Sysdig i Falco

Sysdig:

El programa Sysdig (https://www.sysdig.org) és un programa similar a *strace* però molt més versàtil i complet. No obstant, cal tenir en compte que fa servir un mòdul del kernel (anomenat "sysdig_probe" i responsable, entre altres coses, de generar el dispositiu de captura especial /dev/sysdig0), el qual no està integrat (encara) dins d'ell.

Si Sysdig s'executa (com a root) simplement "tal qual", així, *sysdig* , es mostrarà per pantalla (concretament, a "stdout") tots els events (els quals solen ser crides al sistema, encara que no només) generats en temps real. Cada event és mostrat en una línia diferent amb el següent format

%evt.num %evt.time %evt.cpu %proc.name (%thread.tid) %evt.dir %evt.type %evt.args

on:

evt.num is an incremental event number
evt.time is the event timestamp
evt.cpu is the CPU number where the event was captured
proc.name is the name of the process that generated the event
thread.tid is the TID that generated the event, which corresponds to the PID for single thread processes
evt.dir is the event direction, > for enter events and < for exit events
evt.type is the name of the event, e.g. 'openat' or 'read'
evt.args is the list of event arguments. In case of system calls, these correspond to the system call's ones.

NOTA: An interesting argument which some system call have is the "file descriptor" (fd). A "fd" is a numeric ID that uniquely identifies a file inside a process. This means that you can see the same "fd" number used more than once, but it will have to be in different processes. By following a process/FD combination, you can track specific I/O activity. It's very important to note that in a Linux system, a "file" can be many different things: a regular file, a network connection (socket), the stdout, stderr and stdin streams, a pipe, a timer, a signal, etc

In Sysdig file descriptors are resolved by default. This means that, whenever possible, the FD number is followed by a human-readable representation of the FD itself: the tuple for network connections, the name for files, and so on. The exact format used to render an FD is the following: num(<type>resolved_string) where: *num* is the FD number; *resolved_string* is the resolved representation of the FD (e.g. 127.0.0.1:40370->127.0.0.1:80 for a TCP socket) and *type* is a single-letter-encoding of the fd type, and can be one of the following: "f" for files, "4" for IPv4 sockets, "6" for IPv6 sockets, "u" for unix sockets,"s" for signal Fds, "e" for event FDs, "i" for inotify FDs and "t" for timer FDs

NOTA: For most system calls, sysdig shows two separate entries: an enter one (marked with a '>') and an exit one (marked with a '<'). This makes it easier to follow the output in multi-process environments.

Alguns dels paràmetres més comuns per utilitzar amb sysdig són:

-w /ruta/nomfitxer : Guarda la sortida en el fitxer indicat (en comptes de per stdout). En el cas de combinar aquest paràmetre amb els paràmetres -C o -G o -e (veure més avall) cal establir un criteri per nombrar de forma diferent als múltiples fitxers generats; un podria ser la data, així:
-w /ruta/nomfitxer-$(date +"%Y-%m-%d %H:%M:%S")
-C nºMB : Parteix el fitxer de captura en fitxerets diferents amb un tamany cadascú dels MB indicats
-G nºs : Parteix el fitxer de captura en fitxerets diferents cada número de segons indicat
-e nº : Parteix el fitxer de captura en fitxerets diferents cada número d'events detectats
-W nº : Limita el número de fitxerets petits creats amb -C, -G o -e, realitzant una rotació de logs.
-n nº : Limita el número d'events capturats a la quantitat indicada
-t {h|a|r|D}: Canvia la manera com es mostrarà el camp *evt.time*. Els valors indiquen, respectivament: format humà, format "timestamp" absolut des de 1-1-1970, format "timestamp" relatiu des de que va començar la captura, format "delta" des de l'event anterior
-r /ruta/nomfitxer : Llegeix (i mostra per pantalla) les captures guardades prèviament al fitxer indicat

