Stadrin

Protecting Access

Everyday we face the problem of securing access to services on our servers. In our personal life we can recognise the other party by their face or some form of ID document. When it comes to electronic access, recognition is harder as the identification may be real or stolen. BY MILAN GIGEL

Static passwords are normally based on an account name and a matching password. These days this is not ideal and so we will look at the authentication mechanism – Stadrin.

For securing the identity of the user we usually use static passwords. The main problem is the way it which most users try to keep their passwords secret. As the passwords are the only factor used for checking authentication, it is important to ensure, that nobody, except the user knows the password or even the account/password combination.

As passwords are usually maintained by the users, we frequently see that they are often weak, mainly based on a names of children, pets, date of birth, etc., so it is quite easy to guess them and so lead to misuse.

Sometimes the situation is made a little bit better and the users selects their passwords from a combination of words with some numeric suffixes or prefixes. Unfortunately it is still possible to find the password using brute force techniques supported by large dictionaries.

Static passwords are only secure if no backup copy is kept, either digitally or written on paper and in some cases this is even on the PostIt note stuck on the monitor.

Even if we have a strong password and it is securely stored, the problem still remains that security may still be compromised due to other factors. We can easily imagine a Trojan capturing keystrokes and sending them to a potential violator or someone sniffing the Ethernet traffic to get the passwords transported by either clear text or encrypted.

Sometimes it is sufficient to have a quick look over someone's shoulder to watch and remember what they type as a password. The situation is similar if we use magnetic strip cards, smart cards or even biometrics devices. This is because the passwords is still a static form while it is being transferred through communication channel to a server. Is it possible to solve this problem?

Sure it is. So have a look at how this can be simply done. Whenever we need to increase security we can use a One Time Password (OTP). We typically find this in banking and other financial sectors where the data has a defined monetary value. Even if the password is stolen, security is not compromised as for each use a new OTP is generated.

We can use several policies of assigning OTPs including pre-printed OTP sheets, specialised software components or even the hardware authentication calculators, which are now the most preferred method.

Additional Securing of OTPs

To increase the level of OTP security it is possible to extend the authentication process to use some additional factors. We usually use PINs and "challenges" for this.

To ensure that the OTP received really comes from a trusted party, the server after receiving the account name sends the user a "challenge" which then takes part in an OTP calculation process.

The challenge is normally transported using the same communication channel, but for increasing safety levels it is possible to use other means of transport such as GSM Short Message Service etc.

Possible Compromise

As a second level of access and authentication, the compromise method of authentication may be used.

The most common system being used is based on a set of predefined passwords (different for each user account) which are organised in a matrix array. This usually called a Grid Card, where each cell is represented by a specific row and column and contains a Quasi One Time Password (QOTP). The server after receiving the account name sends a challenge, as an example, the C4 field and after the user picks the respective QOTP it acts as a dynamic password for authentication.

Available Mechanisms

Rekonix has recently introduced the Stadrin mechanism *http://www.stadrin.com* on a Linux platform using the Pluggable Authentication Module (PAM) architecture. This can aid in the implementing of OTPs on existing systems without any need to modify applications and services.

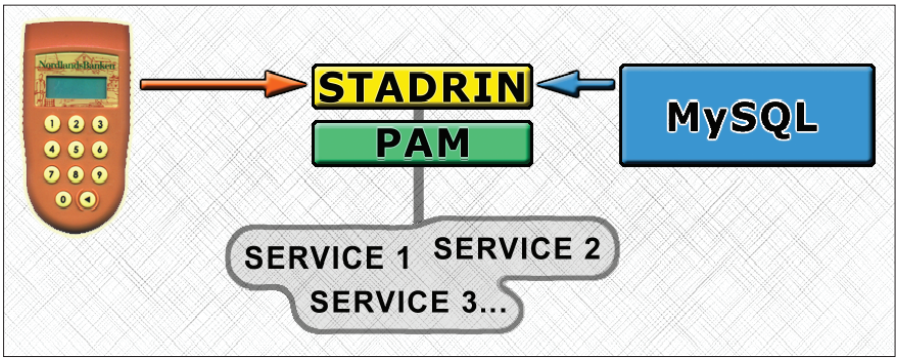

Figure 1: Stadrin architecture

PAM was chosen because of its wide usage. The system is based on using a Stadrin module and Vasco enduser tokens.

