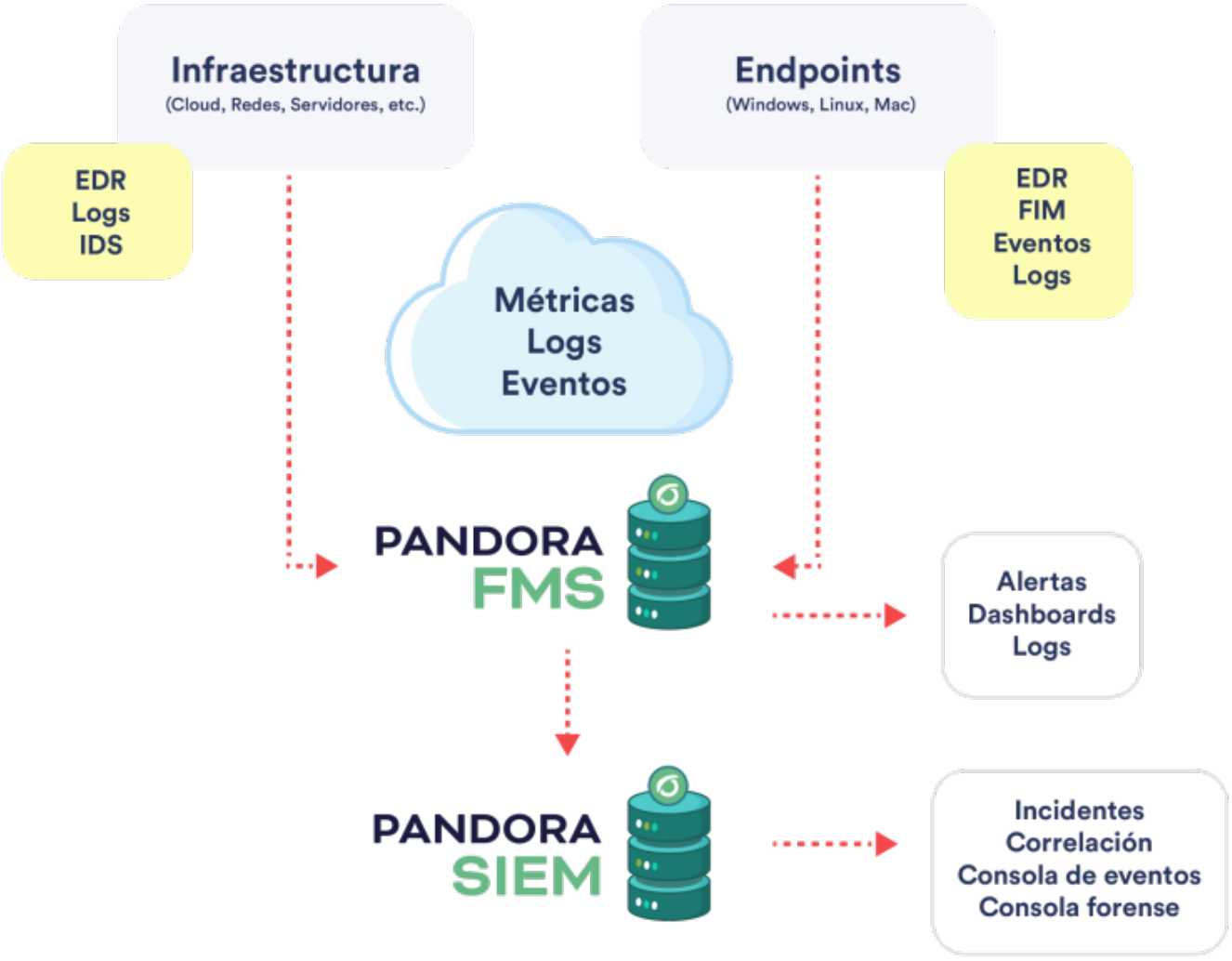
Monitorización SIEM

Monitorización SIEM

Introducción

Las siglas SIEM significan *Security Information and Event Management*. Un SIEM describe las funcionalidades de monitorización de seguridad a través de la recolección, filtrado, normalización, correlación y visualización de eventos de seguridad. Combina la gestión de información de seguridad (SIM) con la gestión de eventos de seguridad (SEM).



Un SIEM gestiona eventos de seguridad, que pueden venir de un *log* ordinario (por ejemplo *logs* de un Apache), del *log* de una herramienta específica de seguridad (*logs* de un *firewall*, IDS, *honeypot*, EDR, etcétera) o eventos enriquecidos de otra herramienta (otro SIEM).

Para aprovechar al máximo todas las características de SIEM PFMS se recomienda tener

instalado como mínimo la versión 783 en los EndPoints.

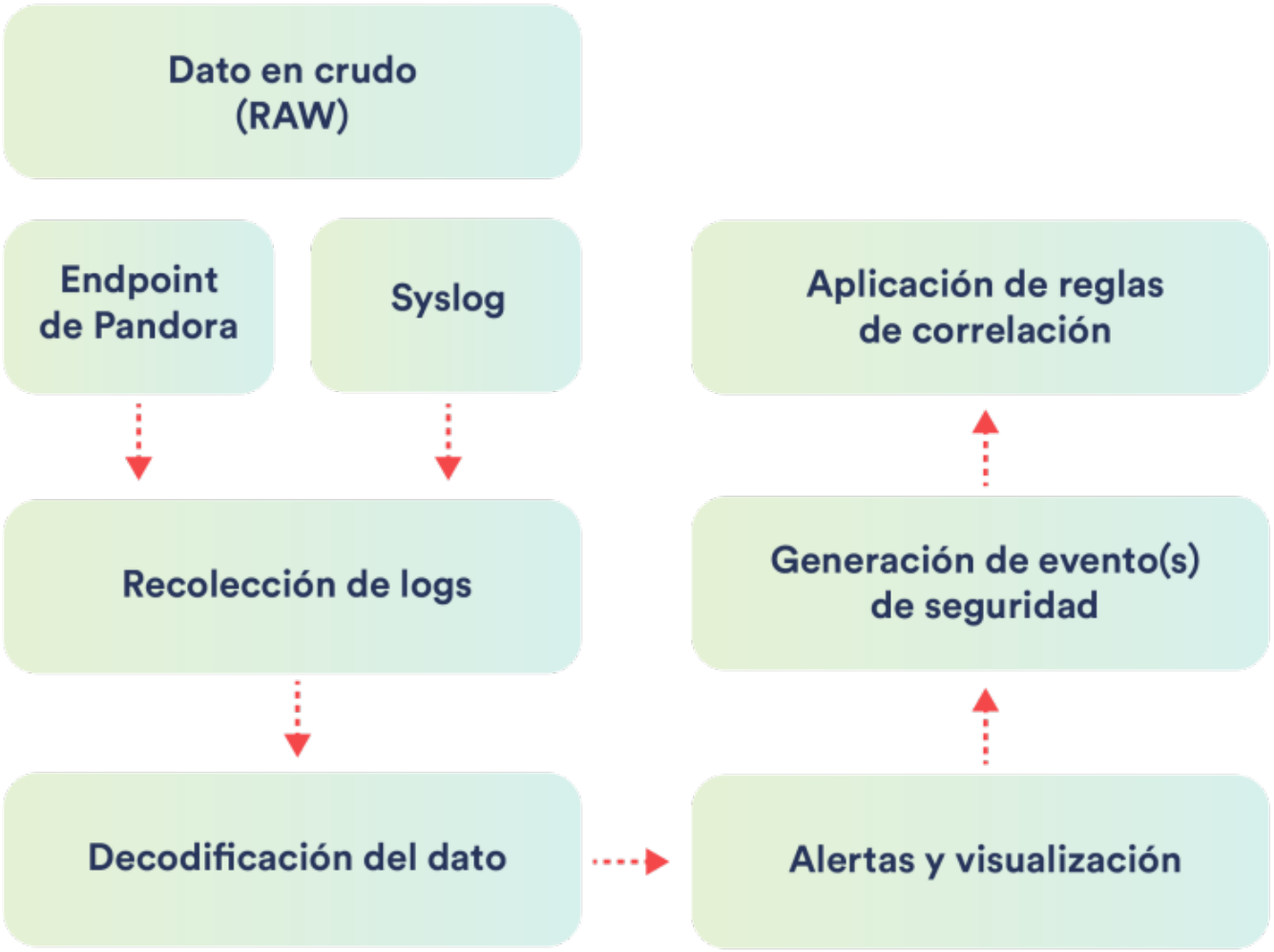
Cómo funciona

Pandora FMS SIEM se encarga de procesar las entradas de *logs* obtenidas mediante su recolección, normalizar los datos y generar eventos de seguridad en base a dichas entradas.

