Guide to Third-Party AutoSSL Provider Modules

Guide to Third-Party AutoSSL Provider Modules

- Module development work
- Authentication deployment workflow
- AutoSSL provider workflow
- Module function interfaces
- Example

Introduction

Warning:
Only advanced users should use this feature.

Note:
We added AutoSSL functionality to cPanel & WHM version 58, and custom AutoSSL provider modules in version 60.

AutoSSL provider modules allow your server's users to automatically secure locally-hosted domains on their accounts with certificates from that SSL certificate provider. We ship the cPanel (powered by Sectigo$^{®}$) provider module with cPanel & WHM, and you can download a plugin to add the Let's Encrypt™ provider module. This document explains how to create your own provider module.

Module development work

When you develop your provider module, we recommend the following workflow:

1. Research the supported parameters for your desired SSL certificate provider.
2. Configure a module that subclasses the `/usr/local/cpanel/Cpanel/SSL/Auto/Provider.pm` module with overrides that match the supported parameters for your certificate provider.

   Warning:
   We advise that you **do not** directly edit the `/usr/local/cpanel/Cpanel/SSL/Auto/Provider.pm` file.

Authentication deployment workflow

After you develop and configure your provider module, we recommend the following workflow to deploy the module:

1. Navigate to WHM's *Manage AutoSSL* interface *(WHM >> Home >> SSL >> Manage AutoSSL)*.
2. Select the provider module.
3. Test the provider module with an account on a non-production server.
4. Review the log files to confirm that an SSL certificate provided by the provider secures the account's domains.

AutoSSL provider workflow

Locations

AutoSSL provider modules reside in the following directories:

- The `/usr/local/cpanel/Cpanel/SSL/Auto/Provider/` directory — cPanel-provided modules.
- The `/var/cpanel/perl/Cpanel/SSL/Auto/Provider/` directory — Third-party modules.

For example, a module for the third-party ExampleSSL's module would reside in the `/var/cpanel/perl/Cpanel/SSL/Auto/Provider/ExampleSSL.pm` location.

AutoSSL

Related documentation

- Certificate Signing Requests - CSR
- Certificates - CRT
- Market Provider Manager
- Generate an SSL Certificate and Signing Request
- SSL Storage Manager

Module function interfaces

The tables below contain the required, recommended, and inherited methods.

Required

Warning:
You **must** configure the following methods in the `Cpanel::SSL::Auto::Provider` class. If you **do not** configure a required method, it will fail with a `Cpanel::Exception::NotImplemented` exception.

Method name	Description	Example

`renew_ssl_for_vhosts(USERNAME, VHOST1 => \@DOMAINS1, VHOST2 => DOMAINS2, ...)`	Key-value pairs that declare each virtual host and which domains within those virtual hosts to secure.
	
	Note:
	The `vhost#.name` key represents the Apache server's name. However, this key may change in a future version.
		`'vhost1.name' => \@list1_of_domains_including_www_subdomains,`
		
		`'vhost2.name' => \@list2_of_domains_including_www_subdomains`

You can provide the following optional methods in your module:

Method name	Description	Example
`DAYS_TO_REPLACE()`	This method declares when to begin the renewal process. If the certificate expires in this number of days or **fewer**, the system starts the renewal process. If you **do not** set this value, the system waits until the certificate expires before it attempts to replace it.	`return 15 ;`
`ON_START_CHECK()`	This method executes immediately after AutoSSL prints a provider name prints into the log file.	

You can override the following optional methods in your module:

Method name	Description	Example

`MAX_DOMAINS_PER_CERTIFICATE()`	The maximum number of domains to request per certificate. This depends on the Certificate Authority's (CA) domain limits. If you **do not** set this value, the system assumes that the CA does not limit the number of domains on a certificate.	`return 100 ;`
`PROPERTIES()`	This method returns a list of additional key-value pairs that define additional properties for the provider module. For example, `terms_of_service` defines the URL at which the API caller needs to accept in order to enable the module, which they do through the `terms_of_service_accepted` parameter.	
`EXPORT_PROPERTIES (`*`NAME1 => VALUE1, NAME2 => VALUE2, ...`*`)`	This method sends information to the external provider, such as registration data.	
`RESET()`	This method resets the server's registration with the remote provider.	
`CERTIFICATE_IS_FROM_HERE (` *`CERTIFICATE_PEM`* `)`	This method indicates whether the PEM-encoded certificate that you send to it comes from a valid AutoSSL provider and **not** a valid, non-AutoSSL provider. This method varies depending on the CA and the type of certificate that they issue. If you **do not** define this method, the system assumes that nothing comes from this module.	`return ( $parsed_certificate->{'issuer'}{'organizationalUnitName'} && $parsed_certificate->{'issuer'}{'organizationalUnitName'} eq $provider_name ) ? 1 : 0;`
`DISPLAY_NAME()`	This method defines the provider's name that the interface displays.	`return 'Bogus SSL Provider for Testing Purposes';`
`ON_ACCOUNT_RENAME (OLDNAME, NEWNAME)`	This method declares what to run when an administrator renames the account. The `OLDNAME` value represents the previous domain, while the `NEWNAME` value represents the new domain.	`return oldexample, newexample;`

`ON_ACCOUNT_TERMINATION (OLDNAME)`	This method declares what to run when the administrator terminates the account. The `OLDNAME` value represents the terminated account.	`return oldexample ]`
`ON_DOMAIN_REMOVAL (OLDNAME)`	This method declares what to run when a user or administrator removes a domain from the account. The `OLDNAME` value represents the username that you removed.	`return oldexample ;`

`get_dcv_errors(OPTIONS)`	This method performs Domain Control Validation (DCV) as part of the AutoSSL `vhost` processing. The options for this method are:
	• `username` — A username.
	• `domains` — A list of domains.
	• `dcv_method` — A hash whose keys are entries in the `domains` argument. Each hash value displays the local DCV methods (for example, `http`) that succeeded for the associated domain.
	
	With this method, AutoSSL will **not** pass domains to the `renew_ssl_for_vhosts` method that fail the provider's DCV. This can mitigate certain issues that arise if cPanel & WHM's local DCV succeeds but the provider or CA's DCV fails.
		`username => 'username',`
		`domains => [ 'example.com', 'www.example.com ],`
		`dcv_method => { 'example.com' => 'http', 'www.example.com' => 'http',`
		
		},

The following methods are inherited, and you should not override them:

Method name	Description	Example
`start_logging (USERNAME)`	This method starts the log for the declared user. If you **do not** set the `USERNAME` value, the system notes that this is an AutoSSL run for **all** users.	`'example'`

`resume_logging (START_TIME)`	This method appends to an existing log. The `START_TIME` value is an ISO 8601 time value. If a log does **not** exist for the `START_TIME` time value, the system throws an exception.	`'2016-05-09T05:34:12Z'`
`log (LEVEL, MESSAGE)`	This method enters the `MESSAGE` text in to the log file. The `LEVEL` value can be one of the following: • `success` • `info` • `warn` • `error`	`'success'`, `'I'm making a note here'`
`increase_log_indent_level()`	This method indents the entries in the log by one level.	
`decrease_log_indent_level()`	This method outdents the entries in the log by one level.	
`get_log_start_time()`	This method returns the time that this class instance started to log, in ISO 8601 time value.	
`keep_log_in_progress()`	When AutoSSL finishes a check run, it sets that run's log to completed. However, this method flags the log as in progress. This is useful when the module uses a separate queue to fetch the AutoSSL certificates, as the cPanel module does.	
`install_certificate (%OPTS)`	This method installs an SSL certificate for Exim, Apache, and Dovecot®. **Note:** In cPanel & WHM version 60, this method also installs an SSL certificate for `cpsrvd` and `cpdavd` modules. We may expand this method to install certificates for other services in future versions.	`'web_vhost_name' => $vhost, 'certificate_pem' => $res->{ 'cert' }, 'key_pem' => $key );`

You **must** pass the following required arguments through this method:

- `web_vhost_name` — The name of the virtual host on which to install the certificate. For more information about virtual host names, read our UAPI Functions - WebVhosts::list_domains documentation.
- `certificate_pem` — The PEM-encoded certificate.
- `key_pem` — The PEM-encoded key.

