Nagios: Active Directory Authentication

by Alan Pipitone and Fabio Frioni

http://www.alan-pipitone.com

Thanks for cover images: http://www.iconarchive.com/ , http://vargas21.deviantart.com/ , http://www.artua.com/ , http://www.icons-land.com/

Introduction:

This document explains the work we have done to integrate Nagios with Microsoft's Active Directory. In this way, users can connect to nagios with their domain account. And if they are member of the correct Domain Group, They will see hosts and services enabled for that group. Also, They can even use the single sign on if enabled on their browser.

We tried to create an authentication system (based on the Domain Group as I said) that could be **reused easily** with other Web interfaces and with more then one Nagios server (balancing configuration, distributed system etc.)

Our method is not quite standard but it works fine :)

Scenario:

1. We have to integrate Nagios with more than one Windows Domain (all with Active Directory).
2. Our users use Windows with Internet Explorer to view Nagios.
3. We want to enable users to connect Nagios with their Domain Account.
4. We want to assign permissions based on the Domain Group.
5. We want to enable single sign on.
6. We want to do point 3, 4 and 5 in the simplest (and efficient) possible way.
7. For Company Policy, We can not login to our Domain Controller and We can not change anything on our Domain. We can create only Domain Groups.

The Solution:

Because of point 6 and 7, we can not use kerberos module for apache on Nagios server. In fact it needs a domain account with a password that never expires (our Company Policy doesn't allow that), also you have to run some command (It's easy but We can not do it) on Domain Controller. With apache kerberos module It isn't simple manage role with Domain Groups as We want. So, implement the kerberos module for our purposes is not possible.
For the authentication process we decided to use a second server with Microsoft IIS. In fact with IIS is much easier to configure the single sign-on with Domain Account (and you don't have to change anything in your Domain, you just have to create Domain Groups). We can consider the IIS server as the Nagios Authentication server. In fact It will contact the Active Directory and will check if the user is (or not) member of a Nagios Domain Group

Our Environment:

Obviously I can not refer to the production environment where I work. So for this tutorial I will use fictitious names. So, we have two Domains called **alpha** and **delta**, both in the same forest called **for**. The two Domain strings are **alpha.for** and **delta.for**. We have also a Domain (called **gerfra**) that isn't in the same forest but it's trusted with alpha and delta, the domain string is **gerfra.euro.world**.
The name of Nagios Authentication server (the IIS web server) is nagiosauth, and it is member of alpha domain, so its fqdn is **nagiosauth.alpha.for**. For the nagios server we have created a DNS record called **nagios.alpha.for**. It's important to note that **nagios and nagiosauth must have the same suffix** (we will see later why), in our case it is “alpha.for”.

The Configuration:

We have created a domain group for each account defined in nagios. For the Domain Groups We used a naming convention: the name of each group is **grp_nagios_<nagios_account>**, where <nagios_account> is the name of the account defined in nagios.
To make it work properly, in the file of nagios contacts we have separated the contacts between those who should receive the e-mail and who should see the pages. The domain group is associated with the contact that should see the pages. If you have a group who should see the pages and receive notifications you must create a contact group (in Nagios) where to place both contact inside.

Example:

We want that all member of the domain group **grp_nagios_application_admin** can see (in Nagios) the server host-one.alpha.for, host-two.alpha.for and host-three.alpha.for and the notifications should be sent to app.admin@*some-domain-for-email.ext* (the e-mail of the application_admin group)

In our Nagios contacts file we have included the contacts as shown below:

```
# template for all account contact
define contact{
        name                    account-template              ; use it as template in contact definition for JUST ACCOUNT USER
        service_notification_period     local-period    ; See timeperiods.cfg service notifications can be sent anytime
        host_notification_period        local-period    ; See timeperiods.cfg host notifications can be sent anytime
        service_notification_options    n
        host_notification_options       n
        email                           root@localhost  ;don't send the e-mail tho other people
        service_notification_commands   notify-service-by-email
        host_notification_commands      notify-host-by-email
        register                0                       ; DONT REGISTER THIS DEFINITION - ITS JUST A TEMPLATE!
 }

# template for notification of application_admin group
define contact{
        name                    appadmin-contact              ; The name of this contact template
        service_notification_period     local-period    ; See timeperiods.cfg service notifications can be sent anytime
        host_notification_period        local-period            ; See timeperiods.cfg host notifications can be sent anytime
        service_notification_options        w,u,c,r,f,s             ; send notifications for all service states
        host_notification_options           d,u,r,f,s                       ; send notifications for all host states
        service_notification_commands       notify-service-by-email        ; send service notifications via email
        host_notification_commands          notify-host-by-email          ; send host notifications via email
        register                0                       ; DONT REGISTER THIS DEFINITION - ITS JUST A TEMPLATE!
}

#application_admin account
define contact{
        contact_name    application_admin               ; Related to: application_admin_notify
        use             account-template
        alias           Application Admin for access only
}

#application_admin notification contact
define contact{
        contact_name    application_admin_notify
        use             appadmin-contact
        alias           Application Admin for notifications
```

```
            email                   app.admin@some-domain-for-email.ext
}

#contact group for application admin, we use this in server that application admin have to see
define contactgroup{
            contactgroup_name          applications_admins
            alias                      Nagios Administrators
            members                    application_admin,application_admin_notify
}
```

Note that the contact_name's value (in our example application_admin) should be in lowercase. This is because the Nagios Authentication server set the cookie's value in lowercase and Nagios is case sensitive.

