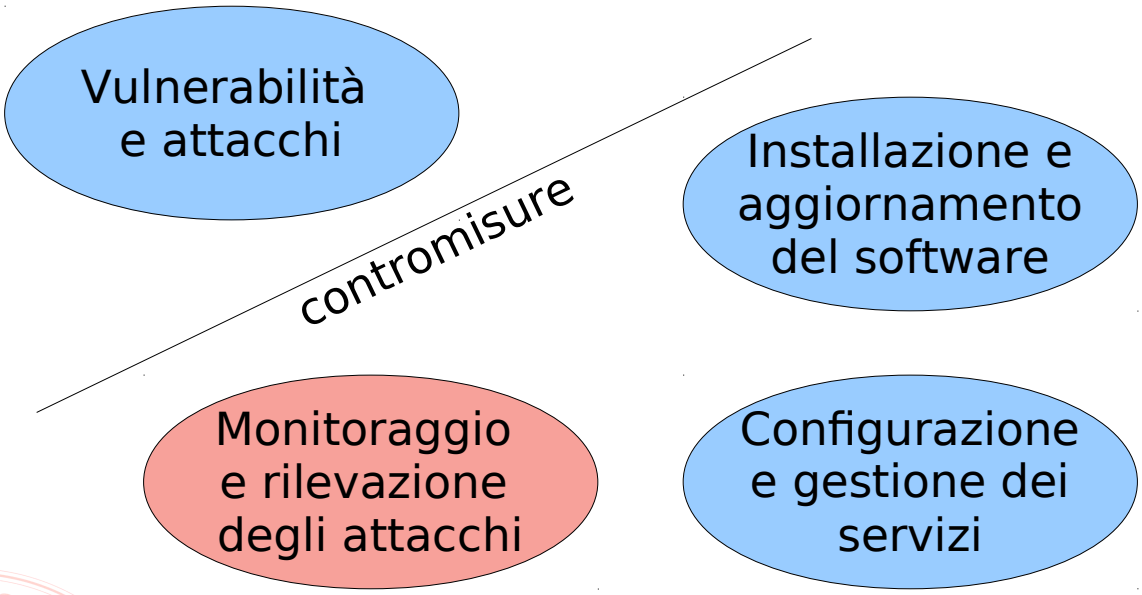
Sicurezza dei sistemi

Qualche esempio delle tecniche usate per violare i sistemi, e delle più comuni contromisure

Marco Prandini

Panoramica

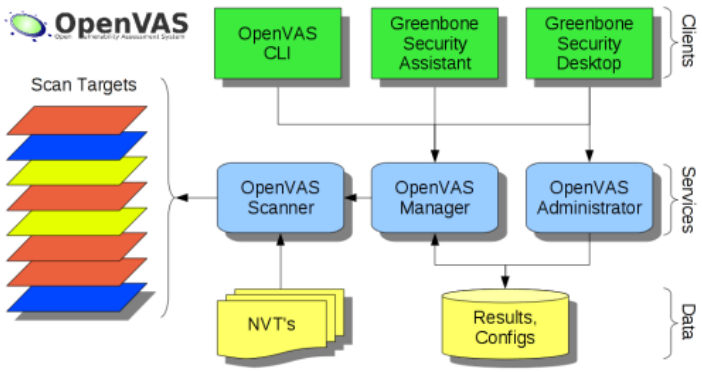


Conoscere i propri punti deboli

- Le vulnerabilità vengono pubblicate secondo il principio della *responsible disclosure*
  - su varie mailing list (es. bugtraq, full disclosure, si veda http://seclists.org/)
  - su archivi e database organizzati
    - Common Vulnerabilities and Exposures http://cve.mitre.org/
    - National Vulnerability Database http://nvd.nist.gov/
    - Open Sourced Vulnerability Database http://osvdb.org/
    - SecurityFocus http://www.securityfocus.com/vulnerabilities
    - US-CERT http://www.kb.cert.org/vuls/
- A livello azienda/organizzazione: CSIRT (Computer Security Incident Response Team)
- A livello nazionale: CERT (Computer Emergency Readiness Team)
  - coordinano lo scambio di informazioni critiche per la sicurezza
  - stabiliscono programmi e procedure di reazione agli attacchi
  - gestiti a livello nazionale, coordinati da FIRST http://www.first.org/

