Supervision SIEM

Supervision SIEM

Introduction

L'acronyme SIEM signifie *Security Information and Event Management* (gestion des informations et des événements de sécurité). Un SIEM décrit les fonctionnalités de surveillance de la sécurité par la collecte, le filtrage, la normalisation, la corrélation et la visualisation des événements de sécurité. Il combine la gestion des informations de sécurité (SIM) et la gestion des événements de sécurité (SEM).

Un SIEM gère les événements de sécurité, qui peuvent provenir d'un *log* ordinaire (par exemple *logs* d'un Apache), du *log* d'un outil de sécurité spécifique (*logs* d'un *firewall*, IDS, *honeypot*, EDR, et cætera) ou d'événements enrichis provenant d'un autre outil (un autre SIEM).

Pour profiter pleinement de toutes les fonctionnalités du SIEM PFMS, il est recommandé d'avoir au moins la version 783 installée sur les EndPoints.

Comment cela fonctionne-t-il?

Pandora FMS SIEM est responsable du traitement des entrées *logs* obtenues par votre collection, de la normalisation des données et de la génération d'événements de sécurité basés sur ces entrées.

Comme pour la collecte *logs*, les événements SIEM générés sont stockés dans OpenSearch®, donc la première condition pour que cette surveillance fonctionne est d'avoir une Installation OpenSearch.

Pandora FMS SIEM génère les événements de sécurité à l'aide de deux composants dans le serveur, de sorte que les événements sont également générés en deux étapes:

- Normalisation des données: Toutes les entrées de la collection *logs* sont décodées, générant de nouvelles entrées *logs* normalisées qui sont stockées temporairement.
- Génération d'événements: La conformité des *logs* normalisés à un ensemble de règles est vérifiée, auquel cas un événement SIEM est produit avec toutes les informations relatives aux règles et le *log* normalisé à l'origine de l'événement.

Le flux complet de la surveillance SIEM est donc le suivant:

- Les agents, *plugins* et autres sources d'information surveillent ou génèrent des *logs* qui sont envoyés au `dataserver` ou au `syslogserver`.
- Le `dataserver` et le `syslogserver` se chargent de traiter ces entrées *logs* et de les stocker sur le serveur OpenSearch configuré pour la collecte des *logs*.
- `siemserver` décode tous les *logs* obtenus par la collecte des *logs* pour générer des *logs* normalisés sur le serveur OpenSearch configuré pour la surveillance SIEM. Ces *logs* normalisés sont stockés temporairement.
- `siemevents` traite chacun des *logs* normalisés et s'ils répondent à un ensemble de règles prédéfinies, des événements SIEM sont générés et stockés sur le serveur OpenSearch configuré pour la surveillance SIEM.

Avec les événements générés, il est possible de les consulter pour leur fonctionnement à partir de la console Pandora FMS.

Configuration de la Console

Une licence SIEM spécifique doit être installée sur le PFMS Server. Si le Command Center est activé, Pandora SIEM ne sera disponible que sur les nœuds.

Pour utiliser la surveillance des événements SIEM, il faut d'abord l'activer à partir de la configuration principale.

Menu Management → Settings → System Settings → SIEM → Activate SIEM,remplissez complètement le formulaire et cliquez sur Update pour enregistrer les modifications.

Cela créera les modèles nécessaires sur le serveur OpenSearch spécifié pour la normalisation des *logs* et la génération d'événements SIEM.

Il activera également ce type de surveillance dans les serveurs Pandora FMS où `siemserver` et `siemevents` ont été lancés.

Configuration du serveur

Pour effectuer la surveillance des événements SIEM, il est nécessaire d'activer les serveurs `siemserver` et `siemevents` dans la configuration du serveur.

Pour décoder et normaliser les entrées de la collection *logs*, il sera nécessaire de disposer de fichiers de décodage XML qui établissent la manière d'obtenir les informations nécessaires dans chaque cas (ci-après, “decoders”). Ces fichiers XML seront situés dans le chemin indiqué par le paramètre `siem_decoders` de la configuration du serveur, par défaut dans:

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

Afin de générer des événements SIEM, il sera nécessaire de disposer de fichiers XML de règles qui établissent les conditions pour générer un événement sur la base des informations obtenues dans les *logs* standardisés (ci-après, “rules”). Ces fichiers XML seront situés dans le chemin indiqué par le paramètre `siem_events_rules` de la configuration du serveur, par défaut dans:

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

Les fichiers XML *decoders* et *rules* de Wazuh® sont compatibles avec la surveillance SIEM de Pandora FMS.

Configuration des agents

La majeure partie de la collecte des *logs* est effectuée par les EndPoints Pandora FMS. Ces agents, tant dans les systèmes Linux® que MS Windows®, disposent de types de modules spécifiques pour effectuer cette tâche.

Pour profiter pleinement de toutes les fonctionnalités du SIEM PFMS, il est recommandé d'avoir au moins la version 783 installée sur les EndPoints.

La surveillance SIEM dépend fortement du type de *log* collecté, il sera donc nécessaire de spécifier `module_source_type` dans les modules de collecte *logs* pour indiquer ce type.

Le type est utilisé par les *decoders* et les *rules*, il convient donc de consulter les *decoders* et les *rules* actifs pour savoir quel type indiquer dans chaque *log*.

Les types *logs* les plus couramment utilisés sont les suivants:

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

Agents Linux®

La collecte des *logs* sur les systèmes Linux se fait principalement par la lecture des fichiers *log*. Ceci peut être réalisé en utilisant des configurations de modules avec cette structure minimale:

```
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
```

Par exemple, pour collecter toutes les entrées de connexion *log* d'un serveur 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
```

Les entrées collectées à partir d'un *log* comme ci-dessus seraient normalisées par des, entre autres:

- *decoders* tels que `web-accesslog`,
- `web-accesslog-ip` ou,
- `web-accesslog-domain`.

Les *logs* décodés d'un *log* comme ci-dessus pourraient générer des événements tels que (entre autres):

- `Common web attack`,
- `XSS (Cross Site Scripting) attempt` ou,
- `SQL injection attempt`.

Agents MS Windows®

Sous MS Windows®, la collecte des *logs* se fait principalement en surveillant les événements du système, bien qu'elle puisse également se faire en lisant les fichiers *log* comme sur les systèmes Linux.

En utilisant le système d'événements Windows, ces *logs* peuvent être collectés en utilisant des configurations de modules avec l'une des deux configurations minimales.

Dans le cas d'événements de type Application, System ou Security:

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

Ou s'il s'agit d'événements sur un autre canal:

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

Par exemple, pour collecter toutes les entrées d'événements de l'application Security et 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
```

Les données recueillies à l'occasion d'événements tels que ceux mentionnés ci-dessus sont normalisées par *decoders* como `windows_eventchannel`.

