Supervision Web

Supervision Web

Surveillance de l'expérience de l'utilisateur Web

Introduction

En la versión Enterprise es posible monitorizar una web utilizando el componente WEB Server (Goliat Server).

Dans la version Enterprise, il est possible de surveiller un Web en utilisant le composant WEB Server, aussi appelé Goliat Server.

Cette fonctionnalité est issu d'un ancien projet du fondateur de Pandora FMS. Goliat F.I.S.T a été un projet opensource pour réaliser des audits dynamiques sur des serveurs Web. Le code source (de 2002) est toujours accessible mais a été abandonné vers 2010.

Sur Pandora FMS, il fonctionne comme un serveur indépendant, semblable au Serveur réseau, au Serveur WMI ou au serveur de plugins distants. Ce système opère sous le principe de transaction WEB, où chaque transaction réussie contre une ou plusieurs pages WEB est définie par une ou plusieurs étapes consécutives, qui doivent être conclues de façon satisfaisantes pour considérer que la transaction a été effectuée avec succès. L'exécution d'une transaction web reproduit fidèlement le processus de navigation complet, qui peut inclure des divers aspects comme s'authentifier dans un formulaire, cliquer sur une option du menu ou remplir un formulaire, en vérifiant que chaque étape renvoie une chaîne de caractères concrète.

Toute erreur au cours d'une étape du processus donnerait comme résultat une erreur dans la vérification. La transaction réussie inclut le téléchargement de toutes les ressources (graphiques animations, etc) que prévoit la navigation réelle. En plus de réaliser des vérifications de fonctionnement et de temps de réponse, il est possible d'extraire des valeurs des pages web pour ensuite les traiter.

Goliat est capable de surveiller aussi bien en HTTP qu'en HTTPS de façon transparente pour l'utilisateur. Il supporte la gestion de sessions grâce aux cookies, à l'étape de paramétrages et

bien-sûr, le téléchargement des ressources associées à chaque page. Il a aussi des limites importantes telles que la gestion dynamique de JavaScript pendant l'exécution. Pour des transactions web plus complexes, Pandora FMS dispose d'un autre composant beaucoup plus puissant (et complexe), appelé Supervision WUX (Web User Experience).

Installation et configuration

Pour pouvoir utiliser Goliat, vous devez tout d'abord l'activer dans le serveur Enterprise de Pandora FMS :

```
webserver 1
```

Selon le nombre de requêtes à réaliser, il est possible que vous deviez augmenter le nombre de threads et le timeout par défaut :

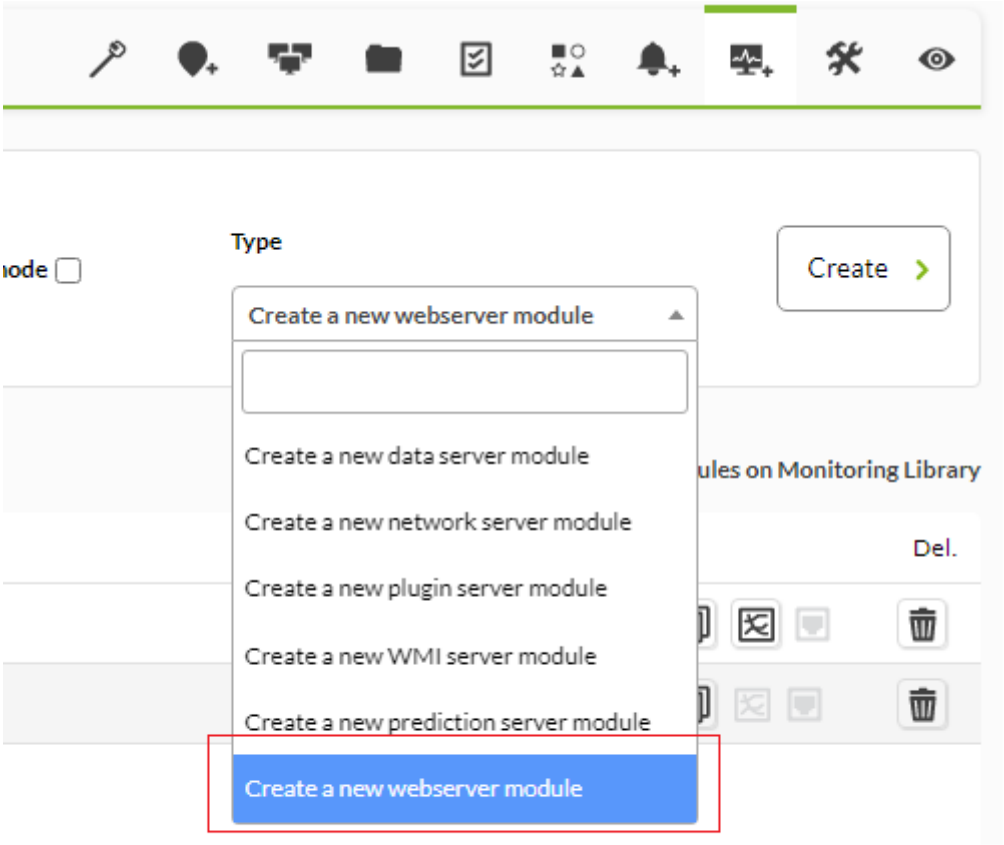
```
web_threads 1
web_timeout 60
```

Il existe un token de configuration avancé qui vous permettra de changer le type de librairie utilisée sous Goliat, LWP ou CURL. Par défaut, LWP est utilisé, bien que si vous rencontrez un problème (connections, timeout longs ou problèmes de concurrence si beaucoup de threads), vous pouvez passer en CURL :