Tenere sotto controllo i sistemi

- Una pratica consigliata è “attaccare” i propri sistemi con gli stessi test di vulnerabilità documentati nei db citati
- Ad esempio: Open Vulnerability Assessment System (OpenVAS) - http://www.openvas.org/



OpenVAS – Caratteristiche principali

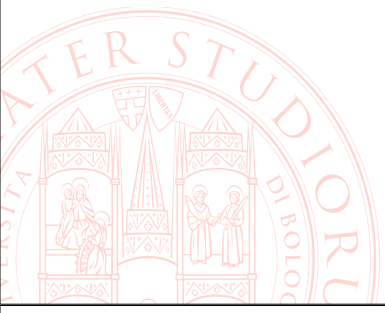
- architettura distribuita che separa
  - il motore di scansione (applica i test ai target)
  - il manager (coordina i task di scansione)
  - l'interfaccia (pianifica i task sul manager e mostra i risultati raccolti)
    - diverse interfacce (cli, grafica, web) usano lo stesso protocollo verso il manager
  - l'amministrazione (gestisce gli utenti e i database)
- database di vulnerabilità
  - Network Vulnerability Tests
    - descrizione della vulnerabilità
    - piattaforme colpite
    - procedura di verifica
  - aggiornato quotidianamente!

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
  - un nome un po' commerciale per racchiudere 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

HIDS vs. NIDS

- Due strategie di rilevazione
  - HIDS / Host-based IDS
  - NIDS / Network-based IDS
- NIDS usa i dati intercettati sui canali di comunicazione
- HIDS può interagire con sistema operativo
  - monitorare il filesystem
  - esaminare in tempo reale i log file,
  - verificare periodicamente contenuti e metadati dei file



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

NIDS

- Vantaggi
  - Visibilità di tutto il traffico, entrante e uscente
  - Richiede un solo punto di installazione
    - vero solo se rete semplice
    - con più sonde: possibilità di ragionare su flussi
  - Un malfunzionamento non incide sugli endpoint
- Svantaggi
  - Maggior tasso di FP
    - processi legittimi posson generare occasionalmente traffico anomalo
  - Più soggetto a sovraccarico o evasione
    - es. pacchetti frammentati
  - Non può esaminare il traffico cifrato

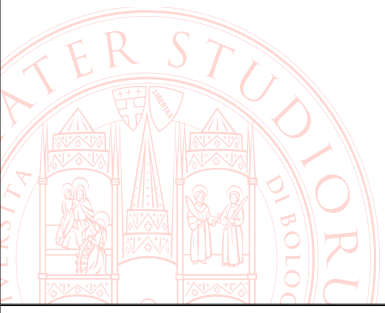
Host IDS

- 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 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

