

Securing the Channel

At some point in building our web application we reach a stage where we need to allow users to input data. In a perfect world this wouldn't be a problem. The issue is that as soon as we place a web page on the web with data entry fields it will inevitably attract the attention of people wanting to illegally obtain secure data e.g. passwords and credit card details.

Consider the following...

25 April 2011 Last updated at 13:00

7.2K  Share    

PlayStation outage caused by hacking attack

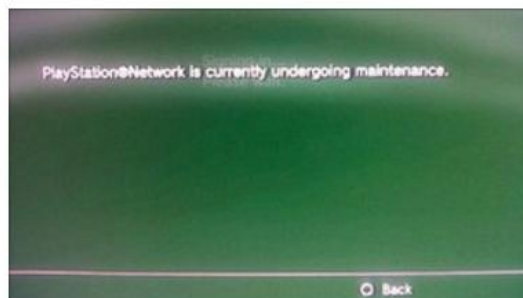
Sony has confirmed that a hacking attack was to blame for its PlayStation Network being taken offline.

The company said that it had taken the PSN down voluntarily while it investigated an "external intrusion".

The system is still unavailable five days after the hack was discovered.

Users trying to connect have been met with error messages stating that the network is "undergoing maintenance" or is "suspended"

<http://www.bbc.co.uk/news/magazine-20493572>



An error message like this one greeted users trying to log-in to the PlayStation Network

Related Stories

Security breaches are becoming a regular feature of weekly news.

Introduction

This module makes no claims to providing a definitive guide to securing your web applications. I plan to cover as much relevant material as I can with practical solutions to certain problems.

The plan is to take apart the widget swap application and look at weaknesses making recommendations on how the security could be improved.

Security Considerations

We have three main areas to consider as we look at security:

1. Secure communication

Secure communication involves preventing third parties from accessing the data. This is not only a matter of securing the communication path but also the physical security of the server and data.

2. Authentication

Authentication involves identifying who is accessing the system and proving they are who they claim to be.

3. Authorisation

Authorisation involves only allowing access to parts of the system that the authenticated user has rights to.

We shall also consider some common types of attack.

- Injection attacks either by use of a script or by SQL
- Social engineering attacks
- Intercepting data on the channel e.g. packet sniffing

This lecture aims to look at some issues to consider when securing the channel, some are technical, and others are common sense. The problem with common sense though is that there is so little of it about!

What is the Channel?

In this lecture I want to cover issues related to data passing between the human beings in the system and the technology. As soon as a person sits down at the keyboard there is potential for intercepting that data and making use of it.

We need to be aware of these potential weak spots and make sure that problems are at least reduced.

Be Careful Who You Talk to!

Not too long ago I heard a story about a guest speaker visiting the University who was giving a lecture on security. Prior to the talk he chatted to the member of staff who told me the story and entered into a friendly conversation. Impressed with how friendly and interested the guest speaker was the member of staff thought nothing of it. At the start of his talk the guest speaker then pointed out that in conversation he had found out a great deal about the member of staff, such things as the name of his pet, mother's maiden name and place of birth.

We tend to select as passwords strings that we can remember which is normal and natural. However the more people learn about us it increases the chances of guessing a password.

Guess the Password

Big fan or Doctor Who

6 letter password

T - - - - S

Unfortunately some security issues are caused by the users of our system and to some extent out of our hands as developers.

The Login

The most obvious mechanism for authentication is providing some sort of login system where the user is identified by a unique user name and verified by means of a password.

Email Address

Password

As you will see there is far more to security than making sure that the password is masked by the means of the asterisks *****!

Something as simple as having somebody look over your shoulder could compromise the security of your web system.

One of the problems with passwords is their potential to be guessed, as above. But even without such inside knowledge there are other means by which a password might be compromised.

Dictionary Attacks

A dictionary attack is pretty much what it sounds like. If a valid user name is known then a dictionary of passwords is sent to the login in the hope that the one of them is the user's password.

The following dictionary is taken from a WiFi cracking tool..

```
12345
abc123
password
passwd
123456
newpass
notused
Hockey
internet
```

It is depressing to consider that many people use obvious everyday words as their passwords. A number of years ago a network where I worked was hacked. One of the junior staff had the name of his new girlfriend as their password. A managing director had the word "password" to log into their email. These two accounts used together allowed for illicit emails to be sent looking as if they came from their accounts.

