Desarrollo de plugins de agente

From:
https://pandorafms.com/manual/!776/
Permanent link:
https://pandorafms.com/manual/!776/es/documentation/pandorafms/technical_reference/06_anexo_agent_plugins
Last update: 2024/06/10 14:34

Desarrollo de plugins de agente

Características básicas del plugin de agente

El *plugin* de agente es ejecutado por el Agente Software de Pandora FMS por lo que tiene que tener unas características especiales:

- Cada ejecución del *plugin* podrá devolver uno o varios módulos con sus correspondientes valores. La salida deberá tener un formato XML.
- Podrá acceder tanto a recursos locales a la máquina o a recursos de otra máquina de forma remota.
- Es posible usar cualquier tipo de lenguaje de programación soportado por el sistema operativo en el que está instalado el agente software de Pandora FMS.
- Todas las dependencias o software necesario para ejecutar el *plugin* deberá estar disponible o ser instalado en la misma máquina que ejecuta el agente de Pandora FMS.

Los plugin de agente pueden realizar una especie de *tarea de reconocimiento* ya que el plugin puede devolver varios módulos en una ejecución y el número puede cambiar entre distintas ejecuciones.

En UNIX y GNU/Linux el resultado de la ejecución del plugin debe ser `0`, en otro caso se ignorará el resultado del *plugin*.

Ejemplo desarrollo plugin de agente

A continuación se explica un ejemplo de un *plugin* sencillo. Este plugin de agente devuelve el porcentaje de uso de los *filesystems* del sistema. El código es el siguiente:

```
#!/usr/bin/perl

use strict;

sub usage() {
        print "\npandora_df.pl v1r1\n\n";
        print "usage: ./pandora_df\n";
        print "usage: ./pandora_df tmpfs /dev/sda1\n\n";
}

# Retrieve information from all filesystems
my $all_filesystems = 0;

# Check command line parameters
if ($#ARGV <0) {
```

```
        $all_filesystems = 1;
}

if ($ARGV[0] eq "-h") {
        usage();
        exit(0);
}

# Parse command line parameters
my %filesystems;
foreach my $fs (@ARGV) {
        $filesystems{$fs} = '-1%';
}

# Retrieve filesystem information
# -P use the POSIX output format for portability
my @df = `df -P`;
shift (@df);

# No filesystems? Something went wrong.
if ($#df <0) {
        exit 1;
}

# Parse filesystem usage
foreach my $row (@df) {
        my @columns = split (' ', $row);
        exit 1 if ($#columns <4);
        $filesystems{$columns[0]} = $columns[4] if (defined
($filesystems{$columns[0]}) || $all_filesystems == 1);
}

while (my ($filesystem, $use) = each (%filesystems)) {

        # Remove the trailing %
        chop ($use);

        # Print module output
        print "<module>\n";
        print "<name><![CDATA[" . $filesystem . "]]></name>\n";
        print "<type><![CDATA[generic_data]]></type>\n";
        print "<data><![CDATA[" . $use . "]]></data>\n";
        print "<description>% of usage in this volume</description>\n";
        print "</module>\n";
}

exit 0;
```

Una parte importante del código es la función usage:

```
sub usage() {
```

```
        print "\npandora_df.pl v1r1\n\n";
        print "usage: ./pandora_df\n";
        print "usage: ./pandora_df tmpfs /dev/sda1\n\n";
}
```

En esta función describe la versión y cómo usar el *plugin*, es muy importante y siempre se debe mostrar al ejecutar el *plugin* sin ningún tipo de parámetro o bien con una opción tipo `-h` o `–help`. En este ejemplo se ejecuta cuando se usa el parámetro `-h`, lo verifican las siguientes líneas:

```
if ($ARGV[0] eq "-h") {
        usage();
        exit(0);
}
```

Respecto a los valores devueltos por el plugin se puede observar que una vez se han recogido los datos de los siguientes ficheros se crea e imprime una parte de XML por la salida estándar para cada uno de ellos, esta labor se realiza en las siguientes líneas:

```
while (my ($filesystem, $use) = each (%filesystems)) {

        # Remove the trailing %
        chop ($use);

        # Print module output
        print "<module>\n";
        print "<name><![CDATA[" . $filesystem . "]]></name>\n";
        print "<type><![CDATA[generic_data]]></type>\n";
        print "<data><![CDATA[" . $use . "]]></data>\n";
        print "<description>% of usage in this volume</description>\n";
        print "</module>\n";
}
```