- 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
```

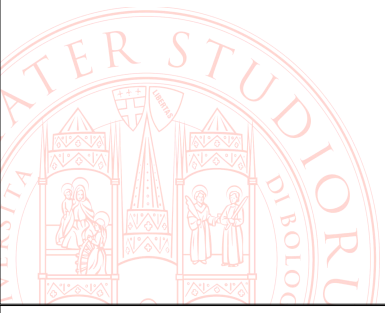
- Caratteristiche
  - Compilato, molto veloce
  - Integrabile con permessi estesi (acl, selinux)

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

AFICK

- Configurazione molto simile ad AIDE ma con un numero minore di check supportati
- Caratteristiche
  - Può essere eseguito da media ottici read only per garantire la sua stessa integrità
  - Report CSV semplici e facili da importare in altri sw
  - Scritto in Perl, molto portabile

http://afick.sourceforge.net/



Log di sistema

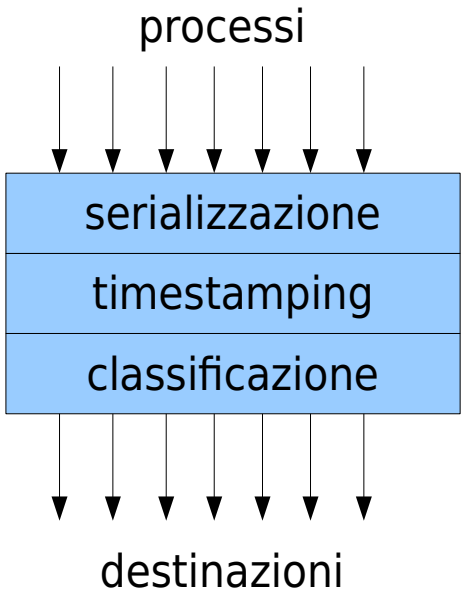
- I log (diari) tenuti dal sistema sono indispensabili per la diagnostica in generale, e in particolare per rilevare attività malevole o sospette
- La loro stessa sicurezza va garantita!
  - Usare appropriatamente un integrity checker
  - Replicarli su macchine remote
- Logging su server remoto
  - Vantaggio 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`

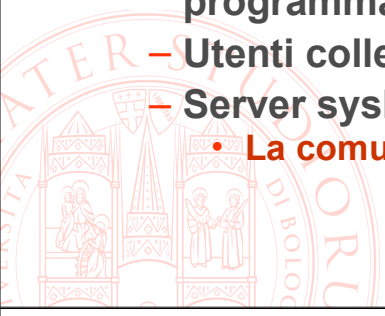
syslog

- 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

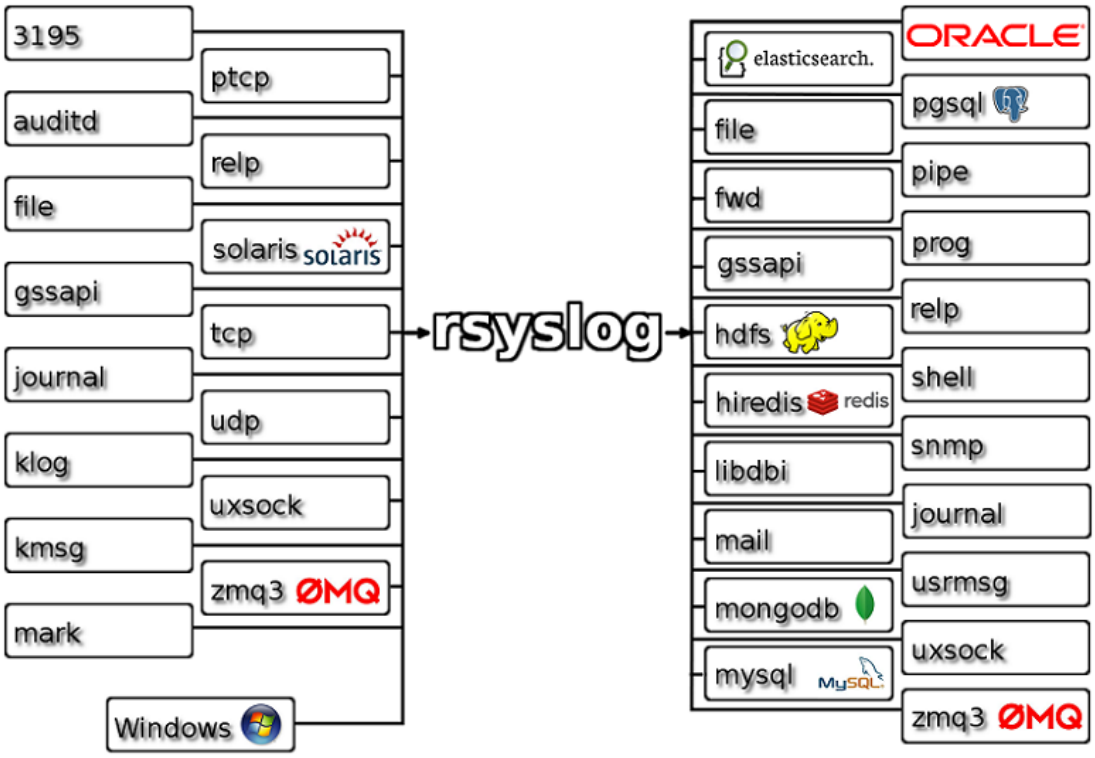


syslog: selettori

- `/etc/syslog.conf` contiene le regole di smistamento dei messaggi
- Ogni riga = una regola
  - [etichetta di interesse] [destinazione]
  - 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



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`
- 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
  - Es:

