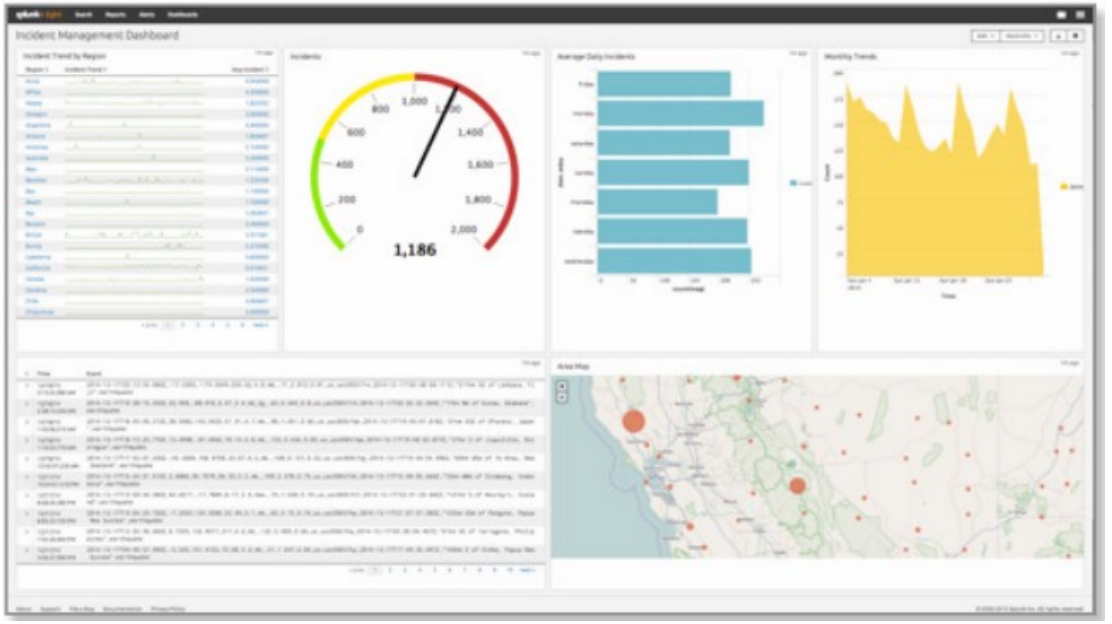
SIEM

- Aggregazione dei dati
  - raccoglie log / eventi da più fonti
  - normalizza e consolida i dati
  - sistema di interrogazione centralizzato
- Correlazione
  - collega eventi con attributi comuni in pacchetti significativi
  - genera automaticamente avvisi in base a condizioni specifiche
- Monitoraggio
  - grafici per visualizzare lo stato corrente
  - grafici relativi alla conformità
- Ritenzione
  - conservazione a lungo termine
  - analisi forense

SIEM

- Stanno evolvendo verso gestione integrata
  - qualsiasi tipo di "dato macchina" → **normalizzazione**
  - algoritmi di machine learning
  - reportistica avanzata
  - on premise o SaaS





Analisi dei dati di ispirazione cloud

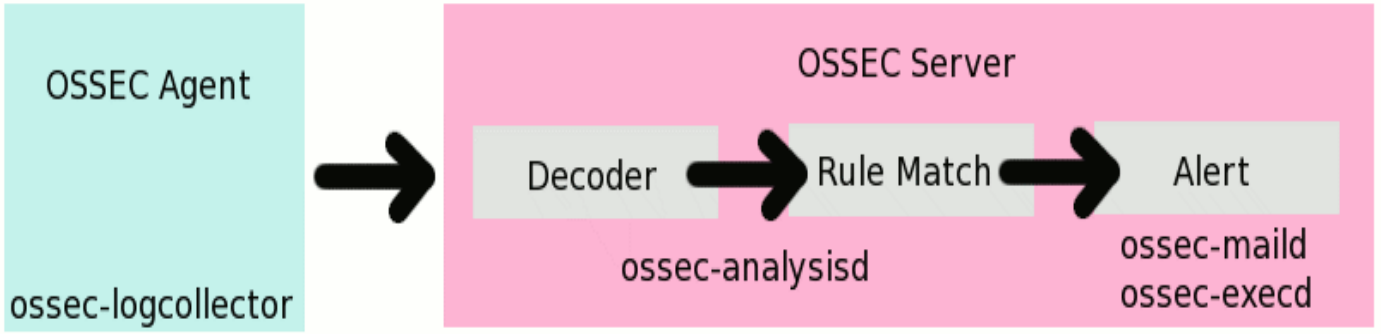
- Stack Elastic, Open Source
  - https://www.elastic.co/
  - Raccolta e esplorazione dei dati di log
  - definizione di un Elastic Common Schema per normalizzare
    - aperto, semplice, estendibile
    - limitato all'ambiente Elastic, verboso, non facile manutenzione delle customizzazioni
- Composto dai tre programmi:
  - Logstash: Pipeline di elaborazione dei log
  - Elasticsearch: Database Nosql
  - Kibana: Visualizzatore web-based per documenti in Elasticsearch