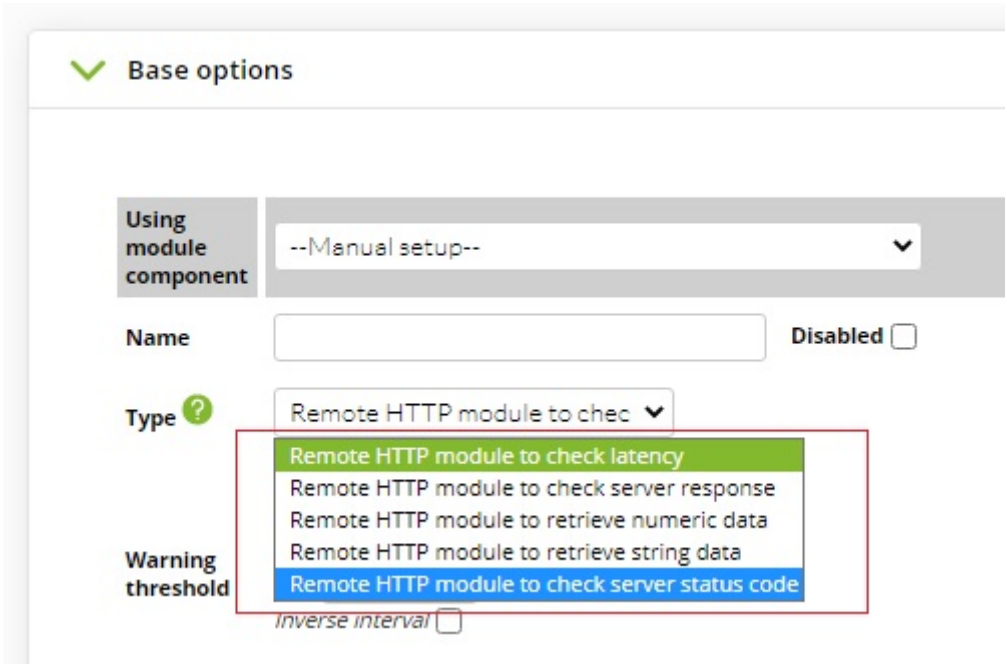
```
web_engine curl
```

Création de modules web

Pour surveiller à distance une page web, une fois l'agent créé, appuyez sur l'onglet des modules. Sélectionnez ensuite créer un nouveau module de serveur web (webserver module) et cliquez sur Create :



Clickez Create et un formulaire apparaîtra dans le quel les champs nécessaires pour surveiller un web devront être remplis:



Seléctionnez le type de check WEB :

- Remote HTTP module to check latency : il permet d'obtenir le temps total qui s'écoule, depuis la

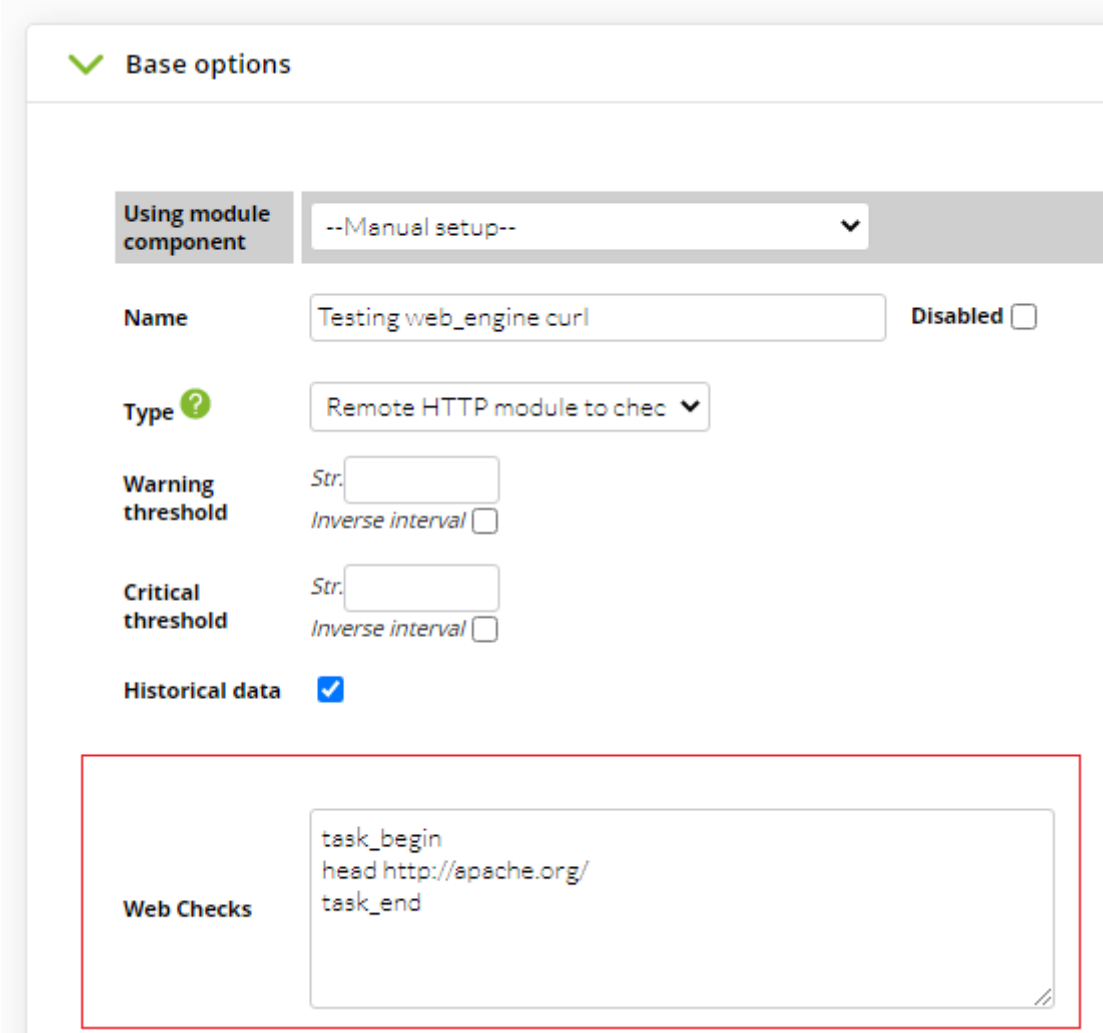
première requête jusqu'à la vérification de la dernière (dans un essai WEB, il existe une ou plusieurs requêtes intermédiaires qui achèvent la transaction). Si dans la définition du test, il est mentionné que la transaction se réalise plus d'une fois, le temps moyen de chaque requête sera alors utilisé.
- Remote HTTP module to check server response : Il obtient un 1 (OK) ou un 0 (CRITICAL) comme résultat de la vérification de toute la transaction. Si, au cours de plusieurs essais, au moins un d'entre eux présente une erreur, l'essai dans son ensemble sera alors considéré comme défaillant. Par ailleurs, le nombre de tentatives peut être utilisé pour éviter les faux positifs, pour ça utilisez le champ *nouvelle tentative* dans des champs avancés.
- Remote HTTP module to retrieve numeric data : Il obtient une valeur numérique, en analysant la réponse HTTP en utilisant une expression régulière pour obtenir cette valeur.
- Remote HTTP module to retrieve string data : Semblable au précédent, mais avec une chaîne de caractères.
- Remote HTTP module to check server status mode : Grâce à l'outil curl correctement activé avec la token de configuration `web_engine curl`, les entêtes HTTP peuvent être retournés.