For the monitoring of server host-one.alpha.for, host-two.alpha.for and host-three.alpha.for we have created a new configuration file in nagios (and we use the contact_groups applications_admins as we have previously described), its content is shown below:

```
#host base template - This is NOT a real host, just a template!
define host{
       name                        server-alpha-base          ; The name of this host template
       notifications_enabled          1                       ; Host notifications are enabled
       event_handler_enabled          1                       ; Host event handler is enabled
       flap_detection_enabled         1                       ; Flap detection is enabled
       failure_prediction_enabled     1                       ; Failure prediction is enabled
       process_perf_data              1                       ; Process performance data
       max_check_attempts             4                       ; Check each server 10 times (max)
       retain_status_information      1                       ; Retain status information across program restarts
       retain_nonstatus_information   1                       ; Retain non-status information across program restarts
       notification_period         default-period             ; Send host notifications at any time
       register                    0                          ; DONT REGISTER THIS DEFINITION - ITS JUST A TEMPLATE!
       }

# host definition template - This is NOT a real host, just a template!
define host{
       name                        server-alpha               ; The name of this host template
       use                         server-alpha-base  ; Inherit default values from the generic-host template
       check_period                default-period             ; By default, Windows servers are monitored round the clock
       check_interval              5                          ; Actively check the server every 5 minutes
       retry_interval              1                          ; Schedule host check retries at 1 minute intervals
       check_command               check-host-alive           ; Default command to check if servers are "alive"
       notification_period         default-period             ; Send notification out at any time - day or night
       notification_interval       30                         ; Resend notifications every 30 minutes
       notification_options        d,u,r                      ; Only send notifications for specific host states
       contact_groups              applications_admins        ; Notifications get sent to the admins by default
       hostgroups                  alpha-grp                  ; Host groups that Windows servers should be a member of
       icon_image                  server.gif
       statusmap_image             server.jpg
       register                    0                          ; DONT REGISTER THIS - ITS JUST A TEMPLATE
       }

# SERVICE TEMPLATES

# Generic service definition template - This is NOT a real service, just a template!
define service{
       name                        server-alpha-service       ; The 'name' of this service template
       active_checks_enabled          1                       ; Active service checks are enabled
       passive_checks_enabled         1                       ; Passive service checks are enabled/accepted
```

```
        parallelize_check               1                               ; Active service checks should be parallelized
        obsess_over_service             1                               ; We should obsess over this service (if necessary)
        check_freshness                 0                               ; Default is to NOT check service 'freshness'
        notifications_enabled           1                               ; Service notifications are enabled
        event_handler_enabled           1                               ; Service event handler is enabled
        flap_detection_enabled          1                               ; Flap detection is enabled
        failure_prediction_enabled      1                               ; Failure prediction is enabled
        process_perf_data               1                               ; Process performance data
        retain_status_information       1                               ; Retain status information across program restarts
        retain_nonstatus_information    1                               ; Retain non-status information across program restarts
        is_volatile                     0                               ; The service is not volatile
        check_period                    default-period                  ; The service can be checked at any time of the day
        max_check_attempts              3               ; Re-check the service up to 3 times in order to determine its final (hard)
state
        normal_check_interval           10                              ; Check the service every 10 minutes under normal conditions
        retry_check_interval            2               ; Re-check the service every two minutes until a hard state can be determined
        contact_groups                  applications_admins             ; Notifications get sent out to everyone in the 'admins' group
        notification_options            w,u,c,r         ; Send notifications about warning, unknown, critical, and recovery events
        notification_interval           60                              ; Re-notify about service problems every hour
        notification_period             default-period                  ; Notifications can be sent out at any time
        register                        0                               ; DONT REGISTER THIS DEFINITION - ITS JUST A TEMPLATE!
        }

# HOST DEFINITIONS
define host{
        use                             server-alpha
        host_name                       host-one.alpha.for
        alias                           APPS – DC: host-one.alpha.for
        address                         192.168.10.11
      }

define host{
        use                             server-alpha
        host_name                       host-two.alpha.for
        alias                           APPS – DC: host-two.alpha.for
        address                         192.168.10.11
      }

define host{
        use                             server-alpha
        host_name                       host-three.alpha.for
        alias                           APPS – DC: host-three.alpha.for
        address                         192.168.10.11
      }

# HOST GROUP DEFINITION
define hostgroup{
        hostgroup_name          alpha-grp
        alias                   host group alpha.for
        }

# SERVICEDEFINITION
define service{
        use                             server-alpha-service
        host_name                       host-one.alpha.for, host-two.alpha.for and host-three.alpha.for
        service_description             CPU Load
        check_command                   check_cpu_nrpe!85!95
        }
```

```
define service{
       use                          server-alpha-service
       host_name                host-one.alpha.for, host-two.alpha.for and host-three.alpha.for
       service_description           Memory in use
       check_command                 check_virtual_mem_nrpe!90!95
       }
```

Now that the nagios configuration files have been described, we proceed with the group on the domain:

Some of our members who belong to the domains alpha.for, delta.for and gerfra.euro.world have access nagios as application_admin. So in each domain we have to create a group called **grp_nagios_application_admin** (as described above) and we insert users in that group of their domain.

