Auth Proxy Authentication

 › [Administration](https://grafana.com/docs/grafana/latest/administration/) › [Authentication](https://grafana.com/docs/grafana/latest/auth/) › Auth Proxy

You can configure Grafana to let a HTTP reverse proxy handling authentication. Popular web servers have a very extensive list of pluggable authentication modules, and any of them can be used with the AuthProxy feature. Below we detail the configuration options for auth proxy.

你可以配置Grafana，然后让一个HTTP反身代理来处理认证。流行的网页服务器都有非常大量的可播入式的认证模块，它们中的任何一个都可以用于AuthProxy功能。下面我们详细的介绍认证代理的配置选项。

[auth.proxy]

# Defaults to false, but set to true to enable this feature

# 默认为false，设置为true可以启用此功能

enabled = true

# HTTP Header name that will contain the username or email

# 包含了用户名或邮件的HTTP头部的名字

header\_name = X-WEBAUTH-USER

# HTTP Header property, defaults to `username` but can also be `email`

# HTTP头部属性，默认为`username`（用户名）也可以是`email`（邮箱）

header\_property = username

# Set to `true` to enable auto sign up of users who do not exist in Grafana DB. Defaults to `true`.

# 设置为true可以启用此功能，自动注册不在Grafana数据库中的用户。默认为true

auto\_sign\_up = true

# Define cache time to live in minutes

# 定义缓存存活的分钟数

# If combined with Grafana LDAP integration it is also the sync interval

# 如果与Grafana LDAP集成在一起，那么这个值也是同步的间隔。

sync\_ttl = 60

# Limit where auth proxy requests come from by configuring a list of IP addresses.

# 通过配置一个IP地址列表，限制代理请求从哪里来

# This can be used to prevent users spoofing the X-WEBAUTH-USER header.

# 这个功能可以用来防止用户欺骗X-WEBAUTH-USER的头部

# Example `whitelist = 192.168.1.1, 192.168.1.0/24, 2001::23, 2001::0/120`

whitelist =

# Optionally define more headers to sync other user attributes

# 可选的定义更多的头部去同步其他的属性

# Example `headers = Name:X-WEBAUTH-NAME Email:X-WEBAUTH-EMAIL Groups:X-WEBAUTH-GROUPS`

headers =

# Check out docs on this for more details on the below setting

# 使用下面的设置，为了更多详情检查这个文档。

enable\_login\_token = false

Bash

Interacting with Grafana’s AuthProxy via curl

使用curl与Grafana的授权代理进行交互

curl -H "X-WEBAUTH-USER: admin" http://localhost:3000/api/users

[

{

"id":1,

"name":"",

"login":"admin",

"email":"admin@localhost",

"isAdmin":true

}

]

Bash

We can then send a second request to the /api/user method which will return the details of the logged in user. We will use this request to show how Grafana automatically adds the new user we specify to the system. Here we create a new user called “anthony”.

我们可以发送一个请求到/api/usr方法，然后返回一个登录用户的详情。我们可以使用这个请求去展示，Grafana如何自动添加一个我们指定的用户到系统中。下面我们创建一个叫”anthony”的用户。

curl -H "X-WEBAUTH-USER: anthony" http://localhost:3000/api/user

{

"email":"anthony",

"name":"",

"login":"anthony",

"theme":"",

"orgId":1,

"isGrafanaAdmin":false

}

Bash

Making Apache’s auth work together with Grafana’s AuthProxy

让Apache的授权与Grafana的授权代理一起工作

I’ll demonstrate how to use Apache for authenticating users. In this example we use BasicAuth with Apache’s text file based authentication handler, i.e. htpasswd files. However, any available Apache authentication capabilities could be used.

我们会演示如何使用Apache来验证用户。在这个例子中，我们使用了Apache基于认证处理的基础授权文本文件，例如：htpasswd文件。然而，可以使用任何有能力的Apache认证。

Apache BasicAuth

In this example we use Apache as a reverse proxy in front of Grafana. Apache handles the Authentication of users before forwarding requests to the Grafana backend service.

