repoze.what-x509 Documentation

Release 0.3.0

Arturo Sevilla

March 22, 2012

CONTENTS

Author Arturo Sevilla.

Latest release 0.3.0

Overview

This plugin enables `repoze.what` to check authorization according to SSL client certificates. It can check the fields (attribute types) in either the subject or issuer 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

ONE

INSTALLING THIS PLUGIN

The minimum requirements for installation are `repoze.what`, `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.what-x509
```

CHAPTER

TWO

SUPPORT AND DEVELOPMENT

The project is hosted on GitHub.

CHAPTER

THREE

QUICK SETUP

In order to protect a resource you must create the corresponding predicate according to what conditions you need to fulfill.

There are two base predicate classes: `X509Predicate` and `X509DNPredicate`, however you will mostly be using the two derived predicates:

- `is_issuer`: This predicate enables you to establish conditions and authorize based on the issuer of the certificate.
- `is_subject`: This predicate enables you to establish conditions and authorize based on the subject of the certificate.

The issuer and the subject are SSL terms corresponding who issued the certificate, and to whom.

For example, if you want to protect a resource when the issuer of the certificate is "XYZ Company", then you create it as follows:

```
from repoze.what.plugins.x509 import is_issuer

predicate = is_issuer(organization='XYZ Company')
```

If you want to allow access only to the user named "John Smith" then you create the predicate as follows:

```
from repoze.what.plugins.x509 import is_subject

predicate = is_subject(common_name='John Smith')
```

Then you can evaluate these predicates according to your system, for example if you are using pylons and the `repoze.what.plugins.pylonshq` plugin then you could use `ActionProtector` or `ControllerProtector` with the created predicates.

You will need to setup Apache or Nginx (or any other server) to work with SSL client certificates. See *Configuration* for examples.

CHAPTER

FOUR

CONTENTS

4.1 repoze.what.plugins.x509 releases

4.1.1 repoze.what.plugins.x509 0.3.0 (2011-03-22)

- Created a r.who plugin and moved the identifier and utils to such repository. This project now contains only the predicates.
- Reflected the change on the docs.

4.1.2 repoze.what.plugins.x509 0.2.0 (2011-03-18)

- Bumped to beta stage.
- Fixed some subtle bugs about order of initialization.
- Finished documentation.

4.1.3 repoze.what.plugins.x509 0.1.3 (2011-03-15)

- First "stable" API of the module, but still considered alpha.
- Full unit test coverage
- First documentation stub

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 or the issuer "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 predicates of `repoze.what.plugins.x509`:

```
from repoze.what.plugins.x509 import is_subject

# we use the subject_key parameter to indicate the key of this variable
# within our WSGI environment.
predicate = is_subject(country='US', subject_key='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.what.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 `subject_key` or `issuer_key` with the predicates. See Headers modification for an example of this configuration.

4.3 Advanced Use

You can customize `repoze.what.plugins.x509` so that it works for your web server. There is a simple customization example in *Headers modification*.

In order to make the best out of the functionality of this plugin, you need to know how it is that it reads the values from the WSGI environment, and the rules for evaluation.

4.3.1 Rules for parameter specification

- When you create any distinguished name based predicate (any subclass of `X509DNPredicate`), you can specify the fields that you need to check upon requests. The constructor accepts `common_name`, `organization`, `organization_unit`, `country`, `state`, or `locality`.
- The constructor can also accept any "custom" field that may be present in the distinguished name of the client certificate. You specify this fields by using the attribute type name as a keyword to the constructor. For example, if there is a field named "A", then you could construct a predicate as `is_subject(A='some value')`.
- Please note that according to the last rule, you may also specify the defined constructor parameters by their equivalent attribute type names, such as, "CN" for common_name, or "O" for organization. However if you specify both of a type, the value that the predicate will check is the one that is present with the defined constructor arguments. For example, `is_subject(organization='ABC', O='XYZ')` will check for an organization named "ABC", not "XYZ".

4.3.2 Rules for predicate evaluation

1. The predicate will first look for the "verified" key in our WSGI environment. By default it will try to locate it in `SSL_CLIENT_VERIFIED`, however you can change this behavior by specifying this key in the predicate constructor through the `verify_key` argument. If value is different than "SUCCESS", it will fail.
2. If the WSGI environment provides the validity time range of the certificate it will be checked. However, not all web servers set this variable in the headers. You can change the keys that the environment tries to check by setting `validity_start_key` and `validity_end_key`.
3. After the first two validations, all `X509DNPredicate` based predicates (`is_issuer` and `is_subject`) will check for server variables that tries to validate it. The keys for these variables will be constructed by appending the `environ_key` parameter (`subject_key` for `is_subject` and `issuer_key` for `is_issuer`) with its corresponding X.509 attribute type. For example, if `environ_key` is `SSL_CLIENT_S_DN`, and you try to check for an organization then the WSGI environment to check will be `SSL_CLIENT_S_DN_O`.
4. **There are various rules to determine if the predicate is valid:**
    - If the distinguished name has one value for an attribute type, then it must equal the value specified in the constructor argument.

- If the distinguished name has more than one value for the same attribute type, and the constructor argument for the predicate is a string (single value), then it will be valid if such argument equals at least one of the values of the distinguished name. The WSGI environment variables that will be checked will follow the same rules as point #3, but suffixed by an index number, for example `SSL_CLIENT_S_DN_O_0`. If there is no such variable, then it will follow the rules of point #5.
- If the distinguished name has more than one value for the same attribute type, and the constructor argument is a tuple or a list, then all of the values of such argument must be present in the distinguished name.

5. If any of the server variables that are tried are non-existent (with the exception of the validity range), then it will try to parse the distinguished name, for which the same rules to point #4 will be applied.
6. If there is an error in the parsing, then the predicate will fail.

4.3.3 API

predicates

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

r