GAA-Apache User's Manual

Author: Li Zhou
Email: zhou@isi.edu
Last Updated: 01 SEP 2003

[PART I] INSTALLATION & CONFIGURATION

(One) How To Install The GAA-Apache On Linux

1. Download the file gaa-httpd.tar.gz onto local directory /usr/src and unzip it:

cd /usr/src
wget http://www.isi.edu/gost/info/gaaapi/source/gaa-httpd.tar.gz
tar xzvf gaa-httpd.tar.gz

Then a new directory "httpd-2.0.47" will be extracted under /usr/src.

2. Make sure that the XML and LTDL libraries have been installed and the following libraries files exist:

ls /usr/lib/libxml2.so.2
ls /usr/lib/libltdl.so

If not, install them or make symlinks from the same share libraries with different file name. Also, these two libraries are also available under "/usr/src/httpd-2.0.40/gaa-api/lib". We could just copy them to "/usr/lib".

3. Go into the apache's source directory and install GAA-Apache Web Server:

cd httpd-2.0.47
./configure
make
make install

The default installation directory for GAA-Apache is /usr/local/apache2. If the user want to install it under another directory, add "--prefix [user-dir]" after the command "./configure".

(Two) How To Configure The GAA-Apache

1. Modify the Apache's Configuration File

Edit the file "/usr/local/apache2/conf/httpd.conf". Here's a typical example of GAA-Apache configuration:

<Directory "/usr/local/apache2/htdocs">
Indexes Includes FollowSymlinks SymLinksifOwnerMatch ExecCGI Multiviews
Options Indexes FollowSymlinks
AllowOverride None
Order allow,deny
Allow from All
Deny from 127.0.0.0/255.0.0.0
AuthName "Protected Files"
AuthType Basic
AuthUserFile /usr/local/apahce2/conf/user.conf
AuthGroupFile /usr/local/apahce2/conf/group.conf
Require valid-user
Satisfy All
EaclFile /usr/local/apache2/conf/1.eacl
GaaOrder eacl,apache expand
</Directory>

Besides the directives defined by Apache Web Server, we have two extra directives for the GAA-API.

The first directive *EaclFile* provides the pathname of local policy file that will be applied to this domain.

- The second directive *GaaOrder* illustrates the relationship between Apache's traditional policy and EACL (Extensive Access Control Language) policy used by GAA-API. Here are the meanings of all possible values.

[Parameter 1]	
eacl,apache	Evaluate EACL policy first.
apache,eacl	Evaluate Apache's traditional policy first.

[Parameter 2]	
expand	"OR" to the following policies.
narrow	"AND" to the following policies.
exact	Stop evaluation, return current answer.

* Since GAA-API is calling the Apache's modules to check the authentication of users and groups. Even when the Apache's traditional policy is disabled by *"GaaOrder eacl,apache exact"*, we should still provides the directives of *AuthName*, *AuthType*, *AuthUserFile*, *AuthGroupFile* and *Require* as well. However, the user/group list given by the directive *Require* does not applies to GAA-API, which uses it own user/group list defined in EACL policy.

2. Define the GAA-Apache's EACL Policy

Edit the file referred by the directive "EaclFile" and customize your own policy. Here's an example of the policy file (/usr/local/apache2/conf/1.eacl). A detailed specification of the EACL policy is available in PART II.

eacl_mode 2

pos_access_right apache "read"
pre_cond_access_host apache "10.0.0.0/255.0.0.0 OR 128.9.0.0/16"
pre_cond_access_time local "05/01/2002-06/30/2003 MON-FRI 8am-11:30am,1:30pm-5pm"
pre_cond_access_user apache "winspring, hellene, file-owner"
rr_cond_append_log syslog "on:success/#[user.ip] is granted"

pos_access_right apache "read,execute"
pre_cond_access_host apache "isi.edu"
pre_cond_access_group apache "group-owner, me"
rr_cond_email_notify apache "on:failure/root@localhost"

***3. Defining Multiple Domains (optional)**

The policy applies to the specific domain (directory or group of files) that's defined by directive <Directory> or <Files>. Defining multiple domains with <Directory> and <Files>, we could apply different GAA policy files for different pages or resources. For instance:

<Directory "/usr/local/apache2/htdocs/mypages">
<Files "a.html">
EaclFile /usr/local/apache2/conf/1.eacl
GaaOrder eacl,apache expand
</Files>
<Files "b.html>
EaclFile /usr/local/apache2/conf/2.eacl
GaaOrder eacl,apache expand

</Files>
<Directory>

Besides defining in Apache's main configuration file (/usr/local/apache2/conf/httpd.conf), we could also define the domain in the per-directory configuration file (.htaccess), which applies to the current directory that ".htaccess" lies in.

