Global Information Assurance Certification Paper

Distributing Passwords for new Applications

By Ankit Singhal
Version 1.4, Option 1 for GSEC

Abstract

During the deployment of any password based application with a large user base, the problem of distributing passwords "securely" becomes one of the major issues. This is even more so if the application to be deployed has more than a few thousand users and the users are spread throughout the world. This paper discusses the various options available to the deployment team, which they can use to securely distribute the passwords for applications. Based on a number of factors, the deployment team can choose any one of the methods listed.

These methods are not limited to the distribution of passwords. Some of them can be used in the distribution of Private Keys, digital certificates, etc. as well. The comparison between various methods for password distribution will help the deployment team in choosing the method best suited to them. The method of choice will, however, depend on a number of factors such as the number of users, distribution of users, over all security required, tools available etc.

Challenges that the Deployment team has to face

Although, technologies like Biometrics and Smart Card based authentication are the cutting edge in secure authentication, we cannot forget the age old authentication method "passwords". Ninety nine percent of the applications requiring authentication are still password based. Even smart cards have a PIN (which is essentially a password) to access the information stored on the card.

The team deploying any application is usually working under a lot of pressure. The Deployment Team is not only responsible for setting the application correctly and ensuring that everything works properly. The team deploying an application (which uses passwords for user authentication) has to take into consideration the following factors as well.

1. The Deadline: This is the most important of all driving factors. Every deployment team has to put in whatever it takes to deploy the system before the deadline expires. Although the time available for deploying the system depends on the kind of system being deployed, but in my experience, it is never enough.
2. The Budget: This is the second most important (if not the first) consideration for the deployment team. They have to complete the deployment and do this in the minimum cost possible.

3. The nature and distribution of users: Based on the nature of application, there may be 10 to 100,000 users. They may be housed in a single room, they may be in the same building or they may be located throughout the world, spread over 120 different countries. The nature of these users is also important. A user base of 50 high level officials (CEO, VPs, etc.) may be more demanding than a team of 1,000 data operators.
4. The overall security of the system: Some systems require the highest security. A good example would be the Financial system of a Bank, which, if compromised could spell doom for not only the bank but a large number of other organizations. On the other hand, there may be applications that may not require such a high level of security. A good example may be your hotmail account. The level of security needed is directly proportional to the value of data handled. Not many would be willing to spend $100, just to have more secure access to their web email account. But U.S. Government will spend millions of dollars, just to have a high security system to protect their nuclear weapons.
5. Would the password be used for any other application: Many companies have a corporate directory, the password for which is used to log on to various applications. Considering the growth in the number of passwords a user has to remember and various Single Sign-On (SSO) Services, the Deployment team may have to consider whether the same password will be used to authenticate the user to other applications.

The User Thought Process

Most of the time, the users are concerned with convenience rather than security. They consider security as a hurdle that they have to pass in order to do their job. For instance, a very few employees think about the importance of identity cards. To them it is just a key to get inside the building. If the guard at the gate did not check those IDs and refuse those who did not have one, no one will bother to carry their badges to work.

We see the evidence of this carelessness towards security all the time. Passwords are written down on stickons which are placed on the computer monitors. User passwords are shared among friends to watch out for each other. I once overheard a coffee room discussion where a person was telling his friends about all the funny passwords he uses for all his logins. If not enforced by the administrator, hardly anyone changes his password regularly.

In order to manage the multitude of passwords, some users have password remembering softwares like Gator, Vego etc. which maintain a repository of the passwords and automatically log you on to password based application. This may again be a security risk if your password repository is not sufficiently encrypted or if someone finds your windows password.

Unlike the above type of user, who unknowingly compromise security, there are a few insiders who deliberately try to defeat it. This, often invisible, inside threat comes in many forms. It could be a disgruntled employee who has received a bad work review and wants to lash out at the company by deleting files or changing information. It could be someone who is struggling financially and has been offered thousands of dollars to e-mail or print out classified information. Or it could be a worker breaking into files to change payroll numbers. *(from" The enemy within", by Sharon Gaudin[1])*

> In their annual joint security survey, the CSI (Computer Security Institute) and the US Federal Bureau of Investigation (FBI) reported that during the year 2000, almost 80% of network abuse and security breaches occurred within the companies surveyed, rather than through external hackers. Of the participating companies that could quantify a financial loss due to security breaches, a total of more than $265 million dollars was reported. Almost 90% of the companies surveyed admitted to being hacked, yet only 25% of those reported that they had been hacked externally. *(from "Focus on Internal Security", by Jon Tullett [2])*