Igual que con la recolección de *logs*, los eventos SIEM generados se almacenan en OpenSearch®, por lo que el primer requisito para hacer funcionar esta monitorización es contar con una instalación de OpenSearch.

Pandora FMS SIEM genera los eventos de seguridad usando dos componentes en el servidor, por lo que los eventos se generan también en dos pasos:

- Normalización de los datos: Se decodifican todas las entradas de la recolección de *logs*, generando nuevas entradas de *logs* normalizados que se almacenan temporalmente.
- Generación de eventos: Se comprueba si los *logs* normalizados cumplen con una serie de reglas, en cuyo caso se produce un evento SIEM con toda la información de la regla y el *log* normalizado que produjo el evento.



De esta manera el flujo completo para la monitorización SIEM es:

- Agentes, *plugins* y otras fuentes de información monitorizan o generan *logs* que son enviados al `dataserver` o al `syslogserver`.
- `dataserver` y `syslogserver` se encargan de procesar esas entradas de *logs* y las almacenan en el servidor de OpenSearch configurado para la recolección de *logs*.
- `siemserver` decodifica todos los *logs* obtenidos mediante la recolección de *logs* para generar *logs* normalizados en el servidor de OpenSearch configurado para la monitorización SIEM. Estos *logs* normalizados se almacenan de forma temporal.
- `siemevents` procesa cada uno de los *logs* normalizados y si cumplen una serie de reglas predefinidas se generan eventos SIEM que se almacenan en el servidor de OpenSearch configurado para la monitorización SIEM.

Con los eventos generados, es posible consultarlos para su operación desde la consola de Pandora FMS.

Configuración de la consola

Para utilizar la monitorización de eventos SIEM, primero se deberá activar desde la configuración

principal.

Menú Management → Settings → System Settings → SIEM → Activate SIEM, rellenar por completo el formulario y pulsar Update para guardar los cambios.

Esto creará las plantillas necesarias en el servidor de OpenSearch indicado para la normalización de *logs* y generación de eventos SIEM.

También activará este tipo de monitorización en los servidores de Pandora FMS donde se hayan iniciado `siemserver` y `siemevents`.

Si se tiene Command Center habilitado, Pandora SIEM solamente estará disponible en los nodos.

Configuración del servidor

Para realizar la monitorización de eventos SIEM será necesario activar los servidores `siemserver` y `siemevents` en la configuración del servidor.

Para decodificar y normalizar las entradas de la recolección de *logs* será necesario contar con ficheros XML de decodificación que establezcan la manera de obtener la información necesaria en cada caso (en adelante, “decoders”). Estos ficheros XML se ubicarán en la ruta indicada por el parámetro `siem_decoders` de la configuración del servidor, por defecto en:

```
/usr/share/pandora_server/util/siem/decoders
```

Para generar eventos SIEM será necesario contar con ficheros XML de reglas que establezcan las condiciones para generar un evento en base a la información obtenida en los *logs* normalizados (en adelante, “rules”). Estos ficheros XML se ubicarán en la ruta indicada por el parámetro `siem_events_rules` de la configuración del servidor, por defecto en:

```
/usr/share/pandora_server/util/siem/rules
```

Los ficheros XML de *decoders* y *rules* de Wazuh® son compatibles con la monitorización SIEM de Pandora FMS.

Configuración de los agentes

Gran parte de la recolección de *logs* se realiza mediante los EndPoints de Pandora FMS. Estos agentes, tanto en sistemas GNU/Linux® como en sistemas MS Windows®, cuentan con un tipos de módulos concretos para realizar esta tarea.

Para aprovechar al máximo todas las características de SIEM PFMS se recomienda tener instalado como mínimo la versión 783 en los EndPoints.

La monitorización SIEM se basa en gran medida en el tipo de *log* recolectado, por lo que será necesario especificar `module_source_type` en los módulos de recolección de *logs* para indicar dicho tipo.

El tipo es usado por *decoders* y *rules*, por lo que se deberá consultar los *decoders* y *rules* activos para conocer el tipo que deba indicar en cada *log*.

Los tipos de *logs* más usados son:

- syslog
- ids
- web-log
- squid
- windows
- host-information
- ossec

Agentes Linux

La recolección de *logs* en sistemas Linux se realiza principalmente mediante la lectura de ficheros de *log*. Esto se puede lograr mediante el uso de configuraciones de módulos con esta estructura mínima:

```
module_begin
module_name <program_name>
module_type log
module_regexp <path_to_log_file>
module_pattern <capture_regexp>
module_source_type <log_type>
module_end
```

Por ejemplo, para recolectar todas las entradas del *log* de acceso de un servidor Apache:

```
module_begin
```

```
module_name apache
module_type log
module_regexp /var/log/httpd/access_log
module_pattern .*
module_source_type web-log
module_end
```

Las entradas recogidas de un *log* como el anterior serían normalizadas por "decoders" como `web-accesslog`, `web-accesslog-ip` o `web-accesslog-domain` entre otros.

Los *logs* decodificados de un *log* como el anterior podrían generar eventos como por ejemplo `Common web attack`, `XSS (Cross Site Scripting) attempt` o `SQL injection attempt` entre otros.

Agentes MS Windows®

La recolección de *logs* en MS Windows® se realiza principalmente mediante la monitorización de eventos del sistema, aunque también se puede realizar mediante la lectura de ficheros de *log* como en sistemas Linux.

Usando el sistema de eventos de Windows, se pueden recolectar estos logs mediante el uso de configuraciones de módulos con una de estas dos configuraciones mínimas.

Si se trata de eventos de tipo Application, System o Security:

```
module_begin
module_name <module_name>
module_type log
module_logchannel
module_source <Application|System|Security>
module_source_type <log_type>
module_end
```

O si se trata de eventos de otro canal distinto:

```
module_begin
module_name <module_name>
module_type log
module_logchannel
module_source <log_channel_path>
module_source_type <log_type>
module_end
```

Por ejemplo, para recolectar todas las entradas del eventos de Security y Windows Defender:

```
module_begin
```

```
module_name Windows_LogEvents_System
module_type log
module_logchannel
module_source Security
module_source_type ossec
module_end

module_begin
module_name Windows_LogchannelEvents_WindowsDefender
module_type log
module_logchannel
module_source Microsoft-Windows-Windows Defender/Operational
module_source_type ossec
module_end
```

Las entradas recogidas de eventos como los anteriores serían normalizadas por *decoders* como `windows_eventchannel`.

Los *logs* decodificados de eventos como los anteriores podrían generar eventos como por ejemplo `Windows error event`, `Short-time multiple Windows Defender warning events` o `Multiple Windows Defender error events` entre otros.

