repoze.who-x509 Documentation

Release 0.2.0

Arturo Sevilla

December 05, 2016

Contents

Author Arturo Sevilla.

Latest release 0.2.0

Overview

This plugin enables `repoze.who` to identify (not completely authenticate) according to SSL client certificates.
It can check the fields (attribute types) in the subject distinguished name.
It supports "out of the box" `mod_ssl` if `mod_wsgi` is also activated in Apache, and Nginx SSL functionality.
However, this documentation also includes configuration examples for both Apache and Nginx for when both are working as reverse proxies.
This plugin was developed independently of the repoze project (copyrighted to Agendaless Consulting, Inc.).

CHAPTER 1

Installing this plugin

The minimum requirements for installation are `repoze.who`, and `python-dateutil`. If you want to run the tests, then Nose and its coverage plugin will also be installed. It can be installed with `easy_install`:

```
easy_install repoze.who-x509
```

CHAPTER 2

Support and development

The project is hosted on GitHub.

CHAPTER 3

Quick setup

If you want to use the `IIdentifier` object, then you can build it as follows, and the pass it to the `identifiers` parameter of `repoze.who.middleware.PluggableAuthenticationMiddleware`:

```
from repoze.who.plugins.x509 import X509Identifier

identifer = X509Identifier('SSL_CLIENT_S_DN')
```

The required parameter of `X509Identifier` is the WSGI environment key of the "distinguished name" of the client certificate subject. By default the credentials are based on the "Email" field, but it can be customized as follows:

```
from repoze.who.plugis.x509 import X509Identifier

identifier = X509Identifier('SSL_CLIENT_S_DN', login_field='CN')
```

In this case it will try to get the credentials from the common name of the client certificate subject.

CHAPTER 4

Contents

4.1 repoze.who.plugins.x509 releases

4.1.1 repoze.who.plugins.x509 0.2.0 (2011-03-22)

- Split between the r.what plugin

4.2 Configuration

In order to get this plugin to work, the web server must enabled in its configuration to accept and verify client certificates. This module requires that at least the subject "distinguished name" is present in the WSGI environment dictionary, though it prefers similar `mod_ssl` variables (sharing the same prefix and only different that the proper suffix is an underscore and the name of the attribute type). We chose to prefer this over the unparsed distinguised name to avoid making a double calculation (parsing done by both `mod_ssl` and our Python code), and to avoid possible bugs.

4.2.1 Apache

With mod_wsgi

You will need to enable `mod_wsgi` and `mod_ssl`:

```
$ a2enmod wsgi
$ a2enmod ssl
```

Note: This document will not cover how to configure Apache's mod_wsgi.

Then will need to modify your configuration file to include the following directives to your site:

```
<VirtualHost yoursite:443>

    # your directives here

    # to turn on the mod_ssl engine
    SSLEngine On
    SSLCertificateFile /path/to/your/server/certificate.crt
```

```
    SSLCertificateKeyFile /path/to/your/server/key.key

    # this CA will check your client certificate
    SSLCACertificateFile /path/to/the/ca/certificate.crt

    # this will turn on client certification verification
    SSLVerifyClient require

    # This depth will allow us to check for self signed certificates and
    # with our CA already specified
    SSLVerifyDepth 2

</VirtualHost>
```

As reverse proxy

The configuration is very similar, but in this case you want to reissue the request to a backend or application server. First, enable the modules:

```
$ a2enmod ssl
$ a2enmod proxy
$ a2enmod proxy_http
```

