Using GDMA with HTTPS

Contents

1.0 Using HTTPS

Some customers prefer the security of HTTPS when GDMA clients contact the GroundWork Monitor server to auto-register the client, and to download configuration files and plugins from the server to the client. Starting with GDMA 2.3.2, a GDMA client configured to use HTTPS for these purposes insists on having a valid server certificate file installed on the client, so the client can ensure that it has a trustworthy connection to the server. This is the basic protection required to prevent a man-in-the-middle (MITM) attack. Previous releases of GDMA only supported the encryption capabilities of SSL, and did not use certificates to otherwise protect the data on the wire. So any site using an older release that wishes to enforce this extra level of protection should upgrade to the GDMA 2.3.2 release.

1.1 Dealing with certificates

Before you begin to address GDMA client setup for HTTPS, you must take the first steps toward configuring Apache and Java within GroundWork Monitor to support HTTPS. Instructions for doing so can be found in How to enable SSL support page in the Bookshelf. For purposes of supporting GDMA over HTTPS, the key parts are the creation of the server private key file (`/usr/local/groundwork/apache2/conf/server.key`), an initial server certificate (`/usr/local/groundwork/apache2/conf/server.crt`), and (optionally) an initial empty revocation list, though of course you will need to complete the rest of the server setup as well.

A server certificate can be either generated locally and self-signed, or obtained from a Certificate Authority. The additional steps for dealing with a Certificate Authority, making a certificate request, and dealing with the file you get back from them, are beyond the scope of this document.

For GDMA purposes, you will need a certificate for your GroundWork Monitor server, as generated via the instructions noted above, and (if desired) a Certificate Revocation List (CRL) file. The CRL file is allowed and used if present, but is not required by the GDMA client software. You may put one in play even if you know of no certificates that should be revoked. The GDMA software uses the CRL, if present, so that customers who might actually want to revoke certain certificates have the built-in capability to do so.

Managing SSL certificates with OpenSSL, as we will do below, is a broad topic. Here, we will only scratch the surface. If you want more details, dive into the OpenSSL documentation as well as other sources on the Internet.

Specifics of the server-side configuration for supporting GDMA are given in the GroundWork Monitor server setup for HTTPS section below.

Forewarned should be forearmed
Managing SSL certificates in your infrastructure is a long-term process. Realizing that and acting on that realization beforehand will save you considerable grief later on.

- Certificates and certificate revocation lists will eventually be replaced. Partly because they are distributed across a large number of machines, it pays to have well-understood and practiced procedures already established when you need to do this in a hurry. See the **Be Prepared** note near the end of this document for particular advice on procedures you should put in place early on.
- You must carefully specify certificate and revocation-list lifetimes, be aware when they are about to expire, and be proactive about replacing them on all your machines before that happens.
- Particularly because those lifetimes are often specified in very long time periods, such as several years, people are likely to forget about upcoming expiration times by the time they finally do occur. Establish some way to keep track of such infrequent maintenance activities in a way that it won't be a sudden surprise when the expiration date arrives. Or establish and test automated procedures for upgrading such files, and replace them more frequently.
- Certificate management is made a lot easier if you establish certain conventions from the beginning, such as numbering your certificate and revocation-list files so you can tell which is which when new ones appear in your infrastructure. The commands we show in this document adopt a simple standard for such numbering. You can adopt whatever policies and conventions you want for your own site, but you should definitely think about such issues in advance of deploying SSL in your infrastructure.

1.2 Direct use of the GroundWork server name as the Target Server

Previous instructions on setting up GDMA clients sometimes recommended use of the virtual `gdma-autohost` name for the GroundWork Monitor server in the `Target_Server` parameter on the GDMA client, this being the default Target Server hostname provided by the GDMA installer. Use of the virtual hostname, which would need to be resolved by your DNS to point to the GroundWork server, allows a level of indirection in how that name is interpreted.

With the use of SSL certificates, that convention is no longer possible because the hostname in the `Target_Server` parameter must exactly

match the hostname used in the SSL certificate, and the hostname in the SSL certificate must match the hostname of the GroundWork server in order for the certificate to be seen as valid for accessing that server. (Use of the `gdma-autohost` name as the value of the `GDMA_Auto_Host` parameter can continue as before, because this use of the name is not tied to the use of SSL certificates. It relates to content passed over the connection to the server, not the establishment of the connection.)

Because matching the name is so critical, setting of the `Target_Server` value to the appropriate server name is called out later in this document, for both existing GDMA clients and for new GDMA clients. "Matching" means an exact match: either fully qualified or unqualified, exactly as contained within the certificate. If you're not sure of the name, you can see the exact hostname contained within a certificate named `server_01.pem` with the following command where the displayed `CN` field will contain the hostname:

```
cd /usr/local/groundwork/gdma/certs
/usr/local/groundwork/common/bin/openssl x509 -noout -text -in server_01.pem \
    | fgrep Subject:
```

2.0 Transitioning your Infrastructure to HTTPS

Customers who have many GDMA clients will naturally want to know how to transition all of them from HTTP to HTTPS, and whether a big-bang approach is required. The transition plan depends on your current setup. For details see the Bookshelf document How to enable SSL support.