Les *logs* décodés d'événements tels que ceux mentionnés ci-dessus pourraient générer des événements tels que (entre autres):

- `Windows error event`,
- `Short-time multiple Windows Defender warning events` ou,
- `Multiple Windows Defender error events`.

En utilisant la surveillance du fichier *log*, la configuration est identique à celle des systèmes Linux. Une configuration minimale comme celle-ci est nécessaire:

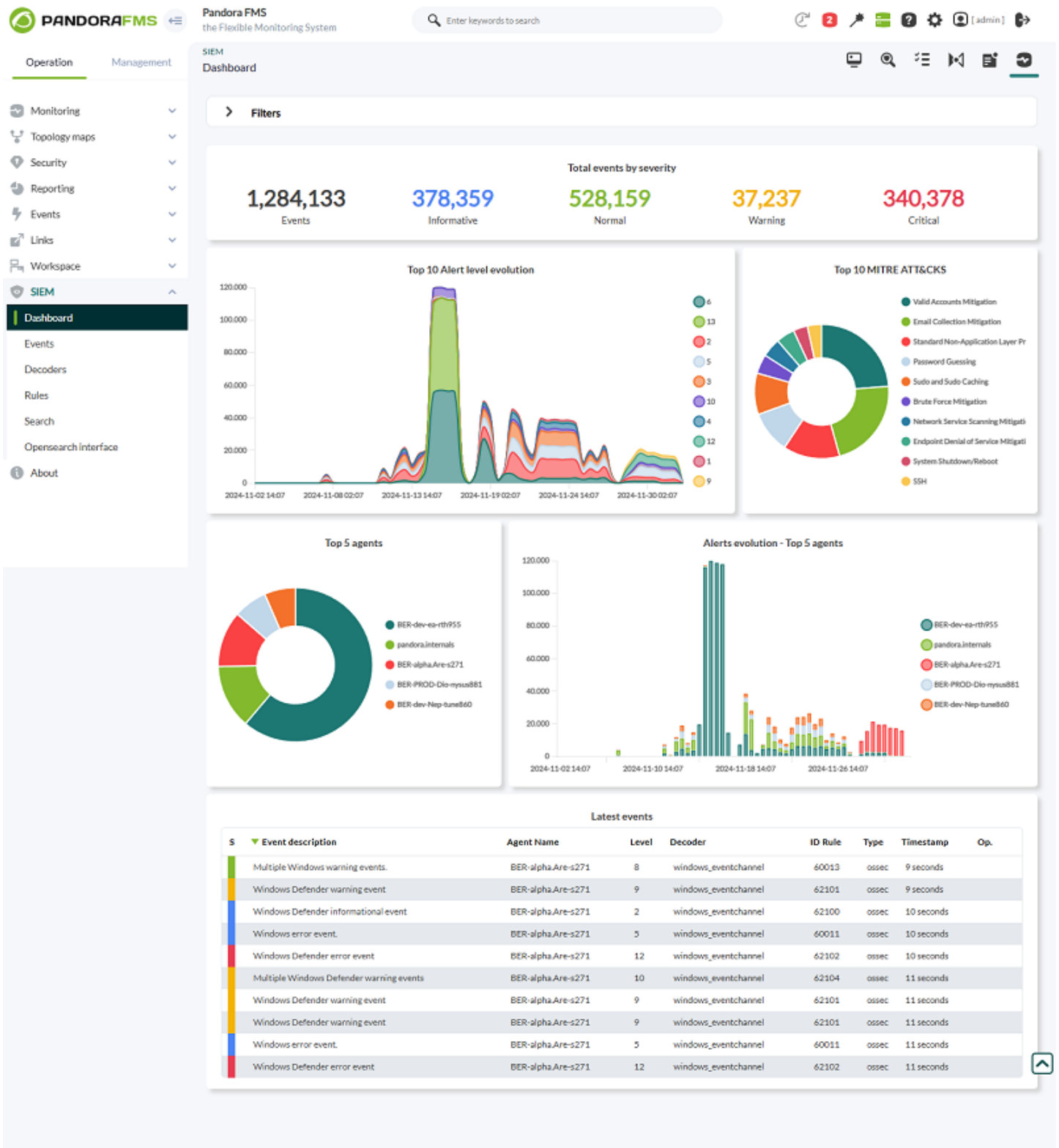
```
module_begin
module_name <program_name>
module_type log
module_regexp <path_to_log_file>
module_pattern <capture_regexp>
module_source_type <log_type>
module_end
```

Par exemple, pour collecter toutes les entrées de journal d'un serveur X:

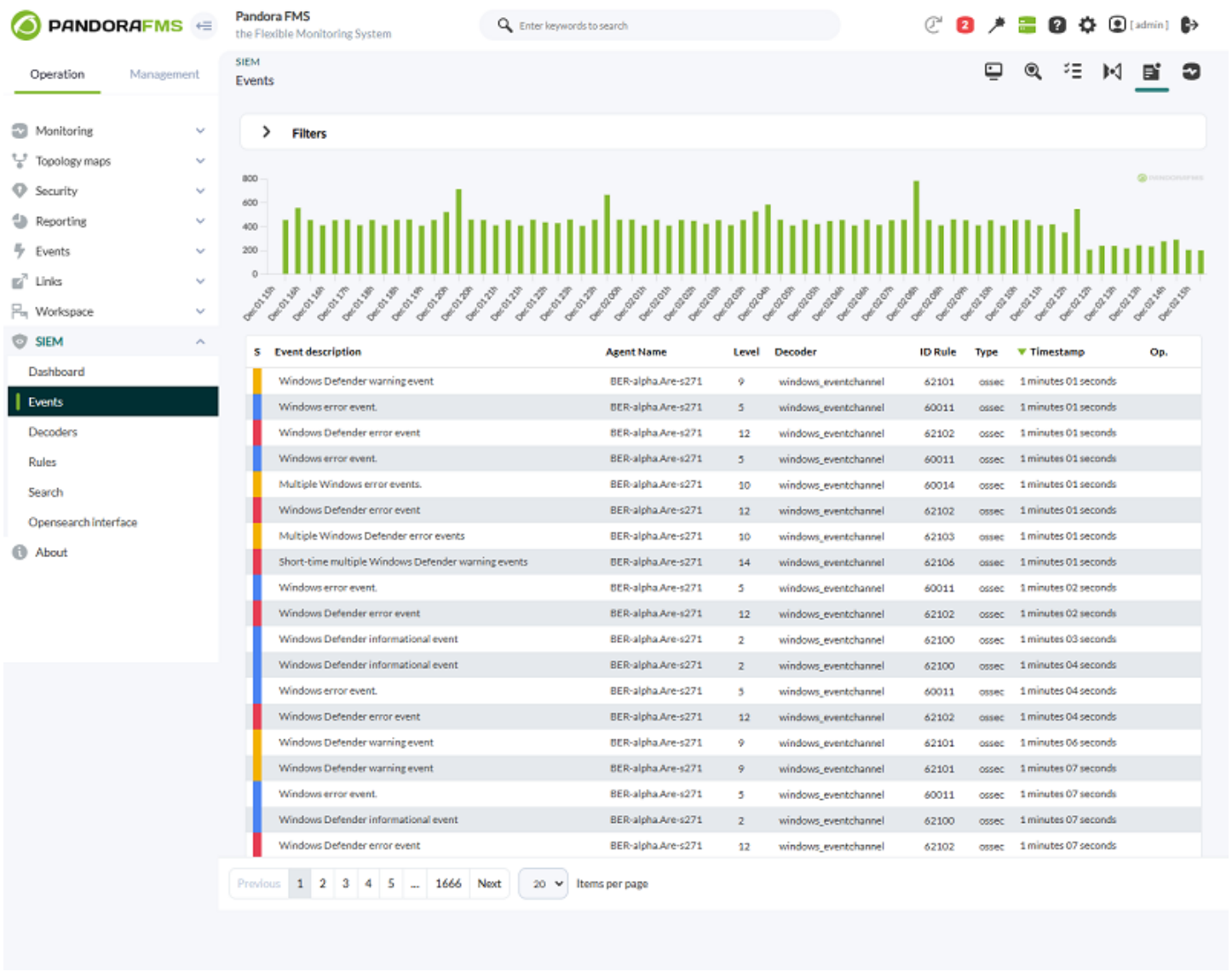
```
module_begin
module_name xserver
module_type log
module_regexp C:\server\logs\xserver.log
module_pattern .*
module_source_type xserver
module_end
```

Événements SIEM

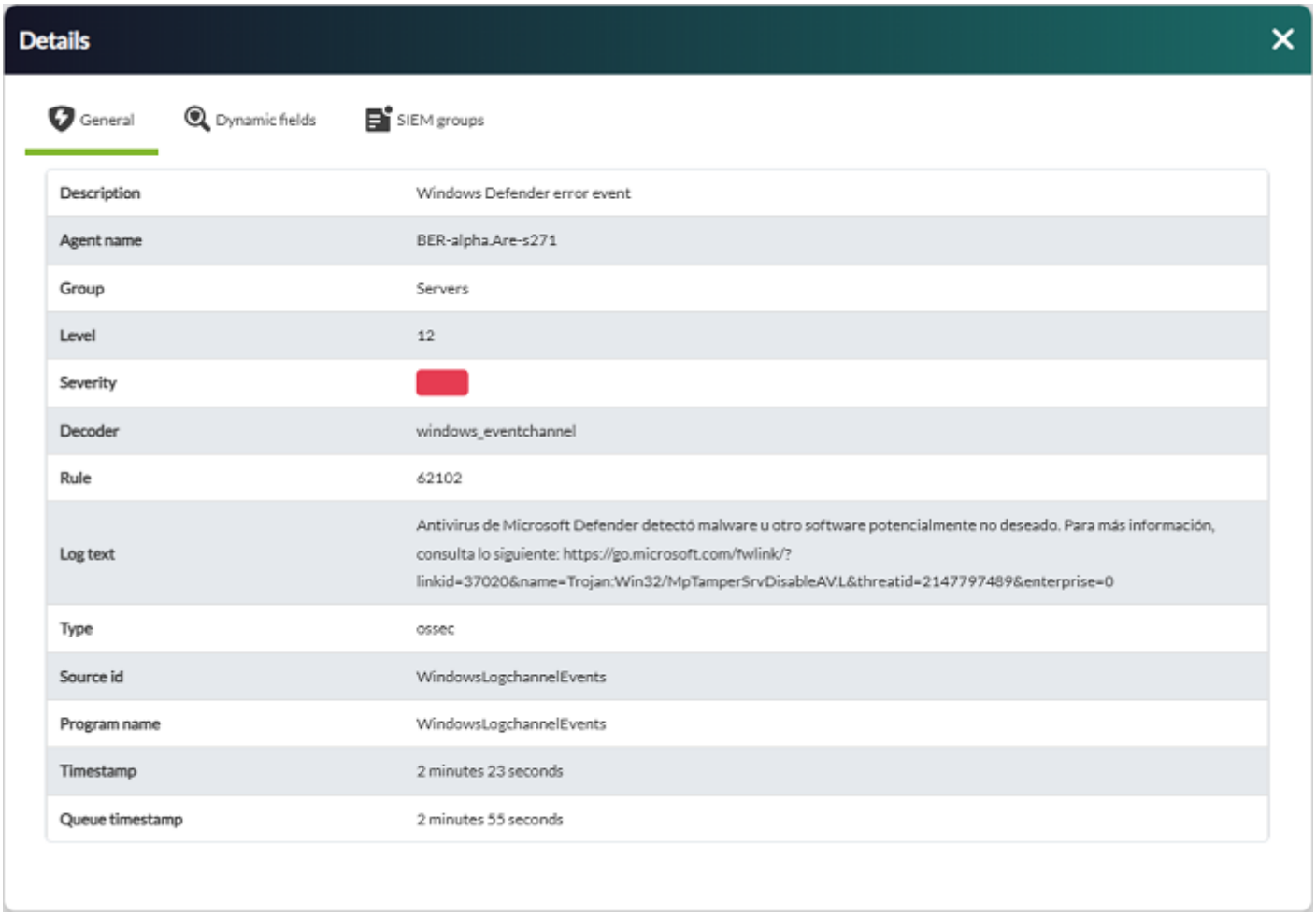
Lorsque la surveillance SIEM est activée et configurée, un aperçu de l'état de la surveillance est accessible dans le menu Operation → SIEM → Dashboard.