Web checks

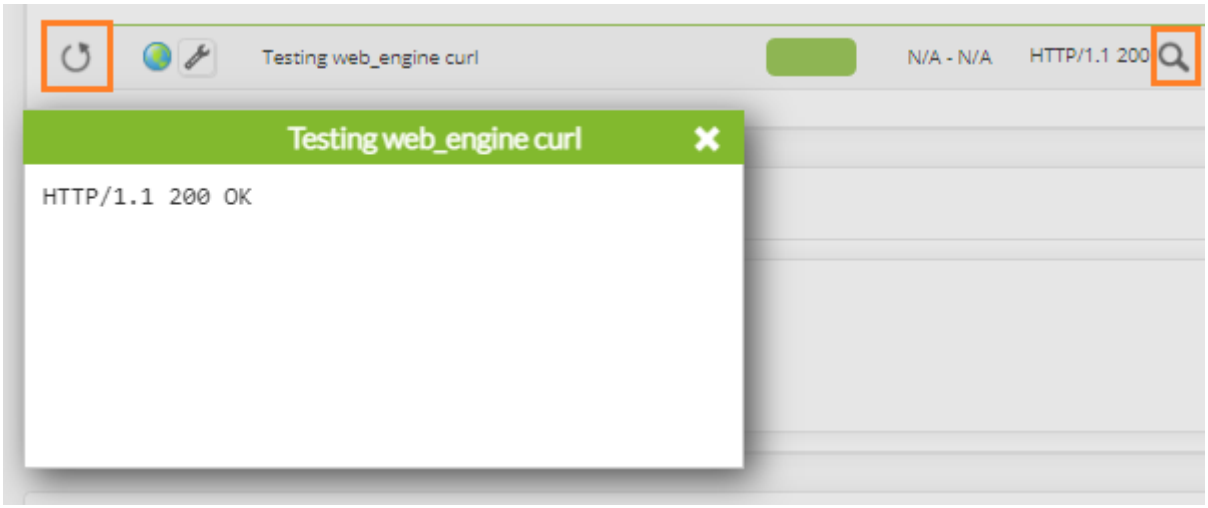
Ce champs primordial permet de définir la vérification WEB qui va se réaliser. Cette dernière se définit en une ou plusieurs étapes, ou par de simples requêtes. Elles doivent être écrites dans un format spécial dans le champ Web checks. Les vérifications se lancent avec l'étiquette `task_begin` et se terminent avec l'étiquette `task_end`.

Un exemple achevé d'une transaction facile serait similaire à celle-ci :

```
task_begin head
http://apache.org/
task_end
```



Après sauvegarder, forcez l'exécution du Module et visualisez son résultat :



Un autre exemple avec plus de commandes:

```
task_begin
get https://apache.org/
cookie 0
resource 0
check_string Apache Software Foundation
```

```
task_end
```

Dans cet exemple de base, nous vérifions s'il existe une chaîne sur une page web. C'est à cela que la variable `check_string` sert. Elle ne permet pas de vérifier le code HTML en soi, elle ne cherche que des sous-chaînes de caractères. Nous cherchons « Apache Software Foundation » sur le web http://apache.org. S'il cette chaîne de caractères existe, le test renverra OK (en cas de *Remote HTTP module to check server response*).

Pour s'assurer qu'une chaîne n'existe pas sur une page web, vous pouvez utiliser la variable `check_not_string` :

```
check_not_string Section 3
```

Las arguments pris par la syntaxe de `check_string` ne sont pas des chaînes de texte normales, ils sont des expressions regulières. C'est à dire, si vous chrechez la chaîne `Pandora FMS (4.0)` vous devrez le chercher avec un expression regulière, par exemple : `Pandora FMS \(4.0\)`. Cela vous permet de faire des recharches plus puissants, mais vous devez avoir sur compte que n'importe quel charactère qui n'est pas une lettre ou une chiffre doit être *échappé* avec `\`.

Pour la vérification des formulaires, il existe plusieurs variables supplémentaires :

- resource (1 ó 0) : Télécharge toutes les ressources du web (images, vidéos, etc).
- cookie (1 ó 0) : Conserve un cookie, ou une session ouverte pour des vérifications futures.
- variable_name : Nom d'une variable dans un formulaire.
- variable_value : Valeur de la variable précédente dans le formulaire.

Avec ces variables, des données pourront être envoyées aux formulaires et pourront vérifier qu'elles fonctionnent correctement.

Dans certains cas de redirections de domaines, les tests pourraient ne pas fonctionner. Pour résoudre le problème, vous devrez créer le module ciblant le dernier domaine.