How It Works:

When the user accesses the pages of Nagios, this last checks if the browser has two particular session cookies. If the user doesn't have the cookies then Nagios will redirect the user to the Nagios Authentication server (IIS Web Server).
If the PC of the user is joined to Domain and the browser is configured for the single sign on then the Nagios Authentication server will be able to take the user's account without requiring any credential. If the the PC of the user isn't joined to Domain or the browser isn't configured for the single sign on then the Nagios Authentication server will prompt the request for the Domain account and password.
With the Domain Account the Nagios Authentication server will search in the Domain of the user if he is member of a Nagios group (in our example **grp_nagios_application_admin**). If the user belongs to the Nagios Domain group then Nagios Authentication server will set the cookies and redirect the user to the Nagios Web Page that he was looking for before the authentication process.

The two cookies are:

1. **nagios_group:** this cookie contains the Nagios account (taken from the name of the group, do you remember the above naming convention?!). In our example the domain group is **grp_nagios_application_admin** so the nagios account is **application_admin**. This is the cookie used to authenticate the user in nagios.
2. **nagios_account:** this cookie contains the domain account of the user. This is not used by nagios, but it could be used in future (for log etc.)

<u>Configure Apache:</u>

First of all we need to delete (or comment) the part that requires the user authentication. When you install Nagios it adds its configuration file (in our case nagios.conf) into Apache folder. The nagios.conf contains the rules for the nagios user authentication. Here the contents of the file nagios.conf after commenting (red color) the lines about the authentication:

```
ScriptAlias /nagios/cgi-bin "/usr/local/nagios/sbin"

<Directory "/usr/local/nagios/sbin">
#  SSLRequireSSL
   Options ExecCGI
   AllowOverride None
   Order allow,deny
   Allow from all
#  Order deny,allow
#  Deny from all
#  Allow from 127.0.0.1
#   AuthName "Nagios Access"
#   AuthType Basic
#   AuthUserFile /usr/local/nagios/etc/htpasswd.users
#   Require valid-user
</Directory>

Alias /nagios "/usr/local/nagios/share"

<Directory "/usr/local/nagios/share">
#  SSLRequireSSL
   Options None
   AllowOverride None
   Order allow,deny
   Allow from all
#  Order deny,allow
#  Deny from all
#  Allow from 127.0.0.1
#   AuthName "Nagios Access"
#   AuthType Basic
#   AuthUserFile /usr/local/nagios/etc/htpasswd.users
#   Require valid-user
</Directory>

# --------------------------------------------
# This Virtual Directory is necessary for news
# news and email reader
#
Alias /repository "/usr/local/nagios/repository"

<Directory "/usr/local/nagios/repository">
   AllowOverride None
   Options None
   Order allow,deny
   Allow from all
#
# Authenticated
#
#   AuthName "Nagios Access"
#   AuthType Basic
#   AuthUserFile /usr/local/nagios/etc/htpasswd.users
#   Require valid-user
</Directory>
```

You must do the same thing with the configuration file that nagvis put into the apache folder (if you have installed nagvis).

The Code (Nagios CGI):

To implement the authentication with the Domain we had to make some changes to source code of the nagios cgi. First of all, to apply this new feature we have to act with some code modification in these 3 files (We have included them in the zip):

1. cgiauth.c
2. cgiutils.c
3. cgiauth.h

The main changes are in the file cgiauth.c file and cgiauth.h header file, in order to allow the HTTP_COOKIE environment variable. The modification is very simple and smarter, inserting the control code in the authentication section and acting on the reading of the environment variable HTTP_COOKIE. A procedure will read the contents of the variable and will check if there is the expected cookie. If no, the cgi will redirect through a remote website for the authentication, until we will receive the correct cookie.
We have achieved this with a little trickery: if the cookies aren't present, we call this function:
sprintf(query_redirect,"<SCRIPT type='text/javascript'>window.location='%s?target=http://%s%s'</SCRIPT>",use_cookie_authentication_site,getenv("HTTP_HOST"),getenv("REQUEST_URI"))
that prints (instead of printing the user name) a redirect to the Nagios Authentication Server (for details see the sources). This is because the user name is printed on every page. In this way you have only to change this part to check the cookie on every page.
We modified cgiutils.c too in order to extend the received information and to display them in the login information box.
If you want to see all changes in the three file, you have to look the part that starts with *patch by Alan Pipitone & Fabio Frioni* in the three sources.
There is nothing embedded in the source files, but we exported all useful information into cgi.cfg definition file. Here is the settings of extended features:

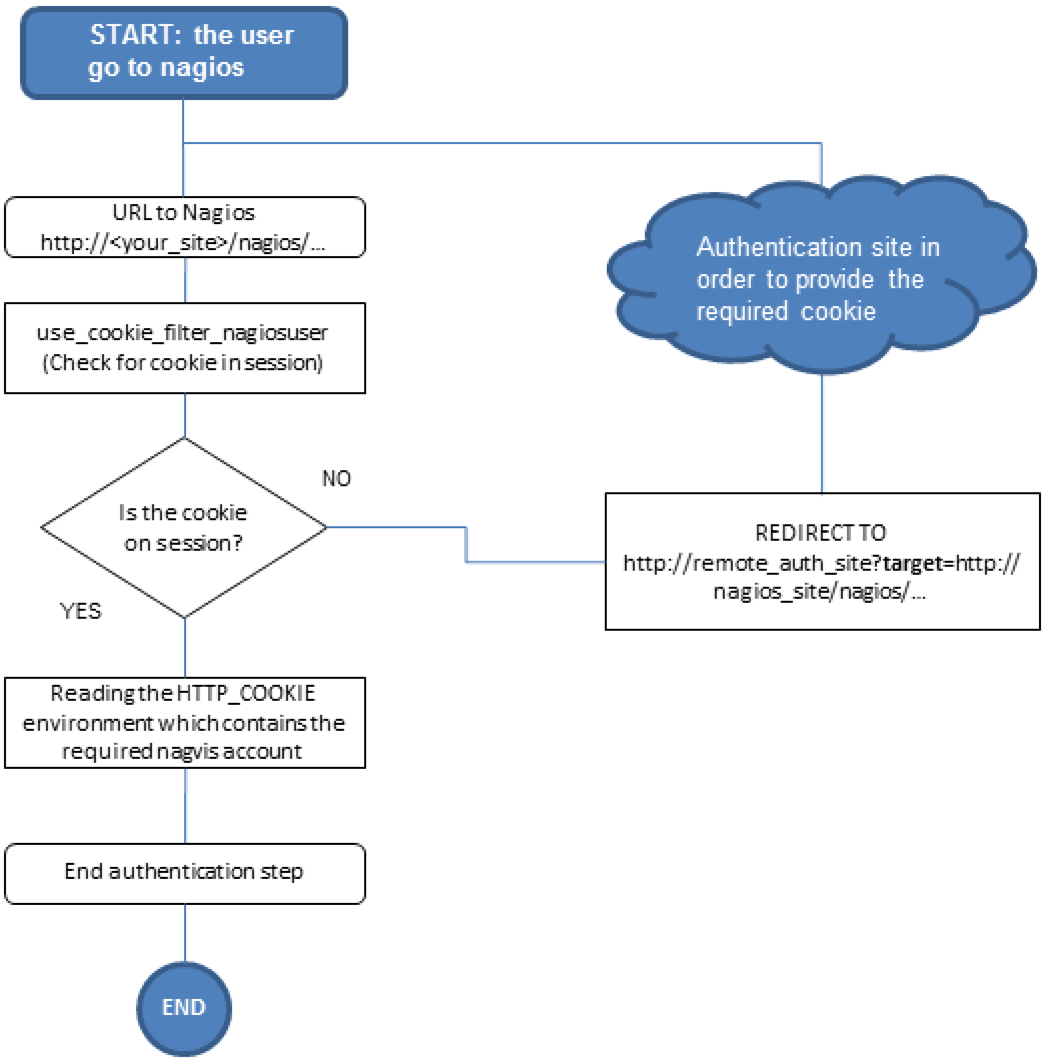
```
# SUPPORT COOKIE AUTHENTICATION
# This tells the CGIs that Nagios will check the Apache HTTP_COOKIE environment variable
# to authenticate the contact.
# use_cookie_authentication=1   enable cookie check authentication
# use_cookie_authentication=0   disable cookie check authentication
# Default is 0

use_cookie_authentication=1

# COOKIES GROUP FILTERS
# These 2 cookies tell the CGIs the nagios contact group and domain user account ID
# feel free to set the proper name to identify these 2 cookies
# Note: the "nagiosuser" cookie is requested, otherwise no authentication is permitted
#
# - Applied filter: nagios_group
# - Cookie found: nagios_group=admins; ...
# - Nagios contact in contacts.cfg: admins
#
use_cookie_filter_nagiosuser=nagios_group
use_cookie_filter_domainuser=nagios_account

# WEBSITE PROVIDER OF COOKIES AND AUTHENTICATION
# Use this value to identify the remote site to generate the requested cookies
# the remote site will authenticate the user

use_cookie_authentication_site=http://<your_remote_authentication_site>
#in our example: use_cookie_authentication_site=http://nagiosauth.alpha.for
```

use_cookie_authentication
This setting enables the cookie authentication method in Nagios CGI server.

use_cookie_filter_nagiosuser and use_cookie_filter_domainuser
These settings define the name of cookie where is stored the Nagios contact and the name of cookie where is stored the real user in Active Directory. In other words we will wait 2 cookies, but nagiosuser is the relevant cookie.

use_cookie_authentication_site
This settings defines the remote site for authentication method. This site will provide 2 cookies, defined in the previous settings.

