
# Version 1.1

# User's Guide

IMPORTANT NOTICE:

The zip file (or any other file) containing the scripts is not to be uploaded to your web server! Doing so invites theft of the software from your site and you become a party to the theft of copyrighted material. Instead, un-compress the files onto your own local hard drive and upload the individual files as described in the Installation section.

DISCLAIMER AND/OR LEGAL NOTICES:

The information presented herein represents the view of the author as of the date of publication. The report is for informational purposes only. While every attempt has been made to verify the information provided in this report, neither the author nor his affiliates/partners assume any responsibility for errors, inaccuracies or omissions. Any slights of people or organizations are unintentional. This report is not intended for use as a source of legal or financial advice. If advice concerning legal, financial, or related matters is needed, the services of a fully-qualified professional should be sought. You should be aware of any laws that govern business transactions or other business practices in your country and state.

# INTRODUCTION

This guide describes the SecureDL (Secure Download) WordPress Plugin, designed for obfuscating and protecting standard download links and creating authenticated Amazon S3 links, all using a simple tag within your posts and pages.

# INSTALLATION

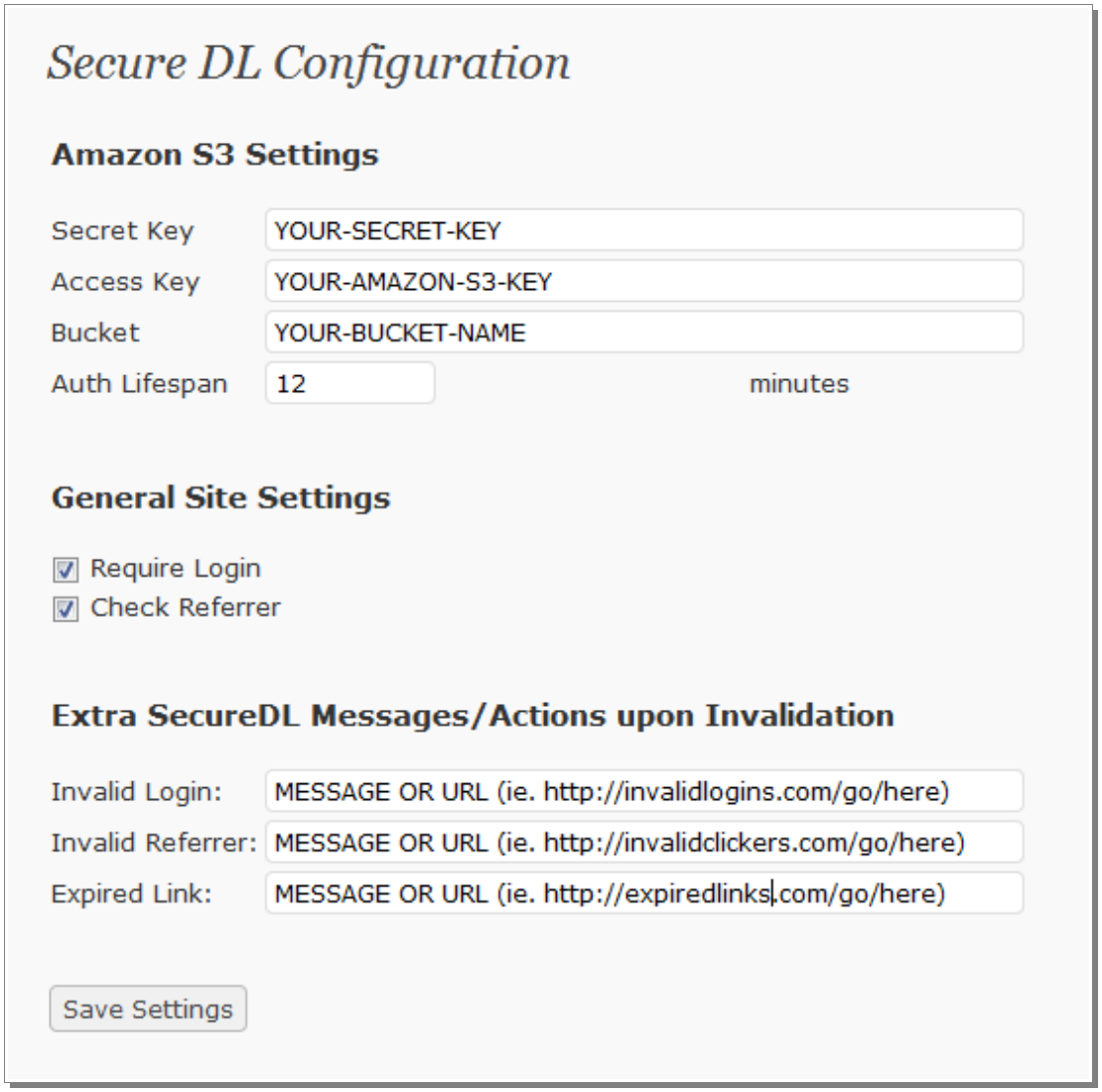
Using an FTP client, upload the SecureDL folder to your blog's wp-content/plugins directory.

From your WP admin panel, navigate to the Plugins screen and click the Activate link for the Secure DL plugin.

Click the link for the plugin's settings panel, either from the plugins screen or under the Settings menu.  Register using the PayPal email with which you purchased the plugin.

# SETTINGS

From the plugin's administration panel, you'll be presented with the following small set of default options.  If you're using the S3 feature of SecureDL, the secret key and access key are required.  Other than that, all settings can be overridden on a per link basis.




There's one implicit default not shown on this screen, namely, that of obfuscating links and redirecting them through download.php in order to handle the login and referrer checks. For standard links, this is enabled by default. However, in the case of S3 links, since the plugin generates a fully authorized and expiring S3 link, leveraging Amazon's security and saving your server's bandwidth.