Un esempio di SIEM OSS: Wazuh

- Open source fork di OSSEC
  - per vedere rapidamente cosa è cambiato
    https://wazuh.com/migrating-from-ossec/
  - nel seguito si fa riferimento alle caratteristiche originali di OSSEC
- Funzionalità principali
  - Verifica della compliance a policy di sicurezza OpenSCAP
  - Raccolta dei log, analisi e conservazione centralizzata
    - Integrazione con Elastic stack
  - Filesystem integrity checking
  - Host-based IDS – non servono sonde in rete
    - monitoraggio del Windows registry
    - scansione per malware, rootkit, anomalie, processi offuscati, syscall inconsistenti, ...
  - Reazioni attive
    - built-in: RTBL (Real Time Black Listing)
    - qualsiasi cosa via scripting

OSSEC

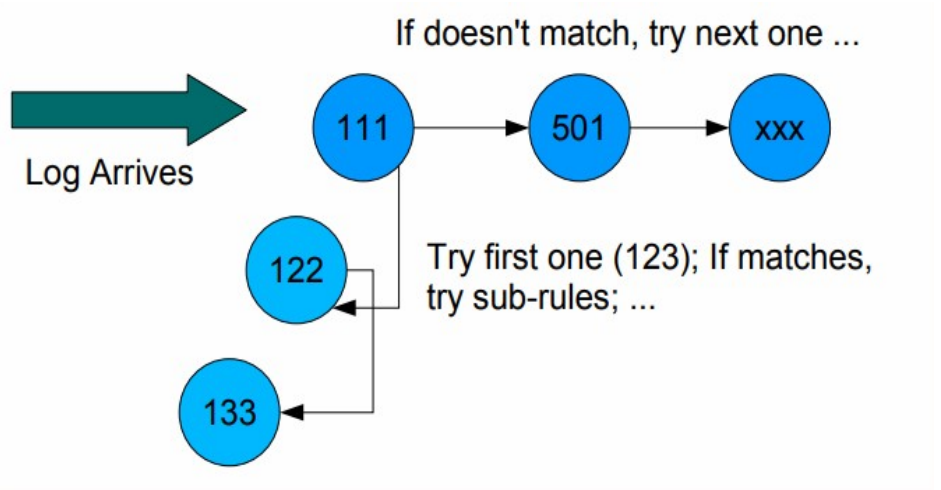
- Due modalità di funzionamento
  - locale, client-server
- Modalità client-server
  - i client ricevono la configurazione da un server
  - i client inviano i log al server su canale cifrato



- Comunicazione
  - standard syslog (UDP:514)
  - compressione
  - cifratura simmetrica (blowfish con chiavi scambiate manualmente)

OSSEC config / decoders e rules

- Parsing dei log file configurabile per mezzo di *decoders*
  - monitoraggio di file multipli
  - regole di parsing ed estrazione scritte in XML
  - forniscono i campi utili per l'attivazione delle *rules*
- Analisi dei dati per mezzo di *rules*
  - scritte in XML
  - componibili in gerarchia
    - livelli di priorità 1-15
  - ruleset pre-configurati per i servizi più diffusi