Pour le cas antérieur, la commande curl a le paramètre

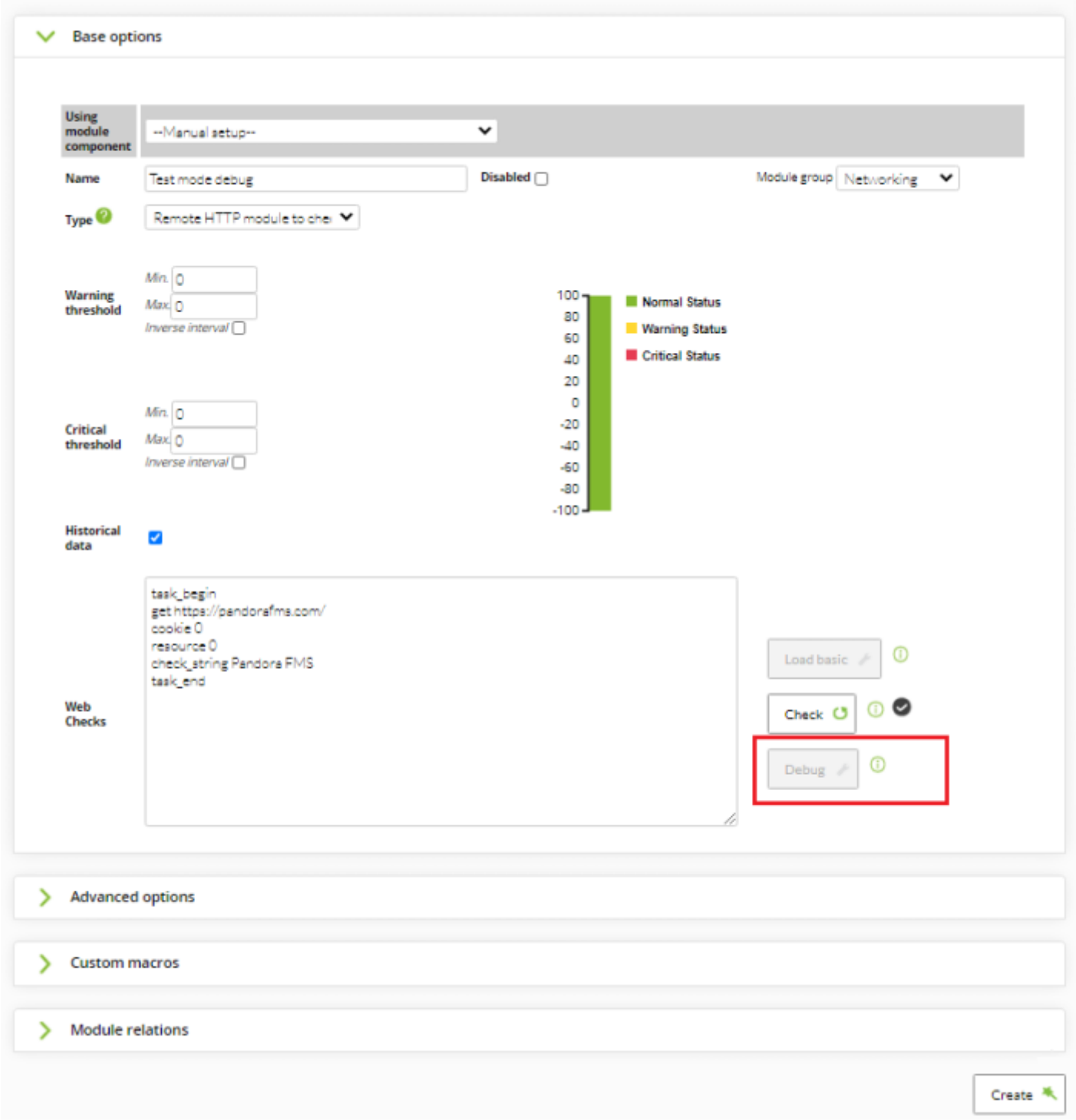
```
-L
```

dans sa version courte et

```
--location
```

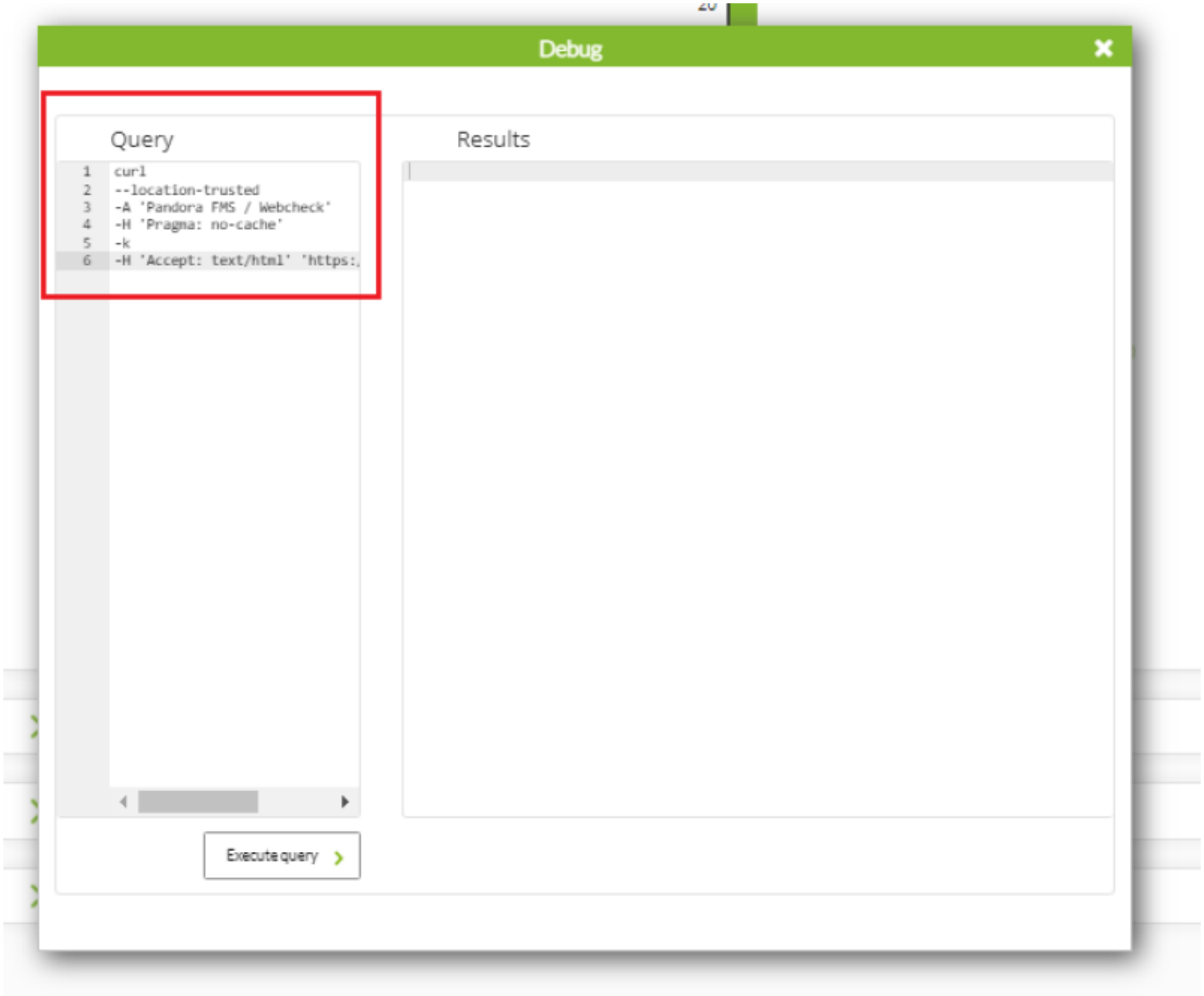
dans sa version longue, lorsque il recève des rédirections HTTP 3XX il l'éxecute à nouveau contre le domaine rédirectionné. Cependant la fléxibilité de Pandora FMS permet d'utiliser le bouton de

dépuration.