Els filtres s'han d'escriure al final de la invocació de *sysdig* , com a últims paràmetres. Els filtres consisteixen simplement en indicar el nom del camp i el valor concret d'aquest que es vol observar. Per exemple, per veure l'activitat d'una determinada comanda (que pot estar ja funcionant o bé encara no) cal fer:

sysdig proc.name=cat

Als filtres es poden usar els següents operadors de comparació : =, !=, <, <=, >, >=, "contains", "icontains" (case-insensitive), "in" i "exists" . També es poden combinar entre ells mitjançant operadors booleans com: "and", "or" i "not" (i parèntesis). Per exemple, la següent comanda captura l'activitat de dos programes a la vegada, *cat* i *vi*:

sysdig proc.name=cat or proc.name=vi

El següent exemple mostra tots els fitxers que són oberts per programes que no siguin el *cat:*

sysdig proc.name!=cat and evt.type=openat

El següent exemple mostra totes les connexions de xarxa rebudes per processos diferents de Apache:

sysdig proc.name!=apache and evt.type=accept

El següent exemple mostra totes les crides "execve" només si les realitza el procés pare "bash":

sysdig evt.arg.ptid=bash and evt.type=execve

NOTA: La diferència entre el filtre *evt.arg* i el filtre *evt.rawarg* és que el segon no fa resolució de PIDs, fds o codis d'error, deixant el valor en la seva forma numèrica "pelada" . Per exemple, es podria usar el filtre *evt.arg.res=ENOENT* per observar un codi de retorn d'error I/O específic o bé, ja que els codis d'errors són números negatius, es podria fer alternativament *evt.rawarg.res < 0 or evt.rawarg.fd < 0*

Per conèixer tots els filtres possibles, es poden llistar fent *sysdig -l*

Per conèixer tots els events reconeguts, es poden llistar fent *sysdig -L* . En general, una llista de les crides al sistema més usades (amb una breu explicació de què fan i com funcionen) es troba a https://sysdig.com/blog/fascinating-world-linux-system-calls. Una llista molt més completa es troba a https://github.com/gregose/syscall-table .No obstant, si es vol accedir a la documentació completa per saber què fa, quins paràmetres té, quins valors de retorn té cadascuna de les crides al sistema existents, no hi ha res millor que consultar les pàgines del manual (concretament la secció 2, així: *man 2 read*, pe exemple).

NOTA: La secció 3 del manual està reservada per a les funcions de la llibreria estàndar. Es poden veure les diferents seccions del manual fent *man man*

Es pot personalitzar la sortida de *sysdig* es pot usar el paràmetre *-p* , el qual té una sintaxis similar a la de la funció printf() de C. Per exemple:

*sysdig -p"*Usuari:%user.name Carpeta on s'entra:%evt.arg.path" evt.type=chdir*

NOTA: L'asterisc del principi de la cadena serveix per indicar a Sysdig que imprimeixi la frase igualment encara que l'event no aporti els valors de tots els camps indicats a la frase (per defecte Sysdig no ho fa)

La personalització de la sortida es pot combinar amb els filtres que ja coneixem. Per exemple, la comanda següent mostra la sortida estàndar del procés cat (el paràmetre **-A** serveix per mostrar només la informació "humanament" llegible i no el contingut binari a pèl, i el paràmetre **-s** serveix per indicar que es volen capturar de cada crida a write() més dels usuals 80 bytes que Sysdig s'autoimposa)

sysdig -A -s 65000 -p"%evt.buffer" proc.name=cat and evt.type=write and fd.num=1

Un altre cas: podríem adaptar un exemple vist en línies anteriors per fer-ho més complet:

sysdig -p"%user.name) %proc.name %proc.args" evt.type=execve and evt.arg.ptid=bash

Per veure qui i amb quin programa està escrivint als fitxers ubicats dins de /etc podríem fer:

sysdig -p"User:%user.name Prog:%proc.name Fitx:%fd.name" evt.type=write and fd.name contains /etc