- Esempi

http://ossec-docs.readthedocs.io/en/latest/manual/rules-decoders/create-custom.html

https://sevenminuteserver.com/post/2010-09-25-writing-custom-ossec-rules/

OSSEC config / alerts

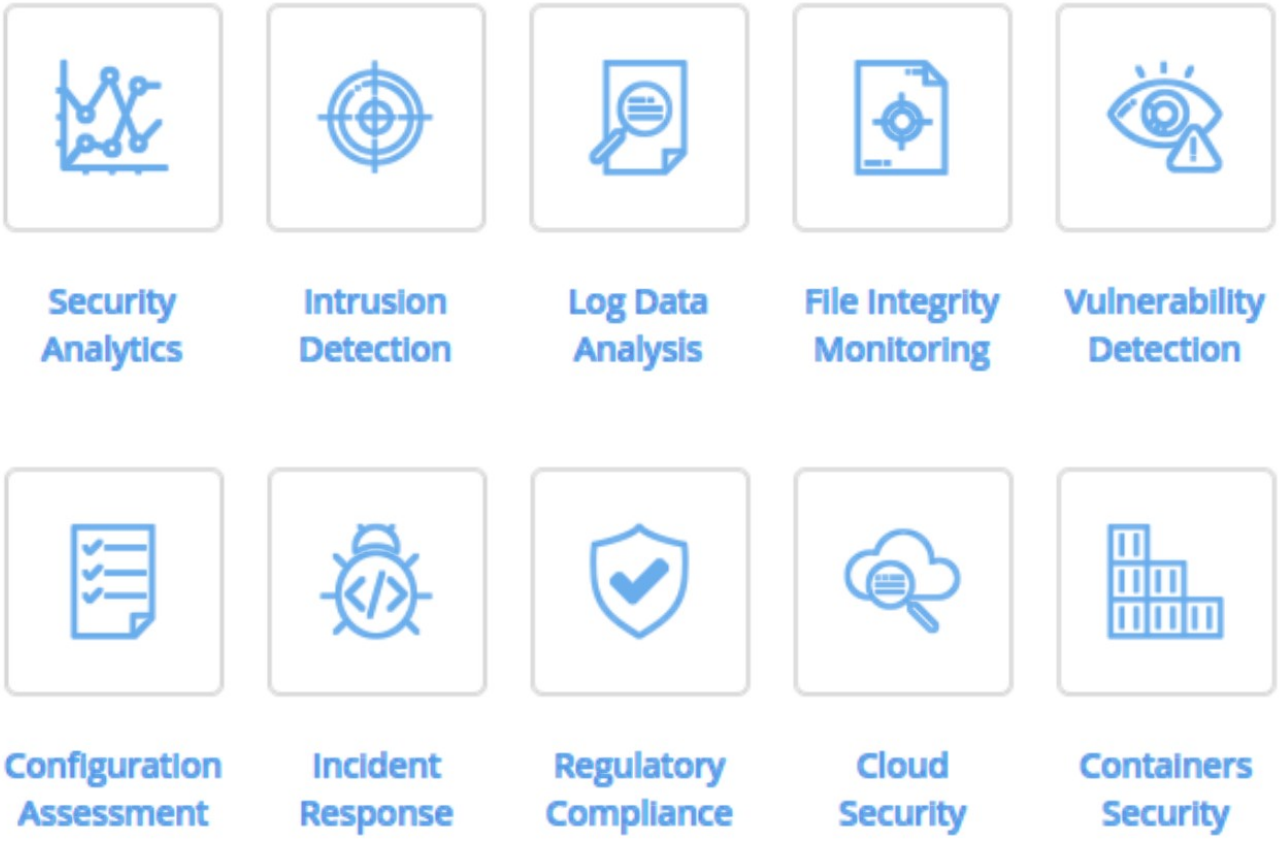
- Azioni predefinite – molte tra cui
  - Vari tipi di attacco ad applicazioni web
  - Attacco di forza bruta agli account via SSH
  - Buffer overflow e terminazioni anomale di processi
  - Eccezioni alle regole di controllo dell'accesso del traffico
  - Utilizzo di sudo
- Creazione di alert personalizzati
  - scattano in funzione delle *rules*
  - possono eseguire
    - logging dell'evento
    - invio di e-mail, sms, …
    - esecuzione di uno script
  - possibilità di **esecuzione su host multipli**

