Global Information Assurance Certification Paper

Name: Nancy Loveland
Version of Assignment: GSEC Practical Assignment v.1.4
Descriptive Title: Single Sign On Through Password Synchronization

ABSTRACT:

This paper is a case study on a project to provide a Single Sign On (SSO) solution to web based applications that use the mainframe as the data store. The paper begins with a thorough description of the high level business requirements. There is a high level overview of Single Sign On and the underlying technologies. The reader is taken through the steps of developing the detailed technical requirements, researching products, trialing products, product selection, customization, implementation and after implementation review.

This paper should provide guidance to other security professionals faced with providing a Single Sign On solution. The Single Sign On solution covered is based on password synchronization. The lessons learned from managing this project are shared, as well as the additional benefits of our password synchronization solution.

Introduction:

This paper is a case study on a project to provide a Single Sign On solution to web based applications that access mainframe data through remote procedure calls (RPCs). The stage is set with a description of our environment and the business objectives of the project. The action section details the steps taken in building requirements for the solution, researching, trialing products and finally implementation. The final section describes the results of the implementation, lessons learned and other benefits realized through this solution.

SETTING the STAGE

My company is a large service bureau running in-house developed applications. The client's data resides and is processed on a mainframe. We provided a (Graphical User Interface) GUI to the client's mainframe data through fat client applications that allowed them to run queries, setup work flow, review reports and other features. In order to stay competitive in today's electronic world, my company decided to develop an application to provide a web based delivery of these GUI applications. This application was to serve as a portal to our other applications by providing security and navigation through a single unified interface. The Portal application needed to provide a Single Sign On for the end user for all of the applications it supported, including the mainframe.

My task was to find a solution to provide end user Single Sign On access to this Portal application. The solution had to be low in cost, have a low impact on the operations environment and be secure.

There are two schools of thought when it comes to Single Sign On; tokens

versus password synchronization.

SearchSecure.com, a target site for security professionals defines Single Sign On as this:

> In any **client/server** relationship, single signon (pronounced SING-uhl SAIN-awn) is a session/user **authentication** process that permits a user to enter one name and password in order to access multiple applications. The single signon, which is requested at the initiation of the session, authenticates the user to access all the applications they have been given the rights to on the server, and eliminates future authentication prompts when the user switches applications during that particular session.

The different methods of providing Single Sign On are:

- Kerberos Protocol (this is what I call “true Single Sign On”)
- Password Synchronization

The first method to provide Single Sign On is using the Kerberos Protocol. This is where a user authenticates to an authentication server that creates a token (or ticket). This token is actually sent to the application which can recognize (or trust) the token and the user is granted access. There is a great explanation of the Kerberos protocol in a paper written by Gary Tagg, titled “Implementing a Kerberos Based Single Sign-on Infrastructure” which can be found at http://www.chi-publishing.com/isb/backissues/ISB_2000/ISB0509/ISB0509GT.pdf.

The second method of providing Single Sign On is by implementing password synchronization where the password is captured and then sent to the different applications so the user can access many applications without having to sign on more than once. In order for this to be a secure solution, the password that is captured must be stored in volatile memory and be passed encrypted via secure channels.

Taking into consideration the requirement of the solution having a low impact on the environment, the Kerberos solution was not an option. The Kerberos solution would require a great infrastructure change, which would be expensive, and would require our applications to be “Kerberised”. Our applications access the mainframe via remote procedure calls which require an actual user-ID and password be sent to the host for authentication. This made it clear to me that my focus should be to find a password synchronization solution in order to provide sign on to our new application.

The objective was clearly defined. I needed to research and implement a password synchronization solution that would provide Single Sign On for our end users to our Portal application.

The challenge was now in place to determine a solution which would balance cost and security. There was nothing budgeted at that time for a password synchronization tool. My solution had to comply with our company’s high security standards as well as my own standards as a security veteran of 14

years.

HOW IT WAS DONE

This section will describe the step by step process I took to meet my objective of researching and implementing a password synchronization solution for our Portal application.