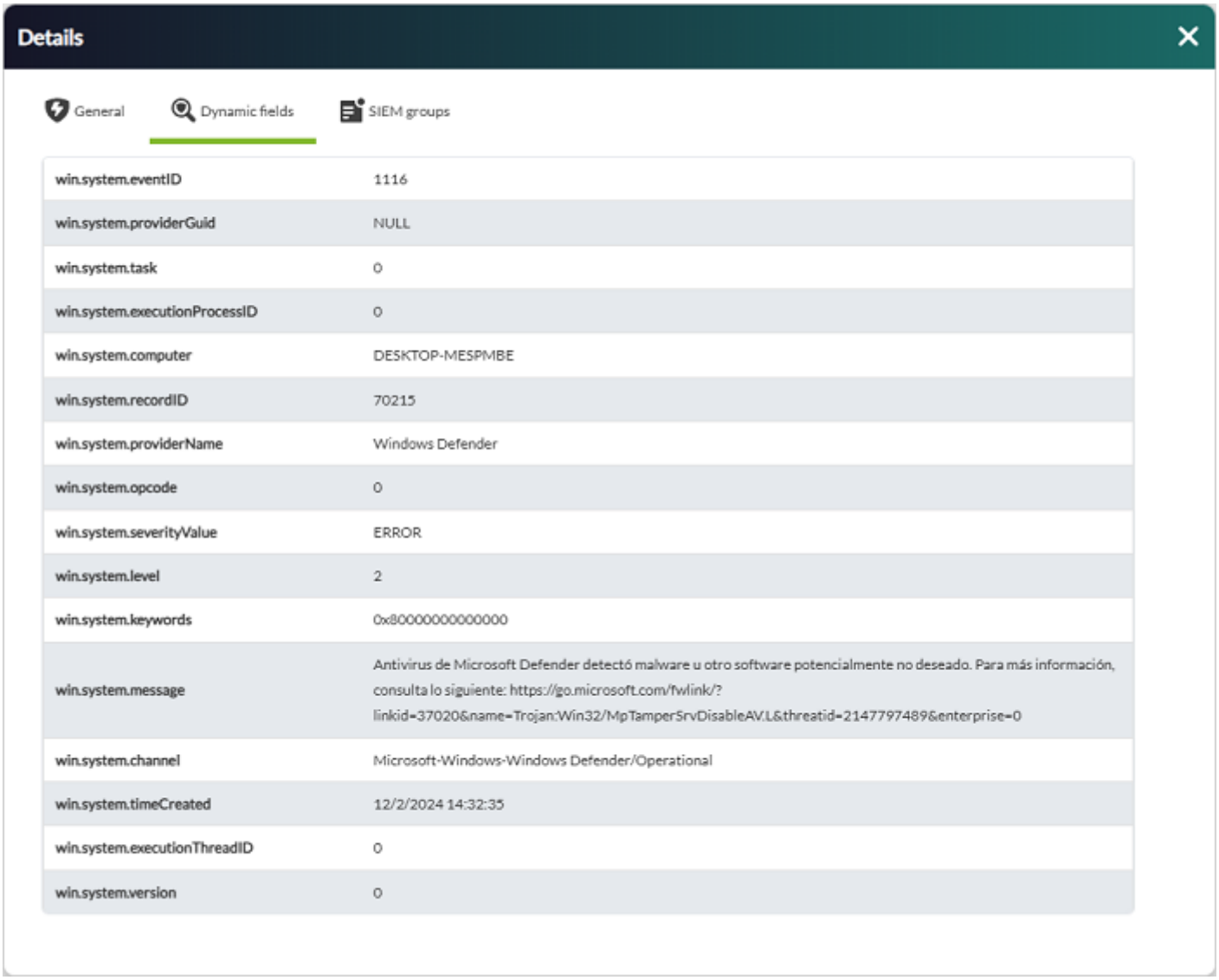
Les événements SIEM générés peuvent être consultés dans leur intégralité dans le menu Operation → SIEM → Events.



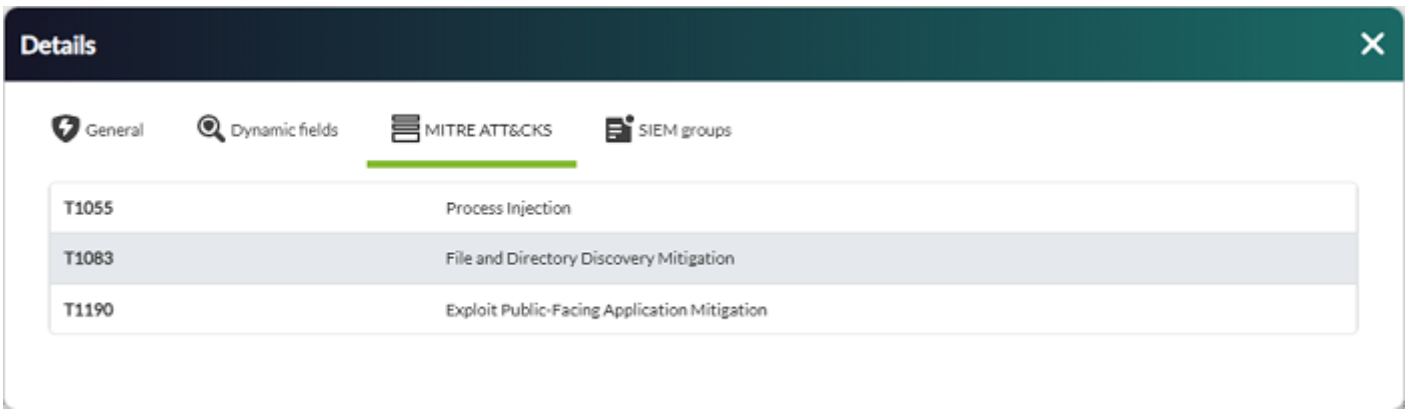
Chaque événement SIEM dispose d'une fenêtre de détails sur l'événement SIEM, affichant des informations sur le *log* normalisé et la règle qui a généré l'événement.