> Industry analysts estimate that in-house security breaches account for 70% to 90% of the attacks on corporate computer networks. And the percentage is probably even higher than that because most insider attacks go undetected. In fact, Dennis Szerszen, director of security strategies at The Hurwitz Group in Framingham, Mass., says for every in-house attack reported, there could be as many as 50 that go unreported or undetected. *(from "The enemy within", by Sharon Gaudin[1])*

Considering this, the Application Deployment team has to come up with a suitable way of distributing passwords. It has to be convenient enough and secure enough to deter both kinds of users. Only distribution of these passwords will not solve the problem. The user must be forced to change their password at least the first time they log in.

Methods for distributing passwords

Considering all the factors mentioned so far, here are a few methods that may be used in order to securely distribute the passwords for users. Note that each method has its own advantages and disadvantages. Decision as to which method is the best to use depends on a large number of factors.

A. Have a password based on public knowledge

The user accounts may be created with an easy password. For example it might be the same as the login id, it may be the last name of the person, it may be his date of birth, etc. This is the easiest of all methods to distribute the passwords. You just have to circulate a memo about this and every one will know what his password is. This method is cheap and the fastest. The only thing that has to be enforced is that the user changes his password once he logs in for the first time.

Changing password at first login can be enforced for UNIX or Windows Accounts easily, but when it comes to applications, this capability must be built into the application itself. Unfortunately, a very few applications prompt the user to change their passwords. Most of the time, the user continues to use the original password and never bothers to change it.

Another security breach occurs when someone else logs in to the account before the user himself. The unauthorized user can do whatever he pleases at this point. Most applications require such a higher level of security and so cannot allow this. For example Banking, insurance, finance, etc. just cannot have this kind of setup.

This scheme may work for less secure applications but for applications that require a high level of security, this is not a solution.

B. Go to them individually

This is the most basic form of password distribution. The administrator has to go, meet the users personally and give them their account information. This is one of the easiest methods to distribute passwords but is applicable only when the application has to be deployed for a very few number of users and the users are located in the same facility. This becomes prohibitively expensive and slow when the number of users increases or the users are spread beyond one building.

This method provides the most secure way of distributing the passwords. The administrator can make absolutely sure that he gives the password to the right person. In other words, he can verify the identity of the user. This is the method used for distributing passwords and codes in nearly every Military Installation.

C. Have more System Administrators.

If the application is such that it will require a system administrator at each office, then we can still distribute the passwords individually. This is just an extension of the previous method. But by increasing the number of system administrators, we have increased the cost. Also, more administrators mean a bigger security risk. If one of the administrators turns against the system, there is no one to stop him. But if the nature of the application requires the presence of an administrator at each office, this is probably the best method to use.

Most of the companies deploying Public Key Infrastructure (PKI) have a Local Registration Authority (LRA) that is responsible for verifying the users and distributing the codes to them. Although they add to the cost, they are a necessary part of the whole PKI system.

D. Mail them

The deployment team can use the facility of the age old Postal system and send out sealed letters containing the account information. They just have to print these individual letters, place them in the right envelop, mail them and hope that they reach their destination in time. This is a very slow and expensive method of distributing passwords. Consider if 1000 users have to be deployed in 4 countries, one in each continent. The cost of printing a 1000 letters and the cost of postage and handling will come out to be $10-15 per user. This is not bad, but you have to consider the security issues. The mail may be easily stolen or wrongly delivered.

Even if the mails are delivered correctly, it is still going to take a long time to send the password to a user, especially if the users are in Siberia, Sahara, Himalayas, etc. For example, just think of how much time would it take for Shell (the oil company) to send out passwords to all the people in all the desolate places where oil is found.

This method is commonly used for distributing passwords by Internet Service Providers. When you sign up for an internet account, the will usually mail you your login name and password. Most of the time, if you request a password change, they will mail you your new password instead of telling it to you over the phone.

E. Call them up

This is a much more practical method if the user base is sufficiently large and the users are spread out. The deployment team can make telephone calls to people individually and tell them their passwords. This is what most of the administrators do when it comes to support. People, who have forgotten their passwords, call up the help desk and ask them the new passwords all the time. For a deployment of 500-1000 users spread over different continents, the cost will come out to be around $15-20.

On the downside, everyone may not have a phone. Also they may have a shared line or the phone may be located in a common work area, where they may not be alone. The phone lines may be bugged, etc. Telling the password on the phone, in these cases is not the most secure way of giving out the password.

