Paper 231-28

Security Control System with SAS® Application Dispatcher

Haidong Tang, Xiao Ji, Guofen Hu
Data Strategies Dept., Shanghai Baosight Software Co., Ltd., China
Xinyu Liu
System Innovation Dept., Shanghai Baosteel Co., Ltd., China

ABSTRACT

This paper aims to introduce a practical way of security control with SAS application dispatcher. The method takes macro variables, such as _RMTUSER, transferred from Web server to decide whether user has the access right to specific resources. To take most advantage of the application dispatcher, its original architecture should be changed slightly. It will not only show the insights of designing the security control system, but also discuss some security issues concerning the architecture itself. Besides, the paper will introduce briefly in Shanghai Baosteel enterprise data warehouse system a real-world security control system, which is based on the described method.

INTRODUCTION

SAS application dispatcher available with SAS/IntrNet software has proved to be an efficient and flexible Web authoring tool. In order to share various SAS generated reports to different customers, security control might be a big issue. Usually resources may include html report files (static reports) and application programs (dynamic reports) dynamically called by application server. Therefore it is important to control these two kinds of basic resource.

An authenticated SAS Broker transfers some environmental variables from Web server to application program. These variables, such as _SRVNAME, _RMTADDR, _RMTUSER, appear as macro variables in application program. Then the system can take some of these macro variables to implement security control. This paper will take _RMTUSER as a basis to develop security system.

SYSTEM ARCHITECTURE

1. OVERVIEW

In order to satisfy the need of security control, the original architecture of SAS application dispatcher should be updated slightly. A new architecture has been illustrated in Figure 1.

There are two subsystems which have been added to the new architecture. One is the resource granting subsystem, the other is the authenticating subsystem. The resource granting subsystem is an independent application, while the authenticating subsystem is embedded in application server. The illustration is a simplified one, though it can be a little bit more complicated according to real situation.

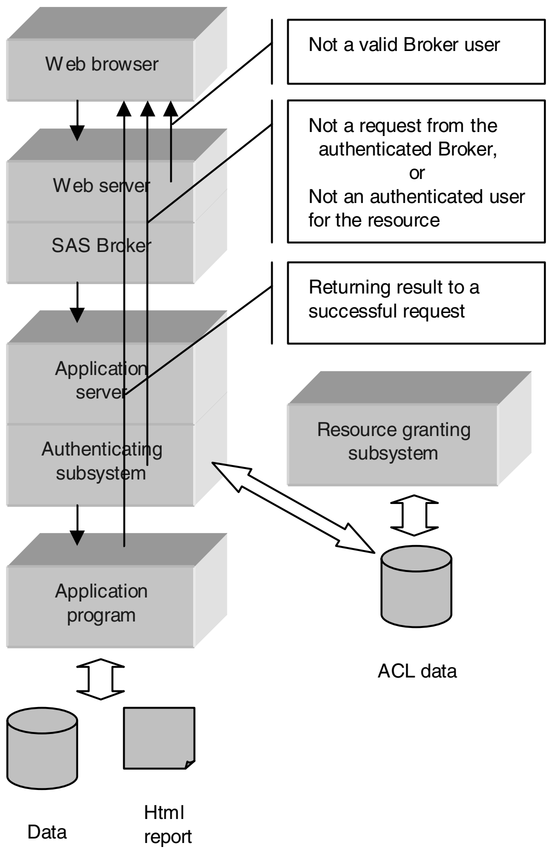
Web browser
Web server
SAS Broker
Application server
Authenticating subsystem
Application program
Data
Html report
Not a valid Broker user
Not a request from the authenticated Broker, or Not an authenticated user for the resource
Returning result to a successful request
Resource granting subsystem
ACL data

Figure 1

2. RESOURCE GRANTING SUBSYSTEM

This subsystem is to grant users the access right to various resources in the application systems. These resources can be either html report files, or application programs.