The steps were:

- Define requirements
- Research
- Conduct a proof of concept
- Make a product selection
- Customize and test
- Implement solution
- Perform Implementation review
    - Validate success criteria
    - Compile lessons learned

Define Requirements

We compiled a list of initial requirements for the solution. Our initial requirements list is shown below.

Initial Requirements:

- Must be able to synchronize password between the following platforms
    - RACF
    - Top Secret
    - ACF2
    - Microsoft NT
    - Novell LDAP server
- Must be able to map user-IDs from one platform to another
- Must provide password status synchronization between platforms
- Must be able to enforce host platform password format rules across the various server platforms
- All synchronization should be bi-directional

Since a password synchronization product was not budgeted at the time we needed to implement, we looked at the possibility of developing it in-house. We are an IT company with the talent required to do in-house development. We scoped the project to determine the feasibility of in-house development of the Single Sign On solution. Initial development costs did not prohibit in-house development but the ever changing technologies would require on-going development. Support would be limited by the developers that wrote the solution, since it would be very complex and span multiple platforms and operating systems. The decision was made to look at vendor products. The

next step was to research vendors that provided password synchronization, which would be our solution to the Single Sign On requirement.

Research

Initial research was done on the internet to find vendor password synchronization tools that would fill our absolute requirements. Rutrell Yasin's article, Password Pain Relief, in the April 2002 edition of Information Security lists many of the password synchronization tools on the market today http://www.infosecuritymag.com/2002/apr/passwordmgmt.shtml.

We compiled a list of vendors that provide Single Sign On products. A new list of requirements was created to include absolute requirements and other evaluation criteria. Our revised list is shown below.

Initial Requirements:

- Must be able to synchronize password between the following platforms
  - RACF
  - Top Secret
  - ACF2
  - Microsoft NT
  - Novell ldap server
- Must be able to map userids from one platform to another
- Must provide password status synchronization between platforms
- Must be able to enforce host platform password format rules across the various server platforms
- All synchronization should be bi-directional

Other Evaluation Criteria:

- Cost
- Support for Lotus Notes
- Support for Tandem
- Support for other security platforms
- Transparency to application development. How many changes are required to our application to support the product? Does it require that users be defined in server security platforms in a particular way? Are API calls required to interface with the product, or does it execute as a background task?
- Can it force synchronization of host expiration period onto server platforms?
- Ease of ID mapping. It is expected that we will be required to write the ID mapping exit. On what platform does this reside, and how difficult is it to develop?
- Flexibility of ID mapping. Is any one platform required?
- Ease of installation
- Ability to recover if a platform is unavailable
- Can synchronization occur between server platforms (Novell & NT) if

there is no ID on the host?

After comparing the list to the information provided by the vendors, three products were selected to install for proof of concept (trial).

Two of the products were discovered to be the same product marketed by 2 different vendors and the vendors decided which one we would work with. The remaining 2 vendor products were as follows:

- Blockade Synchronization Services - http://www.blockade.com/products/pd_products_pss.html
- Schumann SAM Password Synchronization (now Systor Security Solutions, Inc) - http://www.systor.com/en/index/core/core_sam_home/core_sam_prod_home/core_sam_prod_samps.htm

The next step in the project was to notify the vendors or our interest in their product and invite them to bring their products to our company to do a proof of concept (trial).

Proof of Concept (Trial)

We created a test plan so that each vendor would be equally evaluated. Below is the test plan we used.

Test Plan

- Install the product.
- Define five user-IDs for testing. Three of the IDs will be defined in Novell, NT and RACF. The NT and Novell ID for a user will be the same. The RACF ID will be in our standard format. One ID will be defined in NT and Novell but not RACF. The IDs will be the same. One ID will be defined in NT and Novell but will be different (to test ID mapping on server platforms).
- With the RACF IDs, sign on to RACF and go through a normal user password change. Then sign on to NT and Novell and ensure that the change has been propagated.
- Sign on to NT and go through a normal user password change. Then sign on to RACF and Novell and ensure that the change has been propagated.
- Sign on to Novell and go through a normal user password change. Then sign on to RACF and NT and ensure that the change has been propagated.
- Disable an ID through invalid passwords to RACF. Ensure that the status is propagated to all platforms. Repeat the process by signing on to NT and Novell.
- Administratively, enable and disable an ID on each platform. Ensure that the status is propagated to the other platforms.