Step by step what the modified CGI does


START: the user go to nagios
URL to Nagios http://<your_site>/nagios/...
use_cookie_filter_nagiosuser (Check for cookie in session)
Is the cookie on session?
NO
YES
Reading the HTTP_COOKIE environment which contains the required nagvis account
End authentication step
END
Authentication site in order to provide the required cookie
REDIRECT TO http://remote_auth_site?target=http://nagios_site/nagios/...

1) The user wants to connect for instance to http://<nagios_site>/nagios (in our example http://nagios.alpha.for/nagios)

2) The CGI core checks if is already stored a cookie name defined in the *use_cookie_filter_nagiosuser* configuration variable.
3) If the required cookie name is not stored in the browser session, than the object class acts a ***redirect*ion** through the given site for a regular authentication for instance.

   Note that in our case the redirection is pointing a remote application (IIS website) in the form: http://<remote_authentication_site>?*target*=http://<nagios_site>/nagios (in our example http://nagiosauth.alpha.for/?*target*=http://nagios.alpha.for/nagios)

   remote site will generate the cookie and redirects again to the *target*
4) the user will get the grant to access from the authentication web site which will provide the required cookie in the form:

   <cookie name>=<nagios account>

   where *cookie_name* is name of cookie defined in *logonenvvar* config variable
   In our example:
   nagios_group=application_admin
   (the nagios authentication server set another cookie: nagios_account=sxs0090 because our Domain Account is sxs0090, but this is not necessary for nagios)

   In other words, the authentication site will allows the user by storing the required cookie in the client browser and with the correct Nagios contact.
5) When the client browser will get the cookie, the user can access the Nagios site

<u>The Code (Nagvis Pages):</u>

The Module CoreModLogonCookie.php
This module is the core of the main concept and it is similar to *CoreModLogonEnv.php*, In fact uses the same behavior by reading the HTTP_COOKIE environment variable and extracting the specified cookie name.
There are 3 new classes, the last class is used to extend all properties of the first 2

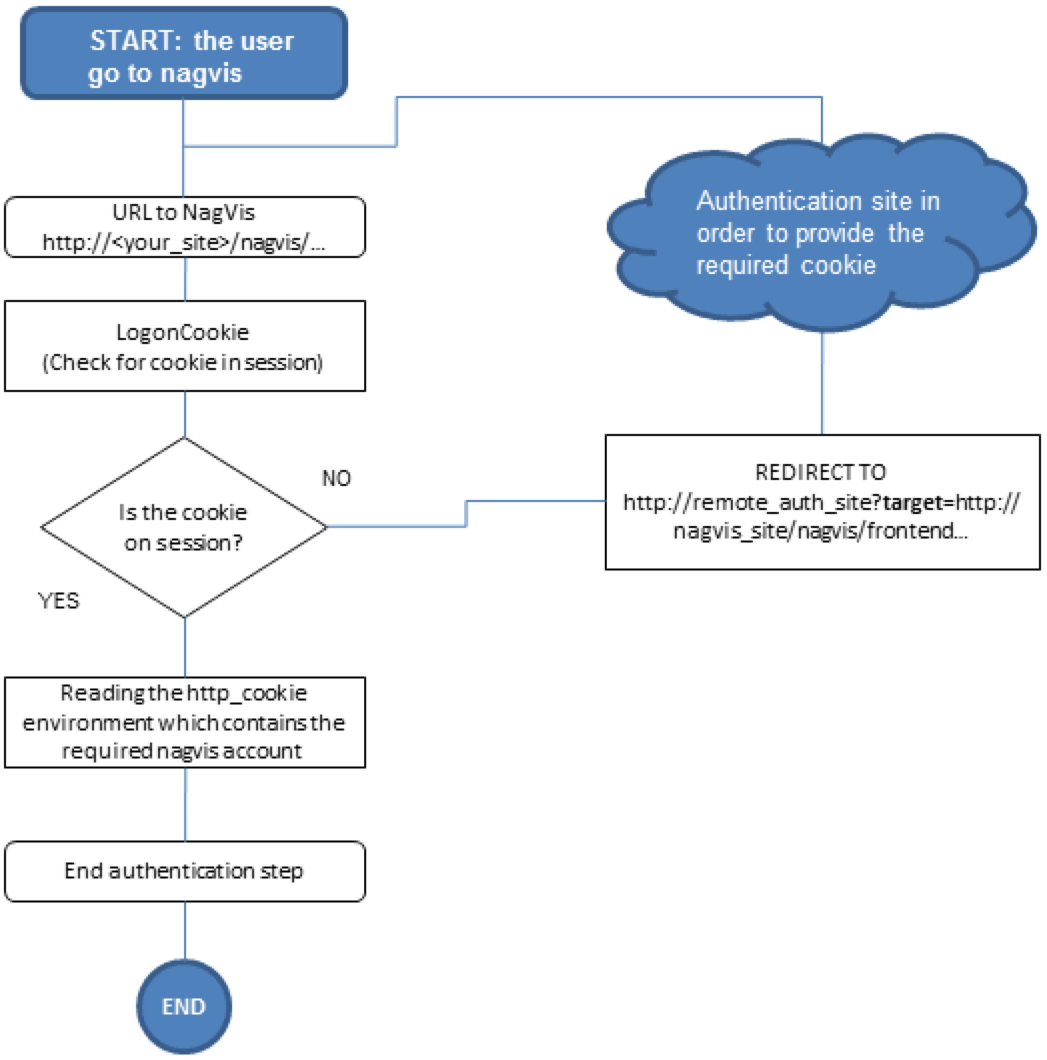
1. WuiModLogonCookie.php
2. FrontendModLogonCookie.php
3. CoreModLogonCookie.php