- **HTTP (GDMA 2.3.1 or earlier) to HTTPS (GDMA 2.3.2 or later)** — Here are the steps:
    1. Upgrade the server to support HTTPS, including installation of SSL certificates on the server. GDMA clients still configured for communication with HTTP will be automatically redirected to use HTTPS (though these older versions of GDMA won't use certificates at this point to validate the server).
    2. Upgrade the GDMA release on individual clients, specifying the use of HTTPS, and installing the server certificate and (optionally) a certificate revocation list, as described below.
- **HTTP (GDMA 2.3.2 or later) to HTTPS (GDMA 2.3.2 or later)** — Here are the steps:
    1. Generate the SSL certificate and (optionally) a revocation list on the server, but *do not otherwise convert the server to run HTTPS at this point*.
    2. Install the SSL certificate and (optionally) the revocation list on the server, so those files will be available there when you later upgrade the server to run HTTPS.
    3. Install the SSL certificate and (optionally) revocation list on each and every GDMA client, so that HTTP clients which are redirected to use HTTPS after the next step will be able to do so. If you are also upgrading the GDMA client to run a newer GDMA release at this time (say, from 2.3.2 to 2.4.0, at some future time when such a release is available), specify that it should use HTTP for the time being, since the server does not yet support SSL at this stage.
    4. Take the remaining steps to upgrade the server to fully implement SSL, per the instructions referenced in the first step.
    5. Modify the `gdma_auto.conf` file on each and every GDMA client so its `Target_Server` parameter specifies the `https://` protocol.
- **HTTPS (GDMA 2.3.1 or earlier) to HTTPS (GDMA 2.3.2 or later)** — Here are the steps:
    1. Ensure that you have SSL certificates installed on your server.
    2. If you did not previously have SSL certificates on the server, restart Apache now (`service groundwork restart apache`), to ensure that they are in play.
    3. Upgrade the GDMA release on individual clients, specifying the use of HTTPS, and installing the server certificate and (optionally) a certificate revocation list, as described below.

You will notice that none of these procedures requires a big-bang approach (converting all of your GDMA clients instantly). Existing GDMA clients will continue to operate and report production results throughout the conversion process. Still, once you've started the conversion, it's best to run it through to completion as soon as possible.

Watch out for conflicting Target_Server definitions
Some customers set a variety of client-configuration parameters (other than the commands to run) in host externals. If your setup does this for the `Target_Server` parameter, you must ensure that the protocol (`http://` or `https://`) in this secondary definition is always in lockstep with the definition in the client's own `gdma/config/gdma_auto.conf` file, at every stage of the transition.

3.0 Managing SSL Certificates

The SSL certificates in use may change over time, for a variety of reasons: limited expiration dates, intentional change of Certificate Authority, unexpected loss of trust in the certificate chain, and so forth. You will therefore likely put in place new certificates from time to time. When this happens, it can be considerably confusing if you always use the same file names for the old and new certificate files and certificate revocation list (CRL) files — how will you know which is the new file and which is the old file? To counter that problem, in the discussion below we adopt a strategy of numbering the certificate and CRL files, so it will be clear what is happening when you upgrade from one certificate to another. We also use certain temporary names during invalidation procedures, both to clearly label the nature of old and new files, and to collapse multiple cases into a single, easy-to-follow process.

4.0 Installing HTTPS on GDMA Clients

Supporting GDMA communication over HTTPS requires both server-side and client-side configuration.

4.1 GroundWork Monitor server setup for HTTPS

These steps must be completed on the server before dealing with any GDMA clients. Run all of these commands as the `nagios` user.

1. If you are doing this setup on GWMEE 6.7.0 or earlier, you will need to edit the `/usr/local/groundwork/common/openssl/openssl.cnf` file on your GroundWork Monitor server, to make one correction before you begin. This line:

   ```
   dir     = ./demoCA                              # Where everything is kept
   ```

   must be changed to:

   ```
   dir     = /usr/local/groundwork/common/openssl     # Where everything is kept
   ```

   That change will allow the rest of the `openssl` commands you execute to find the necessary files that cannot be specified on the command line.

2. Either obtain a trusted certificate from a known Certificate Authority, or generate a self-signed certificate on your GroundWork Monitor server. Basic instructions for this setup are given in the How to enable SSL support page. Keep track of whether you specify the unqualified or qualified hostname in the certificate — this will be important later on.
3. Optionally, generate an empty certificate revocation list (CRL) on your GroundWork Monitor server.

   

   Obtaining CRLs for non-self-signed certificates is more complex
   The process below works fine if you are using self-signed certificates. If you get your certificates from some other source, the procedure will be different; you will likely need to obtain the CRL, if you desire to use it, from the upstream certificate provider. And then you may need to convert it into the `.pem` format which the GroundWork software can deal with. Details regarding the procurement and conversion of a CRL under these conditions are beyond the scope of this article.