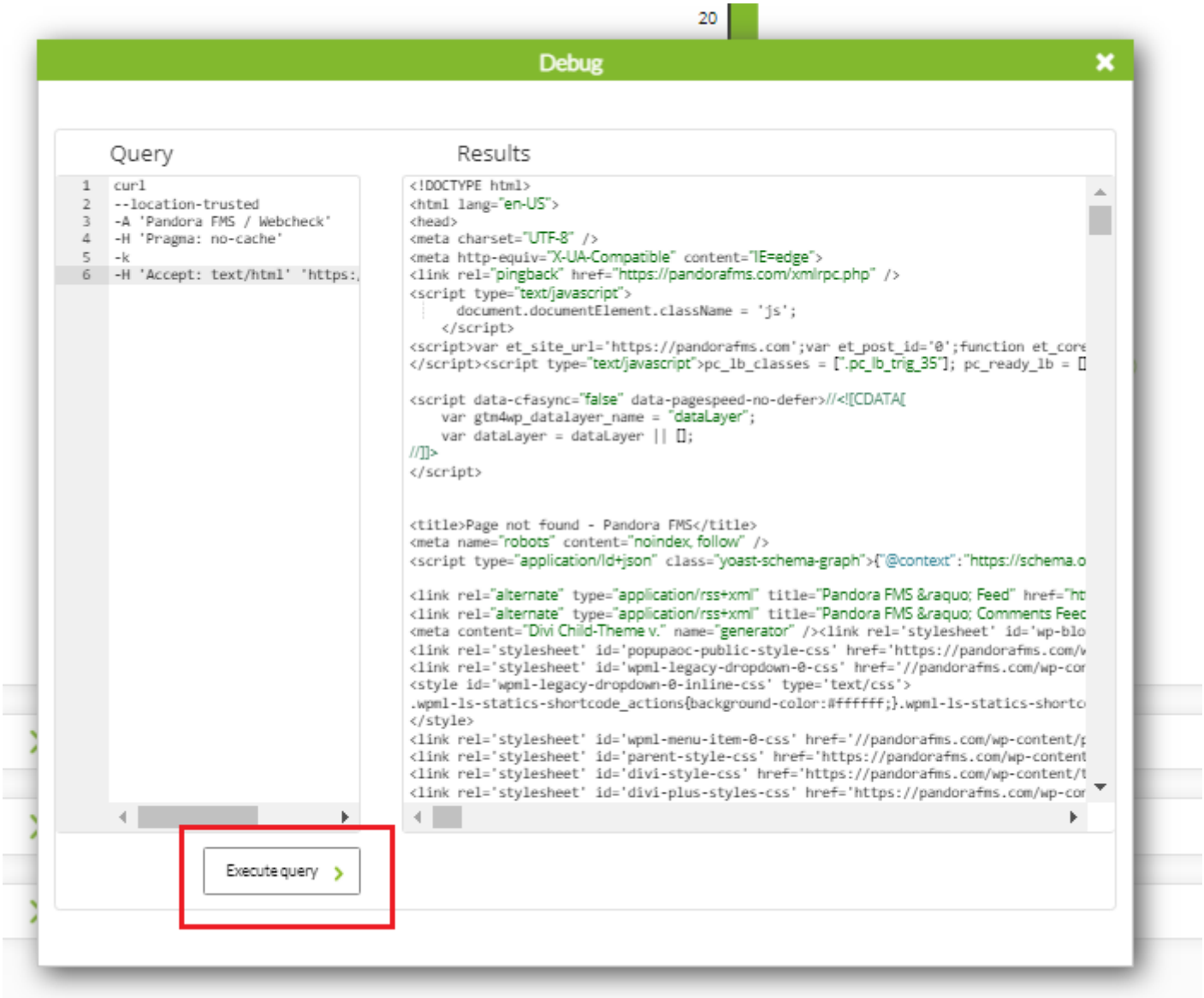


Lorsque le Module est créé il est inactif et vous pouvez l'utiliser après la première verification, dont vous pouvez forcer l'exécution pour gagner du temps.

Lorsque vous modifiez cette Module, cliquez sur Debug et vous pourrez entrer dans le mode de dépuration pour éditer la requête Query :



Bien que vous pouvez exécuter la requête du module avec le bouton Execute query ainsi que modifier et l'exécuter à nouveau avec des autres valeurs jusqu'à que vous obtiendrez le résultat souhaité.



Vérifier le temps de chargement d'un web

Si nous souhaitons vérifier le temps de réponse ou de latence d'une page web, il ne faut que sélectionner le type de module Remote HTTP module to check latency. Par exemple, si nous souhaitons connaître la latence de chargement de la page web http://pandorafms.com :

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

Nous pouvons ajouter les token de configuration `resource 1` pour que le temps de téléchargement qui est calculé soit en téléchargeant toutes les ressources (Javascript, CSS, images, etc), calculant ainsi une donnée plus réelle.

Le temps de téléchargement web N'EST PAS le temps nécessaire pour voir un web dans un navigateur puisqu'il dépend du temps de chargement de Javascript. Goliat

télécharge Javascript mais ne l'exécute pas.

Tests grâce à un Proxy

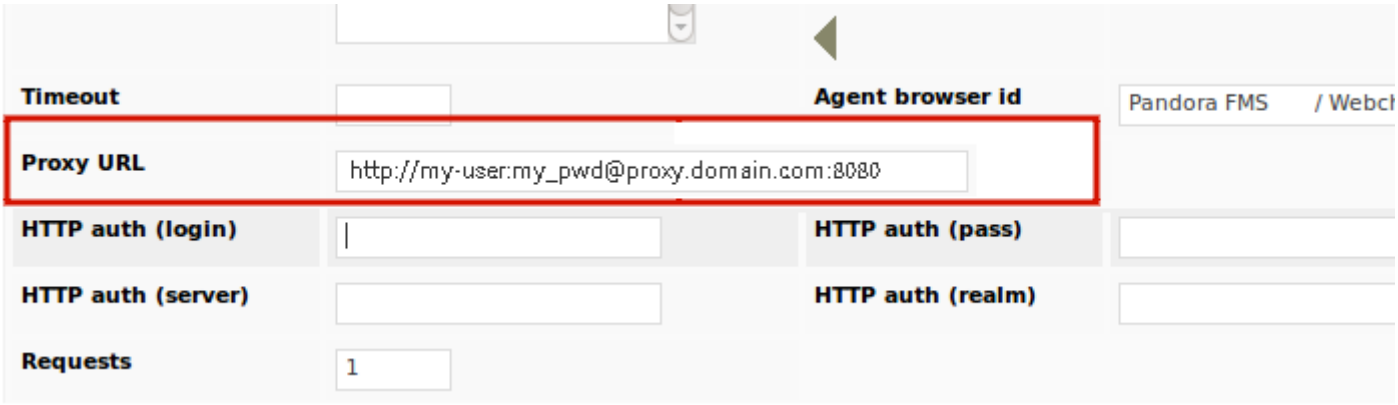
Les tests web supportent aussi l'utilisation de proxy. Pour le configurer, vous devez ajouter l'URL du proxy dans la case que vous trouverez en cliquant sur Advanced options :