Usando la monitorización de ficheros de *log*, la configuración es idéntica a la de sistemas Linux. Es necesaria una configuración mínima como esta:

```
module_begin
module_name <program_name>
module_type log
module_regexp <path_to_log_file>
module_pattern <capture_regexp>
module_source_type <log_type>
module_end
```

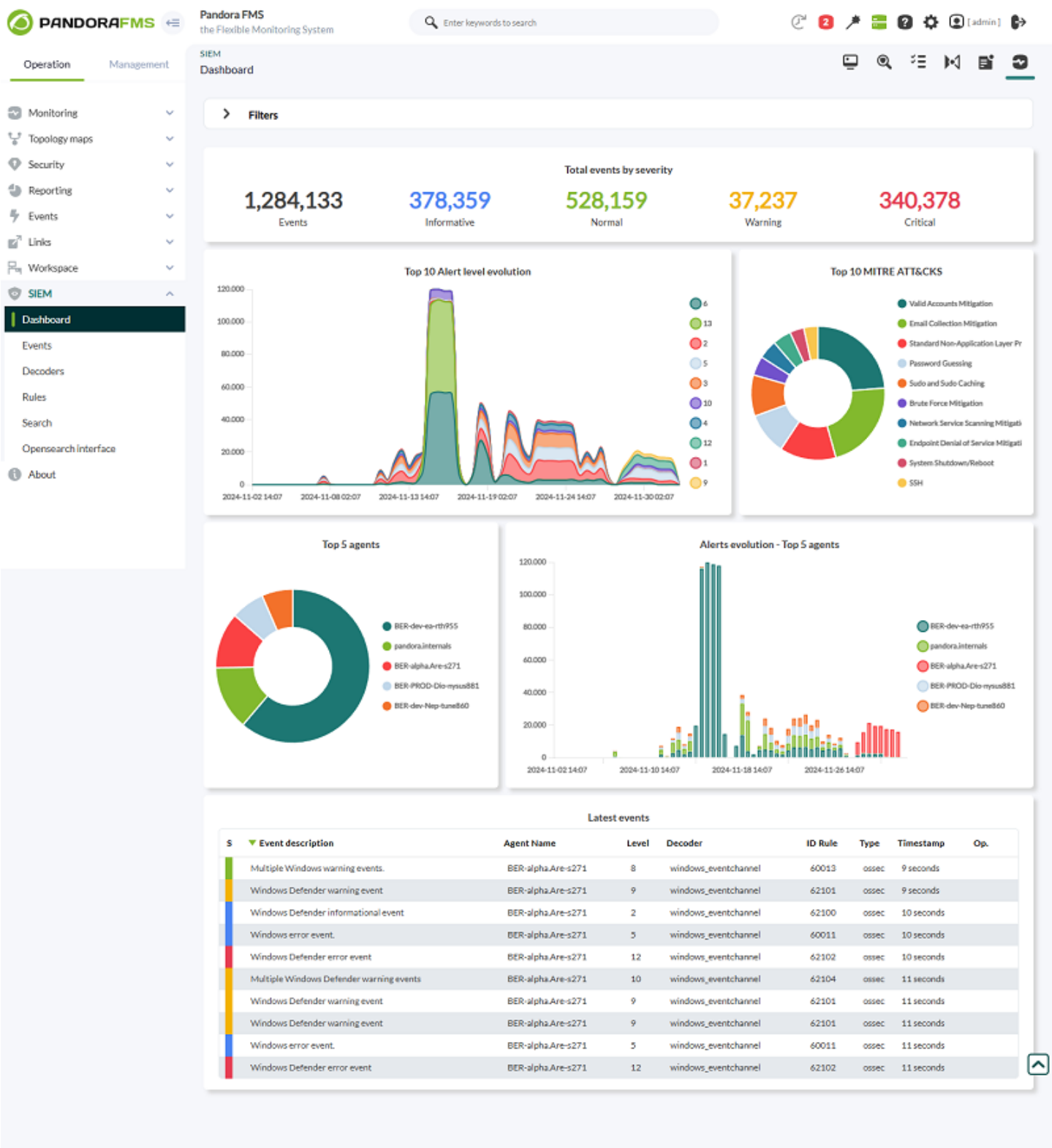
Por ejemplo, para recolectar todas las entradas del log de un servidor X:

```
module_begin
module_name xserver
module_type log
module_regexp C:\server\logs\xserver.log
module_pattern .*
module_source_type xserver
module_end
```

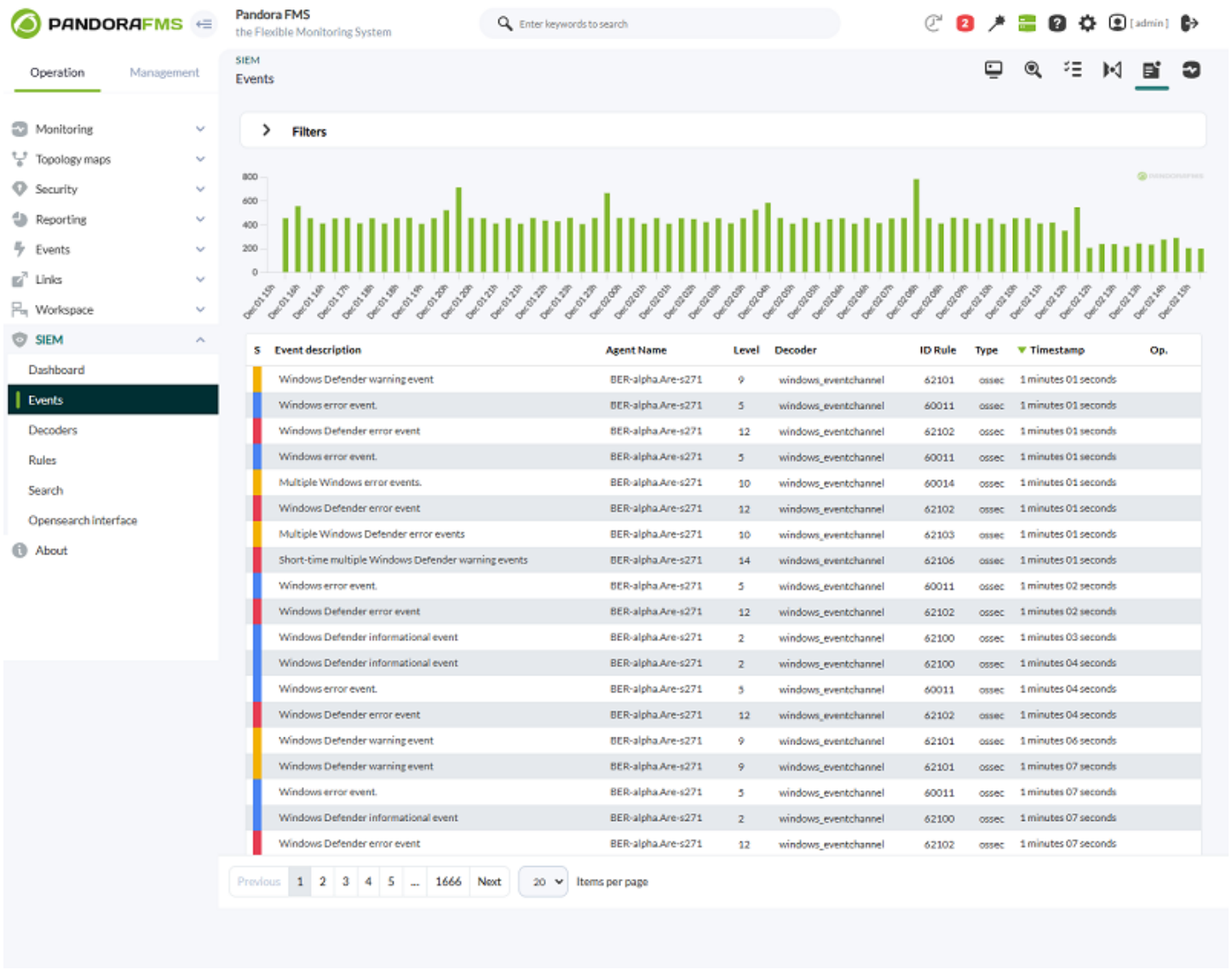
Eventos SIEM

Con la monitorización SIEM activada y configurada, se puede acceder a una vista previa del estado

