

HOW TO USE APPGATE

INTRODUCTION HOW TO UTILISE THE
POWER OF APPGATE.



TABLE OF CONTENTS

Introduction	3
Client Check/Client Command	4
Customised Web page and Interface	8
Customer designed installation packages	12
Creating Personal Networks using Multiple Connections	14
Establish SSL/HTTPS support in the AppGate web server	17
Repack Personal Firewall with a new rules.conf	19
Using starturl frame or javascript to launch applet.	20

Introduction

The AppGate Security Server gives the administrator many opportunities to design and set up the system in such a way that supports the organisations needs and the users wishes.

This manual is intended to help and give ideas to system administrators so they can take advantage of the full power of their AppGate installation

For more information about AppGate Security Server please see the AppGate Manual.

Client Check/Client Command

How to check the connecting device

Summer 2005

INTRODUCTION

The Client Check is a feature which was designed to verify that a client computer is in a certain state before it is allowed to access things protected by the AppGate Server.

The implementation of this Client Check is very general and can probably be used for other purposes as well.

The way it works is that the administrator prepares a file for each client platform. The file should be an executable suitable for that platform. E.g. for the Windows platform it could be an exe file or a bat file.

This executable file is then uploaded to the AppGate Server using the AppGate Console. Navigate to "System Maintenance" -> "Client Check Commands".

Below is an example that use the check.exe (downloadable link). It is a Windows executable binary that can be used. It should be able to serve most Client Check purposes for Windows:

Usage: check.exe

- fileexists <file>
- filesizeis <file> <size>
- filesizelarger <file> <size>
- filesizesmaller <file> <size>
- fileversion <file>
- productversion <file>
- md5sum <file>
- sha1sum <file>
- regexists <root> <key> <value>
- regprint <root> <key> <value>

-process <name>

-isadmin

-issvcrunning <name>

-querysvc <name>

-downloadto <url> [-hash <hash>] <file>

-downloadandrun <url> [<hash>]

-downloadandrunasarg <url> [-hash <hash>] <cmd>

-downloadanddo <url> <op> [<hash>]

where <op> is one of: edit, explore, find, open or print

and <hash> is on the form {md5,sha1}:XXXXX...

<cmd> may contain #f which will be expanded to the file name
of the downloaded copy of the file to which the url points

-filedateis <file> <date>

-filedatenewer <file> <date>

-filedateolder <file> <date>

where <date> is on the form yyyyymmdd[hh[mm[ss]]]

-rfiledatenewer <file> <rtime>

-rfiledateolder <file> <rtime>

where <rtime> is on the form nn['d'|'h'|'m']

Examples:

-fileexists %windir%/notepad.exe

-regexists HKEY_LOCAL_MACHINE "Software/Microsoft/Internet Explorer" Build

-regprint HKLM "Software/Microsoft/Internet Explorer" Build

-process mstask.exe

-filedateis c:/windows/notepad.exe 20040401

-filedatenewer c:/windows/notepad.exe 200404012359

-filedateolder %windir%/notepad.exe 20040401235959

-rfiledatenewer c:/windows/notepad.exe 10d

-rfiledateolder #REG[HKLM,Software/ACME Software,SomeValue] 10h

Observe: that the above examples all use "/" (slash) and not "\" (backslash). This is because the AppGate Console doesn't handle "\" (backslash) very well. The check.exe will handle this in an appropriate way.

HOW TO

1. Fetch the above mentioned check.exe file to your PC.
2. Connect the AppGate Console to your AppGate Server
3. Use the "AppGate Console -> System Maintenance -> Client Check Commands -> Upload..." to upload the check.exe file to the AppGate Server.
4. Do "Add" and set the following:
 - Attribute to "testattr"
 - Chose Platform "pc.*.*"
 - Chose Command "check.exe"
 - Set Arguments to "-fileexists c:/TST.TXT"
 - Press OK
5. Go to "Administration" -> "Access Rules" and chose:
 1. < New access rule >
 2. Set Name to "test"
 3. Set Description to "test"
 4. Chose "Client check...: and:
 - Give Attribute "testattr"
 - Set Regular Expression to "yes". The Regular Expression is a string that is compared to the value of "testattr". In our example the value comes from the output of check.exe and will be equal to "yes" if the check.exe finds the c:/TST.TXT file.
 - Press OK
 - Save
6. Go to any Role you like to test this on:
 - Chose "test" as Access rule

- Save
- 7. Push

Now The AppGate Server is prepared to make Client checks. In This example it will check if a file named "c:/TST.TXT" exists.