```
$template apacheAccess,"/var/log/external/%fromhost%/apache/
%msg:R,ERE,1,ZERO:imp:([a-zA-Z0-9\-]+)\.--end%-access.log"
local6.notice ?apacheAccess
```

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

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

- 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 execute
  - Passa il messaggio come parametro a un programma

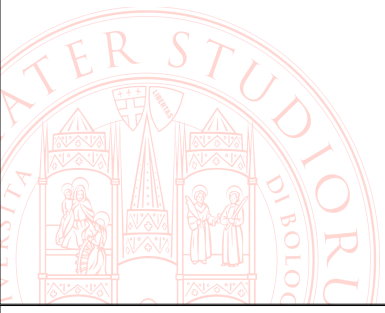
```
• *.*  ^programma;template
```

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!
  - `if expr then destinazione`
  - Diventeranno un sistema completo di scripting, che consentirà di eseguire programmi arbitrari per determinare la destinazione

rsyslog – moduli

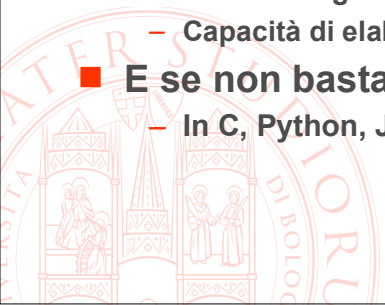
- Troppi per citarli esaustivamente
  http://www.rsyslog.com/doc/v8-stable/
- Tra i più interessanti:
  - RELP logging - per garanzia totale di consegna
    ```
    $ModLoad omrelp
    *.*  :omrelp:indirizzo
    ```
  - Output su tabelle di database
    ```
    $ModLoad ommysql
    ```
  - Acquisizione 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); };
```

Analisi e gestione dei log

- Non basta scrivere gli eventi da qualche parte
- Analisi
  - estrazione del significato dei messaggi
  - serie temporali
  - correlazione file multipli
  - reazione in tempo reale
- Gestione
  - spazio
  - archiviazione

"Vedi la foresta, ma anche gli alberi"
– splunk.com

Software basici di analisi dei log

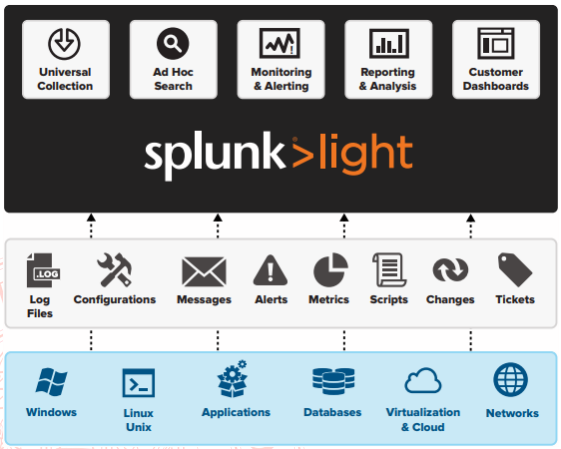
	Logwatch	Swatch
Regular expression support	yes	yes
Real-time monitoring	no	yes
Support for multiple log files	yes	no
Good preconfiguration	yes	no
Modular configuration	yes	no
Reactive	no	yes
Interactive	no	no

Logwatch (System log analyzer and reporter):
http://www.logwatch.org/

Swatch (Simple WATCHer of Logfiles):
http://swatch.sourceforge.net/

Sistemi commerciali

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



Analisi dei log di ispirazione cloud

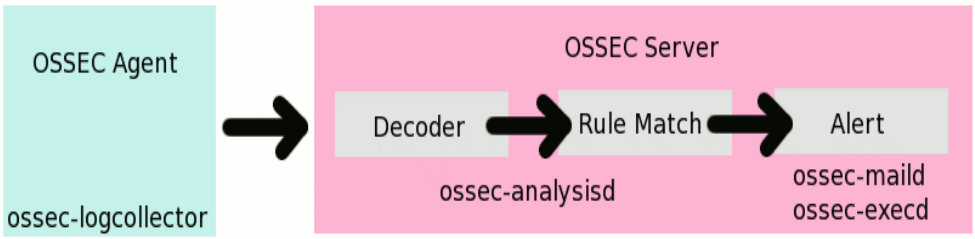
- Stack Elastic, Open Source
  - https://www.elastic.co/
  - Raccolta e esplorazione dei dati di log
- Composto dai tre programmi:
  - Logstash: Pipeline di elaborazione dei log
  - Elasticsearch: Database Nosql
  - Kibana: Visualizzatore web-based per documenti in Elasticsearch