Igualment, per veure les connexions acceptades (Ip:port origen <-> Ip:port destí) per Apache...:

sysdig -p"%fd.name" proc.name=apache and evt.type=accept

A banda dels filtres, una altra funcionalitat molt interessant de Sysdig són els seus (anomenats per ells) "chisels". Els "chisels" són petits scripts (escrits en Lua) que analitzen el flux d'events capturats per Sysdig per respondre a determinats events realitzant determinades accions. Per veure els "chisels" disponibles cal fer:

sysdig -cl

Per obtenir informació sobre què fa i quins paràmetres admet un "chisel" determinat, cal escriure:

sysdig -i nomChisel ***<--Molt interessant!!***

Per executar un determinat "chisel" cal fer servir el paràmetre -c, així (el "chipsel" *topfiles_bytes* mostra en temps real els fitxers més accedits -llegits i/o escrits- a la màquina).:

sysdig –c topfiles_bytes

Si el "chisel" necessités paràmetres, s'han d'indicar després del seu nom. Per exemple, el "chisel" *spy_ip* necessita la direcció IP de l'extrem del qual es monitoritzarà en temps real els paquets intercanviats amb ell:

sysdig –c spy_ip 192.168.1.157

Es poden executar varis "chisels" a la vegada, si s'escau. També es poden combinar amb filtres, indicats al final, opcionalment entre cometes. Per exemple, si no volem que a la sortida del "chisel" *topfiles_bytes* apareguin els accessos a /dev, podríem fer:

sysdig –c topfiles_bytes "not fd.name contains /dev"

O bé voldríem veure només els arxius més accedits en una determinada carpeta, així:

sysdig –c topfiles_bytes "fd.name contains /root"

O bé voldríem veure només els arxius més accedits per un determinat programa:

sysdig –c topfiles_bytes "proc.name=vi"

O bé voldríem veure només els arxius més accedits per un determinat usuari:

sysdig –c topfiles_bytes "user.name=bob"

Un altre "chisel" interessant és *lsof*, el qual mostra els fitxers actualment oberts i quin procés l'obre. Podríem filtrar només, per exemple, els fitxers de registre de l'Apache (error.log, access.log), així ; noteu les cometes simples dins de les dobles (aquest "chilsel" les necessita):

sysdig –c lsof "'proc.name contains apache2 and fd.name contains log'"

O bé mostrar només la llista de connexions de xarxa establertes per l'usuari "root":

sysdig –c lsof "'fd.type=ipv4 and user.name=root'"

Un "chisel" que emula el comportament de la comanda *ps* és l'anomenat ps. Es pot fer servir així (en aquest cas estaríem veient els processos que estiguin escoltant a la xarxa en ports diferents del 80) ; noteu les cometes simples dins de les dobles (aquest "chilsel" les necessita):

sysdig –c ps "'fd.type=ipv4 and fd.is_server=true and fd.sport!=80'"

O bé mostrar la llista de processos que tenen oberts fitxers dins de la carpeta /etc:

sysdig -c ps "'fd.name contains /etc'"

O bé mostrar tots els processos que no s'anomenin "bash":

sysdig -c ps "'not(proc.name contains bash)'"

Els "chisels" stdin i stdout es poden fer servir per mostra l'entrada i sortida estàndar de les comandes executades a partir de llavors (o de les indicades amb filtres). Per exemple:

sysdig -c stdout "proc.name=cat"

NOTA: Si es vol escriure un "chisel" propi, un bon tutorial per començar es troba aquí:
https://github.com/draios/sysdig/wiki/Writing-a-Sysdig-Chisel,-a-Tutorial .
L'API de referència es troba aquí: https://github.com/draios/sysdig/wiki/sysdig-chisel-api-reference-manual

Finalment, dir que una alternativa integrada dins del kernel (més completa però més difícil que Sysdig) és BPF. Una aplicació d'usuari que la fa servir "per sota" que facilita el seu ús és BCC (https://github.com/iovisor/bcc) . Una altra alternativa és Lttng (http://lttng.org).