Do the following to try it out:

1. On a client PC install the AppGate Client 6.1 or later
2. Bring up a window with a continuous AppGate log and watch the log while you do the rest of the steps.
3. Log on as a user who has access to the Role you prepared above.
4. While doing this the following should be visible in the log:
5. Client check attribute: 'testattr' = 'no'
6. Now on the client PC create the file:
7. c:/TST.TXT
8. Log off and log on again.
9. >Now the following should be visible in the log:
10. Client check attribute: 'testattr' = 'yes'
11. and the Role should be available to the user.

Customised Web page and Interface

How to make make the AppGate web server customised.

Summer 2005

INTRODUCTION

This document describes how to create a customized web page on the AppGate web server. Such a web page can serve as the normal entry point for users and assist in a corporate roll out.

This example will be based on a template web page that will give the user only two options:

1. Use the AppGate Applet for access to one simple application, like Outlook Web Access.
2. Use the full AppGate Client packaged together with the IP tunnelling driver and a configuration file for a more full access type of experience.

HOW TO

Step 1 - getting the files

1. Create a working folder on your PC.
2. Fetch this Zip archive which contains a template web page file structure.
3. Unzip the content into you working folder. It should give you the following files and sub folders:
 - applet
 - applet.conf
 - agpkg.exe
 - index.html
 - images
 - style

Step 2 - customize the content

- You can now edit the index.html file to change the look and feel to your liking.
- You may replace the client package agpkg.exe with another one. Please see the note on Building your own installation packages to create a customized package.
- If you are deploying the AppGate Applet you may now edit the applet.conf file to suite your scenario.
- The applet.conf file is an ordinary AppGate client configuration file. The only difference is that it resides on the AppGate server and is fetched each time a user launches the applet from this customized web page.
- In order to streamline the whole user experience you would probably like to set up special AppGate Roles and Services that should be available only when the user is launching the applet from this web page. In order for the AppGate server to detect that it is this customized web page that originated the connection you should tag this applet with a special "ident" value.
- To do this, edit the applet.conf file and find the row that says with ident= applet . Change the value applet to e.g. applet_special.
- The applet.conf can contain a lot of other settings for the applet and you may want to go through it to make more changes.

Step 3 - Upload the files to the AppGate

- Create a new directory on the AppGate.

Example:

1. Use the AppGate Client and connect as admin.
2. Issue "Shell on appgate"
3. Become root using the su command and the root password.
4. `cd /var/opt/appgate`
5. `mkdir webroot-customized`
-
- Transfer all files and sub folders from your working folder to the new directory in the AppGate webroot. You can do this in two ways:
 - Create the sub directories manually using the command:
`mkdir applet images style`

in your webroot-customized directory on the AppGate and then use the AppGate Console to transfer the all the files one at a time. (System Maintenance->File transfer)

- Or you may want to Zip the working folder, transfer the Zip archive using the AppGate Console and then use the unzip command on the AppGate to create all the content in one sweep. You should be located in the newly created webroot-customized directory when unzipping. A free zip tool for Win* is available from www.7-zip.org

-

- Tell the web server where to find the web pages by giving the following commands as root on the AppGate server:

1. `ag_cfggetset -s ag_httpd.root /var/opt/appgate/webroot-customized`
2. `/etc/init.d/ag_httpd restart`

- If you want to reuse files or whole directories from the standard webroot directory the preferred way is to use soft links. E.g. to be able to refer to the standard client `agclient.exe` do as:

- `# cd /var/opt/appgate/webroot-customized`
- `# ln -s /var/opt/appgate/webroot/clients/win32/agclient.exe agclient.exe`

-

- It is also possible to simply copy the desired files. The benefit of soft links is that after any future upgrade the new and upgraded files will be automatically referred.