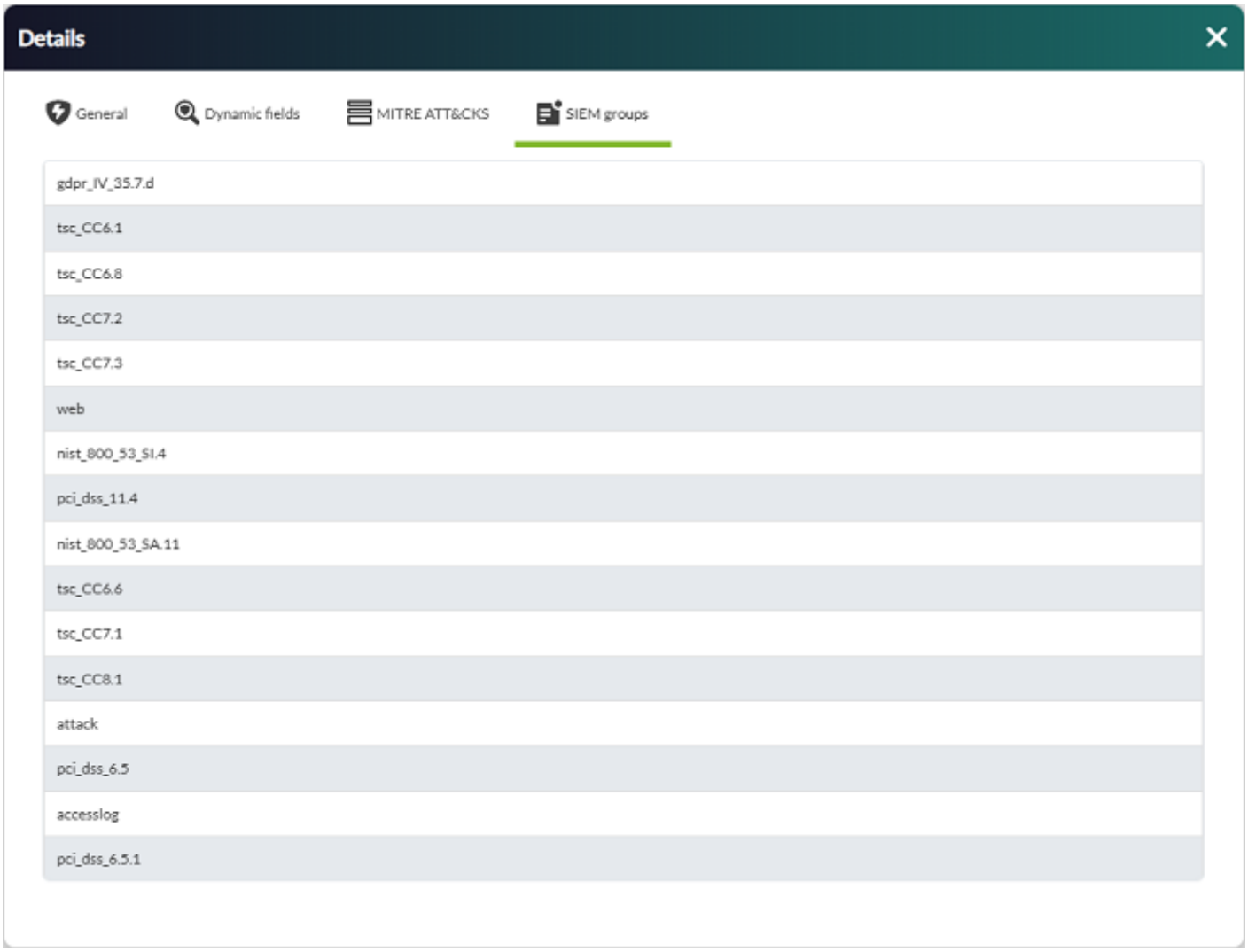


En fonction des *decoders* qui ont normalisé le *log*, l'événement aura un onglet Dynamic fields avec des informations utiles sur le *log*.



En fonction de la règle qui a généré l'événement, des onglets MITRE ATT&CKS® et SIEM groups contiennent des informations utiles sur l'impact de l'événement.





Les informations présentées dans le Dashboard general et dans le tableau des événements peuvent être incluses dans les Pandora FMS Dashboards à travers les *widgets* de SIEM inclus.

Decoders

Pandora FMS inclut par défaut une série de *decoders* pour la surveillance SIEM, mais tout administrateur peut inclure ses propres *decoders*.

Gestion

Pour inclure de nouveaux *decoders*, vous devez d'abord ajouter ou éditer un fichier XML dans le chemin indiqué au Pandora FMS server dans son paramètre de configuration `siem_decoders`.

Les *décodeurs* sont chargés dans l'environnement via des fichiers XML que le serveur Pandora FMS master lit à chaque démarrage du service.

Une fois que les *décodeurs* sont chargés, leur configuration est stockée dans la base de données, et chaque `siemserver` les traite pour les entrées obtenues en collectant les *logs*.

A partir de la console Pandora FMS, il sera possible de visualiser les *decoders* chargés avec leur configuration complète à partir du menu Operation → SIEM → Decoders.

Il est également possible de désactiver les *decoders* de cette vue, afin que `siemserver` ne les prenne pas en compte lors de la normalisation des entrées *log*.

Tous les *decoders* sont entièrement chargés à chaque redémarrage. Cela signifie que les *decoders* qui n'ont pas pu être lus à partir des fichiers XML ne seront pas disponibles (même s'ils l'ont été à un moment donné), et que les *decoders* qui ont été désactivés à partir de la console seront réactivés (s'ils existent).

