AJP with stunnel

Permalink to this page; https://cwiki.apache.org/confluence/x/TCglBg

AJP over stunnel

stunnel is a little more complicated than a normal protocol because it can be used in a number of different ways. I'll give some contrived examples to see how you can set it up in different ways, depending upon the support for encryption of the underlying protocol.

This wiki entry is intended to be a starter-guide and not a replacement for the fine documentation provided by the stunnel team.

Let's say that you have an HTTPS server, but your client can't speak HTTPS for some reason. If you set up stunnel on the *client* side, you can connect locally to the stunnel server and have it establish a secure-connection to the remote server running HTTPS. Like this:

```
client -> localhost:12345 (stunnel)
stunnel -> remote_host:443 (httpd)
```

As far as the client is concerned, it's using HTTP to talk to localhost. But really it's talking to remote_host:443, so everyone is happy. (Yes, there are issues with URLs and redirects produced by the server, but that's out of scope for this discussion).

Let's take another example: you have clients that are HTTPS-capable, but the service you are running can only support HTTP for some reason, and you want to secure it. Set up stunnel on the *server*, then have your remote clients connect to *it* and tunnel to localhost. Like this:

```
client -> remote_host:443 (stunnel)
stunnel -> localhost:8080 (httpd)
```

As far as the client is concerned, it's using HTTPS to communicate with remote_host:443, but really it's connecting to remote_host:8080. (Yes, there are some issues with URLs and redirects but that's out of scope for this discussion.)

So what if the underling protocol doesn't support TLS at all? Well, then you have to set up stunnel on *both sides* of the tunnel, like this:

```
client (mod_jk) -> localhost:12345 (stunnel)
stunnel -> remote_host:12345 (stunnel)
stunnel -> localhost:8009 (Tomcat)
```

The setup for stunnel looks like this for the client (on the web server):

```
sslVersion = all
options = NO_SSLv2
options = NO_SSLv3
client = yes

[ajp13s]
accept=localhost:8009
connect=remote_host:8010
```

On the server, it looks like this:

```
sslVersion = all
options = NO_SSLv2
options = NO_SSLv3
client = no

[ajp13s]
accept=8010
connect=localhost:8009
```

On the web server, set your worker's host to "localhost" and port to 8009. mod_jk will connect to localhost:8009 which stunnel will accept and forward over the network to remote_host:8010 which will be accepted by stunnel on the server and forwarded to localhost:8009 on the server.

stunnel is great because it will auto-reconnect if the connection is dropped for some reason. Remember a few things with stunnel:

1. Depending upon the version, you might only be able to use TLSv1 (and not e.g. TLSv1.2)

2. stunnel generally ignores certificate issues, such as expiration, etc. You might want to configure it with a little more care than the default. **THIS ALSO MEANS IT DOES NOT AUTHENTICATE THE SERVER BY DEFAULT**. You could accidentally connect to a malicious server.

This should be enough to get you started. Please refer to the official stunnel documentation for specifics.