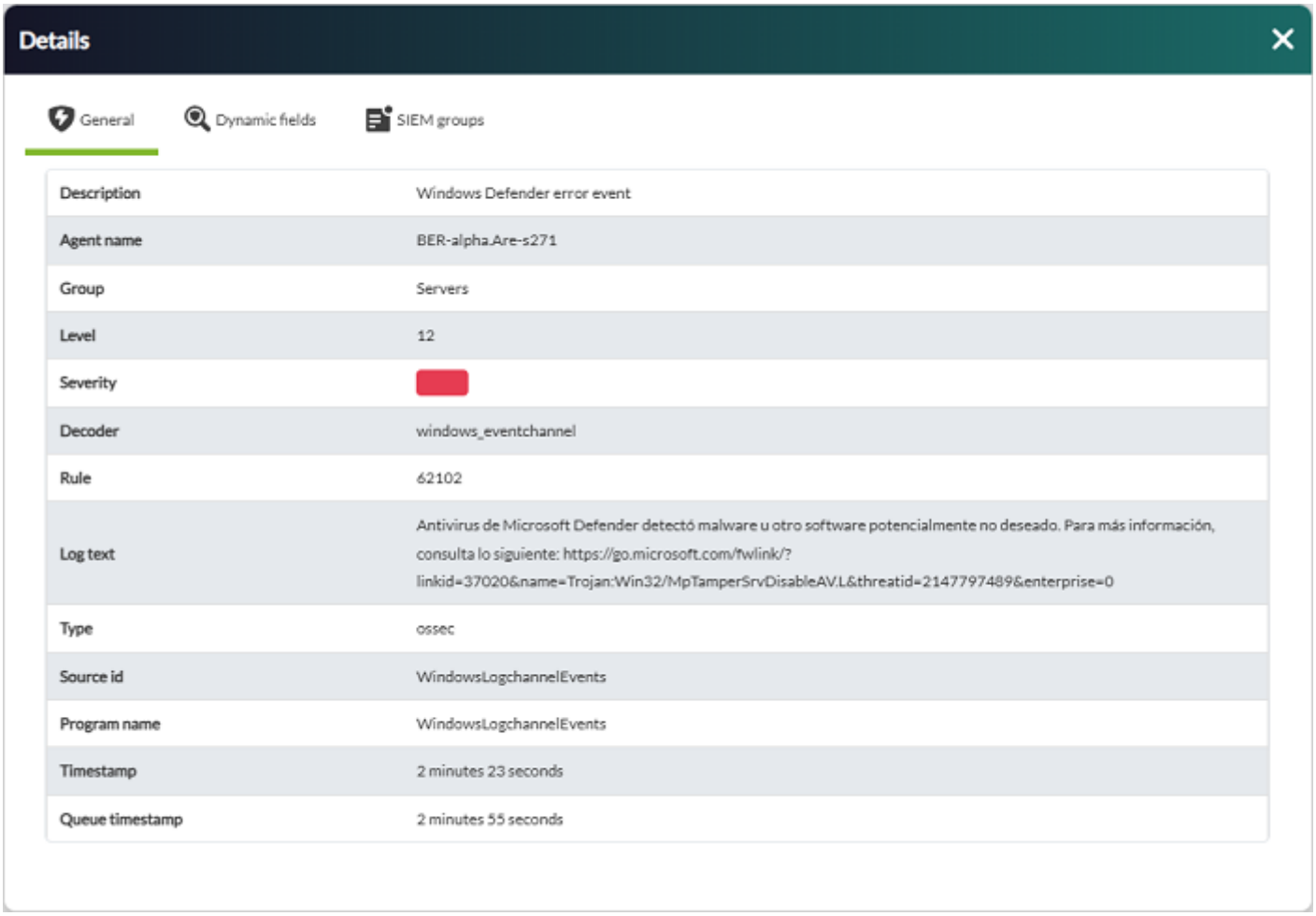
de la monitorización en el menú Operation → SIEM → Dashboard.



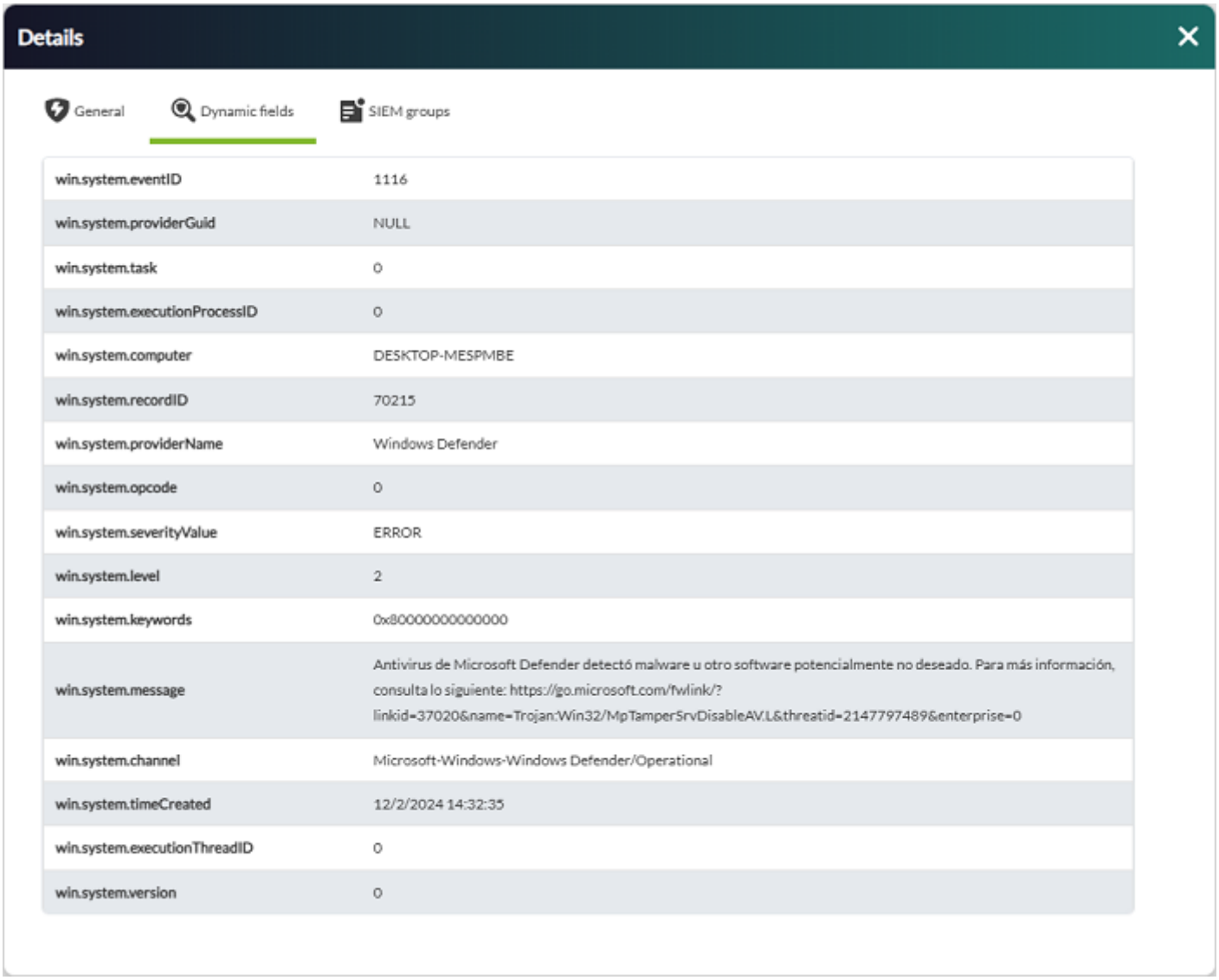
Los eventos SIEM generados se pueden visualizar en su totalidad en el menú Operation → SIEM → Events.