Brute Force Attack

Dictionary attacks are a kind of brute force approach where we make enough guesses until the password is found.

If we have some idea of the length of the password we create strings with different combinations of characters until the password is found.

For example

If the password is CC but all we know is that it is two characters long **

We could use the following strings...

AA

AB

BA

BB

BC

CB

CC

Until we guess correctly.

Obviously the longer the password the more time we need to crack it.

Some Countermeasures to Password Issues

Some simple countermeasures can be applied to the above.

Education

Educate users where possible to make good choices about password selection.

One strategy I employ is to have a standard prefix for my password e.g. nn88gg! and then modify that depending on the site I am using. For example nn88gg!amazon or nn88gg!facebook. This is fairly easy to remember and also ensures that I have a different password for each site I use. (Many people use the same password for all accounts!)

Don't use a password in the dictionary – force use of non alpha numeric characters.

password – very bad

pass_word – slightly better

pass_word!7 – even better

Enforce Rules via Code

Insist on minimum password length.

A two letter password is much easier to guess than an eight letter one.

Consider setting an expiration date on passwords. This will make users change their passwords at regular intervals.

Limit the number of login attempts making it harder for brute force approaches to work.

Examine server logs to see if there are signs of attack.

Writing Login Systems for Web Applications

In writing our applications we have a couple of options available to us.

1. We can use an off the shelf security solution
2. We can create our own

A couple of examples of off the shelf systems are Windows authorisation and Forms authentication.

Windows Authentication

Windows authorisation makes use of the users and groups set up on the Windows machine that the web application is running from. This means that in order to add new users it must be handled by the administrator for that machine. For a desktop application this would be fine however for a web application this is less helpful. In the case of many web applications there is typically a web page allowing them to sign up for a new account, e.g...



The image shows a web form titled "New Account Sign up" on a yellow background. The form contains three text input fields: "First Name", "Last Name", and "E Mail Address". To the right of the "E Mail Address" field is a button labeled "E-mail me my password".

Windows authentication is not best suited to this feature of allowing users to create their own accounts.

Forms authentication

Forms authentication is built in to .NET and provides a set of tools allowing you to create login, sign-up and change password pages with the minimum amount of effort.

Do it yourself

Another option when building your applications is to create your own security systems from scratch.

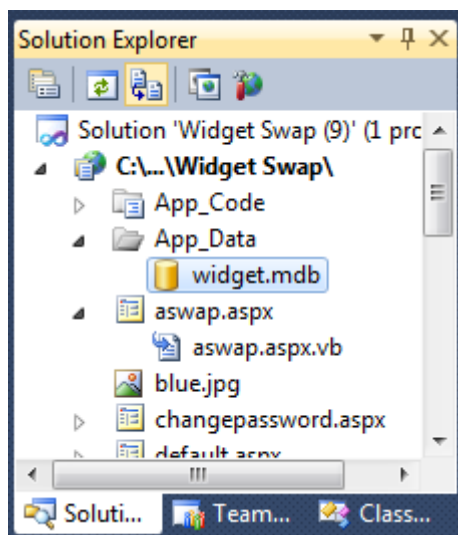
The advantage of this approach is that you know exactly how it works and make it customised to your own requirements.

The down side is that if anything goes wrong then you have nobody to blame but yourself!

Securing Stored Passwords

Unfortunately we don't just have to consider password security at the presentation layer we also need to think about what happens when the data is stored in the database.

If we examine the data store for the widget swap application we will see that it is an unsecured Access database.



Examining the tables also reveal that the passwords are stored in plain text format in the table Users...

Users						
UserNo	FirstName	LastName	EEmail	UserPasswo	DOB	
Will	Widget	widget@dmu.ac.uk	password123			
*(New)						

App_Data Folder

One thing that has been done correctly on this application is the database file has been stored in the App_Data folder.

The App_Data folder is one of a number of secure folders stopping a person simply downloading the entire database.

Entering as the URL...

http://localhost:50933/Widget%20Swap/App_Data/widget.mdb

Results in the following error...

Server Error in '/Widget Swap' Application.

HTTP Error 403 - Forbidden.

Version Information: ASP.NET Development Server 10.0.0.0

Placing the database in this folder makes it slightly harder to simply download the file.

A simpler approach however would be to go to the server room smash down the door and steal the server!