The configuration file will have the same directives as in *With mod_wsgi* but will include the necessary ones for proxying the request:

```
<VirtualHost yoursite:443>

    # your directives here

    # to turn on the mod_ssl engine
    SSLEngine On
    SSLCertificateFile /path/to/your/server/certificate.crt
    SSLCertificateKeyFile /path/to/your/server/key.key

    # this CA will check your client certificate
    SSLCACertificateFile /path/to/the/ca/certificate.crt

    # this will turn on client certification verification
    SSLVerifyClient require

    # This depth will allow us to check for self signed certificates and
    # with our CA already specified
    SSLVerifyDepth 2

    # Enable the reverse proxy
    ProxyPass / http://yourbackendserver/ retry=5
    ProxyPassReverse / http://yourbackendserver/
    ProxyPreserveHost On

    <Proxy *>
        Order deny,allow
        Allow from all
    </Proxy>

    # In order to prevent HTTP header spoofing set this to empty strings,
    # and then reset them to their correct value.
```

```
    RequestHeader set SSL_CLIENT_S_DN ""
    RequestHeader set SSL_CLIENT_I_DN ""
    RequestHeader set SSL_CLIENT_S_DN_O ""
    RequestHeader set SSL_CLIENT_S_DN_OU ""
    RequestHeader set SSL_CLIENT_S_DN_CN ""
    RequestHeader set SSL_CLIENT_S_DN_C ""
    RequestHeader set SSL_CLIENT_S_DN_ST ""
    RequestHeader set SSL_CLIENT_S_DN_L ""
    RequestHeader set SSL_CLIENT_S_DN_Email ""
    RequestHeader set SSL_CLIENT_I_DN_O ""
    RequestHeader set SSL_CLIENT_I_DN_OU ""
    RequestHeader set SSL_CLIENT_I_DN_CN ""
    RequestHeader set SSL_CLIENT_I_DN_C ""
    RequestHeader set SSL_CLIENT_I_DN_ST ""
    RequestHeader set SSL_CLIENT_I_DN_L ""
    RequestHeader set SSL_CLIENT_I_DN_Email ""
    RequestHeader set SSL_SERVER_S_DN_OU ""
    RequestHeader set SSL_CLIENT_VERIFY ""

    <Location />
        RequestHeader set SSL_CLIENT_S_DN "%{SSL_CLIENT_S_DN}s"
        RequestHeader set SSL_CLIENT_I_DN "%{SSL_CLIENT_I_DN}s"
        RequestHeader set SSL_CLIENT_S_DN_O "%{SSL_CLIENT_S_DN_O}s"
        RequestHeader set SSL_CLIENT_S_DN_OU "%{SSL_CLIENT_S_DN_OU}s"
        RequestHeader set SSL_CLIENT_S_DN_CN "%{SSL_CLIENT_S_DN_CN}s"
        RequestHeader set SSL_CLIENT_S_DN_C "%{SSL_CLIENT_S_DN_C}s"
        RequestHeader set SSL_CLIENT_S_DN_ST "%{SSL_CLIENT_S_DN_ST}s"
        RequestHeader set SSL_CLIENT_S_DN_L "%{SSL_CLIENT_S_DN_L}s"
        RequestHeader set SSL_CLIENT_S_DN_Email "%{SSL_CLIENT_S_DN_Email}s"
        RequestHeader set SSL_CLIENT_I_DN_O "%{SSL_CLIENT_I_DN_O}s"
        RequestHeader set SSL_CLIENT_I_DN_OU "%{SSL_CLIENT_I_DN_OU}s"
        RequestHeader set SSL_CLIENT_I_DN_CN "%{SSL_CLIENT_I_DN_CN}s"
        RequestHeader set SSL_CLIENT_I_DN_C "%{SSL_CLIENT_I_DN_C}s"
        RequestHeader set SSL_CLIENT_I_DN_ST "%{SSL_CLIENT_I_DN_ST}s"
        RequestHeader set SSL_CLIENT_I_DN_L "%{SSL_CLIENT_I_DN_L}s"
        RequestHeader set SSL_CLIENT_I_DN_Email "%{SSL_CLIENT_I_DN_Email}s"
        RequestHeader set SSL_SERVER_S_DN_OU "%{SSL_SERVER_S_DN_OU}s"
        RequestHeader set SSL_CLIENT_VERIFY "%{SSL_CLIENT_VERIFY}s"
    </Location>

</VirtualHost>
```

Headers modification

However, in your backend server the WSGI environment variables will not be named with the default `mod_ssl` key, instead they will be prefixed by HTTP_ (after all they are passed as custom HTTP headers). For example `SSL_CLIENT_S_DN` will become `HTTP_SSL_CLIENT_S_DN`, so you will have to be careful when using the identifier:

```
from repoze.who.plugins.x509 import X509Identifier

# We need to specify the full string for the key.
identifer = X509Identifier('HTTP_SSL_CLIENT_S_DN')
```

4.2.2 Nginx

Note: Nginx does not parse the distinguished name of neither the subject or the issuer in to separate fields, so *repoze.who.plugins.x509* tries its best to parse from the given DN fields.

Warning: This module hasn't been tested with nginx's mod_wsgi.

As reverse proxy

You just to need to specify the following configuration in a readable Nginx configuration file:

```
server {
    # enable ssl engine
    listen 443 default ssl;
    # specify our server certificates
    ssl_certificate /path/to/your/server/certificate.crt;
    ssl_certificate_key /path/to/your/server/key.key;

    # enables client certification validation
    ssl_verify_client on;

    # this depth allows us to check self signed certificates and with the CA
    # that we will specify.
    ssl_verify_depth 2;

    # this CA will enable us to check or "authenticate" our client certificate.
    ssl_client_certificate /path/to/your/ca/certificate.crt;
    ssl_protocols SSLv3 TLSv1;
    location / {
        proxy_pass http://yourbackendserver;
        proxy_set_header Host $host;

        # pass the distinguished name fields
        proxy_set_header SSL_CLIENT_I_DN $ssl_client_i_dn;
        proxy_set_header SSL_CLIENT_S_DN $ssl_client_s_dn;
        proxy_set_header SSL_CLIENT_VERIFY $ssl_client_verify;
    }
}
```

As with Apache's configuration, your headers will not be as specified, but prefixed with HTTP_, and you will need to specify your identifier key accordingly. See *Headers modification* for an example of this configuration.

4.3 API

4.3.1 x509

4.3.2 utils

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

r

R