Syntaxe

Les *decoders* sont configurés et chargés dans Pandora FMS avec des fichiers XML. Ces fichiers peuvent avoir la syntaxe valide suivante:

```
<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

↑Syntaxe complète

Utilisé pour indiquer les variables avec leurs valeurs, pour une utilisation ultérieure dans XML (`$VarName`).

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

decoder

↑Syntaxe complète

Il s'agit de l'information sur le *decoder*, avec son nom. Un seul fichier XML peut contenir plusieurs *decoders*.

- `name`: C'est le nom du *decoder*. Plusieurs *decoders* peuvent avoir le même nom, et tous sont évalués.
- `discard`: Avec une valeur `yes` ou `no`, permet aux *logs* d'être écartés de l'évaluation des *decoders* s'ils correspondent à celui-ci. Les *decoders* avec `discard="yes"` sont évalués avant les autres (tant qu'ils n'ont pas de *decoder parent*).

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

parent

↑Syntaxe complète

Spécifie le nom du *decoder* parent pour générer une structure hiérarchique.

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

program_name

↑Syntaxe complète

Le nom du programme à trouver dans les en-têtes *log* ou dans le `source_id` *log*.

- `type`: Permet de spécifier le type d'expression régulière. S'il n'est pas spécifié, OS Regex est utilisé.

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

Le cas échéant, il est spécifié pour PCRE2:

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

type

↑Syntaxe complète

Le type de *logs* pour lequel le *decoder* doit être comparé. Il doit correspondre au `type` du *log*.

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

prematch

↑Syntaxe complète

Si le contenu de *log* correspond, il génère un *log* normalisé.

- `type`: Permet de spécifier le type d'expression régulière. S'il n'est pas spécifié, OS Regex est utilisé.
- Voir offset.

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

regex

↑Syntaxe complète

Les groupes capturés avec cette expression régulière sont des informations normalisées pour *log*.

- `type`: Permet de spécifier le type d'expression régulière. S'il n'est pas spécifié, OS Regex est utilisé.
- Voir offset.

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

order

↑Syntaxe complète

Les valeurs capturées par les regex sont stockées dans ces noms de champs, dans l'ordre des groupes de capture.

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

json_null_field

↑Syntaxe complète

Les valeurs nulles dans les groupes de capture sont stockées sous forme de *string* vide ou rejetées.

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

offset

↑Syntaxe complète

Utilisé pour indiquer l'ordre dans lequel les vérifications sont effectuées et pour écarter du contenu de *log* pour les vérifications suivantes le texte correspondant à l'expression régulière : `after_parent`, `after_regex`, `after_prematch`. Par exemple:

```
<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>
```

Il vérifiera que le texte *log* correspond à l'expression régulière `^\d\d\d\d/\d\d/\d\d \d\d:\d\d:\d\d` , écartera ce contenu du texte et, lors de l'évaluation de l'expression régulière `regex`, il capturera ce qui est approprié. Dans l'exemple, je supprimerais une date du texte lors de l'évaluation pour simplifier les groupes de capture.

Rules

Pandora FMS inclut par défaut une série de *rules* pour la surveillance SIEM, mais tout administrateur peut inclure ses propres *rules*.

Gestion des règles

Pour inclure de nouvelles *rules*, d'abord ajoutez ou éditez un fichier XML dans le chemin indiqué au

serveur Pandora FMS dans son paramètre de configuration `siem_rules`.

Les *rules* sont chargées dans l'environnement via des fichiers XML que le serveur Pandora FMS master lit à chaque démarrage du service.

Une fois que les *rules* sont chargées, leur configuration est stockée dans la base de données, et chaque serveur `siemevents` les traite pour les entrées de supervision SIEM normalisées.

Depuis la console de Pandora FMS, il sera possible de visualiser les *rules* chargées avec leur configuration complète à partir du menu Operation → SIEM → Rules.

Il est également possible de désactiver des *rules* de cette vue, afin que `siemevents` ne les prenne pas en compte lors du traitement des *logs* normalisés.

Toutes les *rules* sont entièrement chargées à chaque redémarrage. Cela signifie que les *rules* qui n'ont pas pu être lues à partir des fichiers XML ne seront pas disponibles (même si elles l'ont été à un moment donné). Contrairement aux *decoders*, les *rules* ont un identifiant qui leur permet d'être désactivées à chaque rechargement.

Les *rules* qui n'ont pas pu être lues à partir des fichiers XML seront marquées dans la console comme « non actives » et ne seront pas prises en compte lors de la génération d'événements SIEM. Les *rules* qui ont été désactivées manuellement par un administrateur ne seront pas non plus prises en compte. Par conséquent, pour que les *rules* soient évaluées, elles doivent être actives et activées.

Une fois les règles activées et habilitées, si une règle doit dépendre de l'évaluation et du résultat d'une autre, la première règle doit avoir un identifiant numérique inférieur à la seconde (c'est-à-dire qu'elles s'exécutent dans l'ordre numérique croissant, voir l'étude de cas correspondant).

Classification

Les *rules* sont classées en plusieurs niveaux, allant du plus bas (`0`) au plus haut (`15`). Le tableau suivant décrit chaque niveau et fournit des informations sur la gravité de chaque événement généré par la surveillance SIEM.

Niveau	Titre	Description
0	Ignoré.	Aucune action n'est entreprise. Utilisé pour éviter les faux positifs. Ces règles sont analysées avant toutes les autres règles, incluent les événements sans rapport avec la sécurité et n'apparaissent pas dans le volet des événements de sécurité.

Niveau	Titre	Description
2	Notification système de faible priorité.	Notifications système ou messages d'état. Ils n'ont aucune incidence sur la sécurité et n'apparaissent pas dans le panneau des événements de sécurité.
3	Événements réussis/autorisés.	Il s'agit des tentatives de connexion réussies, des événements autorisés par le *firewall*, et cetera.
4	Erreur de faible priorité du système.	Erreurs liées à des configurations erronées ou à des dispositifs/applications inutilisés. Ces erreurs n'ont aucune incidence sur la sécurité et sont généralement dues à des installations par défaut ou à des tests de logiciels.
5	Erreurs générées par l'utilisateur.	Il s'agit notamment des mots de passe oubliés, des actions refusées, et cetera. En soi, ils n'ont pas d'importance pour la sécurité.
6	Attaque de faible importance.	Il s'agit d'un ver ou d'un virus qui n'a aucun effet sur le système (comme le code rouge pour les serveurs Apache, etc.). Cela inclut également les événements fréquents du *Intrusion Detection System* (IDS) et les erreurs fréquentes.
7	Correspondance des “mauvais mots”.	Il s'agit de mots tels que “mauvais”, “erreur”, et cetera. La plupart de ces événements ne sont pas classifiés et peuvent avoir une certaine importance du point de vue de la sécurité.
8	Première consultation.	Inclure les événements consultés pour la première fois. La première fois qu'un événement IDS est déclenché ou la première fois qu'un utilisateur se connecte. Il comprend également les actions relatives à la sécurité, telles que l'activation d'un *sniffer* ou d'autres activités similaires.
9	Erreur de source invalide.	Inclut les tentatives de connexion en tant qu'utilisateur inconnu ou à partir d'une source non valide. Peut avoir une incidence sur la sécurité (surtout si elle est répétée). Cela inclut également les erreurs liées au compte “admin” (*root*).
10	Plusieurs erreurs générées par l'utilisateur.	Il peut s'agir de plusieurs mots de passe incorrects, de plusieurs échecs de connexion, etc. Cela peut indiquer une attaque ou simplement le fait qu'un utilisateur a oublié ses informations d'identification.
11	Avertissements de contrôle d'intégrité.	Il s'agit notamment de messages relatifs à la modification de binaires ou à la présence de *rootkits* (pour *Rootcheck*). Ces messages peuvent indiquer une attaque réussie. Sont également inclus les événements IDS qui seront ignorés (grand nombre de répétitions).
12	Événement de grande importance.	Il s'agit de messages d'erreur ou d'avertissement provenant du système, du noyau, et cetera. Ils peuvent indiquer une attaque contre une application spécifique.
13	Erreur inhabituelle (importance élevée).	Correspond à un schéma d'attaque courant la plupart du temps.
14	Événement de sécurité de grande importance.	Il est le plus souvent déclenché par corrélation et indique une attaque.
15	Attaque sévère.	Possibilité de faux positifs. Une attention immédiate est nécessaire.

En fonction de ces niveaux, les événements auront une gravité spécifique qui sera affichée dans la Console:

- Informational: Niveaux 0 à 6.
- Normal: Niveaux 7 à 8.
- Warning: Niveaux 9 à 11.
- Critical: Niveaux 12 à 15.

Syntaxe

Éléments syntaxiques détaillés