Password Hashing

One big problem with the system as it stands is that we are storing passwords in plain text. A step in the right direction is to encrypt the passwords however the problem here is that there is a possibility that an encrypted string could be decrypted.

It is also really easy for a dishonest system administrator to see the passwords and make use of them. Remember many people use the same password for multiple accounts.

A better approach not just with passwords but other sensitive information e.g. credit card details is to create a hash value.

.NET Cryptography

.NET provides a name space that handles (amongst other things) hashing of data using System.Security.Cryptography.

Hashing is different to encrypting because it is harder to calculate the original value of the string from the hash value.

For example

password

might become...

5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8

Rather than storing passwords in plain text or encrypted form we would store the hash value.

When the user enters their password at a later date we take the original password string, recalculate the hash value and then compare this with the one in the database.

Adding Salt

So far with hashing our passwords there is still the problem of what happens if a brute force approach is applied to the data?

Once a hacker has access to the database file they simply apply the hashing algorithm to their own dictionary and eventually deduce the password by comparing results.

(This is not as easy as it sounds since the hacker will have to guess at which hashing algorithm you used to hash your data. But if they have access to your code they can soon figure this out.)

One way to make it harder to do this is to add salt to the hash.

Salt consists of extra data added to the data such that we are not just hashing the password. For example we could store the password + email address.

This also helps to reduce the problem of two users having the same password.

If the passwords for John and Fred without salt look like this...

John 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8

Fred 5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8

We may assume that these two accounts share the same password giving us a clue as to how the data has been hashed.

Adding salt would change the hash values like so...

John C0C1664CC490D0FD9CE0BA81035B9199248AD16426301F03A70A4DECCDDD5990

Fred 9Wo0irC6+92F45AC854246D5A272B8FDB620B48F32230D3FF509299AB07BC62C7396BAFB7

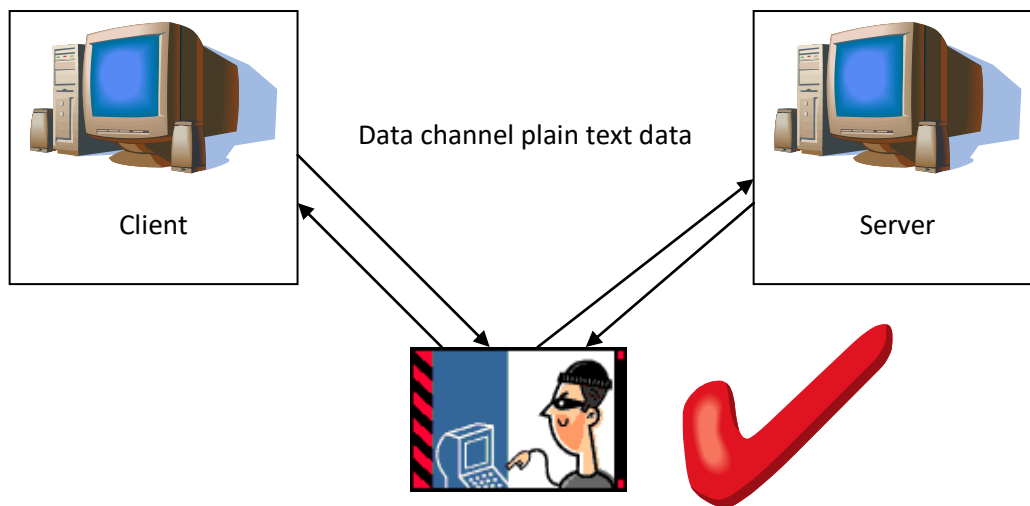
Even though they have the same password, with added salt this is now no longer obvious.

Securing the Communication Channel

If we have thought about password security at the presentation layer and at the data layer we also need to make sure that the channel of communication is secure.

In an HTTP connection data is passed between client and server in an unencrypted form.

This being the case it is quite possible for a third party to intercept the data stream and examine the packets of data passing over the network.



A better approach is to encrypt the data between client and server.

Public and Private Keys

One big problem with encrypting data is the question of what to do with the key?