(Three) How To Run GAA-Apache

1. We could run the following command to start, stop or restart the GAA-Apache server:
   /usr/local/apache2/bin/apachectl start
   /usr/local/apache2/bin/apachectl stop
   /usr/local/apache2/bin/apachectl restart

2. The log file about the running status of GAA-Apache is stored on:
   /usr/local/apache2/logs/error_log

[PART II] EACL POLICY SPECIFICATION

The security policy of GAA-API is specified via EACL (Extensive Access Control Language). In this part, we will further introduce the format of EACL policy file and the currently supported conditions (either the constrains checked by the policy or the actions performed by the policy).

(One) Format of EACL File

Here's the Backus-Naur Form specification of the EACL policy file:

EaclFile ::== ["mode" "0"| "1" | "2"] { Policy }
Policy ::== PositiveAccessRight { Condition }
| NegativeAccessRight { Condition }
PositiveAccessRight ::== "pos_access_right" AccessAuthority AccessValue
PositiveAccessRight ::== "neg_access_right" AccessAuthority AccessValue
AccessAuthority ::== "apache"
AccessValue ::== "read" | "execute" | "*"
Condition ::== ConditionType ConditionAuthority ConditionValue

The field "mode" regulates the relationship between the EACL policy and the original Apache policy defined in "httpd.conf".

mode 0: (expand) the request is granted if either EACL or Apache policy is satisfied.
mode 1: (narrow) the request is granted only if both EACL and Apache policy is satisfied.
mode 2: (exact) the request only applies to EACL policy.

If this field is not defined in the EACL file, it's set to 0 by default.

Each EACL policy consists of the header and a list of conditions. GAA-API has 4 types of condition: pre-condition, rr-condition(request-result-condition), mid-condition and post-condition. Pre-conditions correspond to the constraints checked by the policy. Rr-conditions regulate the actions performed by the policy. Mid-conditions and post-conditions are not used in current version of GAA-Apache.

The fields "AccessAuthority" regulates the identity that a policy applies to. For GAA-Apache it's always set to "apache". The fields "AccessValue" regulates the type of access right that a policy applies to. For GAA-Apache, there're three possible values: "read" applies to the request whose HTTP method is GET or HEAD, "execute" applies HTTP method POST. And "*" applies to any HTTP methods.

Receiving each request, GAA-API evaluates from the first EACL policy towards the last one. For each policy, when all constraints defined by its pre-conditions are satisfied, for "pos_access_right", the request is granted, and for "neg_access_right", the request is denied. If not all the pre-conditions are satisfied. GAA-API will go on evaluating the next policy for access control decision.

Besides making access control decision, GAA-API could also perform actions such as writing log info, sending alert email by rr-conditions. If the corresponding *ConditionValue* in rr-condition starts with "on:success/", this action is executed when all pre-conditions before are satisfied. Otherwise if the corresponding *ConditionValue* starts with "on:failure/", this action is when these exists any pre-condition unsatisfied before this rr-condition.

(Two) Available Conditions

1. Host Condition:

This condition checks if the remoted user is within the *HostList* specified.

- Format:
  pre_cond_access_host apache *HostList*

- *HostList* can be expressed in 5 formats:
  Domain name: [eg.] isi.edu -- matches all domain in *.isi.edu
  IP address: [eg.] 162.105.203.94 -- matches a single IP address
  IP range: [eg.] 166.111. -- matches subdomain 166.111.*.*
  IP/Mask: [eg.] 128.9.1.0/255.255.255.0 -- matches subdomain 128.9.1.*
  IP/CDR: [eg.] 128.64.36.0/22 -- matches subdomain 128.9.36.0-128.9.39.255

- And they could be integrated with the following operators: (from the highest priority on right to the lowest on left)
  ( ), NOT, AND, SUB, OR

- Example:
  pre_cond_access_host apache "(128.9.0.0/16 SUB 128.9.16.0/255.255.240.0 OR 162.105. AND pku.edu.cn) AND NOT 127.0.0.1"

2. Time Condition

This condition checks if the current time is within the TimeSpan specified, which could be either Local Time (*ConditionAuthority*="local") or Greenwich Mean Time (*ConditionAuthority*="gmt")