```
<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>
```

Voir aussi l'étude de cas.

var

↑Syntaxe complète

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

Utilisé pour indiquer les variables avec leurs valeurs, pour une utilisation ultérieure dans XML (`$VarName`).

group

↑Syntaxe complète

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

Permet de regrouper des règles. Il est également utilisé pour les conditions des mêmes règles.

rule

↑Syntaxe complète

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

Il s'agit de l'information contenue dans la règle:

1. `id`: Identifiant de la règle. Il doit être unique (à moins qu'une autre règle ne soit écrasée).
2. `level`: Niveau de l'événement lorsqu'il est généré (`0` à `15`). Les règles de niveau `0` ne génèrent pas d'événement.
3. `frequency`: Nombre de fois qu'une simultanéité doit se produire pour générer un événement. La fréquence indiquée dans les règles est vérifiée pour les *logs* du même agent, et non pour n'importe quel *log*.
4. `timeframe`: Fenêtre temporelle dans laquelle les concordances doivent être données (secondes).
5. `ignore`: Cette règle remet à zéro les compteurs de `frequency` après ces secondes.
6. `overwrite`: Avec une valeur `yes` ou `no`, elle permet d'écraser la configuration d'une règle avec le même ID. Si avec `overwrite="yes"` la règle a `level="0"`, elle est évaluée avant toute autre règle et en cas de correspondance avec le *log*, elle écarte l'évaluation du reste des règles pour ce *log*.

if_matched_sid

↑Syntaxe complète

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

La règle est satisfaite si une autre règle avec l'ID indiqué a déclenché une alerte les fois indiquées par `frequency` dans le `timeframe`.

if_matched_group

↑Syntaxe complète

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

Comme if_matched_sid mais pour les groupes.

same_id

↑Syntaxe complète

```
        <same_id />
```

Les devises ayant le même numéro d'identification doivent être indiquées.

different_id

↑Syntaxe complète

```
        <different_id />
```

Les devises ayant des identifiants différents doivent être indiquées.

same_field

↑Syntaxe complète

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

Il doit y avoir des concomitances avec les mêmes valeurs dans le champ.

different_field

↑Syntaxe complète

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

Les concomitances doivent se produire avec des valeurs différentes dans le champ.

description

↑Syntaxe complète

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

Le texte de l'événement à générer.[1)]

match

↑Syntaxe complète

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

Si le contenu de *log* correspond, un événement SIEM est généré.

1. `type`: Permet de spécifier le type d'expression régulière. S'il n'est pas spécifié, OS Regex est utilisé..
2. `negate`: Avec une valeur de `yes`, il permet de refuser la correspondance.

regex

↑Syntaxe complète

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

Identique à ''match''.

1. `type`: Permet de spécifier le type d'expression régulière. S'il n'est pas spécifié, OS Regex est utilisé..
2. `negate`: Avec une valeur de `yes`, il permet d'annuler l'expression régulière.

decoded_as

↑Syntaxe complète

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

La règle est satisfaite si le *log* a été décodé par le *decoder* spécifié.

category

↑Syntaxe complète

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

La règle est satisfaite si le type du *decoder* correspond.

field

↑Syntaxe complète

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

Si le champ indiqué correspond à la valeur, il est conforme à la règle.

1. `negate`: La valeur `yes` permet de refuser la correspondance avec le champ.

program_name

↑Syntaxe complète

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

Si l'origine de *log* correspond, la règle est respectée.

1. `negate`: Avec une valeur de `yes`, il permet de refuser les correspondances avec l'origine *log*.

time

↑Syntaxe complète

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

Si l'événement est généré dans l'intervalle de temps indiqué, la règle correspond.

weekday

↑Syntaxe complète

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

Si l'événement est généré ces jours-là, la règle est conforme aux dispositions suivantes (`monday - sunday`, `weekdays`, `weekends`).

if_sid

↑Syntaxe complète

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

La règle est satisfaite si l'une des règles parentes est satisfaite.

if_group

↑Syntaxe complète

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

La règle est satisfaite si l'enregistrement satisfait à toute autre règle du groupe spécifié.

if_level

↑Syntaxe complète

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

La règle est satisfaite si une autre règle de même niveau a été satisfaite.

info

↑Syntaxe complète

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

Informations supplémentaires sur l'événement généré.[2)]

group

↑Syntaxe complète

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

Liste des groupes dans la règle.

mitre

↑Syntaxe complète

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

Liste des ID MITRE de la règle.

Étude de cas

Le code suivant décrit une règle pour le SELinux *logs* lié à PHP. Cette règle crée des événements pour toute tentative de connexion à des ports autres que les ports habituels d'un serveur 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>
```

Ainsi, les connexions aux ports 9200, 3306, 80 et 443 ne créeront pas d'événements. Il en va de même pour ceux de type *directory* ou *file*.

*Decoder*général utilisé pour 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>
```

Ordre d'évaluation des règles

Le cas d'étude suivant à résultat négatif illustre le cas où une règle « consulte » d'autres règles pour vérifier si celles-ci ont trouvé une correspondance et évaluer elle-même la correspondance afin de générer un événement.

Il existe une série de règles utilisées pour détecter les échecs de connexion sur un pare-feu matériel, afin de créer ensuite les alertes correspondantes :

```
<rule id="81641" level="1">
    <decoded_as>fortigate-firewall-v6</decoded_as>
    <description>Fortigate v6 messages grouped.</description>
</rule>
```

- La règle 81641 effectue une correspondance avec le journal décodé pour commencer à extraire les données.