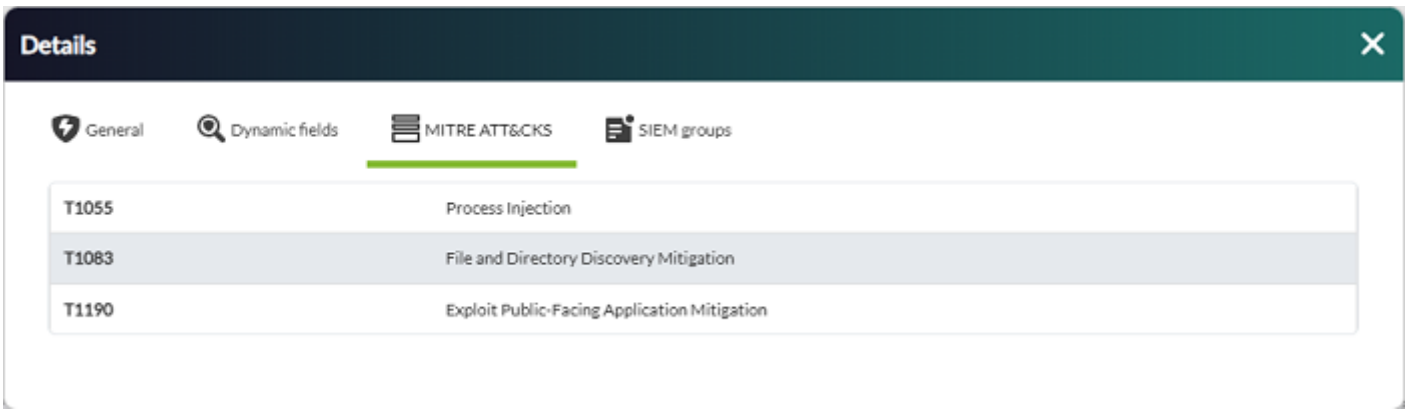
Cada evento SIEM tendrá una ventana con los detalles del mismo, mostrando información del *log* normalizado y de la regla que generó el evento.

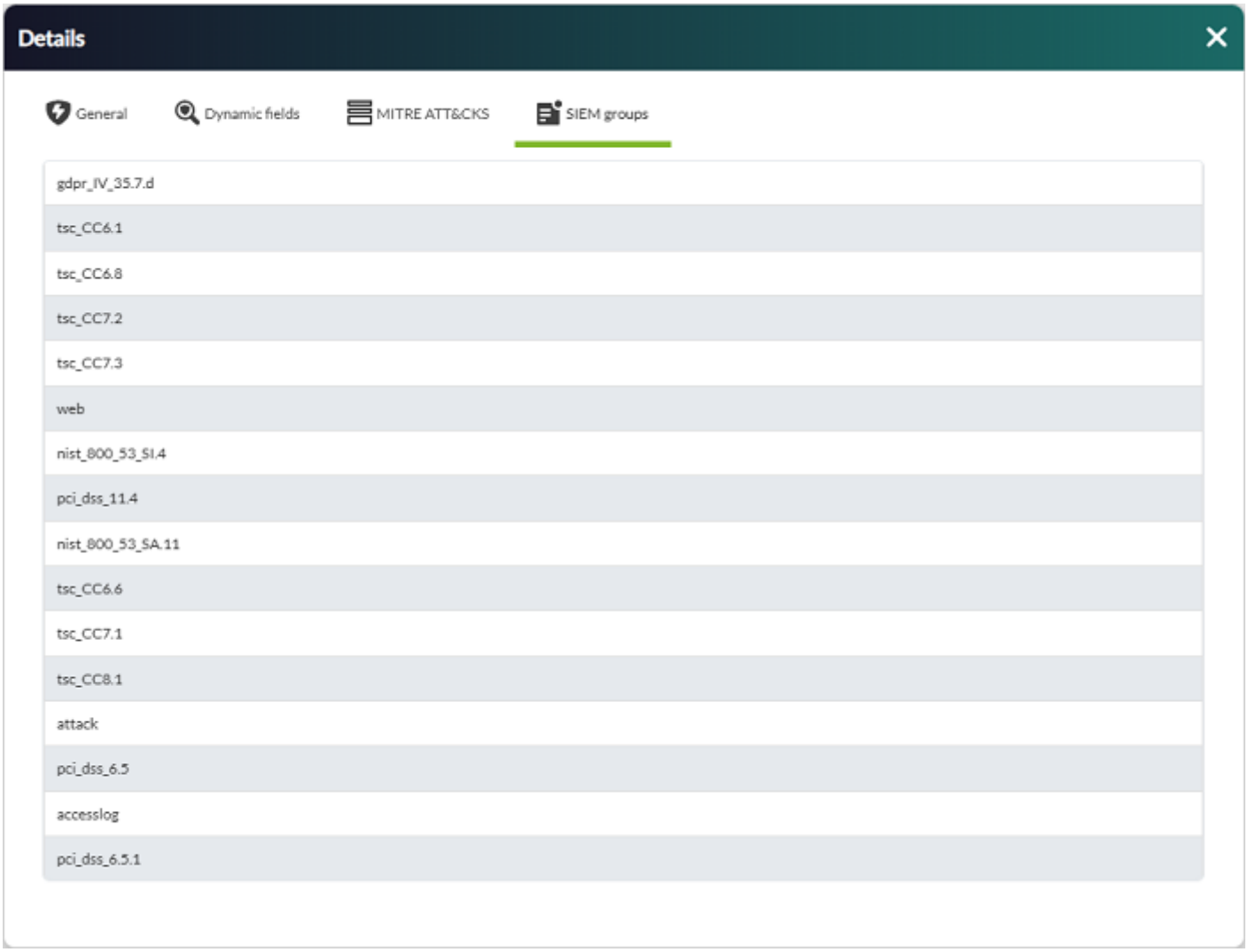


Dependiendo de los *decoders* que hayan normalizado el *log*, el evento contará con la pestaña Dynamic fields con información útil del *log*.



Dependiendo de la regla que haya generado el evento, se contará con las pestañas MITRE ATT&CK® y SIEM groups con información útil del impacto del evento.





La información mostrada tanto en el Dashboard general como en la tabla de eventos se puede incluir en los Dashboards de Pandora FMS mediante los *widgets* de SIEM incluidos.

Decoders

Pandora FMS incluye una serie de *decoders* por defecto para la monitorización de SIEM, pero cualquier administrador puede incluir los suyos propios.

Gestión

Para incluir nuevos *decoders*, primero se debe añadir o editar un fichero XML en la ruta indicada al Pandora FMS server en su parámetro de configuración `siem_decoders`.

Los *decoders* se cargan en el entorno mediante ficheros XML que el servidor de Pandora FMS master lee en cada inicio del servicio.

Una vez los *decoders* son cargados, su configuración se almacena en la base de datos, y cada servidor `siemserver` los procesa para las entradas obtenidas mediante la recolección de *logs*.

Desde la Consola de Pandora FMS, será posible visualizar los *decoders* cargados con su configuración completa desde el menú Operation → SIEM → Decoders.

También es posible deshabilitar *decoders* desde esta vista, de manera que `siemserver` no los tenga en cuenta a la hora de normalizar las entradas de *log*.

Todos los *decoders* se cargan al completo en cada reinicio. Esto significa que los *decoders* que no se hayan podido leer de los ficheros XML no estarán disponibles (aún si en algún momento lo estuvieron), y que los *decoders* que se hayan deshabilitado desde la Consola volverán a habilitarse de nuevo (en caso de existir).