EXERCICIS:

1.-a) Instal.la Sysdig a una màquina virtual qualsevol executant aquest conjunt de comandes:
curl -s https://s3.amazonaws.com/download.draios.com/stable/install-sysdig | sudo bash

NOTA: Altres maneres diferents de fer la instal.lació de Sysdig (fent servir paquets .deb/.rpm, compilant des del codi font, etc) es poden estudiar a https://github.com/draios/sysdig/wiki/How-to-Install-Sysdig-for-Linux

b) Segueix els passos indicats a l'enllaç següent per aconseguir que no calgui executar *sysdig* directament com root: https://github.com/draios/sysdig/wiki/How-to-Install-Sysdig-for-Linux#user-content-use-sysdig-as-non-root

c) Prova els següents exemples (tots mostrats a la teoria) i digues què has de fer per a què mostrin els events pertinents a pantalla:

sysdig proc.name=cat
sysdig proc.name=cat or proc.name=vi
sysdig proc.name!=cat and evt.type=openat
sysdig proc.name!=apache and evt.type=accept
sysdig evt.arg.ptid=bash and evt.type=execve
*sysdig -p"*Usuari:%user.name Carpeta on s'entra:%evt.arg.path" evt.type=chdir*
sysdig -A -s 65000 -p"%evt.buffer" proc.name=cat and evt.type=write and fd.num=1
sysdig -p"%user.name) %proc.name %proc.args" evt.type=execve and evt.arg.ptid=bash
sysdig -p"User:%user.name Prog:%proc.name Fitx:%fd.name" evt.type=write and fd.name contains /etc

NOTA: The `switch` event is generated every time there is a context switch, i.e. when the process scheduler puts a thread to sleep to execute another one. Observing scheduler activity is useful for scenarios such as determining when a process/thread runs with high granularity, debugging process synchronization issues, and monitoring when a thread is migrated to a different CPU. But switch event can also be annoying at times, because sysdig tends to print many of them

even on systems with relatively low activity. Fortunately, it's easy to filter it out if you don't need it by writing : *sysdig evt.type!=switch*

d) Encara que existeixen eines més especialitzades en aquest àmbit (com ara el Wireshark), Sysdig també pot servir per esbrinar què passa amb les connexions de xarxa en el nostre sistema. En aquest sentit, prova els següents exemples i digues què has de fer per a què mostrin els events pertinents a pantalla:

sysdig fd.type=ipv4
sysdig fd.l4proto=tcp
sysdig fd.sip=127.0.0.1
sysdig fd.sport=22

e) Vés a https://github.com/draios/sysdig/wiki/Sysdig-Examples i escull tres exemples que et cridin l'atenció. Prova'ls i explica què fan

2.-a) Prova i digues què mostren els següents "chisels" (tots mostrats a la teoria)

sysdig –c topfiles_bytes
sysdig –c topfiles_bytes "not fd.name contains /dev"
sysdig –c topfiles_bytes "proc.name=vi"
sysdig –c topfiles_bytes "user.name=bob"
sysdig –c spy_ip 192.168.1.157
sysdig –c lsof "'proc.name contains apache2 and fd.name contains log'"
sysdig –c lsof "'fd.type=ipv4 and user.name=root'"
sysdig –c ps "'fd.type=ipv4 and fd.sport!=80'"
sysdig -c ps "'not(proc.name contains bash)'"
sysdig -c stdout "proc.name=cat"

b) Prova i digues què mostren els següents "chisels"

sysdig -c spy_users
sysdig -c spy_file
sysdig -c httptop
sysdig -c netstat

c) Utilitza els paràmetres -cl i -i de *sysdig* per descobrir tres "chisels" més que consideris interessants. Menciona'ls i explica què fan

Sysdig-inspect és una eina complementària que, a partir de les dades emmagatzemades en un fitxer de captura d'events prèviament obtingut per Sysdig, en realitza un estudi estadístic detallat, oferint com a resultat un panell gràfic molt intuïtiu que permet estudiar de forma "forense" l'aparició, comportament i característiques d'aquests diversos events al llarg del temps.