```
<rule id="81603" level="0">
    <if_sid>81600,81601,81602,81641</if_sid>
    <description>Fortigate messages grouped.</description>
</rule>
```

- La règle ID 81603 dépend à son tour de plusieurs règles, dont la règle précédente 81641. Comme cette dernière 81641 est supérieure à 81603, la règle 81603 n'est pas exécutée.

```
<rule id="81614" level="4">
    <if_sid>81603</if_sid>
    <match>ssl-login-fail</match>
    <description>Fortigate: SSL VPN user failed login attempt.</description>
    <group>authentication_failed,
gdpr_IV_32.2,
gdpr_IV_35.7.d,
gpg13_7.1,
hipaa_164.312.b,
invalid_login,
nist_800_53_AC.7,
nist_800_53_AU.14,
pci_dss_10.2.4,
pci_dss_10.2.5,
tsc_CC6.1,
tsc_CC6.8,
tsc_CC7.2,
tsc_CC7.3,</group>
</rule>
```

- Une règle avec l'ID 100700 vérifie la correspondance avec une règle avec l'ID 81641, qui dépend à son tour de la règle ID 81603 (81614 est supérieur à 81603, elle est exécutée).

Le problème dans ce cas d'étude est que le 81614 ne renvoie aucun résultat au 100700 car le 81603 n'a pas été exécuté en raison de l'ordre numérique par ID.

Expressions régulières

Les expressions régulières sont des séquences de caractères qui définissent un motif.

Il existe deux types d'expressions régulières valables pour les *decoders* et les *rules*: OS Regex et PCRE2.

OS Regex

Il s'agit d'expressions régulières simples basées sur une bibliothèque en langage C. Elle est conçue pour être simple tout en prenant en charge les expressions régulières les plus courantes.

Expressions acceptées

Expression	Caractères valides
\w	A-Z, a-z, 0-9, '-', '@', '_'
\d	0-9
\s	Espaces “ “
\t	Tabulation
\p	()*+,-.:;<=>?[]!"'#$%&|{}
\W	Tout ce qui n'est pas \w
\D	Tout ce qui n'est pas \d
\S	Tout ce qui n'est pas \s
\.	Tout ce qui

Modificateurs

Expression	Comportement
+	Faire correspondre une ou plusieurs fois
*	Pour correspondre à zéro ou plusieurs fois

Caractères spéciaux

Expression	Comportement
^	Pour spécifier le début du texte
$	Pour spécifier la fin du texte
|	Pour créer un motif logique “ou” entre plusieurs motifs

Caractères échappés

Pour utiliser les caractères suivants, vous devez les échapper avec \:

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

Limites

- Les modificateurs * et + ne peuvent être appliqués qu'aux expressions avec barre oblique inverse, et non aux caractères simples (par exemple, `\d+` est pris en charge, `0+` ne l'est pas).
- Vous ne pouvez pas utiliser l'alternance dans un groupe, par exemple `(foo|bar)` n'est pas autorisé.
- Les retours en arrière complexes ne sont pas pris en charge, par exemple `\p*\d*\s*\w*:` ne correspond pas à un seul deux-points car `\p*` consomme les deux deux-points.
- `.` correspond à un point littéral, tandis que `\.` correspond à n'importe quel caractère.
- `\s` ne correspond qu'à un espace ASCII (32), pas à d'autres espaces tels que les tabulations.
- Il n'existe pas de syntaxe pour faire correspondre un caret, un astérisque ou un caractère plus littéral (bien que `\p` fasse correspondre un astérisque ou plus, ainsi que d'autres caractères).

PCRE2

Perl Compatible Regular Expression (PCRE2) offre des fonctionnalités telles que les motifs récursifs, les assertions de type “look-ahead” et “look-back”, les groupes de non-capture, les quantificateurs non-voraces, une syntaxe étendue pour les caractères et les classes de caractères, entre autres.

Pour plus de détails, voir la Documentation syntaxique CPRE2.

Expressions acceptées

Expression	Caractères valides
`.`	Tout caractère à l'exception de la nouvelle ligne
`\d`	Tout chiffre décimal, égal à `[0-9]`
`\D`	Tout caractère, autre qu'un chiffre décimal, égal à `[^0-9]`
`\h`	Tout caractère d'espace blanc horizontal
`\H`	Tout caractère autre qu'un espace horizontal
`\s`	Tout caractère d'espacement, égal à `[\t\r\n\f]`
`\S`	Tout caractère qui n'est pas un espace vide, égal à `[^\t\r\n\f]`
`\w`	N'importe quel caractère “mot”.
`\W`	Tout caractère “non-mot”.

Modificateurs

Expression	Comportement
?	0 ou 1, greedy
?+	0 ou 1, possessive
??	0 ou 1, lazy
*	0 ou plus, greedy
*+	0 ou plus, possessive
*?	0 ou plus, lazy

Expression	Comportement
+	1 ou plus, greedy
++	1 ou plus, possessive
+?	1 ou plus, lazy
{n}	Exactement n
{n,m}	Au moins n, pas plus de m, greedy
{n,m}+	Au moins n, pas plus de m, possessive
{n,m}?	Au moins n, pas plus de m, lazy
{n,}	n ou plus, greedy
{n,}+	n ou plus, possessive
{n,}?	n ou plus, lazy

Caractères échappés

Expression	Comportement
\f	Page suivante (hexadecimal 0C)
\n	Nouvelle ligne (hexadecimal 0A)
\r	Retour du chariot (hexadecimal 0D)
\t	Tabulation (hexadecimal 09)
\0dd	Caractère codé en octal 0dd
\o{ddd..}	Caractère codé en octal ddd..
\xhh	Caractère codé en hexadécimal hh
v\x{hhh…}	Caractère codé en hexadécimal hh..

Alertes SIEM

Voir le sujet Système d'alerte SIEM.

SIEM rapports

Voir le sujet Rapports d'événements SIEM.

Retour à l'index de la documentation de Pandora FMS

1) , 2)

Les variables peuvent être utilisées dans la description et dans `info` avec les valeurs des champs personnalisés, par exemple: `$(Field1)`, `$(Field2.Sub1)`, `$(Field2.Sub2)`.