   To generate a CRL, you will execute the following commands as the `nagios` user.

   a. Before you run the `openssl` command below, first make sure you have a `/usr/local/groundwork/common/openssl/index.txt` file on your system. If not, create it this way:

      ```
      touch /usr/local/groundwork/common/openssl/index.txt
      ```

      The `index.txt` file will store details of previously revoked certificates, so they can be included in future CRL files without your needing to specify them again.

   b. Similarly, before you run the `openssl` command below, first make sure you have a `/usr/local/groundwork/common/openssl/crlnumber` file on your system. **If so, leave it alone.** If not, create it this way:

      ```
      echo 01 > /usr/local/groundwork/common/openssl/crlnumber
      ```

      The `crlnumber` file keeps track of the sequence numbers of revoked certificates; the `openssl` command below will henceforth update it as certificates are revoked. We will also use the number in this file to distinguish the CRL files that we generate over time.

   c. **In the following `openssl` command, the `-crldays` option value should be specified to reflect how often you will update the certificate revocation list on all of your GDMA clients.** Here in this example, we set that to a time period of three years, that being the same time period we are using in our examples for generating SSL certificates. (If your CRL expires before you revoke any certificates and have to regenerate a new revocation list for that reason, just generate a new one and distribute it to all your clients that were using the expired list. The CRL is simply a signed copy of your master list of revoked certificates, combined with a validity date and placed into a standard format. You can create a new copy whenever you want.)

      ```
      cd /usr/local/groundwork/apache2/conf
      /usr/local/groundwork/common/bin/openssl ca -gencrl \
          -keyfile server.key -cert server.crt \
          -out crl_`cat ../../common/openssl/crlnumber`.pem -crldays 1095
      ```

      It's not actually necessary to store the output revocation-list file (e.g., `crl_01.pem`) in the same directory as the key and certificate files, but having it there makes it a lot easier to find later on (when you will copy certificate and revocation-list files to your client machines).

4. To make future file copying more convenient, make a copy of your certificate file under the name by which it will be known on the client machines. Embedding a trivial serial number in the copied filename will make it much easier to distinguish which file is which later on, so **change the command shown here to choose the next number in sequence at your site**. The first such copy command might look like this:

```
cd /usr/local/groundwork/apache2/conf
cp -p server.crt server_01.pem
```

5. Finish setting up Apache and Java on the GroundWork Monitor server to use HTTPS, as documented in the How to enable SSL support Bookshelf page. That will include bouncing both `apache` and `gwservices`, as listed at the end of that procedure.

4.2 Upgrading existing clients from HTTP to HTTPS

Installing HTTPS on existing GDMA clients takes the following steps:

1. Transfer the server certificate file (e.g., `server_01.pem`) and (optionally) the certificate revocation list file (e.g., `crl_01.pem`) from the GroundWork Monitor server to the GDMA client. These files must have a `.pem` filename extension on the client, which is why you made a copy of the certificate file under that name on the server, to make these file transfers easier.
2. Put the file(s) into the `gdma/certs/` directory (e.g., on Linux GDMA, this would be in `/usr/local/groundwork/gdma/certs/`).
3. Make sure that each of the files in this directory is owned by either the GDMA user account that owns all the rest of the GDMA client files, or (on UNIX-like systems) by `root`.
4. Make sure that each of the files has no write permissions to anyone other than the file owner.
5. Make sure that none of the files you are installing is completely empty.
6. Create required symlinks, by running the `c_rehash` program. On the Windows platform, the directory argument is defaulted, so you don't have to mention it explicitly. On all other platforms, you must mention the path to the `gdma/certs/` directory explicitly, but it can be given as a simple relative pathname, as we do in the following sample commands.
    - On a Windows system, the commands would be like:

```
cd {path-to}\gdma\certs
..\..\common\bin\c_rehash
```

    - On a Linux or AIX system, the commands would be:

```
cd /usr/local/groundwork/gdma/certs
../../common/bin/c_rehash ./
```

    - On a Solaris system, the commands would be:

```
cd /opt/groundwork/gdma/certs
../../common/bin/c_rehash ./
```

7. Verify that the `gdma_auto.conf` file specifies that you want to use HTTPS, by using that protocol in the `Target_Server` definition. For example:

```
Target_Server = "https://gwserver.mydomain.com"
```

8. In that `Target_Server` value, be sure that you use exactly the same server name, unqualified or qualified, as you used when you created the certificate for that server, for details see Direct use of the GroundWork server name as the Target Server section on this page).
9. Restart GDMA on the client.

4.3 Installing new clients to use HTTPS

If you are just starting to use GDMA and have no clients as yet, follow the instructions above to set up your server, before you install any GDMA clients.

Once your server runs with HTTPS, installing new GDMA clients to use HTTPS is easy.

1. When the installer asks for the Target Server hostname, specify the name of the GroundWork Monitor server exactly as it is included in your SSL certificate, for details see Direct use of the GroundWork server name as the Target Server section on this page).
2. When the installer asks for the protocol to use for communication with the GroundWork server, select HTTPS.
3. When the installer asks whether you wish to start the GDMA service after the installation, select No. (This question is not asked during a Windows GDMA install.)
4. Follow all the steps listed in the preceding section for installing certificate files on the GDMA client.
5. Start (or restart, on Windows) the GDMA client.