Step 4 - Setting up an Access rule

If you are about to use the AppGate Applet for one specific application and you have tagged the `applet.conf` file with the `ident= applet_special` you can now create an Access rule that detects this applet. The expression may look like this:

```
platform {pc.*.applet_special}
```

The above Access rule can be set on a Role to make it available only in the case a user is connecting using the applet on the customized web page. You can also create an inverse of this Access rule and put it on other Roles to disable access to those in this case. Such an Access rule may look like:

```
not platform {pc.*.applet_special}
```

Step 5 - Use it

To access the new web page on the AppGate Server you may now point your browser to:

<http://your.appgate.com/index.html>

Customer designed installation packages

Installation designed tool kits

Updated Dec 2005

INTRODUCTION

With the AppGate system it is possible to design the users experience in way that makes it easier for user based on the organisation special needs.

This information describes how to repackage the AppGate Client together with a agclient.conf file and the IP-tunneling driver. The result will be a file called agpkg.exe that may help in streamlining the user experience.

To be able to do there is need a Windows (2K or XP) machine.

HOW TO.

The following process describes how to design:

- Create a new empty folder - a build folder.
- Put a agpkg.nsi file in the build folder. You'll find it here: agpkg.nsi .
- Put a suitable icon.ico file in the build folder. You can use this one icon.ico . (It is the icon shown for the resulting installation .exe-file)
- Put the template agclient.properties file in the build folder. Typically you can copy such a file from your "C:\Documents and Settings\home folder\appgate\version\agclient.conf" Using this file will give each user the same settings as yours as default. You may also take look at it with e.g. word-pad to make sure it contains only the settings you like. Here is an Excerpt from the client guide on the parameters and syntax for agclient.conf
- Fetch an agclient.exe and an agiptd.exe file and put them into the build folder. You can fetch them from your AppGate's built in web server to get the correct version.
- Install Nullsofts Install System, NSIS which can be downloaded from <http://sourceforge.net/projects/nsis/>. Version 2.0 or later is needed.

- Change directory, possibly using the Windows explorer, into the build folder. Using the right click menu on the agpkg.nsi file you should be able to "Compile NSIS Script".

The last step should create a new agpkg.exe in the build folder. This file could then be uploaded to the AppGate and moved into a suitable position in the /var/opt/appgate/webroot directory or deployed by other means to the client machines.

The installer section in the last 20 lines in the agpkg.nsi is already prepared for other packaging combinations. Simply comment/uncomment the relevant lines to include or exclude other files.

Creating Personal Networks using Multiple Connections

An example how AppGate can reduce firewall costs

Summer 2005

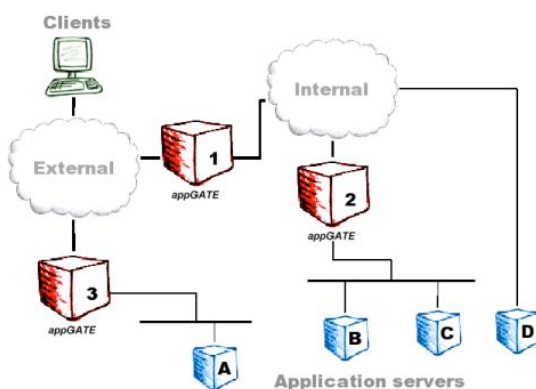
INTRODUCTION

Most VPN solutions will only let a user connect to one VPN server at a time. This is fine for several applications but there are situations where it can be very useful to have users connecting to several AppGate Security Servers at the same time.

A Typical Scenario

A typical scenario is depicted below. Here we have a Client that is situated on an external network, e.g. the Internet. For normal remote access he will then connect to the AppGate Security Server 1. This will possibly enable access to the Application Server D as it is the only server that is connected directly to the Internal network.

However in this scenario another AppGate Security Server (2) is also used. It protects the two application servers B and C. A good reason for a topology like this could be that the B and C servers are extra sensitive and need controlled access even from users directly connected to the Internal network.



With an AppGate solution this presents no problem as it is possible for the Client to make another connection to the AppGate Security Servers 2. This connection will then be handled within the first secure tunnel that is part of the first connection. It is then necessary to ensure that the first connection includes an appropriate service with IP-access components that allows the second connection to take place through the

first.

The user may of course connect to even further AppGate Security Servers. In our scenario it may be that the user needs to connect to AppGate Security Server 3 which could represent a system located at another physical site.

Personal Networks

The above way of letting users utilize many AppGate Security Servers at the same time will create a type of network that can be called a Personal Network. The word Personal is correct because it is actually set up by the user and it is the users credentials that will allow it to be set up. The AppGate system also enforces the actual traffic to be originating from the users own personal computer - so it does not allow traffic to flow between the above networks just because a user have access to all of them.