在这个例子中，我们傅季Apache做为一个Grafana前端的反身代理。在请求传递到Grafana的后端服务之前，Apache会先处理用户的认证。

Apache configuration

<VirtualHost \*:8**0**>

ServerAdmin webmaster@authproxy

ServerName authproxy

ErrorLog "logs/authproxy-error\_log"

CustomLog "logs/authproxy-access\_log" common

<Proxy \*>

AuthType Basic

AuthName GrafanaAuthProxy

AuthBasicProvider file

AuthUserFile /etc/apache2/grafana\_htpasswd

Require valid-user

RewriteEngine On

RewriteRule .\* - [E=PROXY\_USER:%{LA-U:REMOTE\_USER},NS]

RequestHeader set X-WEBAUTH-USER "%{PROXY\_USER}e"

</Proxy>

RequestHeader unset Authorization

ProxyRequests Off

ProxyPass / http://localhost:3000/

ProxyPassReverse / http://localhost:3000/

</VirtualHost>

Bash

* The first 4 lines of the virtualhost configuration are standard, so we won’t go into detail on what they do.

virtualhost配置的前四行是标准的，所以我们无法详细说明他们是做什么的。

* We use a **<proxy>** configuration block for applying our authentication rules to every proxied request. These rules include requiring basic authentication where user:password credentials are stored in the **/etc/apache2/grafana\_htpasswd** file. This file can be created with the htpasswd command.

我们使用了<proxy>配置块对每一个代理请求使用认证规则。这些规则包括了存储在/etc/apache2/grafana\_htpasswd文件中的用户：密码的基础证书。这个文件可以使用htpasswd命令来创建。

* + The next part of the configuration is the tricky part. We use Apache’s rewrite engine to create our **X-WEBAUTH-USER header**, populated with the authenticated user.

配置接下来的部分是棘手的部分。我们使用Apache重写引擎去创建我们带着已认证用户的X-WEBAUTH\_USER头部。

* + - *RewriteRule . - [E=PROXY\_USER:%{LA-U:REMOTE\_USER}, NS]*\*: This line is a little bit of magic. What it does, is for every request use the rewriteEngines look-ahead (LA-U) feature to determine what the REMOTE\_USER variable would be set to after processing the request. Then assign the result to the variable PROXY\_USER. This is necessary as the REMOTE\_USER variable is not available to the RequestHeader function.

*重写规则——[E=PROXY\_USER:%{LA-U:REMOTE\_USER},NS]\*:这一行很神奇，对于每个请求，使用rewriteEngines先行(LA-U)特性来确定在处理请求之后将REMOTE\_USER变量设置成什么。然后指定PROXY\_USER变量的结果。REMOTE\_USER变量不能在RequestHeader方法中使用是有必要的。*

* + - **RequestHeader set X-WEBAUTH-USER “%{PROXY\_USER}e”**: With the authenticated username now stored in the PROXY\_USER variable, we create a new HTTP request header that will be sent to our backend Grafana containing the username.

**RequestHeader set X-WEBAUTH-USER “%{PROXY\_USER}e”**:已认证的用户名现在存储在PROXY\_USER变量中，我们创建一个新的包含着用户名的信息到HTTP请求头部，会发送到Grafana后端。

* The **RequestHeader unset Authorization** removes the Authorization header from the HTTP request before it is forwarded to Grafana. This ensures that Grafana does not try to authenticate the user using these credentials (BasicAuth is a supported authentication handler in Grafana).

**RequestHeader unset Authorization** 在它到达Grafana之前从HTTP请求中移除了认证头部。这样确保了Grafana不会使用它们的证书来尝试认证（在Grafana中支持BasicAuth处理）

* The last 3 lines are then just standard reverse proxy configuration to direct all authenticated requests to our Grafana server running on port 3000.

最后三行是标准的代理配置，去指导所有已认证的请求到我们正在运行的Grafana的3000端口上。

Full walk through using Docker.

使用Docker全过程

For this example, we use the official Grafana docker image available at [Docker Hub](https://hub.docker.com/r/grafana/grafana/)

对于这个例子，我们使用了在DockerHub中的正式的Grafana Docker镜像。

* Create a file grafana.ini with the following contents

使用下面的内容来创建一个grafana.ini文件

[users]

allow\_sign\_up = false

auto\_assign\_org = true

auto\_assign\_org\_role = Editor

[auth.proxy]

enabled = true

header\_name = X-WEBAUTH-USER

header\_property = username

auto\_sign\_up = true

Bash

Launch the Grafana container, using our custom grafana.ini to replace /etc/grafana/grafana.ini. We don’t expose any ports for this container as it will only be connected to by our Apache container.