Note that command-line options are available for unattended-mode installs, for selecting the installation mode, target server name, communication protocol, and (on non-Windows platforms) whether to start the service at the end of the install, as well as other parameters you may wish to set. Run the installer with the `--help` option to see these options listed.

5.0 Maintaining the HTTPS Setup

Starting with the GDMA 2.3.2 release, a GDMA client that wants to perform various configuration-related interactions with the GWMEE server via HTTPS must have a valid SSL certificate and (optionally) a valid certificate revocation list (CRL) in place. Reporting of monitoring results is done via a separate channel, which is not dependent on the SSL certificate files. (This channel can also be encrypted if you wish, but that is a separate setup issue.)

From time to time, you may find it necessary to replace the SSL certificate file in use within your infrastructure. SSL certificates may exceed their useful lifetimes in several different ways, among them:

- The certificate will shortly reach its manufactured expiration time.
- You decide that you want to stop using a particular certificate, for reasons beyond the validity, duration, or current trust level of the certificate. For instance, perhaps you are going to change your master Certificate Authority in a month's time, and you want to switch all your clients to using a certificate derived from the new Certificate Authority. Since you had the foresight and energy to make this happen ahead of time, the old certificate remains valid and useable throughout the transition period.
- The private key on your GWMEE server has been compromised or stolen, or perhaps some upstream key related to your Certificate Authority has been so compromised.

We will classify the first two of those cases as "planned" invalidation of the certificate. We will classify the last case as an "unplanned" invalidation of the certificate. Your response to these two types of invalidation will be very different.

You might also just need to update the CRL file, if it expires but the certificate itself is still valid. In this case, you can just generate a new CRL and install it on your clients, without swapping in a new certificate as well. You will need to run `c_rehash` in the usual manner on each client after the new CRL is in place, to generate the usual necessary symlinks for the CRL to be recognized.

5.1 Revoking an SSL certificate (general discussion)

If a private key gets compromised or stolen, or the chain of trust starting with your Certificate Authority is otherwise compromised, you should revoke any related certificates so the clients can no longer use them. (Based strictly on their manufactured lifetimes, such certificates will otherwise still be valid.) Revoked certificates (e.g., those you wish to disallow because they were manufactured based on the server's compromised private key) are only known to each client to have been revoked because they are stored in a Certificate Revocation List (CRL) on that client. The client references its local CRLs when making a connection to the server. Thus in this situation, the CRL will need to be updated on each and every GDMA client, along with having a new, valid certificate installed.

CRL files are important
If the client is somehow repopulated with an old certificate and still has an out-of-date CRL, a man-in-the-middle (MITM) attack is possible. So for proper security, distributing an updated CRL file is just as important as distributing a new certificate file.

SSL certificate management is critical
Notwithstanding what we said in the note just above, GroundWork is not convinced that CRL files do the job they're supposed to, in preventing unwanted attacks. So it is best not to depend on them, and to ensure that you manage your SSL certificates properly on both the server and your GDMA clients. On the server, you should have only one certificate in play at any one time. On each client, keep only the current valid certificate that the server accepts, and no others except during server transition periods when you briefly have a new certificate installed on the client before the server is transitioned, and an old certificate still installed on the client after the server is transitioned.

The GDMA 2.3.2 client software does not insist on the use of CRL files. If any are present, they will be used, to whatever effect they have on the operation of the SSL libraries. If no CRL files are present, the GDMA client software will simply skip CRL processing.

To revoke a certificate, you must do several things. Details and proper sequencing are provided below this overview.

1. If the certificate to be revoked is your own, remove the certificate from the server, so client connections can no longer validate against it. (This step alone doesn't prevent a MITM attacker from using the certificate, though, which is why you should continue on and update the client CRL.)
2. If your own private server key has been compromised, you will need to generate a new one, and use it for all subsequent operations regarding maintenance of certificate and CRL files.
3. Put in place on the server a replacement certificate, as needed. In particular, if it is your own self-signed certificate that you are revoking, you will need to generate a new self-signed certificate before you can revoke the old one. See the commands below.
4. Add the revoked certificate to your master certificate revocation list (see detailed commands below for how to do this if you are using self-signed certificates). This is especially important if the certificate to be revoked arose from some third-party intrusion, in which case you do not necessarily have access to the intruder's server to remove the certificate there.
5. Tell the clients to use the new certificate (known as `server.crt` on your server), by installing your renamed and numbered copy on every client (for example, as `server_02.pem`, in the `gdma/certs/` directory).