3.-a) Crea, a la màquina virtual que estiguis fent servir, un fitxer de captura amb la comanda *sysdig -n 1000 -w hola.scap*

b) Ara tens dues opcions:

*Si tens entorn gràfic a la màquina virtual on tinguis Sysdig funcionant, descarrega dins d'aquesta màquina virtual el paquet adient (.deb o .rpm, segons sigui la distribució que estiguis fent servir) corresponent al programa Sysdig-inspect (disponible a https://github.com/draios/sysdig-inspect) , i instal.la'l.

*Si no tens entorn gràfic a la màquina virtual on tinguis Sysdig funcionant, descarrega dins del teu sistema Ubuntu real el paquet adient .deb corresponent al programa Sysdig-inspect (disponible a https://github.com/draios/sysdig-inspect), extreu el seu contingut amb un descompressor qualsevol. L'executable del programa el trobaràs dins de l'arxiu "data.tar.gz" (que caldrà descomprimir també),

a l'interior de la carpeta "/usr/lib/" amb el nom de "Sysdig Inspect". En aquest darrer cas, però, hauràs d'aconseguir copiar el fitxer de captura obtingut a l'apartat anterior a la màquina real d'alguna manera (el més fàcil, tenint en compte que si has fet els exercicis anteriors ja hauries de tenir un servidor SSH funcionant a la màquina virtual, és executar una comanda similar a aquesta a la màquina real: *scp* usuari@ip.maq.virt.ual*:/ruta/arxiu/hola.scap /ruta/carpeta/real*)

c) Executa Sysdig-inspect i obre l'arxiu hola.scap.

Veuràs que a la finestra apareixen quatre seccions: la barra superior permet tenir en tot moment clar en quina secció d'informació estem (similar al que seria la barra de direccions d'un gestor de fitxers); la barra inferior mostra la longitud temporal de la captura i permet restringir la vista de la informació a un determinat rang; la zona central és la més important perquè és on es mostren en forma de "quadrets" els valors més important de les mètriques, classificades segons àmbits i és on es pot sel.leccionar el "quadret" (o "quadrets") desitjat/s per explorar més profundament la informació en ell continguda; i finalment, la barra de l'esquerra mostra les diferents vistes d'informació ("Fitxers", "Directoris", "Connexions", "Processos", "Errors",etc) que es poden activar per veure-hi de maneres diferents el mateix contingut sel.leccionat a la zona central, a més del botó "I/O Streams (que permet veure el contingut dels fitxers o paquets de xarxa associats a una sel.lecció) i el botó "Syscalls" (que permet veure directament les crides al sistema associades a una sel.lecció). Trobareu més informació sobre Sysdig-inspect a https://sysdig.com/blog/sysdig-inspect-explained-visually/

d) ¿Què mostra la barra temporal si sel.lecciones un o més "quadrets" de la zona central?

e) ¿Quina informació veus a la vista que es mostra per defecte si fas doble clic sobre el quadret de "Running processes"? ¿I què veus si cliques llavors a la vista "Files" (tot mantenint el quadret "Running processes" activat...això ho pots comprovar a la barra superior)?

f) Torna a la pàgina principal i desel.lecciona tots els "quadrets". ¿Quina informació veus a la vista que es mostra per defecte si ara fas doble clic sobre el quadret de "File Bytes In+Out"? ¿I què veus si cliques seguidament a la vista "System Calls" (mantenint el quadret "File Bytes In+Out" activat)?