Application program can be a SAS program, a source entry, a SCL program, or a compiled macro. To clearly define an application program as a resource, we prefer a full path registration. So we use SC_PROG.INIT.READHTML.SCL instead of READHTML.SCL. Therefore the full path program name will be captured in macro variable _PROGRAM later in authenticating subsystem.

It is simple to define html report files as resources. However, it is actually a little bit complicated in authenticating subsystem. Because there should be a specific application program to read and present it once a valid user sends a request. The specific program name in macro _PROGRAM, together with a user-defined _RPID, should be used in authenticating subsystem.

The granting subsystem finally provides an ACL (Access Control List) table, which may be used by authenticating subsystem while users require a specific resource. Below is a simple ACL table structure.

```
resource_name char(36),
/* can be report id or program name */
resource_spec char(60),
/* specification of the resource */
userid char(36)
/* user who has access right, will be checked
against  _RMTUSER */
```

3. AUTHENTICATING SUBSYSTEM

This subsystem is to authenticate the access right based on ACL data while users' requests arrive. Once a request comes, the authenticating program takes _RMTUSER and _PROGRAM, and _RPID if it requests html report, from application server. Then, if the specific _RMTUSER and resource (_PROGRAM or _RPID) exist in ACL, which means the user has the access right, the application is invoked. Otherwise a specific web page returns to prompt a failure access.

The traditional way to authenticate access right is to call a public routine in the application program. So the developer may consider about security control during developing. But actually there is a sneaky way of implementing this without any security control considerations during coding.

SAS application server has a REQUEST INIT statement, which allows some programs run first before the requested program runs. For example,

```
proc appsrv port=5050 adminpw='pass';
  allocate library pub
'/home2/dev/intrnet';
  proglibs pub;
  request init=pub.init.sas;
run;
```

Therefore all authentication rules can be implemented in pub.init.sas. Such program checks the ACL first. So if it is an authenticated request, the requested program runs. If not, the pub.init.sas returns directly a Web page to user prompting a message of denying, and the requested program will not to be invoked.

For static report, a specific program should be introduced to read and present the html file according to its physical path. A simple program PUB.AUTHEN.READHTML.SCL for reading html file has been shown below.

```
filename in "/home2/dev/rp0001.html";
data _null_;
  infile in lrecl=512 pad end=end;
   file _webout;
   input s $512.;
   put s;
run;
```

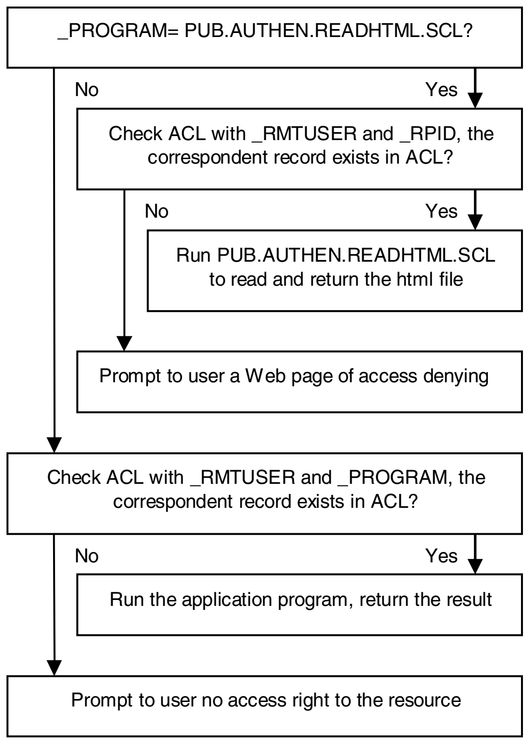
_PROGRAM= PUB.AUTHEN.READHTML.SCL?
No
Yes
Check ACL with _RMTUSER and _RPID, the correspondent record exists in ACL?
No
Yes
Run PUB.AUTHEN.READHTML.SCL to read and return the html file
Prompt to user a Web page of access denying
Check ACL with _RMTUSER and _PROGRAM, the correspondent record exists in ACL?
No
Yes
Run the application program, return the result
Prompt to user no access right to the resource

Figure 2

To put it into production, the html report ID should be transferred from a macro variable _RPID. So once pub.init.sas learns the _PROGRAM with PUB.AUTHEN.READHTML.SCL is requested, it will check the ACL with _RPID as the resource name.