and it is possible to use that class module directly in nagvis.ini.php

```
logonmodule="LogonCookie"
logonenvvar="nagios_group"
```

where *logonmodule* contains the object class module and *logonenvvar* contains the cookie name to be looked up.

Step by step what the class does


START: the user go to nagvis
URL to NagVis http://<your_site>/nagvis/...
LogonCookie (Check for cookie in session)
Is the cookie on session?
NO
YES
Authentication site in order to provide the required cookie
REDIRECT TO http://remote_auth_site?target=http://nagvis_site/nagvis/frontend...
Reading the http_cookie environment which contains the required nagvis account
End authentication step
END

1) The user wants to connect for instance to http://<nagvis_site>/nagvis/frontend/nagvis-js/index.php?mod=Map&act=view&show=<your_map>
2) The core module checks if is already stored a cookie name defined in the *logonenvvar* configuration variable.
3) If the required cookie name is not stored in the browser session, than the object class acts a ***redirection*** through the given site for a regular authentication for instance (***note that the URL redirection to the site which authenticates is embedded in this object class in the file CoreModLogonCookie.php***).

   Note that in our case the redirection is pointing a remote application (IIS website) in the form: http://<remote_authentication_site>?*target*=http://<nagvis_site>/nagvis/frontend/nagvis-js/index.php?mod=Map&act=view&show=<your_map> (for our example, if we are going to the map headquarter without cookie: http://nagiosauth.alpha.for/?*target*=http://nagios.alpha.for/nagvis/frontend/nagvis-js/index.php?mod=Map&act=view&show=headquarter)

   remote site will generate the cookie and redirects again to the *target*

4) the user will get the grant to access from the authentication web site which will provide the required cookie in the form:

   <cookie name>=<nagvis account>

   where *cookie_name* is name of cookie defined in *logonenvvar* config variable

   In our example:
   nagios_group=application_admin
   (the nagios authentication server set another cookie: nagios_account=sxs0090 because our Domain Account is sxs0090, but this is not necessary for nagios)

   In other words, the authentication site will allows the user by storing the required cookie in the client browser and with the correct nagvis account.
5) When the client browser will get the cookie, the user can access the NagVis site

NB: Our module can create into Nagvis an account foreach domain group (read the README.txt in source_and_bin.zip), but you have to set the correct permissions into Nagvis console foreach account.

The Code (Ninja Pages):

Ninja is my favorite web interface. So We decided to use our authentication method also for it. In fact with our method is very simple to implement the single sign-on (and give permissions based on the Domain Groups) on Ninja.

First of all make sure that Ninja (We tried all on Ninja 1.1.0) is properly installed and correctly configured on your server before to set up our authentication system. Then read below:

We modified the file **<ninja_path>/application/controllers/authenticated.php** to force login with the user name taken from the cookie nagios_group. In our example the cookie's value is application_admin (because the domain group is **grp_nagios_application_admin** so the nagios account is **application_admin**, remember our naming convention?!). If our cookies aren't present this page will redirect the user to our Nagios Authentication server (**nagiosauth.alpha.for** in our example).

We modified the file **<ninja_path>/application/config/session.php** to keep the ninja session active only until the browser is closed (like our cookies).

We modified the file **<ninja_path>/application/views/themes/default/template.php** to print the value of nagios_group cookie and the value of nagios_account cookie on Ninja web interface. Also, we have removed the logout link, We do not need it.

Now we have to do a few more steps than Nagios and NagVis. In fact Ninja checks if the user (in our example **application_admin**) is present into the Merlin DB. I didn't try to bypass this check because I didn't want to waste time :)

Ninja provides a tool (the php script auth_import_mysql.php) to import users into the Merlin DB. That script reads the users from htpasswd.users (and from cgi.cfg) of Nagios and then puts those accounts into the Merlin tables.

Note that the htpasswd.users isn't used by our authentication system. It is used only by Ninja's auth_import_mysql.php file to update the users table.

Into the htpasswd.users file you have to insert the exact account (in our example application_admin, because the domain group is **grp_nagios_application_admin** so the nagios account is **application_admin**, remember our naming convention?!) , the password does not matter, You can put anything you want, It will not be used by our authentication system.