Falco:

Falco (https://falco.org) is a behavioral activity monitor designed to detect anomalous activity in your applications. It lets you continuously monitor and detect container, application, host, and network activity... all in one place, from one source of data, with one set of rules. It can detect and alert on any behavior that involves making Linux system calls (collected by underlying Sysdig's infrastructure).For example, you can easily detect things like:

*A shell is run inside a container
*A server process spawns a child process of an unexpected type
*Unexpected read of a sensitive file (like /etc/shadow)
*A non-device file is written to /dev
*A standard system binary (like ls) makes an outbound network connection

By default Falco looks for its configuration in two yaml files: "/etc/falco/falco.yaml" and "/etc/falco_rules.yaml" (but their paths can be dynamically configured using the flags: *falco -c <config file> -r <rules file>*). The first file controls several logging and high-level configuration items (for instance, there are several options for configuring security event output -to a file, to syslog, to stdout/stderr or to a piped spawned program (like an email client, for instance)- along with two options for formats -text vs JSON-, etc). The second file serves to define the default rules which will be used to detect specific events about the system behaviour and, consequently, to trigger associated alerts.

NOTA: As we said, "falco_rules.yaml" file contains a predefined set of default rules designed to provide good coverage in a variety of situations; the intent is that this rules file is not modified as it is replace with each new software version. But if you want to add/override/modify these rules you should use the local falco rules file, installed at

"/etc/falco/falco_rules.local.yaml" (and empty by default). This file will not be replace with each new software version and it's read after "falco_rules.yaml", so rules defined in "falco_rules.local.yaml" take precedence.

Default Falco rules provided by "falco_rules.yaml" file (and customized rules written in "falco_rules.local.yaml" file by us) can be very different in scope but they all follow the same definition pattern. For instance, following rule is one called "Modify binary dirs":

```
- rule: Modify binary dirs
  desc: An attempt to modify any file below a set of binary directories
  condition: (bin_dir_rename) and modify and not package_mgmt_procs and not exe_running_docker_save
  output : >
    File below known binary directory renamed/removed (user=%user.name command=%proc.cmdline
    pcdline=%proc.pcmdline operation=%evt.type file=%fd.name %evt.args)
  priority: ERROR
```

As you can see, there are a few elements to building a rule. Let's look at these fields in detail:

rule: the identifier of the rule.
desc: a description of the rule, e.g. "rule for alerting on network traffic"
condition: a rule written in the Sysdig filtering language and, optionally, using Falco macros
output: The output message for the rule in Sysdig's output formatting.
priority: Nivell de prioritat de la regla (el seu valor pot ser "debug", "info", "notice", "warning", "error", "critical", "alert" o "emergence"). Si el seu valor és igual o més greu que l'indicat a la línia priority de l'arxiu falco.yaml, la regla associada es tindrà en compte; si no, la regla s'ignorarà

A Falco macro is a name assigned to a set of elements which match a specified condition. For instance, in above rule there was the "bin_dir_rename" macro which is defined at the beginning of "falco_rules.yaml" file like this:

```
- macro: bind_dir_rename
  condition: >
     evt.arg[1] startswith /bin/ or
     evt.arg[1] startswith /sbin/ or
     evt.arg[1] startswith /usr/bin or
     evt.arg[1] startswith /usr/sbin
```

There's predefined lists in "falco_rules.yaml" file, too. Its definition begins by – *list:* mark.

NOTA: Teniu molta més informació sobre com definir regles, macros i llistes a la documentació oficial, disponible a https://github.com/falcosecurity/falco/wiki/Falco-Rules

5.- a) Instal.la Falco a una màquina virtual qualsevol executant aquest conjunt de comandes:
curl -s https://s3.amazonaws.com/download.draios.com/stable/install-falco | sudo bash

NOTA: Altres maneres diferents de fer la instal.lació de Falco (fent servir paquets .deb/.rpm, compilant des del codi font, etc) es poden estudiar a https://github.com/falcosecurity/falco/wiki/How-to-Install-Falco-for-Linux

b) Observa el contingut del fitxer de configuració general de Falco (/etc/falco/falco.yaml) i digues per a què serveixen les següents línies:

rules_file:
- /etc/falco/falco_rules.yaml
- /etc/falco/rules.d

json_output: false

log_syslog: true

log_level: info

priority: debug

syslog_output:
enabled: true

program_output:
enabled: true
program: mail -s "Falco notification" destino@gmail.com