Token Cards

For calculating OTPs, the Stadrin module uses a wide range of Vasco tokens. These act in the role of calculation authentication terminals responsible for the whole authentication process on the client side.

Vasco tokens have been around for many years and experience has shown that they are resistant to everyday damage with a long life between servicing. The purpose of the token is to handle the user specific authentication data, and act as a calculation engine. Data used is based on time specific data, unique user specific stored data, manually entered data using keypad and a 3DES algorithm which then displays the current OTP on an integrated LCD. The token itself is protected against unauthorised use by a user changeable PIN, while there is a wide range of models available, we chose to test the DigiPass 300 model.

The Server Side

The server itself implements the Stadrin PAM module which uses a MySQL backend for storing user and token specific data. The information that is stored in the token is handled by a relational database. The admin's job is to make relationships between accounts and services with the token card data. It is possible to assign the same token to several different accounts and so access several different services. We can force the system to use one of the three supported authentication schemes.

Scheme 1

In live running systems providing services and resources there is a possibility of SPWD authentication. This is the standard authentication scheme used in PAM. For the first steps of the implementation process it is possible to authenticate just a selected group of target users using the tokens, while any others can use a previous authentication mechanism. This is very handy for designing mixed authentication schemes and so lowering the implementation costs.

To extend the authentication process so that all users are using the tokens is quite easily and simply done by changing the authentication policy of the selected target users.

Scheme 2

Response Only mode (RO): The first one from the offered authentication schemes using token hardware is the Response Only Mode. In this case the user uses his token for calculating the OTP after its activation, using a PIN code. A specialised algorithm is used for the OTP calculation, which involves several steps starting with fetching token specific info, unique seed values, time stamp and initialisation vectors to be processed by a predefined 3DES algorithm.

The variability of the OTPs is based directly on the timestamp usage within the calculation process, while the generated OTP is usually valid for 38 seconds after the calculation process is finished. This means that the server has to handle the correct time and we can use several mechanisms for this including NTP implementation.

A user turns the token on by pressing a button, enters his pin code and pushes the "1" button for accessing the RO application on his token. The OTP is immediately displayed on the LCD. The tokens are pre-programmed for the Stadrin system default of 8 hexadecimal characters.

Scheme 3

Challenge Response mode (CR): This is the second authentication method using the token backend. This mode builds on the previous method, increasing the security levels and variability of OTPs. The enhancement of this scheme is based on issuing the Challenge by a server PAM module, which is delivered to the user using the same communication channel as the authentication process itself.

Besides the variable timestamp, seed values and initialisation 3DES vectors used in the RO scheme, the challenge also uses, as one of the inputs the 3DES, algorithm used for the OTP calculation process. Input parameters for this challenge are passed to a calculation function with a 3DES backend to get a safe OTP.

The user is granted the generated OTP which is valid for just 38 seconds. From the users point of view the authentication process starts with specifying the account name, receiving the challenge issued by the server, activating the token using the PIN code and entering the received challenge to the token followed by retyping the generated OTP to the authentication system.

Scheme 4

Message Authentication Code (MAC): Apart from the authentication schemes using the token hardware described above, there is also one more authorisation scheme available, supported by the Vasco tokens, which we will describe here. In this case, the authentication process is directly interconnected with the authorisation of transferred data, so done by using just the one schematic transfer.

In one pass the system is able to check the identity of the account in use and validate the consistency of the transferred data throughout the applications transaction data. As within the Challenge Response mode, the challenge is generated on the server side using the specific PAM module, while the user enters, along with the challenge, up to 8 numerical user fields to the token, using keypad.

This means that the token will generate something similar to a digital signature of the transaction, which will include the user's account authentication, with validation data and up to eight system predefinable fields. This is useful in a financial environment.

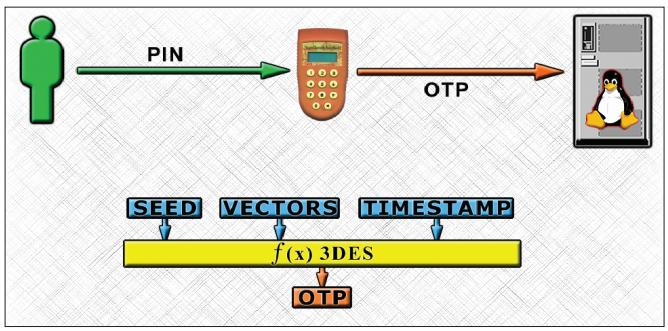

Figure 2: Response Only Authentication scheme

Implementing Stadrin

The installation process of the Stadrin authentication system is quite easy as the package is distributed in the .rpm format. The package requires a Linux kernel greater than 2.2, with a glibc library of 2.2 or better.