- The rule in RACF is that all IDs are disabled after the 3rd invalid password. Ensure that this rule is propagated to all platforms.
- Set the password expiration period for a RACF ID to 1 day. After the expiration date has passed, test the NT and Novell platforms to see if they require a password change.
- Make the host unavailable to the server and change a server password. Ensure that the change is propagated to the host when it becomes available. Repeat the same from the host. Repeat the same between NT and Novell.

The hardware and software requirements were created and a trial environment was created.

Both of the products were brought in house for the proof of concept (trial). We created an issues list for each product and reviewed it in detail with each of the vendors. It was a learning experience. We learned neither product would provide the following:

1) Disabling access after 3 invalid password attempts because this password policy was being used on the mainframe instead of the Portal user store.
2) Synchronize expirations on each user store with the Host user store.

We realized we would have to program our Portal application to compensate for each of these shortcomings. These new programming requirements were passed to the portal development team.

The final evaluation was made and a product was selected.

Product Selection

Blockade Synchronization Services (re-branded to ManageID Syncserv in July 2002) by Blockade Systems Corp was chosen as the product that would provide our Single Sign On solution because it best met the absolute requirements, additional requirements and testing requirements. Blockade Systems Corp's responsiveness was also a deciding factor.

The next step was to start customization and testing. Between our project steps of product selection and customization and testing, the Portal product made a change in the platform they were going to use for end user authentication. Our project adjusted to accommodate the change from Novell Directory Server to the Netscape Directory Server. Since Novell Directory Server seemed to be the most difficult platform to provide bi-direction password synchronization, we felt this change made our project easier. We obtained Blockade System Corp's password synchronization agent for the Netscape Directory Server platform (modified by Blockade Systems Corp to handle a custom attribute we were using) and began our customization and testing phase of the project.

Customization and Testing

Customization was needed in order to accomplish ID mapping for the Portal application. ID mapping is defined as the ability to synchronize the password and password status for one ID on one platform to one or more different user-IDs on one or more other platforms. An example would be having user-ID NLOVELAND on NT and having the ability to synchronize the password for user-ID NLOVE on the mainframe.

We looked into which platform would be best suited for housing this ID mapping functionality. Our options were to put the ID mapping on NT or on Unix Systems Services (USS) on the mainframe. Even though we didn't have a great knowledge base on USS at the time, we chose it to house our ID mapping process since we were already using USS for the Blockade IP Connector (brain). USS would be a lower transaction time than putting it on NT, where we would be faced with additional troubleshooting and availability challenges.

The requirements for the ID mapping exit were defined and given to a development group. The ID mapping exit was developed using C++ using the sample code provided by Blockade Systems Corp as a starting point.

In order to fully test the customization, we created a test environment that best mirrored our production environment. The test environment was extensive because Blockade Synchronization Services has several components. Blockade Synchronization Services (BSS) agents only need to be installed on server components that host the platform security systems. There is no need for client/desktop software to be installed. Along with the different agents Blockade Synchronization Services has an ID mapping exit point and the part that connects it all is called the IP Connector. DB2 was chosen as the data store for the ID mapping table.

The platform scope of our project was to install a BSS agent on the Netscape Directory Server, install a BSS agent on multiple mainframe LPARS, install the IP Connector on USS on one of the mainframe LPARS, install the user-ID mapping exit and work out the communications between USS and DB2, where the ID mapping data store resides.

Our testing was somewhat limited because we could not totally mirror our production environment. The limitations were taken into consideration as we tested.

After testing was complete the next step in the project was implementation.

Implementation

We documented everything we did in the test environment and created a project plan to recreate everything on the production environment. Our synchronization spanned multiple areas of expertise and the tech support staff in each area had to be coordinated to perform their tasks according to the project plan. The key to the success of any implementation is to write extensive documentation in the testing phase and have a clear project plan and clear communications with all of the areas involved.

Our product implementation into production was a success. The password synchronization solution was installed and ready for the Portal application implementation.

