Web Monitoring

From:
https://pandorafms.com/manual/!778/
Permanent link:
https://pandorafms.com/manual/!778/en/documentation/pandorafms/monitoring/06_web_monitoring
2024/12/03 19:30

Web Monitoring

We are working on the translation of the Pandora FMS documentation. Sorry for any inconvenience.

Classic Web Monitoring

Introduction

In Pandora FMS it works as an independent server, similar to the Network server, the WMI server or the remote plugins server. This system operates under the principle of web transaction, where each completed transaction against one or several WEB pages is defined by one or more consecutive steps, which must conclude satisfactorily to consider that the transaction has ended successfully.

- Web server has important limitations such as the dynamic management of JavaScript at runtime.
- For more complex web transactions, Pandora FMS has another much more powerful (and complex) component called WUX Monitoring (Web User Experience).

Installation and configuration

The Pandora FMS server must be activated and restarted with:

```
webserver 1
```

Depending on the number of requests, you may have to increase the number of threads and the default timeout:

```
web_threads 1
web_timeout 60

# Use curl or LWP
web_engine curl
```

Pandora FMS has protection against CSRF and it may happen that web checks, when debugged, get this message: `Cannot verify the origin of the request` Take this protection into account when considering the use of "WUX Monitoring".

Creation of web modules

To remotely monitor a web page, once the agent is created, click on the top modules tab (Modules). In it, select create Create a new webserver module and press the Create button:

The type of WEB check must be selected:

- Remote HTTP module to check latency: Obtains the total time that passes from the first request until the last one is checked (in a WEB test there are one or several intermediate requests that complete the transaction). If the check definition has defined that the transaction be carried out more than once, the average time of each request will be used.
- Remote HTTP module to check server response: Gets a 1 (OK) or a 0 (CRITICAL) as a result of checking the entire transaction. If there are several attempts but at least one of them fails, the test as a whole is considered to also fail. Precisely, the number of attempts is sometimes used to avoid false positives, to do this use the retries field in advanced fields.
- Remote HTTP module to retrieve numeric data: Obtains a numeric value, parsing the HTTP response using a regular expression to obtain that value.
- Remote HTTP module to retrieve string data: Analogous to the previous point but with a text string.
- Remote HTTP module to check server status code: Through the curl tool duly enabled with the `web_engine curl` configuration token, HTTP headers can be paid.

Web checks: This essential field defines the WEB check to be performed. This is defined in one or more steps, or simple requests. Checks start with the tag `task_begin` and end with the tag `task_end`.

For form checking there are several additional variables:

- resource (1 or 0): Download all resources from the web (images, videos, etc.).
- cookie (1 or 0): Maintains a cookie, or an open session for later checks.
- variable_name: Name of a variable in a form.
- variable_value: Value of the previous variable in the form.

In some cases of domain redirection the checks may not work. One way to fix this is to modify the module pointing to the final domain, for example curl -L

Check loading time of a website

To check the response time or latency of a web page, select the module type Remote HTTP module to check latency. Example:

```
task_begin
get https://pandorafms.com
task_end
```

- For the download time to include all resources (JavaScript, CSS, images, etc.), `resource 1` must be added in a line before `task_end`.

- Web checks also support the use of proxy in the Proxy URL token.

The web download time is not the time it takes to display a website in a browser, as this is usually dependent on the JavaScript load time.

Form check on a web page

A form check is much more complex than simply checking text on a web page. In order to carry out this type of checks, you must have the necessary credentials. Additionally, you need to go to the page and get the HTML code to get the variable names, and then you need to have minimal knowledge of HTML to enter the query for the web server.

The practical method to design a WEB transactional test with multiple steps is to test them one by one in debugging mode.

Using Simple HTTP Authentication

Some pages may require simple HTTP authentication. It is generally used as a quick check, a minimal security greeting that allows access to more advanced security checks (encryption, data persistence, etc.).

- Using quotes in the password for `http_auth_pass` is not supported.
- Avoid using single quotes.

Web services and API monitoring

REST type APIs can be monitored, except for more complex types of APIs based on protocols such as SOAP or XMLRPC.

By checking the output with a regular expression, you can verify that everything is correct. Example:

```
task_begin
get https://swapi.dev/api/planets/1/
get_content Tatooine
task_end
```

For more complex responses, other regular expressions and the `get_content_advanced` token should be used.

- It is important to correctly define the capture groups in parentheses so that the call is made properly.
- When making calls to the API it is important to keep in mind that the target API must have permissions to be consulted.

Advanced Options

Modifying HTTP headers

With the header option you can modify HTTP header fields or create custom fields. For example, to change the Host field of the HTTP header:

```
task_begin
get http://192.168.1.5/index.php
header Host 192.168.1.1
task_end
```

Debugging web checks

Web checks can be debugged by adding the `debug <log_file>` option. Two files `log_file.req` and `log_file.res` will be created with the contents of the HTTP request and the response, respectively. For example:

```
task_begin
get http://192.168.1.5/index.php
debug /tmp/request.log
task_end
```

Using Curl instead of LWP

The LWP utility can cause problems when many threads carry out HTTPS requests (due to an OpenSSL limitation). The alternative is to use the curl tool. To solve this problem, edit the file `/etc/pandora/pandora_server.conf` and add the following line:

```
web_engine curl
```

Restart the Pandora FMS server, and the Curl binary will be used to perform web checks instead of LWP.

Advanced transactional monitoring

In addition to the functionality offered by PFMS Web server, there are other ways to perform transactional web monitoring.

- In a distributed manner (UX), deployed in “agent” mode on systems other than the server, even on non-accessible networks.
- Centrally (WUX).

Return to Pandora FMS documentation index