- Format:
  pre_cond_access_time [local | gmt] *TimeSpanList*
  TimeSpanList ::= *TimeSpan* {";" *TimeSpan*}
  TimeSpan ::= { [*DateList*] [*DayList*] [*TimeList*] }

- *TimeSpan* could regulate restrictions on Date, Day and Time. We use ";" to separate them. Within each *DateList*, *DayList* or *TimeList*:
  - use "-" to express an interval, with starting time on left and ending time on right.
    [eg]: 10:30am-20:30pm 1/15/03-5/20/03
  - use "," to separate the intervals, it equals to the relational operator "OR".
    [eg]: MON, WES-FRI 8am-11am,1pm-5pm

- *DateList* could be defined in three eligible formats:
  - mm/dd/yy restricted on year, month and date both.
    [eg]: 01/01/2003-12/31/2004
  - mm/dd restricted on month and date only, regardless which year it is.
    [eg]: 04/01-05/31
  - 00/dd restricted on date only, regardless which year and month it is.
    [eg]: 00/10-00/20

  DayList is defined by the following string: MON TUE WES THU FRI SAT SUN
  [eg]: MON-THU,SAT,SUN
  TimeList could be defined in two eligible formats:
  - hour24
    [eg]: 10:30-22:00
  - hour12 with am/pm
    [eg]: 12am-5:30pm

- Examples:
  pre_cond_access_time local "MON-FRI 8am-8pm;SAT,SUN 1pm-5pm"
  pre_cond_access_time gmt "01/01/03-05/20/03,08/20-12/20"

3. Authentication Conditions

Two conditions "pre_cond_access_id_user" and "pre_cond_access_id_group" checks the identity and group membership that authenticated by the client.

- Format:
  UserCondition ::= pre_cond_access_id_user apache *UserList*
  UserList ::= *UserName* {"," *UserName*}
  GroupCondition ::= pre_cond_access_id_group apache *GroupList*
  GroupList ::= *GroupName* {"," *GroupName*}

- In current GAA-Apache integration, GAA-API is still borrowing the Apache module "mod_auth" to implement authentication. Therefore to enable the authentication conditions, we should also provide corresponding information in Apache configuration files.

- Within your concerned domain (<Directory> or <Files>) of main configuration file "httpd.conf" or per-directory configuration file ".htaccess", add the following lines

```
AuthName  AlertString
AuthType  "Basic" | "Digest" | "DBM" | "Anon"
AuthUserFile  UserDefFile
AuthGroupFile  GroupDefFile
Require valid-user
Satisfy All
```

- Then generate the *UserDefFile* and *GroupDefFile* defined by the directives "AuthUserFile" and "AuthGroupFile" (The following direction applies to "AuthType Basic" only, please refer to the Apache's User Manual for other authentication types.)

  - Use command "htpasswd" to generate user file

  [Execute]: /usr/local/apache2/bin/htpasswd [-c] *UserDefFile UserName*

  UserDefFile is the same full pathname define by "AuthUserFile", *UserName* is the name of user you want to create

  If the *UserDefFile* does not exist and you want to create it, use the command above with parameter "-c". Otherwise, if the *UserDefFile* already exists and you want to append new entry in, do not use the parameter "-c".

  After executing the command above, system will prompt you to input and retype the password for that user. Then the *UserName*/*Password* peer will be added into the *UserFile* with the *Password* encrypted.

  - Create and edit the *GroupDefFile*. In this file, each line defines a group, it starts with the *GroupName*, and followed by a list of *UserName* as the group members.
    [Format]: *GroupFileLine* ::= *GroupName* ";" *UserName* {" " *UserName*}

  - Example:
    [---add the following in main configuration file---]

```
<Directory "/usr/local/apache2/htdocs/mydir">
    ……
    AuthName "Protected File, Please input Username & Password"
    AuthType Basic
    AuthUserFile  /usr/local/apache2/conf/user.conf
    AuthGroupFile  /usr/local/apache2/conf/group.conf
    Require valid-user
    Satisfy All
</Directory>
```

    [---defining user and group files---]

```
/usr/local/apache2/bin/htpassword                    -c
```

```
/usr/local/apache2/conf/user.conf  hellene
/usr/local/apache2/bin/htpassword
/usr/local/apache2/conf/user.conf  michael
/usr/local/apache2/bin/htpassword
/usr/local/apache2/conf/user.conf  diana
cat /usr/local/apache2/conf/user.conf
   hellene:
   michael:
   diana:
vi /usr/local/apache2/conf/group.conf
   groupA:hellene diana
   groupB:michael hellene
```