6. Tell the clients not to use the revoked certificate, firstly by removing the old certificate file (e.g., `server_01.pem`) from every client, and also (if you choose to use CRLs) by distributing the updated certificate revocation list (e.g., `crl_02.pem`) to every client (also in the client's `gdma/certs/` directory). The latter step should block any attempt by a rogue server to use the revoked certificate.
7. On each client, run the `c_rehash` program as you did when initially setting up the client to use HTTPS. This will maintain the necessary symlinks to or copies of the certificate and revocation-list files.
8. You may then choose to stop and start GDMA on the client, so it will reinitialize itself using the new setup. (Strictly speaking, this should not be necessary, because the HTTPS connections are short-lived. Thus there should be no long-lasting connection you need to break by bouncing the client, and the next connection attempt should simply use the new setup, without further ado. Bouncing the client is simply precautionary.)

Commands used to revoke a certificate and to add the revoked certificate to your revocation list are shown in full context within the following two sections, on planned and unplanned certificate invalidation. The commands are sequenced to keep the bad old certificate around long enough to extract information from it for the revoking, and to have in place a good new certificate to use during the revocation processing. For sites using self-signed certificates, the commands result in an updated revocation list (e.g., `crl_02.pem`) that is distributed to your clients along with the new certificate (e.g., `server_02.pem`).

If you're curious about what certificates (if any) have been revoked, you can peek inside the CRL file using commands like this, naming the particular CRL file you are interested in (e.g., `crl_02.pem`):

```
cd /usr/local/groundwork/apache2/conf
/usr/local/groundwork/common/bin/openssl crl -noout -text -in crl_02.pem
```

With that general background, we now present the full series of commands you will need in the two major scenarios of planned and unplanned certificate invalidation.

5.2 Planned certificate invalidation

For a planned invalidation, you have time to make the transition, and should take the following steps:

1. Obtain or create a new server certificate (and perhaps a new private key preceding that, if desired), *but do not install those files yet on the server using the standard filenames which would place these files into active service*. You want your GDMA clients to continue to use their existing copies of the old certificate during the transition period, and for this to happen, the server must continue to use the old files. **Be sure to alter the target filenames in any commands you use to generate such files, so you do not overwrite the existing files.** In the sample commands below, we have altered the filenames in accordance with that dictum, but you must still watch out for filenames with embedded serial numbers, and make appropriate adjustments. Perform these steps on the GroundWork server, as the `nagios` user.

```
# Get to where the certificate files lie, to make these commands easier.
cd /usr/local/groundwork/apache2/conf

# First, copy the old certificate, keeping it around under an obvious
# name so it's clear what you're doing when you formally revoke it
# (that is, when you put it in your certificate revocation list).
cp -p server.crt server_old.crt

# Create a symlink to refer to the server key that will be used once
# the new certificate is in place.  Referring to this symlink simplifies
# the multiple cases we are covering here.
ln -s server.key server.key.future

# If desired, create a new private server key, using these commands.
# Otherwise, skip these commands and go directly to obtaining or
# creating a new server certificate.
rm server.key.future
/usr/local/groundwork/common/bin/openssl genrsa -out server.key.new 2048
ln -s server.key.new server.key.future

# If you obtain a new certificate from a Certificate Authority, install
# it in this directory on your server as the "server.crt.new" file.

# Otherwise (that is, if you do not obtain a new certificate from a
# Certificate Authority), create a new self-signed server certificate
# this way, adjusting the -days parameter (time until certificate
# expiration) as desired.  Here in this sample command, we specify a
# three-year lifetime.
/usr/local/groundwork/common/bin/openssl req -new -x509 \
    -key server.key.future -out server.crt.new -days 1095 \
    -set_serial `date +%s`

# Copy the new server certificate into a serial-numbered name that
# will be more useful later on, on your clients.  When you type this
# command, choose the next unused serial number in sequence at your
# site for such files.  For example, if server_01.pem already exists,
# you might type:
cp -p server.crt.new server_02.pem
```

2. Visit each and every GDMA client in turn, and take these steps:
    a. Install the new certificate (e.g., `server_02.pem`, in this example), leaving the old certificate (and revocation list, if any) still in place for the time being. This file will go into the `gdma/certs/` directory, right next to your existing certificate-related files.
    b. On a UNIX-like GDMA client platform, make sure the certificate file is owned by either `root` or the user that owns all the rest of the GDMA files (typically, `gdma`), and that the new file has only `644` permissions. (Logically, you want to do the same thing on Windows platforms. However, specifying the Windows commands to do so is beyond the scope of this document. At a minimum, you probably want to use "`attrib +r` *filenames*" to make the certificate-related files read-only.)
    c. After you install the new certificate file on each client, run `c_rehash` on the client to generate hash values for all the certificate-related files on the client, both old and new. This will set up the client for the actual transition on the server ("flag day").
        - On a Windows system, the commands would be like:

```
cd {path-to}\gdma\certs
..\..\common\bin\c_rehash
```

        - On a Linux or AIX system, the commands would be:

```
cd /usr/local/groundwork/gdma/certs
../../common/bin/c_rehash ./
```

        - On a Solaris system, the commands would be:

```
cd /opt/groundwork/gdma/certs
../../common/bin/c_rehash ./
```

3. When flag day arrives, perform the following actions on the GroundWork server. Except where noted, run these commands as the

`nagios` user.