Par exemple :

```
http://proxy.domain.com:8080
```

Si le proxy requiert une authentification, il doit utiliser le schéma suivant, où `my-user` est le nom d'utilisateur et `my_pwd` le mot de passe :

```
http://my_user:my_pwd@proxy.domain.com:8080
```

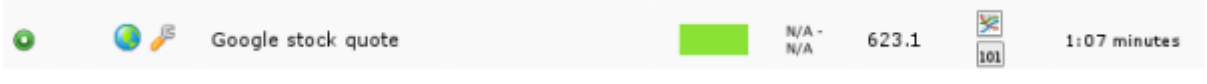


En obtenant des données d'une page web

Nous ne voulons peut-être pas savoir si un site Web en particulier fonctionne ou combien de temps cela prend mais nous souhaitons obtenir une valeur en temps réel, comme par exemple la valeur en bourse de Google. Pour cela, nous utiliserons un module Remote HTTP module to retrieve numeric data avec l'expression régulière ou *regex* appropriée :

```
task_begin
get http://finance.google.com/finance/info?client=ig&q=NASDAQ%3aGOOG
get_content \d+\.\d+
task_end
```

La sortie sera semblable à celle-ci :



Il est aussi possible de préciser une expression régulière plus compliquée pour collecter les

données de réponses HTTP plus complexes, avec le token de configuration `get_content_advanced` :

```
task_begin
get http://finance.yahoo.com/q?s=GOOG
get_content_advanced <span id="yfs_l84_goog">([\d\.]+)</span>
task_end
```

En la renvoyant, la partie de l'expression régulière définie sur `get_content_advanced` doit être écrite entre parenthèses.

Pour configurer les seuils qui déclencheront les états d'avertissement ou critique, nous utiliserons la configuration du module pour vérifier que la chaîne reçue coïncide avec ce qui est attendu.

Vérification de formulaire sur une page web

Une vérification plus intéressante est celle d'un formulaire web. C'est évidemment beaucoup plus complexe qu'une simple vérification de texte sur une page web. Cette vérification prise comme exemple utilisera la console de Pandora FMS, démarrera la session et vérifiera qu'elle en est effectivement capable, en vérifiant un texte dans la section de Workspace qui montre les données de l'utilisateur qui a démarré la session. S'il s'agit d'une console par défaut, l'utilisateur admin possédera la description « Admin Pandora ».

Pour pouvoir utiliser ce type de vérifications, vous devez disposer des informations d'identification nécessaires pour pouvoir démarrer une session. De plus, il faudra aller sur la page et obtenir le code HTML pour pouvoir voir les noms des variables. Ensuite, il faut avoir un minimum de connaissances en HTML pour comprendre comment fonctionne Goliat.

L'idéal, au moment de la conception d'un test transactionnel WEB avec plusieurs étapes, c'est de le tester étape par étape, au cas où quelque chose manquerait lors de l'une d'entre elles.

La Console d'exemple est dans :

```
http://192.168.70.116/pandora_console/
```

Analysant leur code HTML, on observe que les variables du formulaire web et de connexion :

- `nick` : Nom de l'utilisateur.
- `pass` : Mot de passe de l'utilisateur.

Les variables `variable_name` et `variable_value` devront être utilisées ensemble pour pouvoir valider le formulaire. La Console Pandora FMS a par défaut les valeurs `admin` et `pandora`.

La première étape consiste à accéder au formulaire, envoyer l'utilisateur et le mot de passe et s'identifier (nous verrons dans la seconde étape comment déterminer le succès de cette authentification).

```
task_begin
post http://192.168.70.116/pandora_console/index.php?login=1
variable_name nick
variable_value admin
variable_name pass
variable_value pandora
cookie 1
resource 1
task_end
```

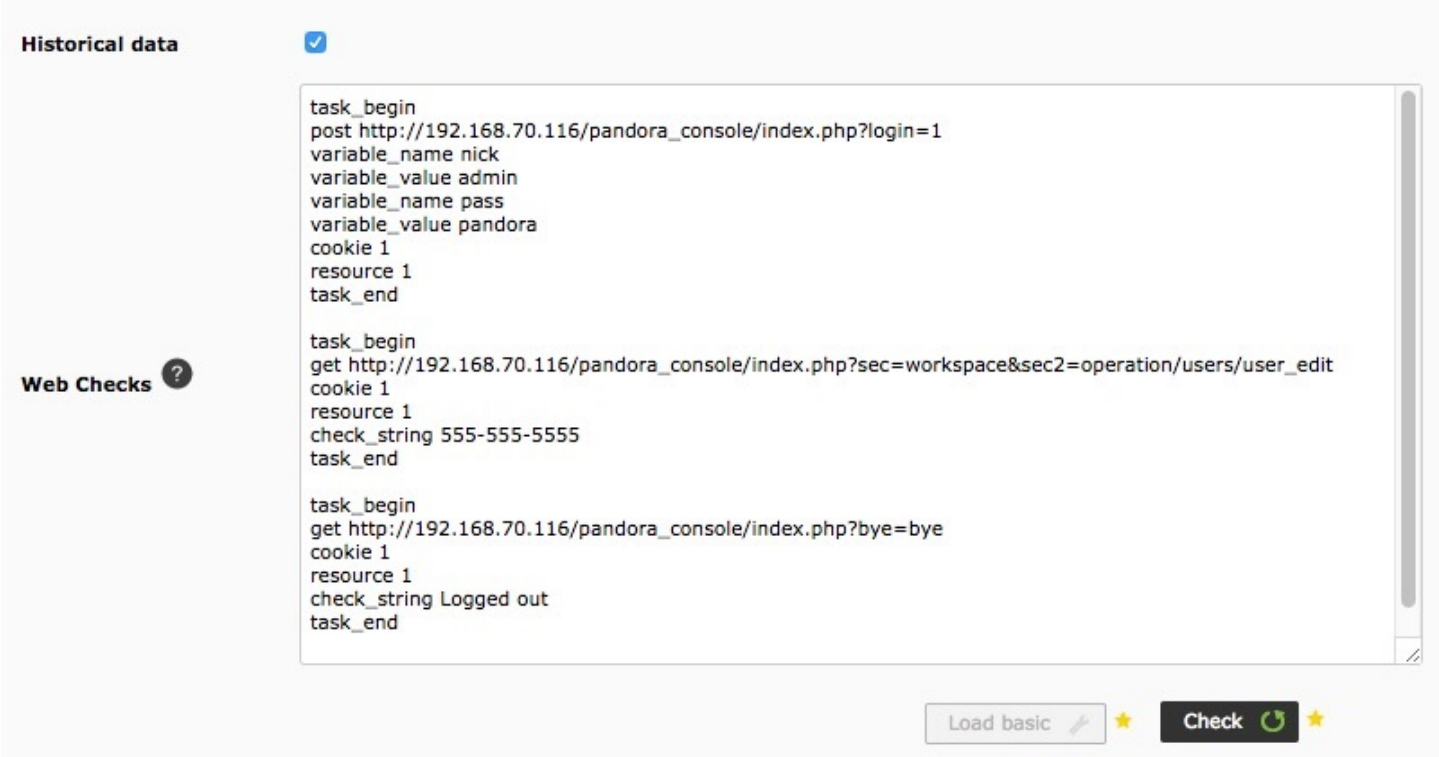
Le *token* `cookie 1` pour maintenir la persistence de valeurs obtenues dans l'étape précédente, si l'authentification a réussi. Après vous accédez à la page de détails de l'utilisateur pour chercher le numéro de téléphone, qui par défaut pour « admin » est 555-555-555. Si vous le pouvez « voir » ça indique que vous avez vous connecté correctement dans la Console (notez l'utilisation de la commande `check_string`) :

```
task_begin
get
http://192.168.70.116/pandora_console/index.php?sec=workspace&sec2=operation/use
rs/user_edit
cookie 1
resource 1
check_string 555-555-5555
task_end
```