Moreover this does not have authentication built in. You can never be sure whom you are talking to on the phone. A person can easily impersonate someone and get the password. Thus it is not entirely secure.

A very common social engineering method to break into systems is to call a help desk, tell then you are user XYZ, you have forgotten your password and ask them to change the password to something else. This works even better if you impersonate someone in the top management. Then the help desk support will not even ask you for verification.

F. Send out an E-Mail

Most of the computer users now have email. It has revolutionized the work place and paper less offices are not a very distant dream. They are considered to be a secure method of disseminating information. It is fast, convenient and cheap. So passwords can be easily sent via email. But email is not the most secure.

It is very common for people to have shared terminals for downloading and checking their emails. The mails are stored unencrypted on these machines and can be easily accessed by prying eyes. One of the very common security breaches that are observed in the Middle East is the internal sharing of e-mail passwords. Executives may ask their assistants or colleagues to access e-mails or other information for them without realizing the inherent security risk they are taking.

SMTP by itself provides neither authentication, nor encryption. Anyone sniffing the network can get all the information in the email. As the mails are stored unencrypted on the mail servers, any one who has sufficient access to it can read the mails. What is more, basic SMTP does not verify the sender either. So a hacker can easily trick someone into believing that the mail came from someone else.

G. Send out a secure email

Basic SMTP is insecure, but the security of the email can be enhanced using PGP. When a user encrypts plaintext with PGP, PGP first compresses the plaintext. Data compression strengthens cryptographic security. Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher. Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis. (Files that are too short to compress or which don't compress well aren't compressed.)

PGP then creates a *session key,* which is a one-time-only secret key. This key is a random number generated from the random movements of your mouse and the keystrokes you type. This session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext; the result is ciphertext. Once the data is encrypted, the session key is then encrypted to the recipient's public key. This public key-encrypted session key is transmitted along with the ciphertext to the recipient. Decryption works in the reverse. The recipient's copy of PGP uses his or her private key to recover the temporary session key, which PGP then uses to decrypt the conventionally-encrypted ciphertext. *(from How PGP Works [4])*

This requires a public key infrastructure to be in place before secure emails can be sent out. Most of the corporate world is still in the primary stage of implementing secure emails. Competing encryption standards has made it even more difficult to implement. Thus sending out a secure email is not always an option. But if available, it is one of the best and easiest methods to use.

H. Using passwords from another application

Most likely the application that is being deployed is not the first password based application for the organization. There must be other application(s) which all or a subset of the users have been using. Examples may range from the windows password used to log in to the domain, email password, Seibel password, etc. The point is that the users should already have some secret that only they know. Most of the times, it is possible to modify an application such that it uses the authentication of some other application. This is like using your windows sign on to access your network resources.

A good example for this may be as follows. Suppose you have a web-based email system. This email system queries a SQL database to authenticate a user. Now you are trying to deploy a web based calendar. So, we can customize the calendar in such a way that it queries the password database for the email system. Hence the user will use the same password for logging in to the new system.

A large number of companies already have a corporate directory, usually based on LDAP. Many applications can be easily customized to get the user authenticated using this corporate directory. We can use the Bind operation of LDAP which is used to initiate a protocol session between a client and a server, and to allow the authentication of the client to the server. *(from RFC 1777 [6])*

This type of customization is not always possible. At the very least, it demands a very good knowledge of the application. More often than not, the applications are off the shelf products which cannot be easily customized.

I. Have a website for getting the passwords

This method again requires the use of some other means of authentication. This is a lot similar to the previous method, but instead of modifying the application to use the password database of some other application, we design a website that uses the password database. Once authenticated, the website accesses the repository of the new passwords and displays it for the user. Thus we do not have to change the applications.

There are, however some issues that need to be taken care of. Simple HTTP traffic is plaintext and so it can be sniffed. Moreover, HTTP does not provide authentication for the server or the client.

An easy way to get the password of hotmail or yahoo account is as follows. Design a website which looks like the login page of hotmail or yahoo and mail the link to the victim. As soon as the victim clicks on the link, he will see the message that he has been logged out of his email account and will be prompted to login again. The unsuspecting user will type in the password which can be stored for later use. You will be surprised at the number of people who can be fooled with this.

This however can be easily solved using Secure Socket Layer (SSL). The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP or IMAP. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection. *(from "Introduction to SSL" [7])*

The passwords at the server also need to be stored safely. If someone could break into the system, he has the complete list of all the user accounts and passwords. It would be advisable to store the passwords in encrypted format and once the role-out is complete, this data should be deleted and cleaned from the hard disk.