IMPORTANT: Please read Amazon's security guidelines carefully. You generally do not want to make your bucket and files readable to just anyone. That's why you have authorization keys.

You'll see the other configurable defaults also start out with a couple basic security measures in place:

The first checkbox, Require Login, simply verifies there is a user is logged in to your blog from the computer accessing a SecureDL link. Nothing fancy, no permission checks, etc., but still a useful option. If the user is not logged in, clicking on such a file returns the error "You must be logged in to download this file."

The other checkbox, Check Referrer, ensures the link has been clicked from your blog versus being manually entered into the address bar or via another site. Failure to pass this check results in the error "You do not have permission to access this file."

To configure S3, only your secret key and access key are required. The bucket and length of a time an Amazon link is active before expiring (specified in minutes since generation) are optional and can be provided on a link-by-link basis.

The last section is for customizing the error messages or redirecting to an URL of your choosing. For example, if you require login, your error handling for someone not logged in might be to redirect them to the login screen or to a member registration. Similarly, for someone who's just pasted your download link into their browser directly.

## USAGE

To create a Secure DL link, you will use the `[secdl]` bbcode-style tag within your post or page with various attributes (some required, some optional) described as below.

For a standard (non-S3) link, the most basic token would look like:

```
[secdl file=http://example.com/downloads/file.zip]
```

For an Amazon S3 link, you would specify at least the path, but quite likely also the bucket, like so:

```
[secdl bucket='example' path='docs/file.zip' hide=1 life=15]
```

Let's take a closer look at this one. First of all, the single quotes are optional. Similarly with bucket, hide, and life, which are all optional. By providing them within the token as here, you simply override the defaults from the plugin's settings panel.

***A word about hide=1*** As discussed in the Settings section, by default, Amazon S3 links do not go thru download.php. To do so would waste your own bandwidth and typically not provide too much extra security over their existing authorization parameters. Furthermore, the download links are not for use in video players. That said, you can opt to include S3 links in the obfuscation (as well as the login and referrer checks) by specifying hide=1 as above.

That leaves the two other settings, Require Login and Check Referrer. To set these on a per-link basis, you would use the reqlogin and checkref attributes. For example,

```
[secdl file=/var/www/vhosts/example.com/files/goodies.zip
reqlogin=0 checkref=1]
```

In this case, the user doesn't have to be logged in to your blog, but they must have clicked from your domain, regardless of how you have them set in the admin panel.

*Note: The full path in this example leads to a file above the public area (typically httpdocs or public_html), providing even more security.*

To recap, please find the complete set of attributes on the next page. These enable you to override the plugin settings on a per-link basis.

## Standard Links

```
[secdl file=http://example.com/blah.zip]
```

| Settings | Values | Description |
|---|---|---|
| file | REQUIRED | Full URL or relative path |
| reqlogin | 1 or 0 | Require login: 1 (yes) or 0 (no) |
| checkref | 1 or 0 | Check referrer: 1 (yes) or 0 (no) |
| life | # (minutes) | Expire after number of minutes (defaults to no timeout) |
| link_expired | | Error message or URL to redirect to when the link has expired |
| refcheck_invalid | | Error message or URL to redirect to when the link was pasted directly into the browser address bar or was clicked from someone else's site (if reqlogin is enabled) |
| reqlogin_invalid | | Error message or URL to redirect to when the user is not logged in |

## Amazon S3 Links

```
[secdl path=files/blah.zip]
```

| Settings | Values | Description |
|---|---|---|
| path | REQUIRED | S3 path |
| bucket | see s3 doc | S3 bucket |

| | | |
|---|---|---|
| `hide` | 1 | Send thru download.php as with standard links (makes reqlogin and checkref available) |
| `life` | # (minutes) | Expire after number of minutes (defaults to setting in admin) |
| `method` | player | Use JavaScript encoding (see Advanced section for details) |

Now, let's discuss how you'd go about actually using these.

Just placing a [secdl] token on the page all by itself may be a good way to get a feel for the replacement that's going on, but ultimately you're going to want to create a download link or play a video.

Each time your post or page is refreshed, the `[secdl]` links are generated on the fly. In the case of S3 links, that means a new expiration date is also created. What you're left with is something that looks like:

```
http://example.com/blog/wp-content/plugins/SecureDL/download.php?
file=amessofcharactershere
```

So, for an href, you'd simply use the token in place of the URL:

```
<a href=[secdl file=http://example.com/myfile.zip]
title='Pretty cool, eh?'>Download</a>
```

Or in a video player, such as WP-SWFObject, an S3 path:

```
[swf][secdl path='files/myvideo.swf'],450,225[/swf]
```

where the 450 and 225 are part of that plugin's `[swf]` tags, indicating width and height, respectively.

## ADVANCED

While SecureDL is designed, as its name implies, for downloads, the authorized URLs it generates for Amazon S3 can also be used within the tokens of several video player WordPress plugins which otherwise would not accept them.

Some, like WP-SWFObject mentioned above, are compatible with the most basic of settings. Others simply require an extra attribute, namely, `method=player` in order to properly encode the URL for use in their JavaScript code. For example, the FLV Embed and Flash Video Player plugins both need this attribute specified on the `[secdl]` link.

And, in the case of one of the most popular video player plugins, FlowPlayer For WordPress, we've had to provide a fix to its token replacement for authenticated URLs.

To apply it, follow the directions in the flowplayer.php file found in the SecureDL/pluginhacks directory. It instructs on where to place the following two lines within the FlowPlayer plugin code:

```
// as per guidelines of SecureDL wordpress plugin
include( ABSPATH . 'wp-content/plugins/SecureDL/pluginhacks/flowplayer.php' );
```