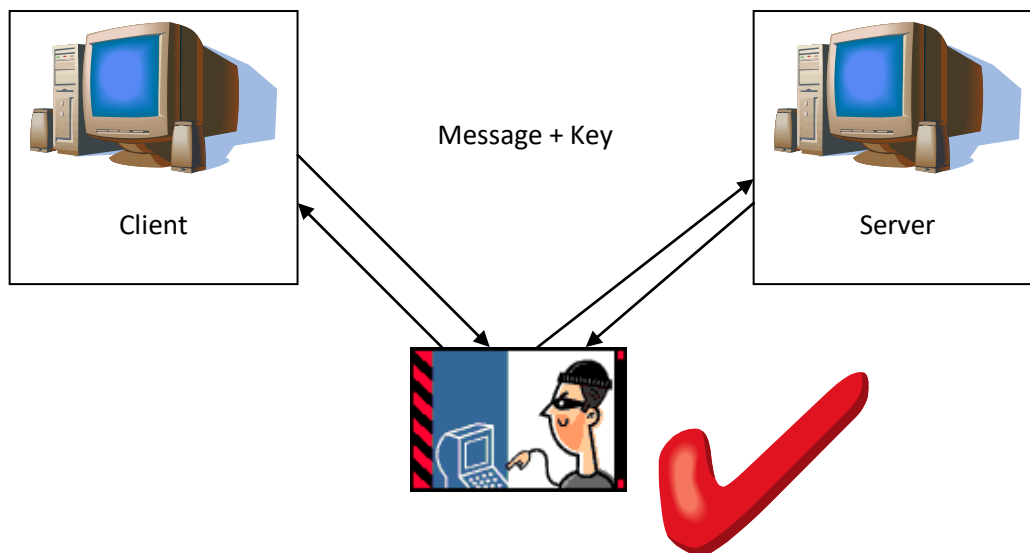
I have some data to send so I encrypt it using a key.

For example password might become qbtxpse (simply moving each letter up one in the alphabet).

I send my message to the recipient who then needs to decode the message.

The recipient gets the message qbtxpse but they also need the key to know how to decrypt the message.

The problem here is that we are forced at some point to send the key over the communication channel so that the data may be decrypted.



This runs the risk of the hacker obtaining both the data and the key making the encryption pointless.

A better approach is to employ a two key system.

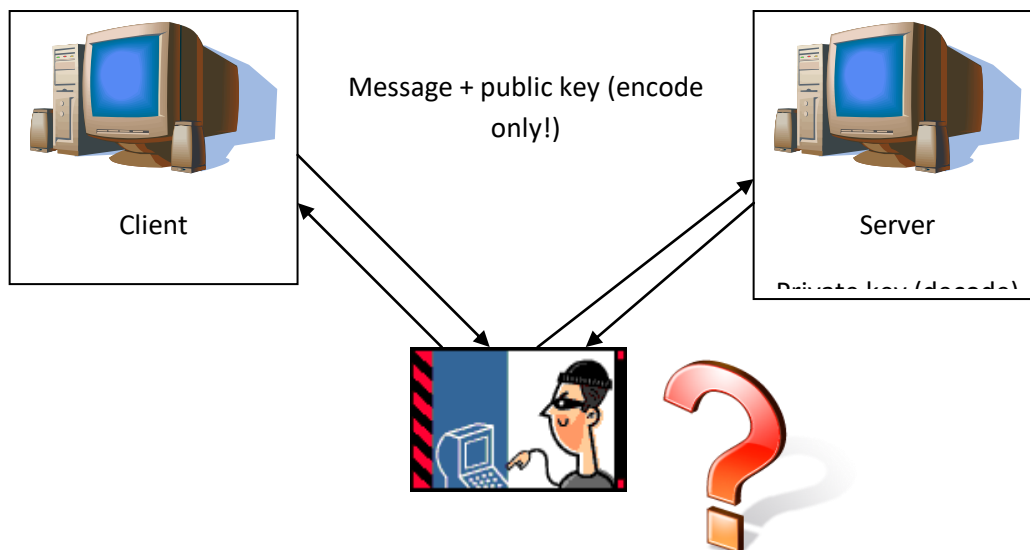
I want you to send me some data in an encrypted form so I send you my public key.

My public key may be used to encrypt the data but is no use for decrypting it.

You encrypt the message and send it to me.

If a hacker intercepts the message along with the public key they are no better off.

I receive the data and using my private key (which never gets sent over the communication channel)
I am able to decrypt the data.



Secure Socket Layer (SSL)


SSL uses a system of public and private keys to encrypt the data.