Reduce high Firewall costs

Many organizations spend large amount of time on administering their firewalls. E.g. there are often weekly meetings on what rules to change or not. After that follows complex and error prone implementations of the rules. An to add stone to burden this firewall circus is often disturbing the whole network infrastructure.

In contrast, the AppGate way of modeling network security has a few advantages over a pure firewall strategy.

- Fewer firewalls and fewer tricky rule sets are needed as all user access is based on individual rights.
- Because of this, policy discussions can be reduced to a mere establishing of facts on whether a certain user should have access to particular system or not. Lengthy firewall discussions regarding which ports, protocols, hosts or networks should be opened up can be reduced.
- Less error prone and time consuming firewall administration.
- More liberal access policy allowing more users to access the systems they need to access, regardless of where they are located.
- Improved security as there are no or fewer unauthenticated access channels.

Firewalls may still be needed to let servers communicate with servers, but user access is best served with a user access system such as the AppGate Security Server.

Personal Networks

The above way of letting users utilize many AppGate Security Servers at the same time will create a type of network that can be called a Personal Network. The word Personal is correct because it is actually set up by the user and it is the user's credentials that will allow it to be set up. The AppGate system also enforces the actual traffic to be originating from the user's own personal computer - so it does not allow traffic to flow between the above networks just because a user has access to all of them.

Reduce high Firewall costs

Many organizations spend large amounts of time on administering their firewalls. E.g. there are often weekly meetings on what rules to change or not. After that follows complex and error-prone implementations of the rules. An added stone to the burden of this firewall circus is often disturbing the whole network infrastructure.

In contrast, the AppGate way of modeling network security has a few advantages over a pure firewall strategy.

- Fewer firewalls and fewer tricky rule sets are needed as all user access is based on individual rights.
- Because of this, policy discussions can be reduced to a mere establishing of facts on whether a certain user should have access to a particular system or not. Lengthy firewall discussions regarding which ports, protocols, hosts or networks should be opened up can be reduced.
- Less error-prone and time-consuming firewall administration.
- More liberal access policy allowing more users to access the systems they need to access, regardless of where they are located.
- Improved security as there are no or fewer unauthenticated access channels.

Firewalls may still be needed to let servers communicate with servers, but user access is best served with a user access system such as the AppGate Security Server.

Establish SSL/HTTPS support in the AppGate web server

How to set up SSL with the AppGate web server.

Summer 2005

HOW TO

1. Install OpenSSL. This can be done on either the AG server or on another machine. Source code is provided on AppGate CD. For this example the OpenSSL 0.9.5a-beta1 20 Mar 2000 was used. For OpenSSL errors and problems please go to the OpenSSL website, www.openssl.org.)
2. Generate a CSR (Certificate Request) to be submitted to your CA. (We used Thawte (www.thawte.com) as CA since they have the ability to generate testcertificates for free. The testcerts are valid for 21 days, which should be long enough for testing.)
3. Do the following to generate the CSR:
“openssl genrsa 512/1024 > appgateserver.my.domain.key”
Important!! Use no passphrase!! (ie no -des3)
“openssl req -new -key appgateserver.my.domain.key > appgateserver.my.domain.csr”
4. The file “appgateserver.my.domain.csr” is your CSR.
5. While generating the CSR you will be asked lots of questions. Enter appropriate answers. The CN or “YOUR name” should be appgateserver.my.domain. (Of course appgateserver.my.domain should be replaced with the appropriate info.)
6. (Your CA should be able to provide you with more details about this process. Use same info as for Apache-SSL. Thawte provides info for this here.)
7. Submit the CSR to your CA, together with all other required documentation, info and payment.
8. You should receive your certificate from the CA. Make sure that it is in the correct format. (Apache-OpenSSL compatible format should work. It is X509 format, base64 encoded. Thawte gives info about it here.)