The whole authenticating process in pub.init.sas can be illustrated in Figure 2.

4. ADVANTAGES OF THE ARCHITECTURE

The security control system based on such architecture takes most advantage of Web server and SAS application server.

a) Do not need particular applications to maintain user's information. The Web server handles all concerning user ID, password, password encryption, password updating, user session control, etc.
b) Should pay no attentions to security control while developing application programs. User validation will be handled automatically in application server as a whole.

SECURE THE ARCHITECTURE

1. STRENGTHENING THE COUPLING BETWEEN SAS BROKER AND APPLICATION SERVER

The SAS Broker and application server can be flexibly deployed on different machines, which is known as a loose coupling system. Though it is flexible, it brings about security problems. For example, user can easily fake a user ID by _RMTUSER from a different Web server to invoke the application program.

To avoid the problem, we should strengthen the coupling between SAS Broker and application server. One effective way is to add a special string in SAS broker and authenticate the string in application server. To set the string in broker, use SERVICESET PASSKEY statement.

```
SocketService default "default socket
service"
   Server 190.2.68.11
   Port 5050
   ServiceSet passkey  "KH#^8694_0+s"
```

In order to authenticate the string in application server, we can still use REQUEST INIT statement. However the scripts of authentication must be put at the very beginning of pub.init.sas. Such process can be illustrated in Figure 3.

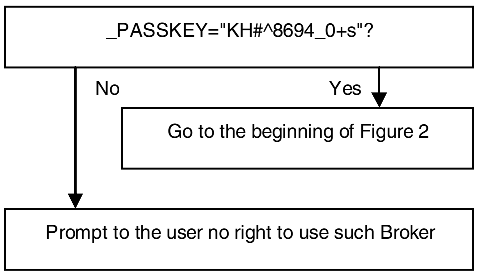
_PASSKEY="KH#^8694_0+s"?
No
Yes
Go to the beginning of Figure 2
Prompt to the user no right to use such Broker

Figure 3

2. OTHER SECURITY ISSUES
In the architecture there are some other common security issues, such as protecting the Broker configuration file, supplying a password when starting the application server, etc. All these can be referred in the document of SAS Application Dispatcher, Release V8.2 for details.

A REAL-WORLD CASE OF SECURITY CONTROL

Shanghai Baosteel enterprise data warehouse has been developing since 1999. There are around 2000 daily reports generated on IBM S85. These reports can be either static reports or dynamic reports. Users from different department may have different access rights to all these static and dynamic reports.

In order to grant the resources easily, we've added user groups. Therefore resources can be granted to groups, and/or users. As the resource granting subsystem is Web-enabled, it is very easy for our customers to manage the granting subsystem themselves.

After the security control was put into production, we've got many positive feedbacks from our customers.

CONCLUSION

The security control architecture discussed in this paper takes the most advantage of Web server and SAS application dispatcher. As the real situation may be varied, the implementation of security control can be varied. However this architecture can be the basis of developing a more complicated security control system.

REFERENCES

SAS Application Dispatcher Document, Release 8.2 - SAS Institute Inc., Cary, NC
SAS Online Document, V8 - SAS Institute Inc., Cary, NC

CONTACT INFORMATION

The authors can be contacted at:
Haidong Tang, Xiao Ji, Guofen Hu
Shanghai Baosight Software Corp., Ltd.
515 Guoshoujing Rd., Pudong District,
Shanghai 201203
P.R.China
Email: tanghaidong@baosight.com
jixiao@baosight.com
huguofen@baosight.com
Xinyu Liu
Shanghai Baosteel Co., Ltd.
Shanghai Baosteel Headquarter, Guoyuan,
Fujin Road, Baoshan District,
Shanghai 201900
P.R.China.
Email: liuxy@baosteel.com