Sintaxis

Los *decoders* se configuran y cargan en Pandora FMS con ficheros XML. Estos ficheros pueden tener la siguiente sintaxis válida:

```
<var name="VarName">VarValue</var>

<decoder name="DecoderName" discard="yes|no">
    <parent>DecoderName</parent>
    <program_name>REGEXP</program_name>
    <type>EventType</type>
    <prematch type="pcre2">REGEXP</prematch>
    <prematch offset="after_parent">REGEXP</prematch>
    <prematch offset="after_prematch">REGEXP</prematch>
    <regex type="pcre2">REGEXP</regex>
    <regex offset="after_parent">REGEXP</regex>
    <regex offset="after_regex">REGEXP</regex>
    <order>Field1, Field2.Sub1, Field2.Sub2</order>
    <json_null_field>string|discard</json_null_field>
</decoder>
```

var

↑Sintaxis completa

Sirve para indicar variables con sus valores, y usarlas posteriormente en el XML (`$VarName`).

```
<var name="VarName">VarValue</var>
```

decoder

↑Sintaxis completa

Es la información del *decoder*, con su nombre. Un mismo fichero XML puede contener varios *decoders*.

- `name`: Es el nombre del *decoder*. Varios *decoders* pueden tener el mismo nombre, y todos se evalúan.
- `discard`: Con un valor `yes` o `no`, permite descartar *logs* de la evaluación de *decoders* si tienen coincidencia con este. Los *decoders* con `discard="yes"` se evalúan antes que el resto (siempre que no tengan un *decoder padre*).

```
<decoder name="DecoderName" discard="yes|no">
  …
  …
  …
</decoder>
```

parent

↑Sintaxis completa

Indica el nombre del *decoder* padre para generar una estructura jerarquizada.

```
    <parent>DecoderName</parent>
```

program_name

↑Sintaxis completa

El nombre del programa que debe ser encontrado en las cabeceras del *log* o en el `source_id` del *log*.

- `type`: Permite indicar el tipo de expresión regular. Si no se indica, se utiliza OS Regex.

```
    <program_name>REGEXP</program_name>
```

Dado el caso se especifica para PCRE2:

```
    <program_name type="pcre2">REGEXP</program_name>
```

type

↑Sintaxis completa

El tipo de *logs* para los que se comparará el *decoder*. Debe coincidir con el `type` del *log*.

```
    <type>EventType</type>
```

prematch

↑Sintaxis completa

Si el contenido del *log* coincide con esto, genera un *log* normalizado.

- `type`: Permite indicar el tipo de expresión regular. Si no se indica, se utiliza OS Regex.
- Véase offset.

```
    <prematch type="pcre2">REGEXP</prematch>
    <prematch offset="after_parent">REGEXP</prematch>
    <prematch offset="after_prematch">REGEXP</prematch>
```

regex

↑Sintaxis completa

Los grupos que se capturen con esta expresión regular es información normalizada para el *log*.

- `type`: Permite indicar el tipo de expresión regular. Si no se indica, se utiliza OS Regex.
- Véase offset.

```
    <regex type="pcre2">REGEXP</regex>
    <regex offset="after_parent">REGEXP</regex>
    <regex offset="after_regex">REGEXP</regex>
```

order

↑Sintaxis completa

Los valores capturados por las regex son almacenados en estos nombres de campos, en el orden de los grupos de captura.

```
    <order>Field1, Field2.Sub1, Field2.Sub2</order>
```

json_null_field

↑Sintaxis completa

Los valores nulos de los grupos de captura se almacenarán como *string* vacío o se descartarán.

```
    <json_null_field>string|discard</json_null_field>
```

offset

↑Sintaxis completa

Sirve para indicar el orden en que se realizan las comprobaciones y descartar del contenido del *log* para las siguientes comprobaciones el texto que coincida con la expresión regular: `after_parent`, `after_regex`, `after_prematch`. Por ejemplo:

```
<decoder name="my_decoder">
  <prematch type="pcre2">^\d\d\d\d/\d\d/\d\d \d\d:\d\d:\d\d </prematch>
  <regex type="pcre2" offset="after_prematch">(\w):(\d+)</regex>
  <order>srcip,srcport</order>
</decoder>
```

Comprobará que el texto del *log* coincida con la expresión regular `^\d\d\d\d/\d\d/\d\d \d\d:\d\d:\d\d ` , descartará ese contenido del texto y al evaluar la expresión regular de `regex` capturará lo que corresponda. En el ejemplo, eliminaría una fecha del texto a la hora de evaluar para simplificar los grupos de captura.

Rules

Pandora FMS incluye una serie de *rules* por defecto para la monitorización de SIEM, pero cualquier administrador puede incluir las suyas propias.

Gestión

Para incluir nuevas *rules*, primero se debe añadir o editar un fichero XML en la ruta indicada al servidor de Pandora FMS en su parámetro de configuración `siem_rules`.

Las *rules* se cargan en el entorno mediante ficheros XML que el servidor de Pandora FMS master lee en cada inicio del servicio.

Una vez las *rules* son cargadas, su configuración se almacena en la base de datos, y cada servidor `siemevents` las procesa para las entradas normalizadas en la monitorización SIEM.

Desde la Consola de Pandora FMS, será posible visualizar las *rules* cargadas con su configuración completa desde el menú Operation → SIEM → Rules.

También es posible deshabilitar *rules* desde esta vista, de manera que `siemevents` no las tenga en cuenta a la hora de procesar los *logs* normalizados.

Todas las *rules* se cargan al completo en cada reinicio. Esto significa que las *rules* que no se hayan podido leer de los ficheros XML no estarán disponibles (aún si en algún momento lo estuvieron). A diferencia de los *decoders*, las *rules* cuentan con un ID que permite mantenerlas deshabilitadas en cada recarga.

Las *rules* que no se hayan podido leer de los ficheros XML se marcarán en la Consola como "no activas" y no se tendrán en cuenta a la hora de generar eventos SIEM. Tampoco se tendrán en cuenta las *rules* que hayan sido deshabilitadas manualmente por un administrador. Por lo tanto para que las *rules* se evalúen es necesario que se encuentren activas y habilitadas.

Clasificación

Las *rules* se clasifican en varios niveles, que van desde el más bajo (`0`) hasta el más alto (`15`). La siguiente tabla describe cada nivel, brindando información sobre la gravedad de cada evento generado por la monitorización SIEM.

Nivel	Título	Descripción
0	Ignorado.	No se ha tomado ninguna medida. Se utiliza para evitar falsos positivos. Estas reglas se escanean antes que todas las demás, incluyen eventos sin relevancia de seguridad y no aparecen en el panel de eventos de seguridad.
2	Notificación de baja prioridad del sistema.	Notificaciones del sistema o mensajes de estado. No tienen relevancia para la seguridad y no aparecen en el panel de eventos de seguridad.
3	Eventos exitosos/autorizados.	Estos incluyen intentos de inicio de sesión exitosos, eventos permitidos por el *firewall*, etcétera.
4	Error de baja prioridad del sistema.	Errores relacionados con malas configuraciones o dispositivos/aplicaciones no utilizados. Estos no tienen relevancia de seguridad y generalmente son causados por instalaciones predeterminadas o pruebas de software.
5	Error generado por el usuario.	Estos incluyen contraseñas olvidadas, acciones denegadas, etcétera. Por sí solos, no tienen relevancia para la seguridad.