使用我们自定义的grafana.ini去替代/etc/grafana/grafana.ini，然后运行Grafana容器。我们不会为这个容器暴露任何的端口，因为它只会被我们的Apache容器连接。

docker run -i -v $(pwd)/grafana.ini:/etc/grafana/grafana.ini --name grafana grafana/grafana

Bash

Apache Container

For this example we use the official Apache docker image available at [Docker Hub](https://hub.docker.com/_/httpd/)

对于这个例子，我们使用了Docker Hub中的正式版的Apache docker镜像。

* Create a file httpd.conf with the following contents

使用下面的内容来创建httpd.conf文件

ServerRoot "/usr/local/apache2"

Listen 80

LoadModule mpm\_event\_module modules/mod\_mpm\_event.so

LoadModule authn\_file\_module modules/mod\_authn\_file.so

LoadModule authn\_core\_module modules/mod\_authn\_core.so

LoadModule authz\_host\_module modules/mod\_authz\_host.so

LoadModule authz\_user\_module modules/mod\_authz\_user.so

LoadModule authz\_core\_module modules/mod\_authz\_core.so

LoadModule auth\_basic\_module modules/mod\_auth\_basic.so

LoadModule log\_config\_module modules/mod\_log\_config.so

LoadModule env\_module modules/mod\_env.so

LoadModule headers\_module modules/mod\_headers.so

LoadModule unixd\_module modules/mod\_unixd.so

LoadModule rewrite\_module modules/mod\_rewrite.so

LoadModule proxy\_module modules/mod\_proxy.so

LoadModule proxy\_http\_module modules/mod\_proxy\_http.so

<IfModule unixd\_module>

User daemon

Group daemon

</IfModule>

ServerAdmin you@example.com

<Directory />

AllowOverride none

Require all denied

</Directory>

DocumentRoot "/usr/local/apache2/htdocs"

ErrorLog /proc/self/fd/2

LogLevel error

<IfModule log\_config\_module>

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

LogFormat "%h %l %u %t \"%r\" %>s %b" common

<IfModule logio\_module>

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio

</IfModule>

CustomLog /proc/self/fd/1 common

</IfModule>

<Proxy \*>

AuthType Basic

AuthName GrafanaAuthProxy

AuthBasicProvider file

AuthUserFile /tmp/htpasswd

Require valid-user

RewriteEngine On

RewriteRule .\* - [E=PROXY\_USER:%{LA-U:REMOTE\_USER},NS]

RequestHeader set X-WEBAUTH-USER "%{PROXY\_USER}e"

</Proxy>

RequestHeader unset Authorization

ProxyRequests Off

ProxyPass / http://grafana:3000/

ProxyPassReverse / http://grafana:3000/

Bash

* Create a htpasswd file. We create a new user **anthony** with the password **password**

**创建一个htpasswd文件。我们会使用这个password的密码来创建一个新的用户anthony**

* htpasswd -bc htpasswd anthony password

Bash

* Launch the httpd container using our custom httpd.conf and our htpasswd file. The container will listen on port 80, and we create a link to the **grafana** container so that this container can resolve the hostname **grafana** to the grafana container’s ip address.

使用我们自定义的httpd.conf和htpasswd文件来运行httpd容器。这个容器监听80端口，我们可以创建一个连接到grafana容器，以便这个容器可以解析主机名为grafana的容器的IP地址。

* docker run -i -p 80:80 --link grafana:grafana -v $(pwd)/httpd.conf:/usr/local/apache2/conf/httpd.conf -v $(pwd)/htpasswd:/tmp/htpasswd httpd:2.4

Bash

Use grafana.

使用Grafana

With our Grafana and Apache containers running, you can now connect to http://localhost/ and log in using the username/password we created in the htpasswd file.

随着我们Grafana和Apache容器的运行，你现在可以连接到<http://localhost/>,并使用我们在htpasswd文件中创建的用户密码来登录。

Team Sync (Enterprise only)

团队同步（企业版可用）

Only available in Grafana Enterprise v6.3+

只在Grafana企业版6.3版本以后可用。

With Team Sync, it’s possible to set up synchronization between teams in your authentication provider and Grafana. You can send Grafana values as part of an HTTP header and have Grafana map them to your team structure. This allows you to put users into specific teams automatically.

用团队同步，需要在团队认证供应者和Grafana之间建立同步。你可以把Grafana值作为HTTP头部的一部分，Grafana会映射他们到你的团队结构中。这允许你自动把用户放入指定的团队。

To support the feature, auth proxy allows optional headers to map additional user attributes. The specific attribute to support team sync is Groups.