Un ejemplo del resultado que devuelve este *plugin* podría ser:

```
<module>
<name><![CDATA[tmpfs]]></name>
<type><![CDATA[generic_data]]></type>
<data><![CDATA[0]]></data>
<description>% of usage in this volume</description>
</module>
<module>
<name><![CDATA[/dev/mapper/VolGroup-lv_home]]></name>
<type><![CDATA[generic_data]]></type>
<data><![CDATA[26]]></data>
<description>% of usage in this volume</description>
</module>
<module>
<name><![CDATA[/dev/sda9]]></name>
<type><![CDATA[generic_data]]></type>
<data><![CDATA[34]]></data>
```

```
<description>% of usage in this volume</description>
</module>
```

El número de módulos devueltos por este plugin dependerá del número de *filesystems* configurados y podrá cambiar entre las diferentes ejecuciones.

El fragmento de XML se añade al XML general que genera el agente software y es enviado al servidor Pandora FMS para ser procesado por el Data Server.

Solución de problemas

Si Pandora FMS no reconoce su plugin de agente, no consigue la información esperada o el agente simplemente no funciona, aquí hay varias cosas que debería tener en cuenta.

Revise el documento pandora_agent.conf

El Agente de Software requiere de una línea en este archivo con la ruta correcta del *plugin*.

Ejemplo:

```
 module_plugin /etc/pandora/plugins/MyMonitor.pl
/etc/pandora/plugins/MyMonitor.conf 2> /etc/pandora/plugins/MyMonitor.err
```

MyMonitor.pl es el plugin de agente, MyMonitor.conf es el archivo de configuración pasado como argumento, y MyMonitor.err es un archivo que recibirá posibles errores de la ejecución del plugin y mantendrá limpia la salida estándar.

Reinicie el pandora_agent_daemon

Si tiene la versión básica de Pandora FMS (no Enterprise), el Agente de Software ejecutará los *plugins* cada 5 minutos. Para aquellas personas que no pueden esperar, es posible reiniciar el Agente de Software desde la línea de comandos.

Ejemplo:

```
/etc/init.d/pandora_agent_daemon restart
```

Revise los permisos del plugin

El plugin y los archivos que va a usar, deben tener los permisos correctos de lectura, escritura y ejecución. En UNIX, esto debería bastar:

```
chmod 755 <ruta_plugin>
```

Valide la salida

Una forma fácil de encontrar los errores, es ejecutar el plugin manualmente en línea de comandos. Por favor revise la salida cuidadosamente.

Por ejemplo:

```
popeye:/etc/pandora/plugins # ./pandora_df
<module>
<name><![CDATA[/dev/sda2]]></name>
<type><![CDATA[generic_data]]></type>
<data><![CDATA[19]]></data>
<description>% of usage in this volume</description>
</module>
<module>
<name><![CDATA[udev]]></name>
<type><![CDATA[generic_data]]></type>
<data><![CDATA[1]]></data>
<description>% of usage in this volume</description>
</module>
```

Valide el XML resultante

El XML que imprime el plugin debe tener una sintaxis de XML válida . El XML, además, debe estar bien formado. Para comprobar si lo está, puede seguir estos dos pasos desde la línea de comandos:

1. Cree un documento XML con la salida del plugin: `./Plugin.pl > Plugin.xml`
2. Compruebe el documento XML: `xmllint Plugin.xml` .

Modo Debug

Puede activar el modo de desarrollo cambiando el valor de la etiqueta *debug* en el archivo `pandora_agent.conf` de `0` a `1`. Una vez haya hecho esto, cuando el Agente de Software ejecute el plugin, los resultados se guardarán en un documento XML con toda la información del agente.

El nombre del documento será el nombre del agente con la extensión *.data*, y estará localizado en el directorio */tmp* (eche un vistazo al *log* del agente PFMS en `/var/log/pandora/pandora_agent.log`). Revisando el documento, podrá ver si los datos de su *plugin* están siendo recogidos y si son como esperaba.

Cuando habilite el modo *Debug*, el agente se ejecuta solo

una vez y posteriormente sale.

Foro

Si después de todo, el error persiste, puede preguntar en el foro PFMS.

Volver al Índice de Documentación Pandora FMS.