NOTA: Teniu més informació a https://github.com/falcosecurity/falco/wiki/Falco-Configuration

c) Es pot executar Falco de dues maneres: o bé en forma de dimoni en segon pla (*sudo systemctl start falco*) o bé en primer pla (*sudo falco*). Fes-ho d'aquesta segona manera (per això hauries de tenir el dimoni apagat) i, en un altre terminal diferent, prova de modificar un fitxer que estigui dins de la carpeta /etc. ¿Què veus al terminal on s'està executant Falco? ¿I si ara crees un nou fitxer dins de la carpeta /bin, què veus?. Finalment, atura Falco amb CTRL+C

d) Observa el contingut del fitxer "falco_rules.yaml" i respon:

NOTA: Els valors que apareguin de "evt.dir" a les macros i regles estudiades no seran importants per nosaltres
NOTA: Trobareu més exemples a https://github.com/falcosecurity/falco/wiki/Falco-Examples

*¿Què representa la macro "etc_dir" i la macro "write_etc_common" (on s'usa la primera)?
*¿Què representa la llista "passwd_binaries" (usada també a la macro "write_etc_common") i què indica el comentari que hi ha just a sobre de la definició d'aquesta llista?
*¿Quin missatge mostra la regla "Write below etc" (on s'usa la macro "write_etc_common") i quan?

*¿Què representa les macros "bin_dir_rename" i "bin_dir_mkdir"?
*¿Quin missatge mostra la regla "Modify binary dirs" (on s'usa la macro "bin_dir_rename") i quan?

*¿Què representa la llista "bin_dir"? ¿I la macro "open_write"? ¿I la macro "package_mgmt_procs"?
*¿Quin missatge mostra la regla "Write below binary dir" (on s'usen la llista i macros anteriors) i quan?

*¿Què representa la llista "sensitive_file_names" i la macro "sensitive_files" (on s'usa aquesta llista)?
*¿Quin missatge mostra la regla "Read sensitive file trusted after startup" (on s'usa la macro anterior) i quan?

*¿Què representa la macro "inbound"? ¿I la macro "outbound"? ¿I la macro "inbound_outbound"?
*¿Quin missatge mostra la regla "Disallowed SSH Connection" (on s'usen les macros anteriors) i quan?
*¿Què indica el comentari que apareix sobre la macro "allowed_ssh_hosts" (usada a la regla anterior)?

*¿Quin missatge mostra la regla "System procs network activity" (i quan)?
*¿Quin missatge mostra la regla "Create files below dev" (i quan)?

e) Escriu el següent contingut dins del fitxer "falco_rules.local.yaml" i digues per a què serviria. Prova'l:

- list: unalistadeprogramas
items: [ls, cat, pwd]
- macro: uneventoconcreto
condition: evt.type=openat
- rule: pepita
desc: adescubrir
condition: proc.name in (unalistadeprogramas) and uneventoconcreto
output: misterio misterio (user=%user.name command=%proc.cmdline file=%fd.name)
priority: INFO

f) Modifica el contingut de l'arxiu "falco.yaml" per tal de què la seva sortida sigui en format JSON i la envïi (via netcat) a un servidor remot escoltant al port 1234 (consulta els comentaris que apareixen en aquest fitxer a sobre de les línies implicades per tenir una pista.

fBIS) ¿Què passaria si, en comptes de fer servir el client netcat a la línia "program" utilitzada a l'apartat anterior, indiquessis la següent comanda: *curl -X POST -H "Content-Type:application/json" -d @- https://ptsv2.com/t/pepito/post*