为支持这个功能，认证代理允许可选的头部去映射额外的用户参数。这个支持团队同步的指定参数是Groups。

# Optionally define more headers to sync other user attributes

# 可选的定义头部去同步其他用户的参数

headers = "Groups:X-WEBAUTH-GROUPS"

Bash

You use the X-WEBAUTH-GROUPS header to send the team information for each user. Specifically, the set of Grafana’s group IDs that the user belongs to.

为每个用户，你可以使用X-WEBAUTH-GROUPS头部去发送团队信息。特别的，用户所属的Grafana组id的集合。

First, we need to set up the mapping between your authentication provider and Grafana. Follow [these instructions](https://grafana.com/docs/grafana/latest/auth/team-sync/#enable-synchronization-for-a-team) to add groups to a team within Grafana.

首先，我们需要去安装你的认证供应者和Grafana之间的映射。通过这些介绍来在Grafana中将组添加到团队中。

Once that’s done. You can verify your mappings by querying the API.

一旦完成这些。你可以通过查询API来验证你的映射。

# First, inspect your teams and obtain the corresponding ID of the team we want to inspect the groups for.

# 首先，检查你的团队，并获取我们想要去检查的组的相应的团队ID。

curl -H "X-WEBAUTH-USER: admin" http://localhost:3000/api/teams/search

{

"totalCount": 2,

"teams": [

{

"id": 1,

"orgId": 1,

"name": "Core",

"email": "core@grafana.com",

"avatarUrl": "/avatar/327a5353552d2dc3966e2e646908f540",

"memberCount": 1,

"permission": 0

},

{

"id": 2,

"orgId": 1,

"name": "Loki",

"email": "loki@grafana.com",

"avatarUrl": "/avatar/102f937d5344d33fdb37b65d430f36ef",

"memberCount": 0,

"permission": 0

}

],

"page": 1,

"perPage": 1000

}

# Then, query the groups for that particular team. In our case, the Loki team which has an ID of "2".

# 然后，查询特定团队的组。在我们的例子里，Loki组的ID是2

curl -H "X-WEBAUTH-USER: admin" http://localhost:3000/api/teams/2/groups

[

{

"orgId": 1,

"teamId": 2,

"groupId": "lokiTeamOnExternalSystem"

}

]

Bash

Finally, whenever Grafana receives a request with a header of X-WEBAUTH-GROUPS: lokiTeamOnExternalSystem, the user under authentication will be placed into the specified team. Placement in multiple teams is supported by using comma-separated values e.g. lokiTeamOnExternalSystem,CoreTeamOnExternalSystem.

最终的，无论什么时候Grafana接收到带有X-WEBAUTH-GROUPS: lokiTeamOnExternalSystem的头部，处于认证的用户会被放在指定的团队中。使用逗号分隔值来使用多个团队定点，如lokiTeamOnExternalSystem,CoreTeamOnExternalSystem

curl -H "X-WEBAUTH-USER: leonard" -H "X-WEBAUTH-GROUPS: lokiteamOnExternalSystem" http://localhost:3000/dashboards/home

{

"meta": {

"isHome": true,

"canSave": false,

...

}

Bash

With this, the user leonard will be automatically placed into the Loki team as part of Grafana authentication.

通过这种操作，用户leonard会自动地被放入Loki团队，作为Grafana认证的一部分。

[Learn more about Team Sync](https://grafana.com/docs/grafana/latest/auth/team-sync/)

Login token and session cookie

登录令牌和session cookie

With enable\_login\_token set to true Grafana will, after successful auth proxy header validation, assign the user a login token and cookie. You only have to configure your auth proxy to provide headers for the /login route. Requests via other routes will be authenticated using the cookie.

当enable\_login\_token被设置为true，Grafana会在认证代理成功后，分配用户一个登录令牌和cookie。为了/login路由，你必须配置你的认证代理到供应者头部。请求其他的路由会自动使用cookie。

Use settings login\_maximum\_inactive\_lifetime\_days and login\_maximum\_lifetime\_days under [auth] to control session lifetime. [Read more about login tokens](https://grafana.com/docs/grafana/latest/auth/overview/#login-and-short-lived-tokens)

使用[auth]下的login\_maximum\_inactive\_lifetime\_days和login\_maximum\_lifetime\_days来控制session的生存期。