Nivel	Título	Descripción
6	Ataque de baja relevancia.	Estos indican un gusano o un virus que no tiene efecto en el sistema (como el código rojo para servidores Apache, etcétera). Esto también incluye eventos de *Intrusion Detection System* (IDS) frecuentes y errores frecuentes.
7	Coincidencia de “malas palabras”.	Estos incluyen palabras como “malo”, “error”, etc. La mayoría de estos eventos no están clasificados y pueden tener cierta relevancia desde el punto de vista de la seguridad.
8	Primera vez visto.	Incluir eventos vistos por primera vez. La primera vez que se activa un evento IDS o la primera vez que un usuario inicia sesión. También incluye acciones relevantes para la seguridad, como la activación de un *sniffer* o actividades similares.
9	Error de fuente no válida.	Incluye intentos de iniciar sesión como un usuario desconocido o desde una fuente no válida. Puede tener relevancia de seguridad (especialmente si se repite). Esto también incluye errores relacionados con la cuenta “admin” (raíz o *root*).
10	Múltiples errores generados por el usuario.	Estos incluyen múltiples contraseñas incorrectas, múltiples inicios de sesión fallidos, etcétera. Estos pueden indicar un ataque o simplemente señalar que un usuario ha olvidado sus credenciales.
11	Advertencia de comprobación de integridad.	Estos incluyen mensajes sobre la modificación de binarios o la presencia de *rootkits* (por *Rootcheck*). Estos pueden indicar un ataque exitoso. También se incluyen eventos IDS que serán ignorados (gran cantidad de repeticiones).
12	Evento de alta importancia.	Estos incluyen mensajes de error o advertencia del sistema, kernel, etcétera. Estos pueden indicar un ataque contra una aplicación específica.
13	Error inusual (alta importancia).	Coincide con un patrón de ataque común la mayor parte del tiempo.
14	Evento de seguridad de alta importancia.	La mayoría de las veces se activa con correlación e indica un ataque.
15	Ataque severo.	No hay posibilidad de falsos positivos. Es necesaria atención inmediata.

En base a estos niveles, los eventos contarán con una severidad concreta que se mostrará en la Consola:

- Informational: Niveles del 0 al 6.
- Normal: Niveles del 7 al 8.
- Warning: Niveles del 9 al 11.
- Critical: Niveles del 12 al 15.

Sintaxis

Elementos de sintaxis detallados

```
<var name="VarName">VarValue</var>

<group name="GROUP1,GROUP2,">
    <rule id="N" level="N" frequency="N" timeframe="N" ignore="N"
overwrite="yes|no">
```

```
        <if_matched_sid>N</if_matched_sid>
        <if_matched_group>GROUP</if_matched_group>
        <same_id />
        <different_id />
        <same_field>Field1</same_field>
        <same_field>Field2.Sub1</same_field>
        <different_field>Field1</different_field>
        <different_field>Field2.Sub1</different_field>
        <description>TEXT</description>
        <match type="pcre2">RREGEXP</match>
        <match negate="yes|no">RREGEXP</match>
        <regex type="pcre2">RREGEXP</regex>
        <regex negate="yes|no">RREGEXP</regex>
        <decoded_as>DecoderName</decoded_as>
        <category>EventType</category>
        <field name="Field1">REGEXP</field>
        <field name="Field2.Sub1" negate="yes|no">REGEXP</field>
        <program_name negate="yes|no">REGEXP</program_name>
        <time>TIME-RANGE</time>
        <weekday>DAYS</weekday>
        <if_sid>PARENT1, PARENT2</if_sid>
        <if_group>GROUP</if_group>
        <if_level>N</if_level>
        <info type="text|link|cve">TEXT|LINK|CVE</info>
        <group>GROUP1,GROUP2,</group>
        <mitre>
            <id>MITRE_ID</id>
            <id>MITRE_ID</id>
        </mitre>
    </rule>
</group>
```

Véase también Caso de estudio.

var

↑Sintaxis completa

```
<var name="VarName">VarValue</var>
```

Sirve para indicar variables con sus valores, y usarlas posteriormente en el XML (`$VarName`).

group

↑Sintaxis completa

```
<group name="GROUP1,GROUP2,">
…
</group>
```

Permite agrupar reglas. Es usado también para condiciones de las mismas reglas.

rule

↑Sintaxis completa

```
    <rule id="N" level="N" frequency="N" timeframe="N" ignore="N"
overwrite="yes|no">
    …
    </rule>
```

Es la información de la regla:

1. `id`: Identificador de la regla. Debe ser único (salvo que se sobrescriba otra regla).
2. `level`: Nivel del evento al generarse (`0` a `15`). Las reglas con nivel `0` no generan evento.
3. `frequency`: Veces que se debe dar una concurrencia para generar un evento. La frecuencia en las reglas se comprueba para *logs* de un mismo agente, no para cualquier *log*.
4. `timeframe`: Ventana de tiempo en que se deben dar las concurrencias (segundos).
5. `ignore`: Esta regla reinicia sus contadores de `frequency` pasados estos segundos.
6. `overwrite`: Con un valor `yes` o `no`, permite sobrescribir la configuración de una regla con el mismo ID. Si junto con `overwrite="yes"` la regla tiene `level="0"`, esta se evalúa antes que cualquier otra y en caso de tener coincidencia con el *log*, descarta la evaluación del resto de reglas para ese *log*.

if_matched_sid

↑Sintaxis completa

```
        <if_matched_sid>N</if_matched_sid>
```

La regla cumple si otra regla con el ID indicado ha disparado una alerta las veces indicadas por `frequency` dentro del tiempo de `timeframe`.

if_matched_group

↑Sintaxis completa

```
        <if_matched_group>GROUP</if_matched_group>
```

Como el if_matched_sid pero para grupos.

same_id

↑Sintaxis completa

```
        <same_id />
```

Deben darse las concurrencias con el mismo ID.

different_id

↑Sintaxis completa

```
        <different_id />
```

Deben darse las concurrencias con distintos ID.

same_field

↑Sintaxis completa

```
        <same_field>Field1</same_field>
        <same_field>Field2.Sub1</same_field>
```

Deben darse las concurrencias con los mismos valores en el campo.

different_field

↑Sintaxis completa

```
        <different_field>Field1</different_field>
        <different_field>Field2.Sub1</different_field>
```

Deben darse las concurrencias con distintos valores en el campo.

description

↑Sintaxis completa

```
        <description>TEXT</description>
```

El texto para el evento que se genere.[1)]

match

↑Sintaxis completa

```
        <match type="pcre2">RREGEXP</match>
        <match negate="yes|no">RREGEXP</match>
```

Si el contenido del *log* coincide con esto, genera un evento de SIEM.

1. `type`: Permite indicar el tipo de expresión regular. Si no se indica, se utiliza OS Regex.
2. `negate`: Con un valor de `yes` permite negar la coincidencia.

regex

↑Sintaxis completa

```
        <regex type="pcre2">RREGEXP</regex>
        <regex negate="yes|no">RREGEXP</regex>
```

Igual que ''match''.

1. `type`: Permite indicar el tipo de expresión regular. Si no se indica, se utiliza OS Regex.
2. `negate`: Con un valor de `yes` permite negar la expresión regular.

decoded_as

↑Sintaxis completa

```
        <decoded_as>DecoderName</decoded_as>
```

La regla cumple si el *log* ha sido decodificado por el *decoder* indicado

category

↑Sintaxis completa

```
        <category>EventType</category>
```

La regla cumple si el tipo del *decoder* coincide.

field

↑Sintaxis completa

```
        <field name="Field1">REGEXP</field>
        <field name="Field2.Sub1" negate="yes|no">REGEXP</field>
```

Si el campo indicado coincide con el valor, cumple la regla.

1. `negate`: Con un valor de `yes` permite negar la coincidencia con el campo.

program_name

↑Sintaxis completa

```
        <program_name negate="yes|no">REGEXP</program_name>
```

Si el origen del *log* coincide la regla cumple.