You can pass the following optional arguments:

- `cab_pem` — The PEM-encoded CA-bundle, with newlines separating each certificate.
- `installing_user` — The user for whom to install the certificate. This option attempts to install the certificate with the permission set of the user (instead of the `root` user). If the user does **not** possess the permission to install on the given virtual host, the system will display an exception. If you **do not** set this option, the system could install a certificate on the wrong user's account.

  Note: We added the `installing_user` option in cPanel & WHM version 68

	Important: We **strongly recommend** that you use the `install_certificate` method instead of an API function to install certificates. This method improves speed and will not restart Apache and Dovecot for each certificate installation.	

Example

The following AutoSSL module outline demonstrates a minimal set of functionality.

Warning:
This is **not** a fully-functional module. This only demonstrates basic workflow. Your implementation will require more internal logic. Also, this module does **not** demonstrate the necessary API calls that would allow your module to hook into your SSL certificate provider.

```
#Name your module properly to be a
submodule of the parent referenced
below:
package
Cpanel::SSL::Auto::Provider::BogusSSLPr
ovider;

use strict;
use warnings;

use parent qw(
Cpanel::SSL::Auto::Provider );

# I use CPAN modules here as much as
possible for clarity of examples, you
can write your own custom
parsers/requesters if you feel like it.
use HTTP::Tiny();

use JSON::MaybeXS();
use Crypt::X509();
use Crypt::OpenSSL::RSA();
use Crypt::OpenSSL::PKCS10();

# Set this value to whatever you think
```

```
is a reasonable for the domain to begin
requesting a new free certificate (as
you may queue the DCV check). This is
mostly to help ensure a seamless SSL
coverage experience for users of your
free certificates (instead of them
having coverage gaps waiting on DCV).
sub DAYS_TO_REPLACE { return 15; }

# Set this to whatever maximum you allow
within your signing infrastructure for
DV certificates. For example, Let's
Encrypt has a limit of 100 domains that
can be on any given CSR they'll sign.
sub MAX_DOMAINS_PER_CERTIFICATE { return
100; }

# Defines what your SSL Provider name
will look like in the cPanel & WHM UIs
and AutoSSL logs.
sub DISPLAY_NAME { return 'Bogus SSL
Provider for Testing Purposes'; }

# The logic in this subroutine needs to
accept an SSL certificate string (in PEM
format) and be able to tell us if that
certificate came from your provider.
# Retuns 1 if yes, 0 if no.
sub CERTIFICATE_IS_FROM_HERE {
    my ( $self, $cert_pem ) = @_;
 # To parse a PEM encoded certificate
file, you may want to use a module like
Crypt::X509 from CPAN. See
http://search.cpan.org/~ajung/Crypt-X50
9-0.51/lib/Crypt/X509.pm

    my $parsed_certificate =
Convert::X509->new($cert_pem);
    # It can be as simple as looking at
what organization signed the cert, but
whatever info you want to look at in the
Certificate is acceptable.
    # Similarly, you may want to check
that the *validity* period for your
certificate matches the product type of
your free certificate offering.
 # Convert::X509 has 'to' and 'from'
subroutines that would be helpful in
this regard.
    my $provider_name = "Internet Widget
Signing Organizaton, pty";
    return ( $parsed_certificate->issuer
=~ m/$provider_name/ ) ? 1 : 0;
```

```
}

# This optional method allows a provider
to do Domain Control Validation (DCV) as
part of the AutoSSL vhost processing.
When this method is in place correctly,
AutoSSL will forgo passing domains to
{{renew_ssl_for_vhosts()}} that fail the
provider’s DCV. This can mitigate
certain issues that arise if, for some
reason, cPanel & WHM’s local DCV
succeeds but the provider/CA’s DCV
fails.
sub get_dcv_errors {
    my ($self, %opts) = @_;
    my @dcv_errors;
    for my $domain ( @{
$opts{'domains'} } ) {
        my @these_dcv_errors = ...;
# “...” being whatever custom external
DCV logic is needed
        push @dcv_errors,
\@these_dcv_errors;
    }
    return \@dcv_errors;