[---write the conditions in GAA-API's policy file---]

```
pre_cond_access_id_user apache "diana,michael"
pre_cond_access_id_group apache "groupA"
```

4. Option & Variable Conditions

- **Options**

In GAA-Apache, we could get the detailed information about Apache Web Server's per-request status through GAA-API's option. An *Option* is expressed in the following formats:

Option ::= "#" *OptionAuthority* "." *OptionType* or
Option ::= "#[" *OptionAuthority* "." *OptionType* "]"

Options could be used within the *ConditionValue* field of every GAA-API condition. GAA-API will translated all definitions of the options to their corresponding *OptionValue* before the checking process of each condition.

Here's the list of options available for GAA-Apache

Definition of the *Option*	**Description of the *OptionValue***
#[apache.remote_host]	The domain name of requested client
#[apache.remote_ip]	The IP address of requested client
#[apache.method]	The HTTP method of this request. (GET,POST,HEAD,etc.)
#[apache.uri]	The requested URI (eg. http://www.isi.edu/gost/index.html?id=anon)
#[apache.uri_path]	The pathname of requested resource (eg. /gost/index.html)
#[apache.uri_filename]	The filename of requested resource (eg. index.html)
#[apache.args]	The argument list from this request (eg. id=anon)
#[apache.content_type]	The content type of this request (eg. IMAGE/JPEG)
#[apache.content_encoding]	The encoding mechanism used in this request
#[apache.user]	The username that has been authenticated by the client
#[apache.auth_type]	The authority type that last authentication uses
#[apache.request_line]	The request line in HTTP protocol
#[apache.status_line]	The status line in HTTP protocol
#[apache.conn_id]	The connection id that assigned by Apache
#[apache.request_rec]	The request_rec structure used in Apache for this request

Except the option #[apache.request_rec], which is a pointer to the structure "request_rec" that is widely used in Apache's modules (defined in $APACHE_SRC_ROOT/include/httpd.h). All other *OptionValue*s are expressed in ASCII string format.

If the *ConitionAuthority* of the condition using this option is the same as *OptionAuthority*. We could also omit the OptionAuthority in the definition of this option. (Eg. #[apache.conn_id] could also be expressed as #conn_id if the *ConditionAuthority* is "apache").

- **Variables**

In GAA-API, we could read and write variables through INI-files. A *Variable* is expressed in the following formats:

Variable ::= "%" [*VariableSection* "."] *VariableName* ["@" *VariableFile*] or
Variable ::= "%{" [*VariableSection* "."] *VariableName* ["@" *VariableFile*] "}"

Like the options, variables could also be used within the *ConditionValue* field of every GAA-API condition. GAA-API will translate all definitions of the variables to their corresponding *VariableValue* before the checking process of each condition. These *VariableValues* are provided in INI format files. Moreover, we could also use condition (rr_cond_set_variable) to set or update the values of all variables.

Here's the example of an INI file (/tmp/apache.var).

```
[GAA_PARAMS]
securitylevel=red
[APACHE]
visitcount=15203
alertmsg=The request from #[apache.remote_ip] is rejected
```

Then the Option %{apache.visitcount@/tmp/apache.var} will be translated to value "15203", and %{GAA_PARAMS.securitylevel@/tmp/apache.var} will be translated to value "red".

Further more, variable could

- If the *VariableFile* is not a full path start with "/", GAA-API will assigned it to the default directory "/tmp/"
- If the definition of *VariableFile* is absent, the INI-file will be assigned as "/tmp/+*ConditionAuthority*+".var". (The value of *ConditionAuthority* is got from the condition using this variable.)
- If the definition of *VariableSection* is absent, GAA-API will assign it to the default section "GAA_PARAMS".

So, in the previous example, the variable %{GAA_PARAMS.securitylevel@/tmp/apache.var} could also be expressed as the following:

%{securitylevel@apache.var} or
%securitylevel (if the condition using this variable has its *ConditionAuthority*="apache")

- **Checking Conditions**

In GAA-API, there're conditions specially used to check the value of options and variables. Their format is:

Condition ::= *CheckCondition* *ConditionAuthority* *CompareEquation*
CompareEquation ::= *Option* | *Variable* "=" *CompareValueList*
CompareValueList ::= *CompareValue* {";" *CompareValue*}

We have the following types of *CheckCondition* available now:

- pre_cond_check_equal

Check if the value of *Option* or *Variable* equals to the any of the *CompareValue*

- pre_cond_check_caseequal

Check if the value of *Option* or *Variable* equals to the any *CompareValue* with letter case ignored

- pre_cond_check_token

Check if any of the *CompareValue* is included in the value of *Option* or *Variable*.