OSSEC funzioni avanzate

- Monitoraggio dell'output di script, ad esempio scan NMAP
  - alert quando un host sotto controllo cambia

  https://ossec.github.io/docs/manual/notes/nmap_correlation.html
- Reportistica
  - summary
  - database per successive elaborazioni
  - web UI (deprecata)
- https://ossec.github.io/

Wazuh functions



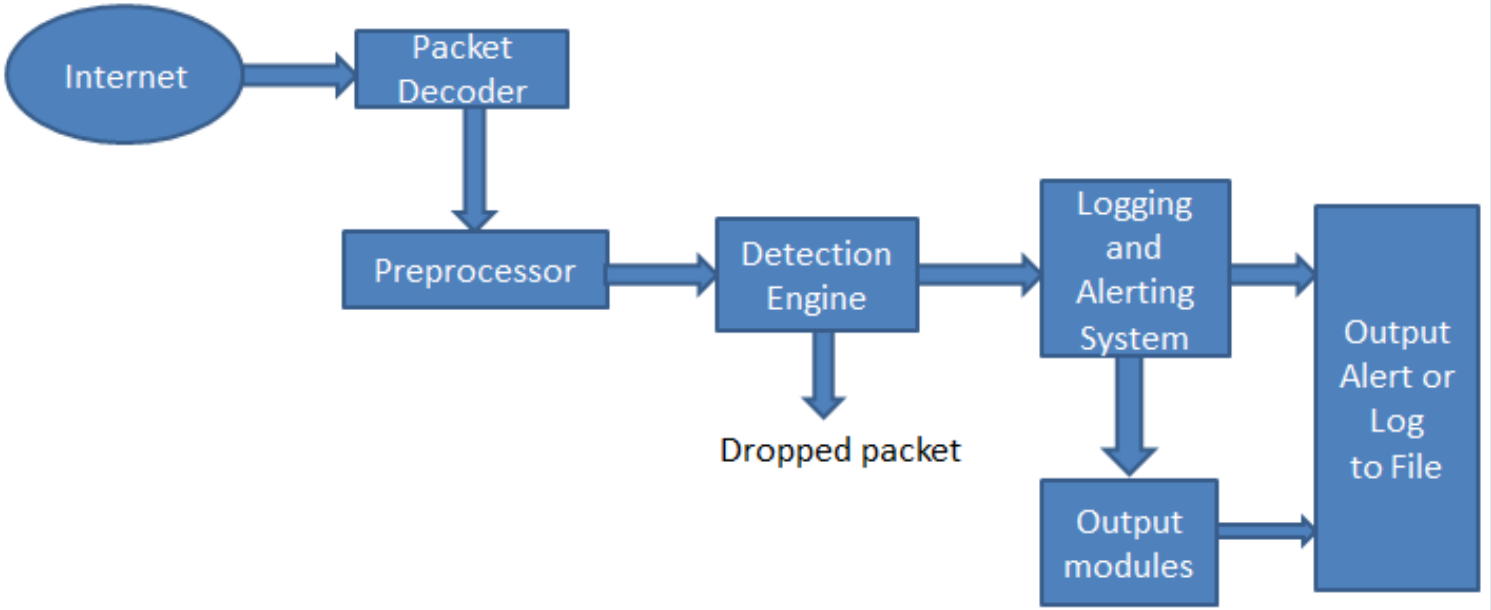
- https://wazuh.com/

Network IDS

- La rilevazione di attacchi che giungono via rete viene svolta analizzando il traffico in entrata/uscita
- Problema essenziale:
  - esaminare tutto il traffico senza rallentarlo
  - generando pochissimi falsi allarmi
  - senza lasciar sfuggire attacchi reali
- Due approcci:
  - Signature based: rileva flussi con caratteristiche notoriamente malevole
  - Anomaly based: rileva flussi che si discostano dalla “normalità”
- Tra i più diffusi
  - Snort
  - Suricata
  - Zeek (ex Bro)