If the web server and the old authentication database are not in the same room, then connection between them should also be encrypted.

An example of this is what company XYZ Inc. (I have changed the name due to security reasons) did when they were deploying a new financial system. XYZ Inc. is a multinational in the true sense of the word. It has offices worldwide with a user base of 3,500 users. XYZ Inc. already had a directory service based on LDAP, which was being used to keep contact information for the employees. Each employee had a password which they could use to change the information in their LDAP record. While deploying the new financial system, the deployment team, came up with a website that use SSL and asked the users to authenticate themselves with their LDAP password. Once authenticated, the website queried another database that maintained the user id and the password for the new financial system.

To create such a website, all one needs is Apache Web Server, with the following modules

1. The *auth_ldap Module* for Apache: This is an authentication module for Apache that allows Apache to authenticate clients using user entries in an LDAP directory.
2. *Open SSL for Apache:* OpenSSL is a cryptography toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) network protocols and related cryptography standards required by them.

The steps to set up these modules are very well documented on the web. *(Ref [8], [9] & [10])*

Password Distribution is not enough

No mater what password distribution method you use, it alone will not be enough to provide security. The system can only be made more secure if the users are educated about the need and the method of maintaining that security. Listed below are few basic

steps that an organization should follow so that a reasonable level of security is maintained.

1. The whole concept of passwords is based on the assumption that the password is known only to the user. But if someone else creates the account then this password is known at least to two people (if not more). Thus it is imperative that the user changes his password at the first login.
2. There should be a well defined password policy. A minimum length for the passwords should be specified. The passwords should be strengthened by using upper and lower case and special characters. It is a good idea to have a password checker in place.
3. The management should ensure that the users are not careless with their passwords. If a manager sees that one of the employees has his password written on his desk, the employee should at least be warned.
4. Not only the management, but all the employees should take it upon themselves to maintain the security of the system. If someone asks you for help by checking his account on his behalf, he should be reported to the management.
5. The organization should have a well documented security policy when it comes to Information technology.
6. Require help desk to verify all users prior to assisting and visa versa require each user to verify that it is in fact the help desk on the phone. Either by calling the user or the help desk back or by enforcing a Pin (personal Identification Number) policy that can be verified.
7. Use your company's intranet or bulletin board to post statistics of losses due to network intrusion. Allow your users to read about when a bad guy gets caught. Let them know that the law is on their side.
8. Security is an ongoing process. Technology is in a constant state of change. The more functionality you add, the more measures have to be taken to enhance the security.

Conclusion

The choice of method to distribute the passwords depends on many factors, some of which are nature of the application, number and spread of user population, tools available, services already present and the overall security level required, time and cost.

Based on these factors, the deployment team can make an informed decision as to which method to choose for distributing the passwords.

The method used to distribute the password is very important in the overall security of the system. But it is not the only thing. It has to be coupled with good system administration practices to ensure a reasonable level of security.

References

1. Gaudin Sharon, "The Enemy Within", for Network World, 8th May, 2000, http://www.nwfusion.com/research/2000/0508feat.html
2. Tullett Jon, "Focus on internal security", 25th April, 2002 http://www.itp.net/features/99450469796016.htm
3. Dublin Lawrence, "The Enemy Within: A system Administrator's Look at Network Security", 7th January, 2002 http://rr.sans.org/social/within.php
4. "How PGP works", taken from the document "Introduction to Cryptography" in the PGP 6.5.1 documentation, Network Associates, Inc. http://www.pgpi.org/doc/pgpintro/
5. Setty Harish, "System Administrator – Security Best Practices", 16th August, 2001. http://rr.sans.org/practice/sysadmin.php
6. W. Yeong, T. Howes & S. Kille, "Lightweight Directory Access Protocol", RFC 1777, March 1995. http://www.umich.edu/~dirsvcs/ldap/doc/rfc/rfc1777.txt
7. "Introduction to SSL", Netscape Communications Corporation, 9th Oct, 1998 http://developer.netscape.com/docs/manuals/security/sslin/contents.htm
8. Engelschall Ralf S., "OpenSSL", part of The OpenSSL Project, 1999 http://www.openssl.org/docs/apps/openssl.html
9. Carrigan Dave, "The auth_ldap Module for Apache", Enbridge Pipelines Inc. 1999. http://www.rudedog.org/auth_ldap/1.6/auth_ldap.html
10. Apache-SSL, 1st March, 2002. http://www.apache-ssl.org/