1. `negate`: Con un valor de `yes` permite negar coincidencia con origen del *log*.

time

↑Sintaxis completa

```
        <time>TIME-RANGE</time>
```

Si el evento se genera en el rango de tiempo indicado la regla coincide.

weekday

↑Sintaxis completa

```
        <weekday>DAYS</weekday>
```

Si el evento se genera estos días la regla cumple (`monday - sunday`, `weekdays`, `weekends`).

if_sid

↑Sintaxis completa

```
        <if_sid>PARENT1, PARENT2</if_sid>
```

La regla cumple si cualquiera de las reglas padre cumple.

if_group

↑Sintaxis completa

```
        <if_group>GROUP</if_group>
```

La regla cumple si el log cumple cualquier otra regla en el grupo indicado.

if_level

↑Sintaxis completa

```
        <if_level>N</if_level>
```

La regla cumple si otra regla con el mismo nivel ha cumplido.

info

↑Sintaxis completa

```
        <info type="text|link|cve">TEXT|LINK|CVE</info>
```

Información adicional para el evento generado.[2)]

group

↑Sintaxis completa

```
        <group>GROUP1,GROUP2,</group>
```

Lista de grupos de la regla.

mitre

↑Sintaxis completa

```
        <mitre>
            <id>MITRE_ID</id>
            <id>MITRE_ID</id>
        </mitre>
```

Lista de los ID MITRE de la regla.

Caso de estudio

El siguiente código describe una regla que sirve para *logs* de SELinux relacionados con PHP. Esta regla crea eventos de cualquier intento de conexión a puertos distintos a los habituales en un servidor web.

```
<rule id="100201" level="6">
        <decoded_as>setroubleshoot_program</decoded_as>
        <match>/usr/sbin/php-fpm</match>
        <field name="object_target"
type="pcre2">^(?!.*\b(9200|3306|80|443)$).*$</field>
        <field name="object_target" type="pcre2">^(?!.*(directory|file)
(conf|data_in|cron.lock)).*$</field>
        <description>SELinux prevented /usr/sbin/php-fpm execution on: $(action)
access on the $(object_target)</description>
        <mitre>
            <id>T1071.002</id>
        </mitre>
        <group>exec,threat,PHP</group>
</rule>
```

De esta manera las conexiones a puertos 9200, 3306, 80 y 443 no crearán eventos. Tampoco los que sean de tipo *directory* o *file*.

Decoder general utilizado para SELinux:

```
<decoder name="setroubleshoot">
    <program_name>^setroubleshoot$</program_name>
</decoder>

<decoder name="setroubleshoot_program">
    <parent>setroubleshoot</parent>
    <prematch type="pcre2">(SELinux is preventing|SELinux está negando
a)</prematch>
    <regex type="pcre2">(\S+) (?:from|de) (\S+) (?:access on the|el acceso a)
(.+?)\. (?:For complete SELinux messages run: sealert -l|Si quiere los mensajes
de SELinux completos, ejecute sealert -l) (\S+)$</regex>
    <order>binary,action,object_target,sealert_id</order>
</decoder>

<decoder name="setroubleshoot_program">
    <parent>setroubleshoot</parent>
    <prematch type="pcre2">failed to retrieve rpm info for path</prematch>
    <regex type="pcre2">failed to retrieve rpm info for path (\S+)</regex>
    <order>path</order>
</decoder>
```

Expresiones regulares

Las expresiones regulares son secuencias de caracteres que definen un patrón.

Hay dos tipos de expresiones regulares válidas para *decoders* y *rules*: OS Regex y PCRE2.

OS Regex

Son expresiones regulares sencillas basadas en una biblioteca hecha en lenguaje C. Está diseñada para ser simple y al mismo tiempo admitir las expresiones regulares más comunes.

Expresiones admitidas

Expresión	Caracteres válidos
\w	A-Z, a-z, 0-9, '-', '@', '_'
\d	0-9
\s	Espacios “ “
\t	Tabulación
\p	()*+,-.:;⇔?[]!”'#$%&|{}
\W	Cualquier cosa que no sea \w

Expresión	Caracteres válidos
\D	Cualquier cosa que no sea \d
\S	Cualquier cosa que no sea \s
\.	Cualquier cosa

Modificadores

Expresión	Comportamiento
+	Para que coincida una o más veces
*	Para que coincida cero o más veces

Caracteres especiales

Expresión	Comportamiento
^	Para especificar el comienzo del texto
$	Para especificar el final del texto
|	Para crear un patrón lógico “o” entre múltiples patrones

Escapar caracteres

Para utilizar los siguientes caracteres se deben *escapar* con \:

$	(	)	\	|	<
\$	\(	\)	\\	|	\<

Limitaciones

- Los modificadores * y + solamente se pueden aplicar a expresiones de barra invertida, no a caracteres simples (por ejemplo, \d+ se admite, 0+ no se admite).
- No se puede utilizar la alternancia en un grupo, por ejemplo (foo|bar) no está permitido.
- No se admite el retroceso complejo, por ejemplo, \p*\d*\s*\w*:, no coincide con solamente dos puntos porque \p* consume los dos puntos.
- . coincide con un punto literal, mientras que \. coincide con cualquier carácter.
- \s coincide solamente con un espacio ASCII (32), no con otros espacios en blanco como tabulaciones.
- No existe sintaxis para que coincida con un signo de intercalación, asterisco o más literal (aunque \p coincidirá con un asterisco o más, junto con algunos otros caracteres).

PCRE2

Perl Compatible Regular Expression (PCRE2) proporciona características como patrones recursivos,

aserciones de búsqueda anticipada y retrospectiva, grupos sin captura, cuantificadores no voraces, sintaxis extendida para caracteres y clases de caracteres, entre otras.

Para obtener más detalles, consulte la documentación de sintaxis de PCRE2.

Expresiones admitidas

Expresión	Caracteres válidos
`.`	Cualquier carácter excepto nueva línea
`\d`	Cualquier dígito decimal, igual a `[0-9]`
`\D`	Cualquier carácter que no sea un dígito decimal, igual a `[^0-9]`
`\h`	Cualquier carácter de espacio en blanco horizontal
`\H`	Cualquier carácter que no sea un espacio en blanco horizontal
`\s`	Cualquier carácter de espacio en blanco, igual a `[\t\r\n\f]`
`\S`	Cualquier carácter que no sea un espacio en blanco, igual a `[^\t\r\n\f]`
`\w`	Cualquier carácter de “palabra”
`\W`	Cualquier carácter “no palabra”

Modificadores

Expresión	Comportamiento
`?`	0 o 1, greedy
`?+`	0 o 1, possessive
`??`	0 o 1, lazy
`*`	0 o más, greedy
`*+`	0 o más, possessive
`*?`	0 o más, lazy
`+`	1 o más, greedy
`++`	1 o más, possessive
`+?`	1 o más, lazy
`{n}`	Exactamente n
`{n,m}`	Al menos n, no más de m, greedy
`{n,m}+`	Al menos n, no más de m, possessive
`{n,m}?`	Al menos n, no más de m, lazy
`{n,}`	n o más, greedy
`{n,}+`	n o más, possessive
`{n,}?`	n o más, lazy

Escapar caracteres

Expresión	Comportamiento
`\f`	Avance de página (hexadecimal `0C`)
`\n`	Nueva línea (hexadecimal `0A`)

Expresión	Comportamiento
`\r`	Retorno de carro (hexadecimal `0D`)
`\t`	Tabulación (hexadecimal `09`)
`\0dd`	Carácter con código octal `0dd`
`\o{ddd..}`	Carácter con código octal `ddd..`
`\xhh`	Carácter con código hexadecimal `hh`
`v\x{hhh…}`	Carácter con código hexadecimal `hh..`

Alertas SIEM

Véase el tema Sistema de alertas SIEM.

Informes SIEM

Véase el tema informes de eventos SIEM.

Volver al índice de documentación de Pandora FMS

1) , 2)
Se pueden usar variables en la descripción y en `info` con los valores de los campos personalizados, por ejemplo: `$(Field1)`, `$(Field2.Sub1)`, `$(Field2.Sub2)`.