9. Transfer the key file (appgateserver.my.domain.key) and the certificate (lets call it "appgateserver.my.domain.crt" to the AppGate server.
10. Concatenate the two files. Ie:
"cat appgateserver.my.domain.crt appgateserver.my.domain.key >
appgateserver.my.domain.all"
11. Point out this file in the /var/opt/appgate/conf/ag_httpd.conf file:
certfile="/var/opt/appgate/certs/appgateserver.my.domain.all"
12. Restart ag_httpd by running "/etc/init.d/ag_httpd restart".
13. You are now done! The AppGate webserver should begin listening on port 443 (SSL port) as well as port 80.
14. By default though, port 80 will be used by the clients. To remedy that, change the links to use "https" instead of "http". Ie: https://appgateserver.my.domain/.

Repack Personal Firewall with a new rules.conf

Package the AppGate Personal Firewall with a custom default
rule set.

Summer 2005

HOW TO

1. Use the AppGate Personal Firewall CD.
2. Install the repackaging tool named `agpfw_pkgtool.exe`
3. Open a command prompt and move to the directory where you installed the tool. Typically:

`C:\Program files\AppGate\Pfw Packaging Tool`

4. Create or copy the desired `rules.cfg` to this directory.
5. Run the `buildpkg.bat` file with arguments like this:

```
buildpkg.bat -d rules.cfg -o pfw.exe
```

This will take your `rules.cfg` file and bundle it with the Personal Firewall and output the file named `pfw.exe`.

The file `pfw.exe` is the resulting and installable Personal Firewall which may now be distributed to the users.

Using starturl frame or javascript to launch applet.

Starting the AppGate Applet in an Popup window or in a Frame.

Summer 2005

INTRODUCTION

Here are two examples of how one can modify how the applet is started. The aim is to have the Applet running in some sort of separate window that can stay on the users desktop for the life of the session.

HOW TO

JavaScript Popup

To use JavaScript to pop up a separate window to hold the applet do the following:

1. On the AppGate Server edit the `/var/opt/appgate/webroot-dist/index.html-dist`
2. Change the two links that are referring the applet/applet.html page to: `<AHREF=javascript:Launch('applet/applet.html')>`
3. Please observe that changes made in the webroot-dist has to be propagated to the actual webroot directory, this is done automatically now and then or you can force it by running `ag_httpd_update`.
- 4.
5. Add the following just before the `</body>`: `<SCRIPT LANGUAGE="JavaScript"> <!-- Begin function Launch(page) { OpenWin = this.open(page, "test", "toolbar=no,menubar=no,location=no,scrollbars=no,resizable=yes,width=550,height=250"); } </SCRIPT>`
6. In the server-block of the applet.conf file you can then use something like this:
7. `starturl= http://localhost:81 starturl_frame= _blank`

8. The `_blank` value for `starturl_frame` will instruct the browser to open the `http://localhost:81` URL in a new blank browser window.

In the above example the Applet will run in a new small window that the above small JavaScript program will open. The `applet/applet.html` is the web page that will be displayed in this window and it would probably be a good idea to modify it a bit to display clear message to the user to say "Don't shut this window down" or something similar.

As soon as the user have been authenticated the AppGate Applet will use the `starturl` and `starturl_frame` to popup another browser window (`_blank`) and and instruct the browser to use that window to load the `http://localhost:81` URL. Obviously, to have this URL to work an appropriate AppGate config must be in place but that is outside the scope of this Application Note.

Using Frames

Change directory to `/var/opt/appgate/webroot`

In `index.html` change the the `href` to the following:

```
<a href=applet/frameset.html>
```

Observe that the `index.html` is sourced from `../webroot/index.html-dist` and it may be wise to actually change it there and do a `ag_httpd_update` to have take effect.

```
Create a corresponding applet/frameset.html: <!DOCTYPE HTML PUBLIC
"-//W3C//DTD HTML 4.0 Frameset//EN"> <HTML> <HEAD> <TITLE>A sim-
ple frameset document</TITLE> </HEAD> <FRAMESET rows="20%, 80%">
<FRAME src="applet.html" name=topframe> <FRAME src="testpage.html"
name=botframe> </FRAMESET> </HTML>
```

Add into the server-block in `applet.conf`:

```
starturl= http://localhost:81 starturl_frame= botframe
```

The above `http://localhost:81` serves just as an example of an URL that should go down into a port-forward and reach the desired application web server.

In `applet/applet.html`:

```
<PARAM NAME="configurl" VALUE="../applet2.conf">
```

Create `applet/testpage.html`:

```
"This page will hold the application"
```