I added this line into htpasswd.users file:

```
application_admin:null.null.null.null
```

It's a nice password, isn't it? :)

Than I ran the command:

<path_of_php_interpreter>/php <ninja_path>/install_scripts/auth_import_mysql.php <ninja_path>

For example, on my server I ran:

```
/usr/bin/php    /usr/local/nagios/addons/ninja/install_scripts/auth_import_mysql.php
/usr/local/nagios/addons/ninja
```

These changes (modify htpasswd.users and then use auth_import_mysql.php) may be a bit boring. Another solution is to manually update the database with the new account. The tables involved should be only **users** and **ninja_user_authorization** but I'm not sure.

Remember to read also the file README.txt

The Code (Nagios Authentication Server):

In this section we describe the code of the web page of Nagios Authentication server.
The page is written in ASP .NET (Framework .NET 4.0) and is hosted by an IIS 6.0 Web Server (our Nagios Authentication Server).
You can change some configuration in the web.config (the configuration file for our Asp .NET page):

cookie_domain: contains the domain of the cookie. It must be the same of nagios server and Nagios Authentication Server (in our example: .alpha.for) otherwise the browser will not read it.

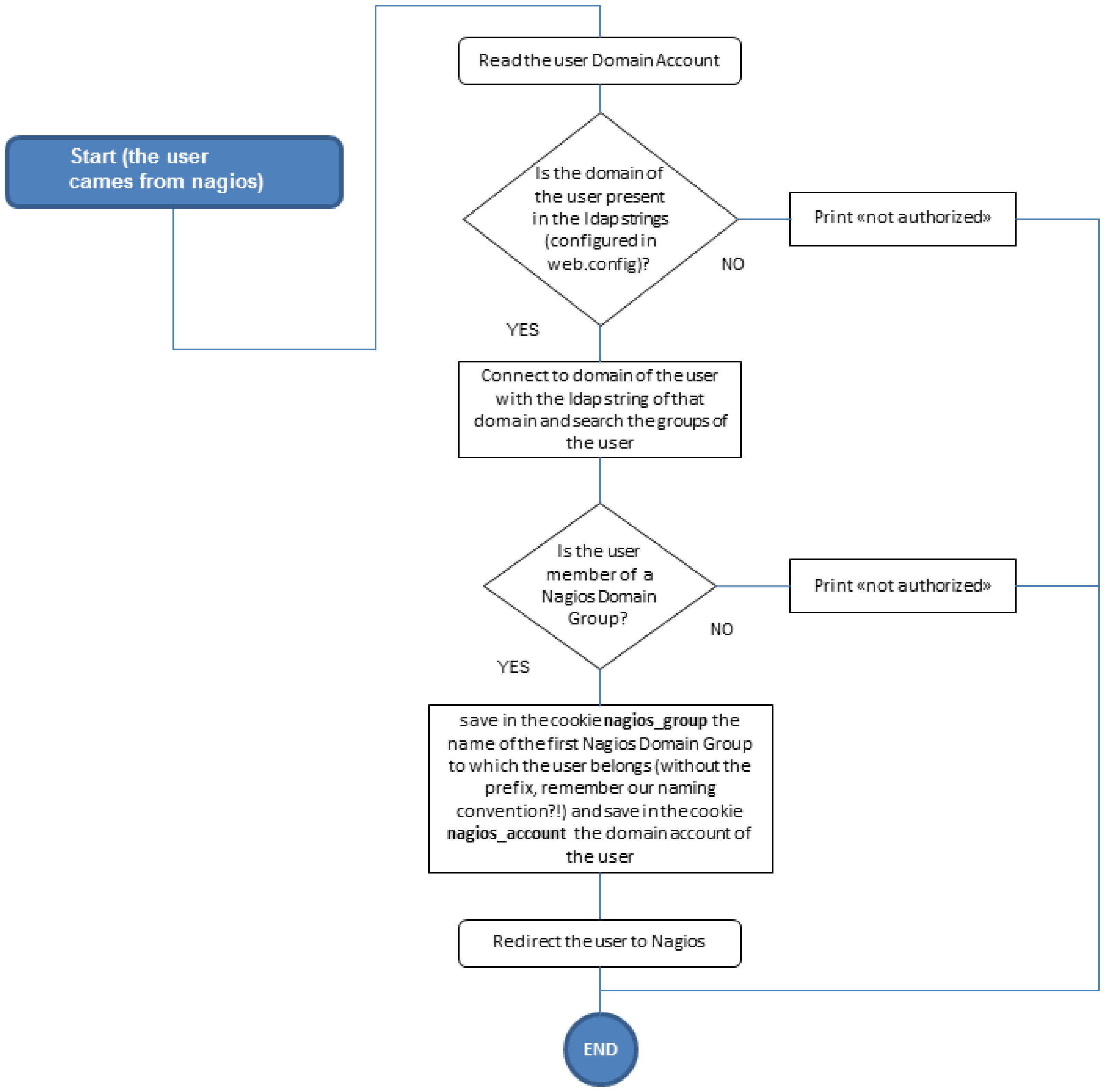
group_prefix: contains the prefix of our Nagios Domain Group (in our example: grp_nagios_). Do you remember the above naming convention?!