- pre_cond_check_token

Check if any of the *CompareValue* is included in the value of *Option* or *Variable* with letter case ignored.

- pre_cond_check_regex

Check if any of the value of *Option* or *Variable* matches the pattern of regular expression defined in *CompareValue*.

- pre_cond_check_caseregex

Check if any of the value of *Option* or *Variable* matches the pattern of regular expression defined in *CompareValue* with letter case ignored.

For a regular expression:
"*" represents an arbitrary string in any length.
"?" represents an arbitrary character.
"[]" represents any character specified in the list. (eg. [1-9,a-z])

- pre_cond_numeric_comp

Check if any of the value of *Option* or *Variable* satisfies the numeric equation. In this condition, the value of *Option* or *Variable* should be a valid number (either integer or float), or the condition will always answers NO to GAA-API.

Moreover, the format of "pre_cond_numeric_comp" is different from the conditions above

Condition ::= pre_cond_numeric_comp *ConditionAuthority* *CompareEquation2*
CompareEquation2 ::= *Option* | *Variable* *CompareOP* *ComparedNumericValue*
CompareOP ::= "**>**" | "**<**" | "**=**" | "**==**" | "**!=**" | "**<>**" | "**<=**" | "**>=**"

From the definition of *CompareOP*, we could find that the comparing relationship of less than (<), greater than (>), equal to (**=** or **==**), not equal to (!= or <>), less than or equal to (<=) and greater than or equal to (>=) are supported by the function. But the definition of option or variable must be put on the left, and the numeric value for comparison must be put on the right.

- **Setting Conditions**

GAA-API also has conditions to set or update the values of variables (but not for options). Their format is:

Condition ::= rr_cond_set_variable *ConditionAuthority* *SetEquation*
Condition ::= rr_cond_inc_variable | rr_cond_dec_variable *ConditionAuthority* *Variable*
SetEquation ::= *Variable* "=" *SetValue*

- rr_cond_set_variable

Update the value of *Variable* to *SetValue*.

- rr_cond_inc_variable

Increase the value of *Variable* by 1. If the original value is not an integer or does not exist, the updated value will be set to 1 by default.

- rr_cond_dec_variable

Decrease the value of *Variable* by 1. If the original value is not an integer or does not exist, the updated value will be set to 0 by default.

- **Examples**

- pre_cond_check_regex apache "#uri_filename=A[0-9]*.html; B[0.9]*.htm"

Check if the option "apache/uri_filename" matches any of the 2 regular expressions.

- rr_cond_set_variable system "%{threatlevel.#[remote_ip]}='under attack' "

Set the variable in section "threadlevel" of INI-file "/tmp/system.var" to the value "under attack". The variable name is dynamically assigned to the value of option "apache/remote_ip".

[eg.] If the remote_ip is 128.9.64.23, then in file /tmp/system.var we will have:

```
[threatlevel]
128.9.64.23=under attack
```

- pre_cond_check_casetoken apache "#content_type=%{allowedtype}"

Check if the token list given by the variable *allowedtype* is included within the option "apache/content_type". This variable *allowedtype* is defined in section "GAA_PARAMS" of

INI-file "/tmp/apache.var".

- pre_cond_access_host apache "127.0.0.1 OR %{permit_host@/var/apache/def}"

Check if the request client is 127.0.0.1 or the hosts defined in variable *permit_host* of section "GAA_PARAMS" in INI-file "/var/apache/def"

5. Other Conditions

- rr_cond_email_notify

Send email notification containing the request client's IP address and username to *EmailAddress*

[format]:
EmailCondition ::= rr_cond_email_notify *ConditionAuthority* *EmailAddress*
[example]:
```
rr_cond_email_notify apache "on:failure/zhou@isi.edu"
```

- rr_cond_append_log

Append a new line into the log file, whose filename is given by the *ConditionAuthority*

[format]:
LogCondition ::= rr_cond_append_log *LogFileName* *LogContent*
[example]:
```
rr_cond_append_log /tmp/apache.log "%{log_msg_1@/var/logmsg}"
```

---The content of /var/logmsg---
```
[GAA_PARAMS]
log_msg_1=Alert/Insecure request from #[apache.remote_ip]
```