Of course the MySQL database is also required and this can reside on the same server, or something more distant, thanks to the support of remote database connection used in the Stadrin backend design.

The current available version of Stadrin is version 1.1, and is distributed on CD ROM for the system and floppy diskette containing the licence information.

So, lets have a look at how the installation process flows. The first step is to mount the CD-ROM drive and invoke the installation process using the rpm package manager:

```
#mount -t iso9660 /dev/cdrom➥
 /mnt/cdrom
#rpm -ivh /mnt/cdrom/➥
stadrin-1.1.rpm
```

The installation script is automatically invoked to start the configuration of Stadrin and the system files that go with its system:

```
Preparing...     ###########
 [100%]
   1:stadrin     ###########
 [100%]
Creating symbolic link for ➥
Vasco shared library
Done.
Install complete.
Configuring database.
```

The first step in the configuration process is to specify the database backend information. Currently just the MySQL backend platform is supported, which, as said, can be present on the same system or on any other server, in which case this can provide us with the possibility of creating central account databases. We have to specify the location and user data first (see Listing 1).

Now that we have entered the basic data into the database, which will be used in the creation of the database storage backend creation process, we can follow on with the next configuration section (see Listing 2). As we have provided the root user information, all the other creation process can be run automatically using the prepared initialisation script.

After following these steps the system is almost ready to run, after providing the licence information and setting up the accounts and PAM system. Before that, the MySQL backend has to be prepared by the init script in order to set the previous definitions, if all is well we get the following message:

```
Creating Stadrin database user
Stadrin user created
successfully
Congratulation! Set-up is done.
Use stadrin utility to set-up
users and tokens.
For instructions see stadrin
documentation.
```

The user licence is distributed on floppy diskette and has to be added to the default configuration directory, */etc/stadrin*:

```
mount /dev/fd0 /mnt/floppy ➥
-t vfat
cp /mnt/floppy/stadrin.lic ➥
/etc/stadrin
chmod 600 ➥
/etc/stadrin/stadrin.lic
```

The whole configuration is in a single file */etc/stadrin/stadrin.conf*.

As we can also see, the database access password is stored here in a clear text form, so it is necessary to make sure, that this file is not readable by any other users accessing the system:

```
database        mysql
dbname          stadrin
user            stadrin
password        opensesame
```

The second half of configuration file is composed of default properties for account data, synchronisation parameters and parameters regarding to root account, so it will be necessary to customise these settings to adapt to a specific server:

```
active_user             yes
active_token            yes
method                  RO/CR/SPWD
max_bad                 5
root_at_login_swpd      yes
```

The default authentication scheme:

```
auth  required ➥
/lib/security/pam_stack.so ➥
service=system-auth
```

can be changed to Response Only mode:

```
auth  required ➥
/lib/security/pam_stadrin.so
```

Listing 1: Specifying location and user data

```
Enter the location of MySQL database (localhost): localhost
Enter name of MySQL administrator (q to quit) [root] : root
MySQL administrator is root
Is it correct? (yes/no/q/quit) [no] : yes
Enter password of MySQL administrator root
!!! WARNING : password will be visible !!! : opensesame
Is it correct? (yes/no/q/quit) [no] : yes
```

Listing 2: Creating the database

```
Enter name of database for stadrin [stadrin] : stadrin
Database name is stadrin
Creating stadrin database with name stadrin
Database created successfully.
Tables created successfully.
Enter name of user for stadrin database [stadrin] : stadrin
Stradrin database user is stadrin
Enter password of stadrin user stadrin :opensesame
Stadrin user will have password opensesami.
```