ldap_strings: contains the ldap connection string for our Domains. Each ldap string is separated by a pipe. This is the sintax: domain1=ldap_string1|domain2=ldap_string2 . It's important to note that You have to put the domain name followed by an equals sign and the ldap connection string (in our example: alpha=LDAP://DC=alpha,DC=for|delta=LDAP://DC=delta,DC=for|
gerfra=LDAP://DC=gerfra,DC=euro,DC=world).

nagios_url: contains the default url of nagios server (in our example: http://nagios.alpha.for/nagios/). As described above, our Asp .NET page should redirect the user to the target passed in the query string. However, if the target has not been filled, the Asp .NET page redirects the user to the link present here (in nagios_url).

log_path: contains the path where the log should be saved. You have to create the folders if they don't exist.

Step by step what the class does


Start (the user cames from nagios)
Read the user Domain Account
Is the domain of the user present in the ldap strings (configured in web.config)?
NO
Print «not authorized»
YES
Connect to domain of the user with the ldap string of that domain and search the groups of the user
Is the user member of a Nagios Domain Group?
NO
Print «not authorized»
YES
save in the cookie **nagios_group** the name of the first Nagios Domain Group to which the user belongs (without the prefix, remember our naming convention?!) and save in the cookie **nagios_account** the domain account of the user
Redirect the user to Nagios
END

1. When the user accesses the pages of Nagios, this last checks if the browser has two particular session cookie. If the user doesn't have the cookies then Nagios will redirect the user to the Nagios Authentication server (IIS Web Server). In our example http://nagiosauth.alpha.for
2. At this point, the IIS web server takes the user's account and user's domain.
3. With the user account and user Domain the Nagios Authentication server verify if the user's domain is present into the web.config.
4. If the user's domain is present in the web.config then our page use the ldap connection string (for that domain) to connect to the user's domain.

5. If the user is member of a Nagios Domain Group then the Asp .NET page sets the cookie with the information of the first Nagios Domain Group to which the user belongs.
6. Finally the user is redirected to the page of nagios that he required before the process of authentication.

NB: in IIS You have to disable the Anonymous Access and enable only Integrated Windows Authentication. Also, you need to configure the browser of your users to enable the single sign on (otherwise the Nagios Authentication server will prompt the request for the Domain account and password). You can find an example of how to configure IIS and Internet Explorer for single sign-on on my web site (that tutorial refers to IIS 7.0 but many of those concepts can also be applied to IIS 6.0) : http://www.alan-pipitone.com/show.php?id=1

Also note that in the source we didn't set any username and password for ldap connection (in the method new DirectoryEntry(ldapPath) we didn't set any credential). Our code runs under Network Service account (the Identity of our Application Pool) so also the ldap queries run under Network Service account. In fact Active Directory by default allows any computer accounts to run LDAP queries. So, if your server has joined to the domain the Network Service account should be able to execute the queries. In my environment it works.
If you need to use another service account for ldap queries, read the comment in the source (search **if you need to specify a domain account for ldap connection**) to find how to connect to ldap with username and password.

One note on Domain Groups

As you can see **you can not insert a domain user in more than one nagios group** (I speak of domain groups). To do that you must define in nagios (and in the domain) one group that sees hosts and services of the various groups to which you want to enable the user.

MORE SECURITY:

If you need a strong security (our cookies and other information travels over the network in clear text) I recommend you to configure https for both web sites (Nagios web site and Nagios Authentication web site) and encrypt the two cookies (nagios_group and nagios_account).

These will be our next steps:

1. **Enable https** on Nagios web site and Nagios Authentication web site. In this way all data will be encrypted to avoid packet sniffing.
2. **Encrypt the cookie's value** (for example every 5 minutes) with **random key** shared between Nagios web site and Nagios Authentication web site. In this way the user cannot save the cookie's value to generate new cookies by himself (for example: if he will be delete from nagios domain group he will not be able to use old cookies value to access Nagios web site).

Conclusion

As you can see our authentication method is very flexible and easy to implement. With cookies you can easily enable Active Directory Authentication (and, if you want, other kind of authentication) with all Nagios web interfaces and other web pages. You should only change the source code of web interfaces (or other web pages) to read the cookies.

Remember that you have to access Nagios (from your browser) with its fqdn. In our example *http://nagios.alpha.for/nagios* (or http://nagios.alpha.for/<your_dir>).
If we access Nagios with only http://nagios/nagios all our system doesn't work. Because it can't read cookies and you will **run into a loop of redirects**.

In our enviroment environment all this work like a charm. If you cannot make it work you are probably doing something wrong or I forgot to write something :)

I hope you enjoyed this article (and that it has been helpful). This work is free and open source (it has no warranty), so if you notice anything wrong on this article, please tell us how to fix it (if you can) and not just criticize us. We worked to this project at home. In our workplace we have only configured it.

Thanks for joining us.

Alan Pipitone and Fabio Frioni http://www.alan-pipitone.com/