SNORT

- http://www.snort.org/



SNORT

- Architettura di base:
  - Libpcap-based sniffing interface
    - cattura i pacchetti e li salva in un formato standard per l'analisi successiva
  - Rules-based detection engine
    - possibilità di utilizzare un vasto set di regole già pronte e di personalizzarle
  - Plug-in system
    - funzionamento estendibile aggiungendo moduli
- CAVEAT: Snort è uno strumento potente, ma per massimizzare la sua efficacia serve una competenza specifica ed un lungo affinamento della configurazione
  - stima dell'autore originale: 12 mesi di formazione per acquisire i fondamenti di intrusion detection, 24-36 mesi per diventare esperti

SNORT – Detection Engine

- Le regole definiscono le “signature” di un attacco, cioè l'insieme di caratteristiche per riconoscerlo
- Possono essere formate combinando più elementi semplici
- Possono riconoscere una molteplicità di scenari
  - Stealth scans, OS fingerprinting, buffer overflows, back doors, CGI exploits, ecc.
- Il sistema è molto flessibile e la creazione di nuove regole è relativamente semplice

```
alert tcp $EXTERNAL_NET 27374 -> $HOME_NET any (msg:"BACKDOOR
subseven 22"; flags: A+; content: "|0d0a5b52504c5d3030320d0a|";
reference:arachnids,485; reference:url,www.hackfix.org/subseven/;
sid:103; classtype:misc-activity; rev:4;)
```

SNORT – Plug-Ins

La struttura di base permette di attivare diversi moduli per le tre fasi principali

- Preprocessor
  - esamina e manipola i pacchetti prima di passarli al detection engine (evitando ad esempio la scansione di cose ovviamente innocue)
- Detection
  - ogni modulo implementa un singolo test semplice su di un singolo aspetto o parte di un pacchetto
  - può anche essere saltata se si vuole usare Snort solo per salvare traffico interessante da processare successivamente, fuori linea
- Output
  - Riporta i risultati degli altri plug-in (quindi consente la personalizzazione delle destinazioni e dei formati dei messaggi diagnostici)

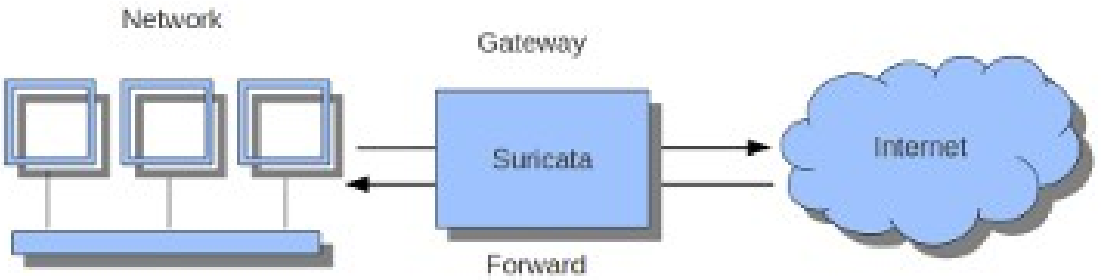
Suricata

https://suricata-ids.org/

- Configurazione:
  - Implementa un linguaggio per la rilevazione di signature
    - Compatibile con SNORT
  - Può essere configurato per rilevare anomalie
  - Può essere esteso con LUA per processing oltre la capacità del linguaggio a regole
- Caratteristiche di funzionamento particolari:
  - Riconosce automaticamente il tipo di traffico e adatta il dettaglio dei log
    - es. salva i certificati X.509 usati nelle connessioni TLS
    - salva l'header a livello applicazione per i protocolli più comuni
      - Deep Packet Inspection
- Output facilmente integrabile con molti strumenti di analisi e visualizzazione
- Molte fonti gratuite di signature
  - Emerging Threats
  - Talos
  - Positive Technology