Un esempio di SIEM OSS: OSSEC

- Filesystem integrity checking
- Monitoraggio del registry su Windows
- Reazioni attive
  - Tipicamente: RTBL (Real Time Black Listing)
  - Qualsiasi comportamento è implementabile via script
- Rilevazione di rootkit
- HIDS – non necessita di sonde di rete
- Capacità di correlare eventi

OSSEC

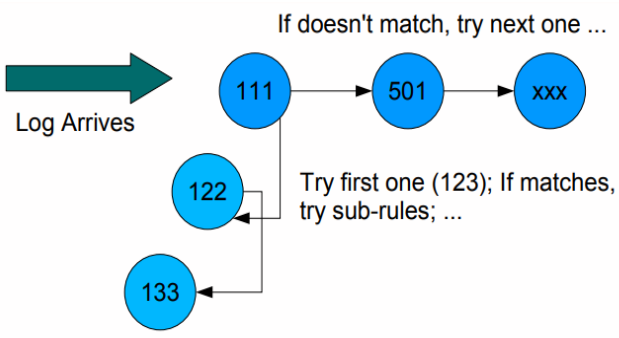
- Due modalità di funzionamento
  - locale, client-server
- Modalità client-server
  - i client ricevono la configurazione da un server
  - i client inviano i log al server su canale cifrato



- 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



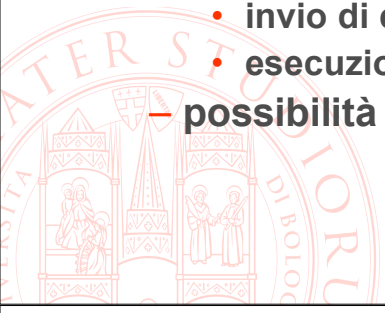
- Esempi

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

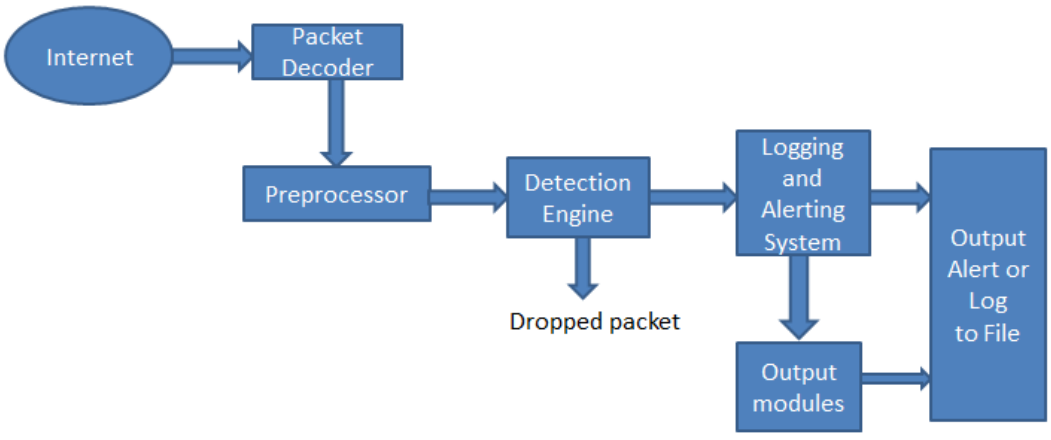
  https://ossec.github.io/docs/manual/notes/nmap_correlation.html
- Reportistica
  - summary
  - database per successive elaborazioni
  - web UI (deprecata)
- https://ossec.github.io/

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
  - 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)