- The browser makes a secure HTTP request HTTPS on port 443
- The server sends back a digital certificate verifying its credentials
- The client verifies the certificate with the issuing agency
- Using the public key data is encrypted between client and server

The certificate is a file verifying that the server is a trusted source. The certificates are issued by a certificate authority.

Setting up SSL on IIS is fairly simple the downside is that setting up a certificate that is approved by a certificate authority will cost money.

IIS does allow for self signed certificates to be created. These still allow for an SSL connection to be established but may generate the following error in a browser...





There is a problem with this website's security certificate.


The security certificate presented by this website was not issued by a trusted certificate authority.
The security certificate presented by this website was issued for a different website's address.

Security certificate problems may indicate an attempt to fool you or intercept any data you send to the server.

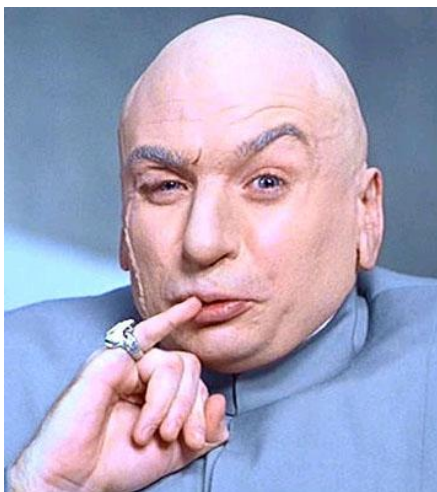
We recommend that you close this webpage and do not continue to this website.

 [Click here to close this webpage.](#)

 [Continue to this website \(not recommended\).](#)

 [More information](#)

Validation – All Input is Evil



One big key question in security is “who do you trust?”

For example you have all signed up for accounts on G677 which means I know a great deal about all of you.

Do you trust me not to make use of that data in some way?

Do you trust my skills enough to write a web application that will not be compromised in any way?

Another dimension to web security is that rather than a simple matter of users entering data into our systems our web applications may be gathering data from external systems.

Not just a matter of what people you trust but what systems do you trust?

Consider a web application that uses a web service on another site? Is that site as conscientious about security as your site? You may have gone to great efforts making your system as secure as

possible but what happens if you download XML from this other site that contains harmful script or SQL injection code?

Securing the Server

Once we have secured the communication channel we need to take steps to secure the server. This includes physical security and software security.

Firewalls

One tool used by hackers is called as port scanner.

Each service running on TCP/IP is allocated a port number.

Port 80 for HTTP etc.

A port scanner takes your IP address and scans each port in order 1 to 65535.

If an open port is found then further probing will be applied to see what sort of service is on the port and if a connection can be made.

Free tools are available to scan your machine to see if there are any open ports you didn't know about.

In the case of a machine running on the DMU network the default position is to hide it behind the University firewall.

If we set up a server on the local area network it will be available to all machines on the DMU network. If we try to access the machine from another location, e.g. from home access will be denied.

One option here is to allow access to specific ports from external machines. This approach reduces the surface area of the machine and reduces options for attack.

Use IP Filtering

As well as encrypting the data stream we may also control which machines may access the server remotely. When I say remotely I don't refer to remote login, what I mean is something as simple as an HTTP request.

IP Filters allow us to allow / disallow specific IP addresses' and IP range's access to the application.

One way I used this was to create an application that only allowed users with a valid DMU IP address access. If a user wanted to access the application from a different IP address e.g. Virgin Media I granted access on a case by case basis.

IIS also enforces a reverse DNS look up when IP Filtering is turned on.

A reverse DNS lookup checks the IP address of the HTTP connection and verifies if it is in fact part of the domain it claims to be in.

For example a machine might connect our site with a domain name of dmu.ac.uk however when the server looks up the IP address for the machine it is found to not be part of the dmu.ac.uk domain.

Turn off Unused Services

The rule here is if you don't use it, turn it off!

The default position with software such as Windows Server is to have pretty much all services unavailable. You turn them on as you need them.

In the case of Windows Server 2008 you cannot even browse the internet without being asked security questions.

Grant Minimum Permissions to Resources

When developing programs I tend to grant my applications maximum permissions on the development machine so that I don't need to be fighting with the security as well as writing the program.

In the case of a deployed application take the opposite approach and only grant the minimum.

For example in SQL server we may specify what a single stored procedure is allowed to do. (Read, update, delete etc.) If the stored procedure only retrieves data then only give it the read permission.