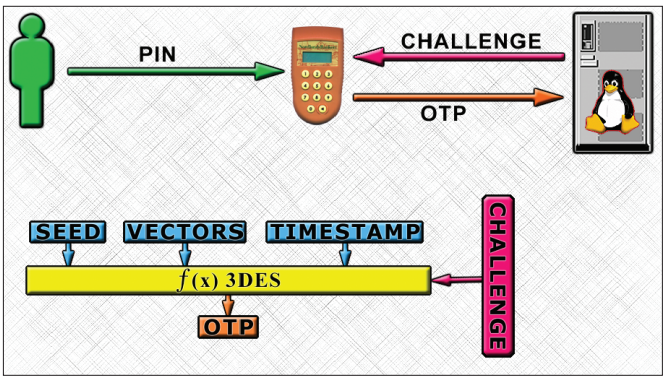

Figure 3: Challenge Response Authentication scheme

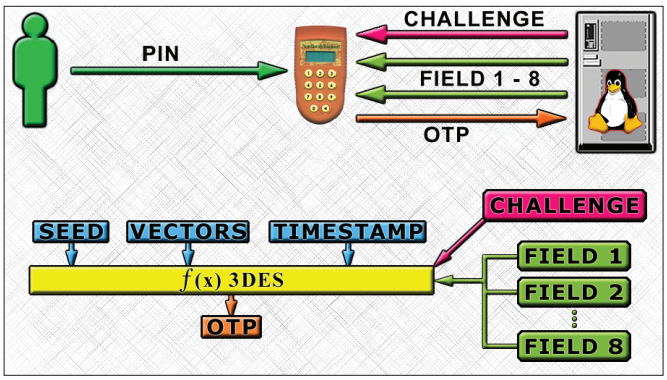

Figure 4: Message Authentication Code Authentication scheme

```
#auth required ➥
/lib/security/pam_stack.so ➥
service=system-auth
```

or to the Challenge Response mode with:

```
auth  required ➥
/lib/security/pam_stadrin.so CR
#auth required ➥
/lib/security/pam_stack.so ➥
service=system-auth
```

This is the right place to notice that all the common services natively support SPWD and RO authentication scheme, while the CR scheme is only supported by the login service.

User Token Definitions

After activating the authentication schemes we have to make definitions for the user accounts and the tokens. The simplest way is to import users from /etc/passwd using the command:

```
stadrin -sync
```

Alternatively, we can use our own importing scripts:

```
cat /etc/passwd | ➥
awk '{split($1,i,":");➥
print "stadrin -au "i[1];➥
print "stadrin -m "i[1]}" ➥
SPWD"' | bash -i
```

Now we have finished the tasks regarding the user definition process. The next step involves importing the token definitions and their assignments to specific users. The tokens comes with digitally signed and encrypted data files containing specific token information. For each token we have a unique definition file and activation key. The typical importation of the tokens is usually done in a single pass with an immediate assignment to the selected user, as shown in Listing 3.

Finally, we have to change the authentication scheme for the users with assigned tokens to a Response Only or Challenge Response mode from the default SPWD scheme:

```
stadrin -m pandoruser RO|CR
```

It is also possible to assign one token to a specific group of users with a single definitions using:

```
stadrin -st useraccount ➥
tokendefinition
```

After finishing the assignment procedure, with a view to making the authentication scheme complete, we have to test it's function on any available service such as login. The challenge response mode using the login service appears as follows:

```
killer login: pandoruser
Challenge is 0495
Enter your response:
Last login.....
killer$
```

Listing 3: Importing token data

```
stadrin -a pandoruser ./pandortoken.dpx
Enter your ini-key, please: ➥
11111111111111111111111111111110097123456 0097123456
Added APPL 1
Added APPL 2
Added APPL 3
```

Conclusion

From am enterprise point of view, Stadrin is an effective tool for raising the security and safety level of existing systems, especially where the static passwords authentication systems are no longer sufficient. Authentication schemes of this type are most commonly used by banks and insurance companies for securing access to electronic funds.

As Stadrin becomes more widespread, we will face OTP authentication systems more often and in more diverse places. We will certainly find it in system and application environments. Although systems like this have been mainly used in a very secure environments in the past, this is bound to change.

With its low cost, (Stadrin starts in the range of EUR 100–156 for single user – including the token) it is possible to implement this higher level of security in more standard computing environments, so preventing any unauthorised accesses and violations. A special discount of 25 per cent is available for all the Linux Magazine readers running for a one month period.

As we are all aware, most unauthorised system access is made from within a company, dynamic passwords are the best way to protect the services, for through-internet access as well as the local ones. For this reason you should consider changing before you find out the hard way, that your static passwords are no longer secure. ■