Et pour terminer, déconnectez-vous de la console et nous chercherons le message de déconnexion `Logged out` :

```
task_begin
get http://192.168.70.116/pandora_console/index.php?bye=bye
cookie 1
resource 1
check_string Logged out
task_end
```

Vérification totale de le *script* :



Comportements des requêtes WEB

Les champs des propriétés avancées sont semblables à ceux des autres types de modules, bien qu'il existe quelques champs différents et propres aux tests WEB :

Timeout

C'est le temps d'expiration pendant la requête. Si ce temps est dépassé, la requête de vérification sera ignorée.

Agent browser id

C'est l'identifiant de navigateur web à utiliser, puisque des certaines pages n'acceptent que quelques navigateurs web (consulter https://www.zytrax.com/tech/web/browser_ids.htm) pour obtenir plus d'informations).

Requests

Pandora FMS répétera la vérification autant de fois que c'est indiqué dans ce paramètre. Si l'une est défaillante, c'est la vérification entière qui sera considérée comme erronée. Selon la quantité de vérifications dans le module, un nombre déterminé de pages sera obtenu. C'est-à-dire que si le module se compose de trois vérifications, trois pages seront téléchargées. Si dans le champ Requests, une valeur est indiquée, alors le nombre de téléchargement sera multiplié par cette dernière. Il est important de prendre cela en compte pour savoir le temps total que prendra le module pour achever les opérations.

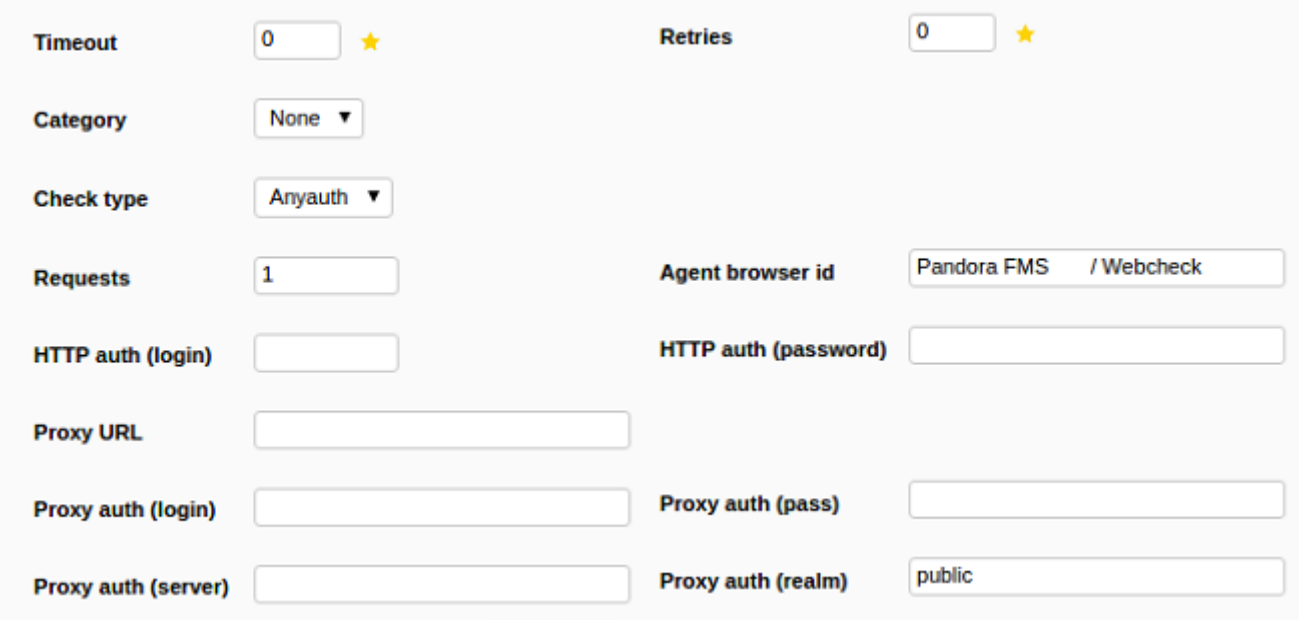
Retries

Le nombre de fois que vous faissez un Request jusqu'à obtenir un résultat qui réussit. Examples :

- Retries = 2 et Requests = 1 : Si le premier essai échouait, il re-tenterait une nouvelle fois. Si à la seconde tentative, le test fonctionne, il serait considéré comme valide.
- Retries = 1 et Requests = 2 : Il réaliserait deux tests mais si l'un des deux échouait, l'essai serait considéré comme un échec.

En utilisant l'authentification HTTP Simple

Quelques pages peuvent requérir une simple authentification simple HTTP. Géneralement elle est utilisée comme une rapide « salutation » de securité qui permet d'accéder à des tests de sécurité plus avancées (cryptage, persistence de données, etc.).



