Documentation

Current Version: 01.02.03
Released: 09/09/08

Table of Contents

Introduction

OneBridge is a specialised enterprise-class bridge/reverse-proxy that is used to connect multiple versions of the OneGate universal Common Gateway Interface (CGI) engine, which itself allows communication between web browsers and "applications" of any kind.

OneBridge essentially allows you to have a public and a private web server, minimising the security risks and exposure of putting your data or application server directly on the network. Regular fields, cookies, and even HTTP File Uploads all work correctly.

Unlike regular proxy servers, you can use the **application-level** firewall in the copy of OneGate on the public server to determine what gets stopped at the public side. Likewise, OneGate's security features keep most malicious CGI attacks from ever reaching the private server, also reducing the load on that server. Combining the power of OneGate with OneBridge is a large boost to security of your data/application server within a secure network configuration.

OneBridge is exceedingly easy to use. If you've used OneGate, you already know everything you need to know to use OneBridge. Installation is a snap, configuration consists of setting two variables, and OneBridge plugs right into the public server's copy of OneGate.

Features

- Supports multiple program sets (applications)
- Supports GET, POST (x-www-application-urlencoded, or multipart/form-data)
- Supports HTTP File Upload with safe filename handling
- Provides internal OneGate with the information the external OneGate received, and sends the reply back with the correct MIME type
- Designed with system security in mind
- Provides application-level virtual firewalling, allowing you to deny or allow wildcarded subnets or explicit IP addresses
- Supports multiple testbeds as well as production configurations
- Source code immediately available for modifications, if necessary or desired

---

Requirements

This program requires Perl version 5.6 or higher to run. It also requires the Perl XML::Parser module installed on your system to operate correctly.

In addition, you will need two servers, each running a copy of OneGate, and one copy of RawQuery on the public server. These products are sold separately.

Both servers should be running an HTTP daemon configured for CGI execution.

OneBridge has been tested on Solaris, Linux, SCO UNIX OpenServer 5.0.6, and Windows 2000. It should encounter no difficulties on any platform on which perl will compile and run correctly.

Licensing

It should be noted that there is no demo version of OneBridge, nor will there be a "Lite" version.

We will provide whatever pre-sales assistance is required to demonstrate whether OneBridge is the right product for your needs. If you have questions or comments, please direct them to sales@fairlite.com without hesitation, and we will answer as promptly as possible.

We realise that software is an investment, and hope that you preview the documentation available for this product and make sure that it will suit your needs before purchasing a license. We also hope you avail yourself of the opportunity to ask any pre-sales questions you might have. All license sales are final and non-refundable.

OneBridge is licensed at a cost of **$2000.00 USD** *per server on which it is installed*. Bulk discounts may be negotiated for purchases of more than five copies at the same time.

Each license fee entitles you to use OneBridge on one server, in any role you require. You may modify the program to further suit your needs, and are under no obligation to release changes back to Fairlight Consulting. However, derivative works and/or modified versions **may not** be resold or otherwise distributed. Similarly, you may not copy OneBridge, modify it, and run the altered version on another machine. You must purchase another license to use it in any form, altered or otherwise, on an additional machine. You may use an altered version and the original version on the same machine under a single license, however. ***"Machine" shall be defined as one instance of an operating system, for the purpose of this license. Machines which run multiple concurrent operating systems (virtual machines) count as multiple machines, and require additional licenses for each instance.*** The licensee agrees to keep the source code confidential and protected.

Upon receipt of payment for a license, access to the program will be generated for the licensee, and such information shall be delivered to the email address associated with the PayPal payment.

Upgrades for the product are *currently* free when moving to new minor and major versions. Fairlight Consulting reserves the right to change this policy in the future, with no prior warning.

There is ***no warranty*** for this software. This software is offered **"AS-IS"** and without warranties as to performance or merchantability or any other warranties, whether expressed or implied.

Good computing practice dictates that any program should be thoroughly tested with non-critical data before deploying it for production. The user assumes the entire risk of using the program. In no event shall Fairlight Consulting be held liable for loss of data, failure of performance, or any other damages, be they real or perceived.

If you agree to these terms, click here to order OneBridge!

Already registered and have your account information?
Download OneBridge!

Support

Extended (non-bug-related) support for OneBridge is available at our standard hourly rates. Because we offer pre-sales assistance in determining if OneBridge is right for your needs, and because documentation is readily available, anything not covered by either of these is deemed an at-cost support issue.

You may request *any* kind of technical assistance with OneBridge by sending email to onebridge@fairlite.com, including bug reports and feature requests.

Feature requests may be commissioned for special functionality, if desired. Any non-commissioned requests are subject to being implemented soley at the discretion of Fairlight Consulting. This may include not being implemented at all, depending on how useful we think the feature would be in general. Commissioned requests can be price-negotiated based on whether the features requested are allowed to be re-integrated into the main product, or whether they shall remain proprietary and exclusive to the commissioning party.

We also offer consulting on how to achieve specific results using the product through this support mechanism, and would be happy to assist you in this regard.

Installation

Installing OneBridge is very straightforward, and involves a few simple steps.

OneBridge should be installed to the same directory as *OneGate*. There will be a program file and a configuration file present after installation.

If you are installing on *nix, make sure you log in as or su to the same user as should be running onegate prior to installation. If you do not, you will need to manually adjust permissions and ownerships accordingly.

To install OneBridge, follow these steps:

1. Copy the distribution tarball to a directory where you have write permission.
2. Untar the distribution tarball.
3. Change directory into *onebridge-install*.
4. Run **perl ./install.pl** to begin the installation process.
5. Answer the location questions.
6. Adjust the ownership of the files **onebridge** and **onebridge.conf** if necessary.
7. Proceed to the section on configuration.