a. If you will be distributing CRL files to your GDMA clients, generate a revised certificate revocation list that includes the old certificate. The process for doing so if you are getting your certificates from an outside source is beyond the scope of this article. The commands for doing so if you are using self-signed certificates are presented here.

```
# Get to where the certificate files lie, to make these commands easier.
cd /usr/local/groundwork/apache2/conf

# Extract details of the old certificate into your permanent store of
# revoked certificates (/usr/local/groundwork/common/openssl/index.txt,
# as specified in the openssl.cnf file in that same directory).
/usr/local/groundwork/common/bin/openssl ca -revoke server_old.crt \
    -keyfile server.key.future -cert server.crt.new

# Generate a new revocation list, one that will now contain a description
# of the certificate you just revoked (in addition to descriptions of any
# previously revoked certificates).  Modify the command shown here to set
# the -crldays parameter to whatever period makes sense, as noted earlier.
/usr/local/groundwork/common/bin/openssl ca -gencrl \
    -keyfile server.key.future -cert server.crt.new \
    -out crl_`cat ../../common/openssl/crlnumber`.pem -crldays 1095
```

The odd construction of the `-out` parameter in the last command will have created a serial-numbered file such as `crl_02.pem` as the new CRL file.

b. Shut down the GroundWork software before you change the certificate, as the change will disrupt ordinary operation.

```
su - root
service groundwork stop gwservices
service groundwork stop apache
exit
```

c. Put the new certificate (and the private key, if you are replacing that as well) into play with Apache.

```
# Get to where the certificate files lie, to make these commands easier.
cd /usr/local/groundwork/apache2/conf

# If you will continue to use the same private server key, just skip
# this command.  Otherwise, you created a new private server key in
# the first step.  Now is the time to put it into place, thereby
# destroying the old one, with the following command.
mv server.key.new server.key

Move the new certificate into place, thereby destroying the old one.
mv server.crt.new server.crt

# Clean up the scaffolding you created in earlier steps.
rm server.key.future
rm server_old.crt
```

d. Delete the old certificate from the Java keystore. **"gwservername" in the `-delete` command here must be the same name you used when you installed the old certificate in the keystore.** Each of the `keytool` commands will prompt for a password; by default, it is `changeit` .

```
# You can verify the existing "gwservername" value in your Java keystore
# by examining the output from the following command.
/usr/local/groundwork/java/bin/keytool -list \
    -keystore /usr/local/groundwork/java/jre/lib/security/cacerts \
    | fgrep `hostname | sed -e 's/\..*//'`

# Delete the old certificate from the Java keystore.  Modify the command
# shown here to substitute in the proper alias for "gwservername" on
# your system, exactly matching the server name in the certificate you
# are deleting.
/usr/local/groundwork/java/bin/keytool -delete -alias gwservername \
    -keystore /usr/local/groundwork/java/jre/lib/security/cacerts
```

e. Put the new certificate into the Java keystore. **As above, modify "gwservername" in the `-import` command here to mention your own server name.**

```
# Modify the command shown here to substitute in the proper alias for
# "gwservername" on your system, exactly matching the server name in
# the new certificate you are adding.
/usr/local/groundwork/java/bin/keytool -import -alias gwservername \
    -file /usr/local/groundwork/apache2/conf/server.crt \
    -keystore /usr/local/groundwork/java/jre/lib/security/cacerts
```

f. Restart the GroundWork software, to operate with the new certificate in play.

```
su - root
service groundwork start apache
service groundwork start gwservices
exit
```

4. Once the server has been so converted, go back to each and every GDMA client in turn, and take the following steps:
    a. Delete the old certificate (e.g., `server_01.pem`) from the `gdma/certs/` directory.
    b. If you are supporting CRL files on your clients, install the updated revocation list (e.g., `crl_02.pem`) in the `gdma/certs/` directory. You can leave older CRLs in place; there is no particular reason to delete them.
    c. If you are supporting CRL files on your clients, then on a UNIX-like platform, make sure the new CRL is owned by either `root` or the usual GDMA user (typically, `gdma`), and that it has only 644 permissions. (Logically, you want to do the same thing on Windows platforms. However, specifying the Windows commands to do so is beyond the scope of this document. At a minimum, you probably want to use "`attrib +r` *filenames*" to make the certificate-related files read-only.)
    d. Run `c_rehash` to establish hashes for all the presently installed files, using the commands shown above for this type of platform.