Elle peut être configurée dans les options avancées du test (ou directement dans la définition de la tâche WEB) avec les token de configuration suivant :

Check type

Type de check avec le serveur HTTP.

http auth (login)

Utilisateur http.

http auth (password)

Mot de passe d'utilisateur *http*.

Proxy auth realm

Nom de la zone (*realm*) d'authentication.

Proxy auth (server)

Domaine web et son port.

Proxy URL

Adresse URL du proxy.

Proxy auth (login)

Utilisateur de connexion du *proxy*.

Proxy auth (pass)

Mot de pass de connexion au *proxy*.

Un exemple terminé serait le suivant :

```
task_begin
get http://artica.es/pandoraupdate4/ui/
cookie 1
resource 1
check_string Pandora FMS Update Manager \(4.0\)
http_auth_serverport artica.es:80
http_auth_realm Private area
http_auth_user admin
http_auth_pass xxxx
task_end
```

L'utilisation de guillemets dans le mot de passe pour `http_auth_pass` n'est pas prise en charge. Évitez d'utiliser des guillemets simples `'` .

Supervision de web services

Avec Pandora FMS et Goliat, il est possible de surveiller API's REST. Ce n'est pas le cas pour les API's plus complexes basées sur des protocoles comme SOAP o XMLRPC.

Par exemple, supposons que nous souhaitions vérifier une API avec cet appel qui renvoie un nombre entier (de 0 à l'infini) dans le cas où elle ne fonctionne pas (utilisant les identifiants `my_user` et `my_pass`) :

```
task_begin
get
```

```
http://artica.es/integria/include/api.php?user=slerena&pass=xxxx&op=get_stats¶ms=opened,,1
check_string \n[0-9]+
task_end
```

Ceci renvoie une réponse du type :

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Connection: close
Date: Mon, 13 May 2013 15:39:27 GMT
Pragma: no-cache
Server: Apache
Vary: Accept-Encoding
Content-Type: text/html
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Client-Date: Mon, 13 May 2013 15:39:27 GMT
Client-Peer: 64.90.57.215:80
Client-Response-Num: 1
Client-Transfer-Encoding: chunked
Set-Cookie: a81d4c5e530ad73e256b7729246d3d2c=pcasWqI6pZzT2x2AuWo602; path=/

0
```

Grâce à la vérification de la sortie avec une expression régulière, vous pouvez vérifier que tout est ok. Pour des réponses plus complexes, vous devrez utiliser d'autres expressions régulières.

Quelques exemples de plus :

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

Dans le cas du module créé pour montrer les données doit être établi au type Remote HTTP module to retrieve string data (web_content_string).

```
task_begin
get https://pokeapi.co/api/v2/pokemon/ditto/
get_content imposter
task_end
```

Pareille au module précédent le type de données doit être « Remote HTTP module to retrieve string data (web_content_string) » pour fonctionner correctement.

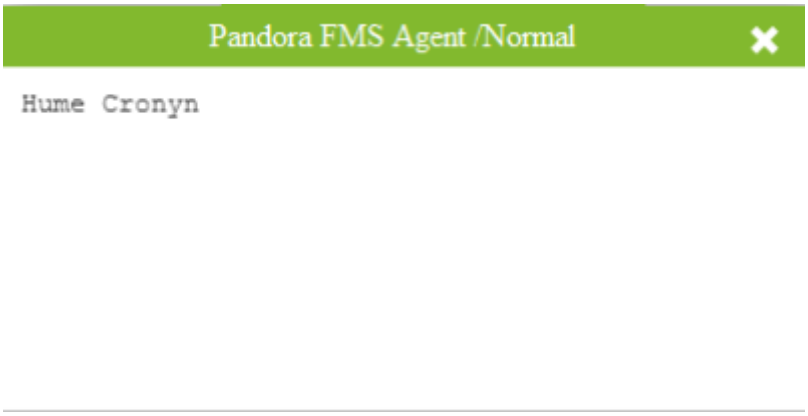
Vous pouvez faire des appels avec get_content_advanced :

```
task_begin
get https://api.hillbillysoftware.com/Awards/ByYear/1990
```

```
get_content_advanced "Nominee":"([A-Za-z ]+)","Year":"1990"
task_end
```

Résultat :

Pour être réflété sur Pandora FMS comme :



Il est important de définir correctement les groupes de capture entre parenthèses pour que l'appel soit correcte.

lorsque vous faissez des appels à l'API, il est important de garder sur l'esprit que l'API de destination doit avoir des permis pour être consultée.

Supervision https

Goliat peut vérifier aussi bien en http qu'en https. Pour pouvoir faire des vérifications sur le Web sécurisé (https), il suffit d'indiquer ce protocole dans l'URL. Par exemple :

```
task_begin
get
https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&
continue=https%3A%2F%2Fmail.google.com%2Fmail%2F%3Fui%3Dhtml%26zy%3Dl&bsv=zpwhty
gjntrz&ss=1&scc=1<mpl=default<mplcache=2
cookie 1
resource 0
check_string Google
task_end
```

Options avancées

En modifiant les en-têtes HTTP

Avec l'option *header*, les champs de l'en-tête HTTP peuvent être modifiés ou personnalisés. Par exemple, pour changer le champ *Host* de l'en-tête HTTP :

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

Débogant les tests web

Les tests web peuvent être débogués en ajoutant l'option `debug <log_file>`. Deux fichiers seront créés `log_file.req` et `log_file.res` avec, respectivement, les contenus de la requête HTTP et la réponse. Par exemple :

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

En utilisant Curl au lieu de LWP

La librairie LWP peut causer des problèmes lorsque beaucoup de threads mènent des requêtes HTTPS (en raison d'une limitation de OpenSSL). L'alternative est d'utiliser l'outil curl. Pour résoudre ce problème, éditez le fichier `/etc/pandora/pandora_server.conf` et ajoutez la ligne suivante :

```
web_engine curl
```

Relancez le serveur de Pandora FMS et le binaire de Curl sera utilisé pour mener les vérifications web au lieu de LWP.

Supervision transactionnelle avancée

En outre la fonctionnalité offert par Goliat, il y a d'autres manières d'effectuer une supervision transactionnelle web.

- De manière distribuée (UX), déployée en mode « agent » dans des systèmes différents á celui du serveur, même en réseaux non accessibles.
- De manière centralisée (WUX).

Retour à l'index de documentation du Pandora FMS