> NOTA: If you start the data posted by -d parameter with the letter @, the rest should be a file name to read the data from, or "-" if you want curl to read the data from stdin
> NOTA: If "keep_alive" line in "falco.yaml" file is true, program's network connection (and program itself) will be kept running forever; if it's false (default value), program's network connection (and program itself) will be closed once data stream is sent.
> NOTA: L'objecte JSON que treu Falco té els següents camps: "time" (el moment quan s'ha emés l'alerta, en format ISO8601), "rule" (el nom de la regla que ha causat l'alerta), "priority" (la prioritat d'aquesta regla), "output" (la sortida de la regla) i "output_fields" (tots els noms i valors dels diferents filtres Sysdig utilitzats a la sortida de la regla)
> NOTA: Trobareu més informació a https://github.com/falcosecurity/falco/wiki/Falco-Alerts

fTRIS) Si volguessis enviar un correu per cada alerta, hauries de deixar a la línia "program" la comanda *mail ...* allà suggerida però substituint la direcció de mostra que hi apareix per una altra: la direcció de destí real on es vulgui rebre els missatges (i, opcionalment, substituir també el valor del paràmetre -s, que indica l'assumpte que tindran tots aquests missatges). No obstant, per a què aquesta comanda *mail* funcioni, cal haver fet prèviament un seguit de passos indicats al quadre següent (els quals consisteixen, bàsicament, en instal.lar i configurar al nostre sistema un servidor enviador de correus -Postfix- i configurar Gmail per a què sigui ell qui reenvïi finalment a la bústia de destí aquests correus originats des de Postfix, ja que Postfix no ho pot fer directament per no ser un servidor reconegut pels demés servidors d'Internet). Fes aquests passos i comprova que, efectivament, cada cop que s'activi una alerta (per exemple, en modificar un fitxer de dins de la carpeta /etc) s'enviarà un correu a la bústia de la direcció indicada com a destinatari.

0.-By default, Gmail doesn't permit regular applications to send mails through it in order to prevent spam and misuse. To disable this protection, you must login manually into the Gmail account you plan to use as sender and, then, you must go to https://myaccount.google.com/lesssecureapps to turn on the "Allow less secure apps" option.

1.-If you're using an Ubuntu system, install these packages: *apt install postfix mailutils libsasl2-2 ca-certificates libsasl2-modules*
In you're using a Fedora system, install these packages instead: *dnf install postfix mailx cyrus-sasl cyrus-sasl-plain*

2.-Edit Postfix's main config file ("/etc/postfix/main.cf") to make it look like this:

```
#587 is the port used by Gmail server. It's the standard in secure email server. Insecure ones use port nº25
relayhost = [smtp.gmail.com]:587
#SASL is the standard way to do authentications in email servers (in this case, to authenticate Postfix in Gmail)
smtp_sasl_auth_enable = yes
#/etc/postfix/sasl_passwd is a file which we'll create to contain password of Gmail account used to send email
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
#TLS is the standard way to send encrypted messages to email servers. Gmail doesn't send non-encrypted ones.
smtp_use_tls = yes
#To use TLS we need a "CA certificate". Several ones are provided by "ca-certificates" package.
#In Fedora, nevertheless, the value of next directive should be /etc/ssl/certs/ca-bundle.crt
smtp_tls_CAfile = /etc/ssl/certs/thawte_Primary_Root_CA.pem
```

3.-Specify Gmail address (and password) that Postfix will use to authenticate into Gmail to be able to send emails through it. This can be achieved by executing (as root) these commands:

```
echo "[smtp.gmail.com]:587 username@gmail.com:password" > /etc/postfix/sasl_passwd
postmap /etc/postfix/sasl_passwd
chown root:root /etc/postfix/sasl_passwd*  &&  chmod 600 /etc/postfix/sasl_passwd*
```

NOTA: *Postmap* command generates a file called "/etc/postfix/sasl_passwd.db", which "caches" the content of "sasl_passwd" file in a binary form to optimize the reading of the information contained within

4.-Restart Postfix and test this configuration by sending an email to some receiver; it should arrive

```
systemctl restart postfix
echo "Text del correu" | mail -s "Assumpte del correu" destinatari@hotmail.com
```