5.3 Unplanned certificate invalidation

For an unplanned invalidation, trust has been lost, so you have a security crisis to deal with, and your response must be immediate and severe. The certificate whose chain of trust has been compromised must be immediately revoked, so it is no longer used by your clients. In this situation, you must take the following steps, as the `nagios` user:

1. Obtain or create a new certificate (and perhaps a new private key preceding that, if appropriate, depending on what has been compromised). If obtaining a new certificate from an external Certificate Authority may take some time, you might consider using a self-signed certificate on a temporary basis, so you can stop using the old certificate immediately but still allow browser-based access to GroundWork Monitor until the situation can be fully corrected. Perform these steps on the GroundWork server.

```
# Get to where the certificate files lie, to make these commands easier.
cd /usr/local/groundwork/apache2/conf

# First, copy the bad certificate, keeping it around under an obvious
# name so it's clear what you're doing when you formally revoke it
# (that is, when you put it in your certificate revocation list).
cp -p server.crt server_bad.crt

# Create a symlink to refer to the server key that will be used once
# the new certificate is in place.  Referring to this symlink simplifies
# the multiple cases we are covering here.
ln -s server.key server.key.future

# If necessary (e.g., if the private server key has been compromised),
# or if otherwise desired, create a new private server key, using these
# commands.  Otherwise, skip these commands and go directly to obtaining
# or creating a new server certificate.
rm server.key.future
/usr/local/groundwork/common/bin/openssl genrsa -out server.key.new 2048
ln -s server.key.new server.key.future

# If you obtain a new certificate from a Certificate Authority, install
# it in this directory on your server as the "server.crt.new" file.

# Otherwise (that is, if you do not obtain a new certificate from a
# Certificate Authority), create a new self-signed server certificate
# this way, adjusting the -days parameter (time until certificate
# expiration) as desired.  Here in this sample command, we specify a
# three-year lifetime.
/usr/local/groundwork/common/bin/openssl req -new -x509 \
    -key server.key.future -out server.crt.new -days 1095 \
    -set_serial `date +%s`

# Copy the new server certificate into a serial-numbered name that
# will be more useful later on, on your clients.  When you type this
# command, choose the next unused serial number in sequence at your
# site for such files.  For example, if server_01.pem already exists,
# you might type:
cp -p server.crt.new server_02.pem
```

2. If you will be distributing CRL files to your GDMA clients, generate a revised certificate revocation list that includes the old certificate. The process for doing so if you are getting your certificates from an outside source is beyond the scope of this article. The commands for doing so if you are using self-signed certificates are presented here; run these commands on the GroundWork server, as the `nagios` user.

```
# Get to where the certificate files lie, to make these commands easier.
cd /usr/local/groundwork/apache2/conf

# Extract details of the bad certificate into your permanent store of
# revoked certificates (/usr/local/groundwork/common/openssl/index.txt,
# as specified in the openssl.cnf file in that same directory).
/usr/local/groundwork/common/bin/openssl ca -revoke server_bad.crt \
    -keyfile server.key.future -cert server.crt.new

# Generate a new revocation list, one that will now contain a description
# of the certificate you just revoked (in addition to descriptions of any
# previously revoked certificates).  Modify the command shown here to set
# the -crldays parameter to whatever period makes sense, as noted earlier.
/usr/local/groundwork/common/bin/openssl ca -gencrl \
    -keyfile server.key.future -cert server.crt.new \
    -out crl_`cat ../../common/openssl/crlnumber`.pem -crldays 1095
```

   The odd construction of the `-out` parameter in the last command will have created a serial-numbered file such as `crl_02.pem` as the new CRL file.

3. In this step, you will switch the certificate (and the private key, if you are replacing that as well) on the server. This will immediately stop

all clients from being able to access the server for configuration-related information. GDMA clients should continue to monitor their respective environments and report back results from this point forward, but their configurations will be frozen for the time being. Ordinary browser clients will probably need to have their local certificate stores manually flushed to drop the old certificate, using whatever preference-editing mechanisms are provided by the browser. Perform the following actions on the GroundWork server. Except where noted, run these commands as the `nagios` user.

a. Shut down the GroundWork software before you change the certificate, as the change will disrupt ordinary operation. (You cannot even remove the bad certificate before this step, as having it missing would prevent you from stopping Apache now!)

```
su - root
service groundwork stop gwservices
service groundwork stop apache
exit
```

b. Put the new certificate (and the private key, if you are replacing that as well) into play with Apache.

```
# Get to where the certificate files lie, to make these commands easier.
cd /usr/local/groundwork/apache2/conf

# If you will continue to use the same private server key, just skip
# this command.  Otherwise, you created a new private server key in
# the first step.  Now is the time to put it into place, thereby
# destroying the old one, with the following command.
mv server.key.new server.key

Move the new certificate into place, thereby destroying the old one.
mv server.crt.new server.crt

# Clean up the scaffolding you created in earlier steps.
rm server.key.future
rm server_bad.crt
```

c. Delete the old certificate from the Java keystore. **"gwservername" in the `-delete` command here must be the same name you used when you installed the old certificate in the keystore.** Each of the `keytool` commands will prompt for a password; by default, it is `changeit` .

```
# You can verify the existing "gwservername" value in your Java keystore
# by examining the output from the following command.
/usr/local/groundwork/java/bin/keytool -list \
    -keystore /usr/local/groundwork/java/jre/lib/security/cacerts \
    | fgrep `hostname | sed -e 's/\..*//'`

# Delete the old certificate from the Java keystore.  Modify the command
# shown here to substitute in the proper alias for "gwservername" on
# your system, exactly matching the server name in the certificate you
# are deleting.
/usr/local/groundwork/java/bin/keytool -delete -alias gwservername \
    -keystore /usr/local/groundwork/java/jre/lib/security/cacerts
```

d. Put the new certificate into the Java keystore. **As above, modify "gwservername" in the `-import` command here to mention your own server name.**

```
# Modify the command shown here to substitute in the proper alias for
# "gwservername" on your system, exactly matching the server name in
# the new certificate you are adding.
/usr/local/groundwork/java/bin/keytool -import -alias gwservername \
    -file /usr/local/groundwork/apache2/conf/server.crt \
    -keystore /usr/local/groundwork/java/jre/lib/security/cacerts
```

e. Restart the GroundWork software, to operate with the new certificate in play.

```
su - root
service groundwork start apache
service groundwork start gwservices
exit
```