Suricata

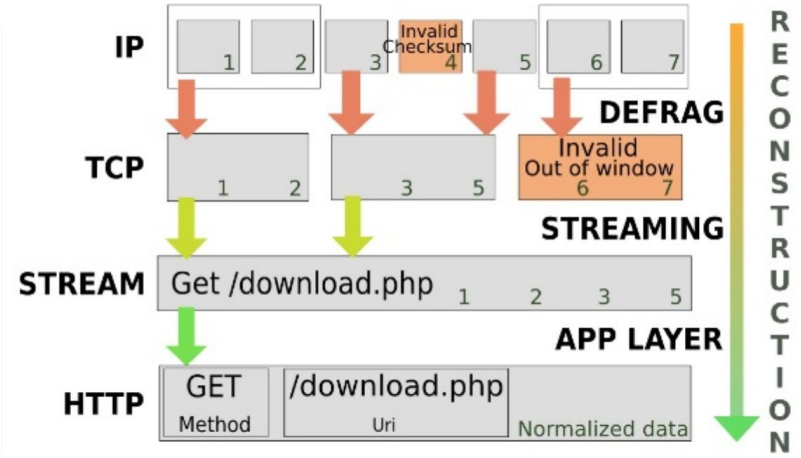
- Suricata può agire da IPS (Intrusion Prevention System)
  - Se interposto tra due reti, può non inoltrare il traffico malevolo

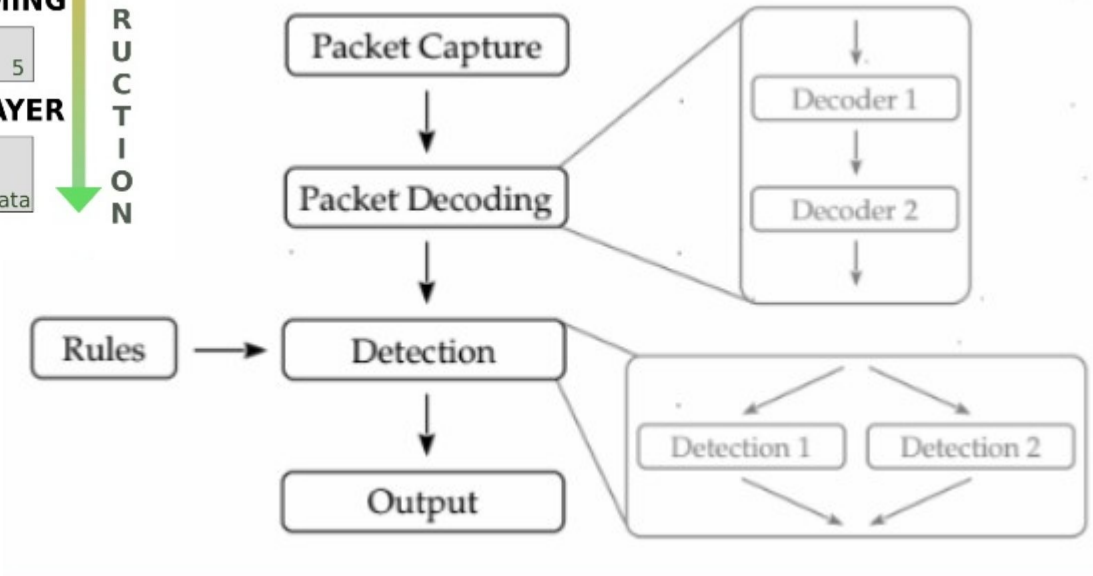


- Regole di pattern matching sul traffico
- Possibilità di operare "sul filo"
  - interfaccia in promiscuous mode
- Possibilità di operare "offline"
  - alimentato da file *pcap*
  - i file possono essere generati in tempo reale da probe integrate su apparati di rete o su host

Suricata

- Sistema modulare per ricostruzione, decodifica ed estrazione dei dati dai pacchetti e successiva classificazione





Zeek

https://zeek.org/

- Decodifica vari protocolli applicativi (HTTP, SSL, DNS, SMB e molti altri)
- Analizza TUTTO il traffico
  - Non utilizza le firme per individuare traffico malevolo
- Ha un motore di scripting molto potente:
  - per estrarre file interessanti
  - usa filtri di Bloom per cercare corrispondenze “intelligenti”
  - supporta la geolocalizzazione
  - blocco attivo della connessione chiamando API del firewall
  - calcolo dell'entropia su letture e scritture SMB per rilevare i ransomware mentre cifrano i file