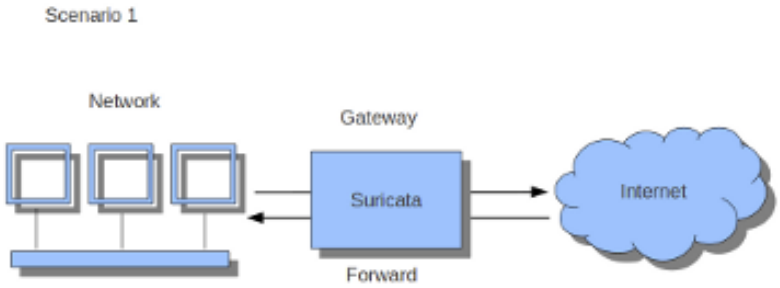
Suricata

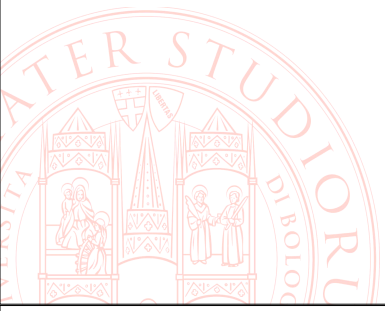
https://suricata-ids.org/

- 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

Suricata

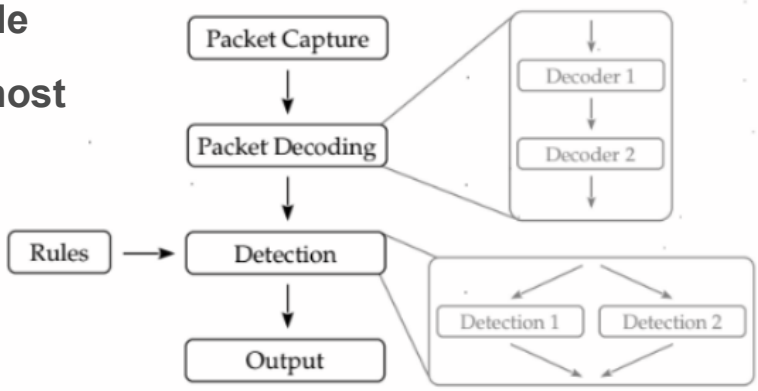
- Suricata può agire da IPS (Intrusion Prevention System)
  - Se interposto tra due reti, può non inoltrare il traffico malevolo





Suricata

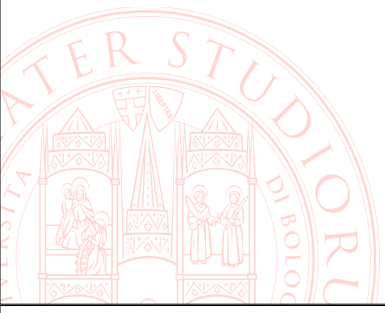
- Regole di pattern matching sul traffico
- Possibilità di operare "sul filo"
  - interfaccia in promiscuous mode
- Possibilità di operare "offline"
  - alimentato da file *pcap*
  - i file possono essere generati in tempo reale da probe integrate su apparati di rete o su host



Bro

https://www.bro.org/

- Un framework per definire piattaforme di analisi del traffico
- Scalabile
  - Sistema distribuito
  - Nodi replicati con load balancing
- Particolarmente usato in ambito accademico



Monitoraggio – in laboratorio

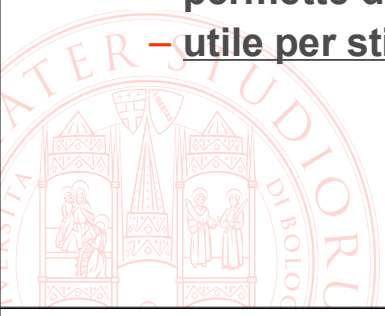
- Comandi essenziali per il monitoraggio

Utenti	File	Processi	Spazio
w last	fuser	ps top uptime	df du free
	lsof		vmstat iostat

- Strumenti essenziali per l'automazione dei task di monitoraggio (ma non solo)
  - esecuzione posticipata: at
  - esecuzione periodica: cron