4. Visit each and every GDMA client in turn, and take these steps:
    a. Delete the old certificate (e.g., `server_01.pem`) from the `gdma/certs/` directory.
    b. If you are supporting CRL files on your clients, install the updated revocation list (e.g., `crl_02.pem`) in the `gdma/certs/` directory. You can leave older CRLs in place; there is no particular reason to delete them.
    c. Install the new certificate (e.g., `server_02.pem` in this example) in the `gdma/certs/` directory.
    d. On a UNIX-like platform, make sure the new files are owned by either `root` or the usual GDMA user (typically, `gdma`), and that they have only 644 permissions. (Logically, you want to do the same thing on Windows platforms. However, specifying the Windows commands to do so is beyond the scope of this document. At a minimum, you probably want to use "`attrib +r` *filenames*" to make the certificate-related files read-only.)
    e. Run `c_rehash` to establish a current set of hashes, using the commands shown above for this type of platform.

During an unplanned invalidation, the GDMA clients will continue to report monitoring results for some period, set by the `Poller_Pull_Failure_Interval` as configured on each client. This parameter is set to `86400` (seconds), or one day, in the standard distributed `gdma_auto.conf` file. For details, see the description of this option in the GDMA Advanced page of the Bookshelf. This is how long you will have to update all of your clients with a new certificate before they go dark from a monitoring-results standpoint.

Be Prepared
Because of this limited time period, you must be ready with a procedure to generate or procure a new certificate, and update **all** your GDMA clients quickly, well in advance of any emergency situation. You may extend this period (preferably in advance of an actual emergency) by changing the `Poller_Pull_Failure_Interval` setting. This can be especially important on GDMA clients which for some reason are hard to reach for maintenance purposes.

6.0 Troubleshooting an HTTPS Connection

Very little information is provided by the HTTPS-support software when the certificate and certificate revocation list do not work as intended. If you enable logging in the GDMA poller, via the `Enable_Local_Logging` option in the `gdma_auto.conf` file, a small amount of information may be gleaned. Here is an example, with the lines wrapped for easier viewing here:

```
[Thu Apr  4 16:55:52 2013] Failed to get https://gwserver/gdma/gwmon_linuxgdma.mydomain.com.cfg --
    500 Can't connect to gwserver:443 (certificate verify failed)
[Thu Apr  4 16:55:52 2013] Failed to get https://gwserver/gdma/gwmon_linuxgdma.cfg --
    500 Can't connect to gwserver:443 (certificate verify failed)
[Thu Apr  4 16:55:52 2013] Failed to fetch the host config file.
[Thu Apr  4 16:55:52 2013] ERROR:  Auto-registration request was not processed;
    HTTP/S response code = 500.
[Thu Apr  4 16:55:52 2013] XML response was:
Can't connect to work.groundwork.groundworkopensource.com:443 (certificate verify failed)

LWP::Protocol::https::Socket: SSL connect attempt failed with unknown error error:
    14090086:SSL routines:SSL3_GET_SERVER_CERTIFICATE:certificate verify failed at
    /usr/local/groundwork/perl/lib/site_perl/5.8.9/LWP/Protocol/http.pm line 51.
```

The phrase "certificate verify failed", repeated several times as shown above, is as much detail as you're going to get. So it's important to follow the certificate-install instructions carefully on every GDMA client. The main things to check are:

- Is the certificate file (e.g., `server_01.pem`) in place on the client, in the `gdma/certs/` directory, with restricted ownership and permissions?
- If you are supporting CRL files on your clients, is the certificate revocation list (e.g., `crl_01.pem`) also in place on the client, in the `gdma/certs/` directory, with restricted ownership and permissions?
- Is the certificate file up-to-date with the current copy on the server?
- If you are supporting CRL files on your clients, is the certificate revocation list file up-to-date with the current copy on the server?
- Does the certificate reflect the actual hostname used by the server?
- Does the `Target_Server` option in the client `gdma_auto.conf` file reflect the exact same form of the hostname as is present in the certificate?
- Have the appropriate hash symlinks or copies been made in the `gdma/certs/` directory, using the `c_rehash` program?

If all of those things check out, perhaps you really do have a Man-In-The-Middle attack going on, and the client *should not* be validating the certificate under that condition! (Proving that such a situation is happening is beyond the scope of this document. Consult your local network folks.)