AFTER PRODUCT IMPLEMENTATION

Validate Success Criteria

The real test of whether the implementation of our password synchronization solution was a success was when the Portal application was installed in production. We learned many lessons in the implementation and discovered other benefits.

Initially we found out our solution fell a little short.

Our multiple LPAR configuration with one IP Connector configuration limited the password policies to be checked only on the system running the IP Connector. We realized we overlooked this requirement and presented the problem to Blockade Systems Corp. Blockade Systems Corp quickly provided the functionality we needed in the newest version of the IP Connector they were developing. We upgraded to the newest version and our problem was solved.

The second problem we encountered was the inability for Blockade Synchronization Services to handle a non-expiring password change. We did not test every possible password parameter on an administrative password reset. This was a shortcoming in our test plan. It is considered a best practice to have a policy where an administrator must expire a password when it gets reset however, this policy is hard to enforce in a service bureau environment with a decentralized security model. This shortcoming does not have a great impact on us; however, it is mentioned in this paper as information in case this is a requirement for someone else doing a similar project. The problem has been reported to Blockade Systems Corp. and is in the product management department for consideration in a future release.

Once the password policy problem on multiple LPARS was solved, the project was considered a great success. Blockade Synchronization Services product with added customization provided the Single Sign On solution for end users into our Portal application.

Compile Lessons Learned

There were many lessons learned during this project and I share them in hopes that anyone doing a similar project can use them.

Have an accurate test plan. Test all possible combination of parameters and ensure the desired results. Test every platform and combination of platforms in your project scope.

Choose a vendor solution that covers a variety of platforms, not just the platforms in your scope. This will make you better prepared in case your scope changes due to events outside of your control. We learned that we made a good choice in products when the portal project user store was changed from the Novell user store to the Netscape Directory Server user store. We were able to immediately install, test and implement a new BSS agent.

Keep communications open with the vendor and your internal project team members. We learned the benefit of keeping in close communications with Blockade Systems Corp and our internal project team members. Be sure to set expectations early on.

Set up automation to trap and do notification in case of any problems with any part of the solution.

Document and train the support staff in how to trouble shoot problems.

Other Benefits

A great benefit installing Blockade Synchronization Services was that it was a way to ensure password policies across multiple platforms. Our solution ensured that the written policy was being enforced automatically with minimum policy configuration.

Another benefit to password synchronization is that it only requires the user to remember one password no matter what system they log onto. Limiting the number of passwords a user must remember reduces the risk of the user writing the password down and reduces the overall helpdesk costs.

Summary

This paper covered the major steps in the project to select and implement a Single Sign On solution for a web based portal application communicating to mainframe data via RPCs. There is a brief description of Kerberos technology vs. password synchronization to provide a Single Sign On solution. The steps in requirements, research, and trialing, selecting, customizing and implementing the solution are defined. And finally the project was considered a success and lessons learned and other benefits were covered.

References

Blockade Synchronization Services [Computer Software]. Blockade Systems Corp. http://www.blockade.com/products/pd_products_pss.html (May 12, 2002).

Connolly, P.J. “Single Sign-on dangles prospect of lower help desk costs.” September 29,2000. URL: http://www.infoworld.com/articles/es/xml/00/10/02/001002esnsso.xml (May 12, 2002).

Crume, Jeff. Inside Internet Security What Hackers Don’t Want You To Know. Great Briain: Pearson Education Limited 2000, 2000.

SAM Password Synchronization. SYSTOR. http://www.systor.com/en/index/core/core_sam_home/core_sam_prod_home/core_sam_prod_samps.htm (September 2, 2002).

Tagg, Gary. “Implementing a Kerberos Based Single Sign-on Infrastructure.” Information Security Bulletin. November 2000. URL: http://www.chi-publishing.com/isb/backissues/ISB_2000/ISB0509/ISB0509GT.pdf .(May 11, 2002).

Waynforth, Chris.”Single Signon.” SearchSecurity.com Definitions. February 14, 2002. URL: http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci340859,00.html (May 12, 2002).

Yasin, Rutrell. “Password Pain Relief.” Information Security. April 2002. URL: http://www.infosecuritymag.com/2002/apr/passwordmgmt.shtml (May 11, 2002).