Upgrading

To upgrade OneBridge, follow the steps for installation.

Please note that OneBridge's installer will overwrite the old program file, but your entire *onebridge* configuration file will remain untouched by the process. All existing configuration will be retained.

Despite this fact, good computing practice dictates that one should make a backup of one's configuration before performing an upgrade. It is not necessary, but it *is* wise.

Configuration Instructions

In this section we will walk you through the relatively simple tasks involved in configuring OneBridge for use with OneGate and your applications.

Before we begin, it should be stated that in the following sections, pathnames will contain certain parts in square brackets. These parts are:

- [maindir] - The full path to the main *onegate* directory for configuration and spooling.

- [progset] - The name of a valid program set, used as a directory name.

Configuration File

The file *[maindir]/onebridge.conf* controls global behaviour for OneBridge. There are four options in all, which are described in detail on the Configuration Options page. The format of this file is one **option=value** assignment per line. Comments may be on lines by themselves, and blank lines are ignored.

Individual OneGate program sets should be configured normally on the internal data server, as if it were the public server. Program sets on the public server side should contain the following line for *nix systems:

```
/path/to/onebridge
```

On Windows systems, you will need the following:

```
\\servername\path\to\perl.exe \\servername\path\to\onebridge.pl
```

That's all you need for the program set command list on the public server. You can apply firewall rules as you like. The field list should be identical to the private server's field list for each program set.

Configuration Options

The following options are valid for OneBridge's configuration file:

debug

When this is activated, with **on**, the program goes into debug mode. This is for OneBridge development and testing only, and should not be used in production.

no_unlink

Similar to debug, this is for OneBridg development and testing only, and should not be used in production. When set to **on** it prevents removal of OneBridge spool directories and their contents.

internal_onegate

This option should have as a value the URL for the private OneGate location. An example value would be *https://204.199.12.2/cgi-bin/onegate*.

cookie_domain

This option sets the domain to resend cookies for. If your private server is configured to believe it is *www.widgets.com*, the domain should be **.widgets.com** (note the leading dot!).

timeout

Timeout in seconds to wait for a response from the internal OneGate.

Configuration Example

This example is meant to illustrate, in layman's terms, how OneBridge is meant to be used. This example will utilise the case of the filePro database/application suite in a web-base context, for the sake of illustration of one of OneBridge's strong points: namely, **disparate server access between the web browser and the application server**.

The first prerequisite to fill is configuration of a private server inside a firewall. This server hosts the application (filePro, in this case), as well as a private web server. The machine should be firewalled with the exception of port 80, port 443, or both if desired. Those ports should be open to access **only** to the public web server described below. On this server, you configure OneGate to serve all your application needs, as normal.

The second phase of using OneBridge is to configure a public web server on the Internet proper. This can be just outside the firewall, or half a world away; it makes no difference. The entire point is that, besides adding security by keeping the application system firewalled and adding another layer to the onion any hacker would have to peel, the data source server and the web server can now be disparate entities. Older applications such as filePro are not network-capable in the terms most *SQL users would think of, and this is a means of achieving a separation that comes normally to most newer applications. This server is configured with a public web server. The OneGate program sets should have identical *fields* files, but the *program.set* file will simply call OneBridge, which will handle the rest of the scenario.

Once configured, when a web browser accesses a OneGate program set on the public server, all security prerequisites are checked. The virtual firewall at the application level is checked. All required and missing field checks are performed. The entire request is pre-screened. You would put your "world-at-large" firewall rules on the public server, allowing or denying whatever you wish.

Assuming all security criterion are met, the request is passed inward to the private web server. At this level, the internal OneGate has its own ability to have different firewall rules per program set. This means that applications can be LAN-only, Internet-only, both, or subsets of the possibilities. See OneGate's virtual firewall documentation for more on this facility.

The request is acted on within the internal server, the output is generated for the web and passed back to the public web server. From there, it is passed back to the browser, preserving the MIME type of the internal OneGate's response.

That, in a nutshell, is that.

What makes this better than a reverse proxy? The added security features ***at the application level*** within OneGate allow for dual application-level firewalling, without having to try and do intricate things with URL rewrites (which would not be possible to distinguish in a POST context), hardware or more general port firewalls, or other security methods. You handle everything with OneGate and let its security features work for you. Your basic hardware firewall rules for the internal machine are configured once and stay the same. You can change the OneGate virtual firewall rules on a whim, and affect either the whole site, a few program sets, or one specific program set--whatever you need. OneGate also prevents malicious attacks from ever even reaching the private server at all. Generally, most requests of that type have missing or extra fields, and that's all caught on the *outside*, preventing gratuitous abuse of resources on your private server.

Moreover, you can "split" your services, using OneGate locally on the web server for some program sets, while allowing other requests to be brokered to the private server. No amount of rewrite rules in a web server or hardware firewall rules will let you do that, as to those it all looks like it's going to the same place. OneBridge gives you the flexibility to do it all at the application level without having to know a cartload of complex configuration.

In short, OneBridge is a perfect companion to OneGate in terms of security and versatility.

Change Log

v01.01.00 - 08/23/05

Initial public release.

v01.02.00 - 06/28/06

Revised error handling so that OneGate locally indicates there was an error if OneBridge fails. Created logging for OneBridge failures (**onebridge.log**).

v01.02.01 - 11/08/07

Changed pathname handling for file uploads.

v01.02.02 - 03/01/08

Fixed data mangling of file uploads and textareas larger than 8KB.

v01.02.03 - 09/09/08

Fixed MIME type RE check. Added timeout config option.