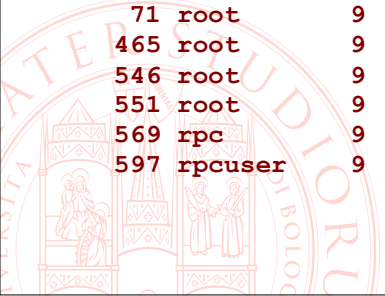
ps – uptime – free → top

- Comandi che scattano un'istantanea del sistema
  - ps: stato dei processi
  - uptime: carico del sistema
    - unità di misura = lunghezza media coda runnable
  - free: occupazione memoria
- Questi comandi sono interfacce verso `proc` filesystem
- 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
```



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:

```
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hdb1               202220    126789     64991  67% /
/dev/hdb4              5558076   3916612   1359124  75% /usr
/dev/hdb3               303336     49822    237851  18% /var
none                    192240         0    192240   0% /dev/shm
/dev/hda1             10231392   9473248    758144  93% /win/c
/dev/hda5              9790032   4247000   5543032  44% /win/d
```

- *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:
  - filesystem speciale **/proc**
    https://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/proc.html

```
# 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:

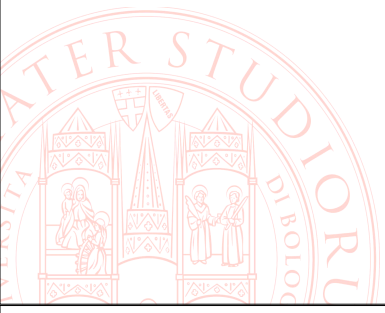
```
# fuser -m /var
/var:                  546   597c   714   714c   879c   898   916   916c
  964  1013  1020  1021  1318  6493  9244  9244m  9249  9249m  9275

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

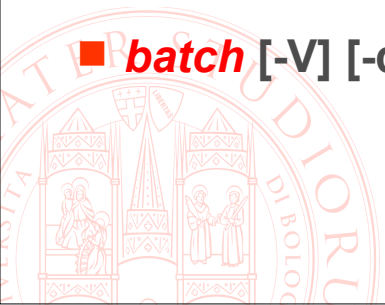
```
COMMAND    PID TID        USER   FD      TYPE             DEVICE SIZE/OFF       NODE NAME

automount  862 870        root    2u      CHR                1,3      0t0       5580 /dev/null
automount  862 870        root    3u      CHR               10,59      0t0       9423 /dev/vboxguest
automount  862 870        root    4uW     REG                0,15        4      12174 /run/vboxadd-service.pid
automount  862 870        root    6u     unix 0xffff88000a746bc0      0t0      12175 socket
rpc.mount  623            root   16u     IPv4               11645      0t0        UDP *:34398
rpc.mount  623            root   17u     IPv4               11794      0t0        TCP *:44328 (LISTEN)
rpc.mount  623            root   18u     IPv6               11798      0t0        UDP *:52821
rpc.mount  623            root   19u     IPv6               11802      0t0        TCP *:36162 (LISTEN)
rpc.mount  623            root   20u     unix 0xffff88000a75ab80      0t0      11809 socket
```



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:

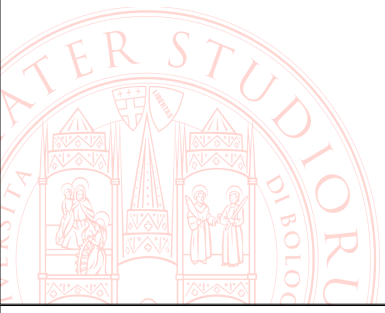
- *at* [-V] [-q queue] [-f file] [-mldbv] TIME
  pianifica un comando al tempo TIME
- *atq* [-V] [-q queue] [-v]
  elenca i comandi in coda
- *atrm* [-V] job [job...]
  rimuove comandi dalla coda
- *batch* [-V] [-q queue] [-f file] [-mv] [TIME]
  esecuzione condizionata al carico



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/doc/at-<versione>/timespec. Alcuni esempi:

```
echo 'wall "sveglia"' | at 08:00
echo "$HOME/bin/pulisci" | at now + 2 weeks
echo "$HOME/bin/auguri" | at midnight 25.12.2018
```



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/<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