}

# This function is where the magic
happens, as we actually make a request
here to your servers, and then
*do_something* with that.
# In this example, we're assuming that
the DCV happens *instantaneously* and
you are delivered a certificate in
return (as with the Let's Encrypt
provider).
# If your provider cannot do this, then
I would suggest you make a companion
script to this that references a queue
of some sort for installing your SSL
certs.

# Anyways, the autossl binary, when run,
will pass in the account and a hash
containing information on all vhosts and
domains contained therein.
# The function can return anything, but
should probably return undef, as nothing
checks the return value. If something
goes wrong here, we'd wan't you to throw
an exception/die.
sub renew_ssl_for_vhosts {
    my ( $self, $account_name,
%vh_domains ) = @_;

    # Generate the key for the cPanel
account. See
https://metacpan.org/pod/Crypt::OpenSSL
::RSA for more information.
 # /dev/random exists on all supported
platforms, so Crypt::OpenSSL::Random's
random_seed function and then importing
that seed should not be needed.
 my $key =
Crypt::OpenSSL::RSA->generate_key(2048);

    my ( $csr, $cert, $payload, $res );

 # Each vhost on the account will need a
separate CSR, as cPanel's Apache stack
is setup to only allow one certificate
per vhost.
    foreach my $vhost ( keys(
%vh_domains ) ) {
        # Create the CSR for the vhost
        $csr = _create_csr_for_vhost(
$key,  @{( $vh_domains{$vhost} )} );

  # Generate any additional data you may
want to send over to your HTTP cert
requesting endpoint.
  # In this example, I'm making an
assumption that you are going to POST
over some data along with your CSR, but
you can do whatever it is you need.
  _generate_dcv_files( $csr, @{(
$vh_domains{$vhost} )} );

        # Request the signed Cert.
        $payload = { 'validation_type'
=> 'dcv', 'csr' => $csr };
        $res =
HTTP::Tiny->new()->post_form(
'https://some.url.endpoint/my_ssl_api',
$payload );
        $res = $res->{content} if
length $res->{content};
        $res =
JSON::MaybeXS::decode_json($res);
        # If we haven't thrown an
exception by now, we've gotten a
certificate. Hooray! Let's go ahead and
install it.
        $res = eval {
$self->install_certificate(
'web_vhost_name' => $vhost,
'certificate_pem' => $res->{'cert'},
'key_pem' =>
$key->get_private_key_string() ); };
        warn $@ if $@;
    }
    # If we've gotten here, we're
groovy. The AutoSSL logger will report
great success to the user regarding the
AutoSSL check *for this account*.
    # Any exceptions/warnings thrown
earlier will be presented to the
administrator in the autossl log.
    return;
}

# A simple skeleton function for
creating a CSR for a vhost.
# See
https://metacpan.org/pod/Crypt::OpenSSL
::PKCS10 for a CPAN module that can help
here.
sub _create_csr_for_vhost {
    my ( $key, @domains ) = @_;
    my $req =
Crypt::OpenSSL::PKCS10->new_from_rsa($k
ey);

 # Add whatever extensions, etc. you'd
need in general for your request
 ...

 foreach my $domain ( @domains ) {
  # Add whatever you might need to add
per domain for your request
  ...
 }

 # Get the CSR in PEM format for us to
return.
 my $csr = $req->get_pem_req();
 return $csr;
}

# Do something here that would generate
the DCV files in the places you would
normally look for DCV files on a domain
on your end.
# In this example, I'm iterating over
the array of domains in a vhost we
passed in above. I've also added the CSR
text in case we wanna use that
# for some reason here. Pass in whatever
you need here. If parsing the CSR is
```

```
needed, use
https://metacpan.org/pod/Crypt::PKCS10
# If you want a CPAN module that can
help for writing your files, use
File::Slurp::write_file -- see
https://metacpan.org/pod/File::Slurp
sub _generate_dcv_files {
 my ( $csr, @domains ) = @_;
 my $something;
 foreach my $domain ( @domains ) {
  #Create your DCV files by whatever
means you deem necessary
 }
    # Presumably whatever you want to
return gets populated within the loop if
you need to consume this information
later.
 return $something;
}

1;
```