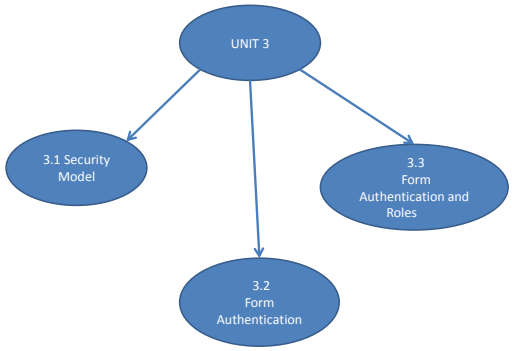
UNIT 3

Security



3.1 Security Model

- The most important factor for creating a secure application architecture and design lies in a good understanding of environmental factors such as users, entry points, and potential possible threats with points of attack.
- That's why *threat modeling* has become more important .
  - Threat modeling is a structured way of analyzing your application's environment for possible threats, ranking those threats, and then deciding about mitigation techniques based on those threats.

Secure Coding Guidelines

- In terms of web applications, you should always keep the following guidelines in mind when writing code:
  - **Never trust user input**
  - **Never use string concatenation for creating SQL statements.**
  - **Never output data entered by a user directly on your web page before validating and encoding it.**
  - **Use SSL**

Con..

- **Never store sensitive data, business-critical data, or data that affects internal business rule decisions made by your application in hidden fields on your web page.**
- **Never store sensitive data or business-critical data in view state.**
- **Enable SSL when using Basic authentication or ASP.NET forms authentication.**
- **Protect your cookies**

Understanding the Levels of Security

- **Authentication**
  - First, you have to authenticate users. Authentication asks the question.
  - It determines who is working with your application on the other end.
- **Authorization**
  - as soon as you know who is working with your application, your application has to decide which operations the user may execute and which resources the user may access.
- **Confidentiality**
  - While the user is working with the application, you have to ensure that nobody else is able to view sensitive data processed by the user.
- **Integrity**
  - you have to make sure data transmitted between the client and the server is not changed by unauthorized actors. Digital signatures provide you with a way to mitigate this type of threat.

Authentication

- *Authentication* is the process of discovering a user's identity and ensuring the authenticity of this identity.
- The process of authentication is analogous to checking in at a conference registration table.
  - First, you provide some credentials to prove your identity .
  - Second, once your identity is verified with this information, you are issued a conference badge, or *token*, that you carry with you when you are at the conference.

CON..

- In an ASP.NET application, authentication is implemented through one of several possible authentication systems:
  - Windows authentication
    - the Windows operating system uses a 96-bit number called an SID (security identifier) to identify each logged-on user
  - Forms authentication
    - The user is given a forms authentication ticket, which is a combination of values that are encrypted and placed in a cookie.
  - A custom authentication process

Authorization

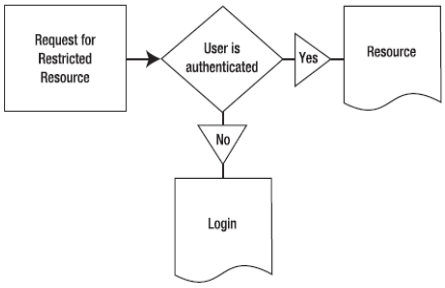
- *Authorization* is the process of determining the rights and restrictions assigned to an authenticated user.

Confidentiality and Integrity

- *Confidentiality* means ensuring that data cannot be viewed by unauthorized users while being transmitted over a network or stored in a data store such as a database.
- *Integrity* is all about ensuring that nobody can change the data while it is transmitted over a network or stored in a data store.
- Both are based on encryption.

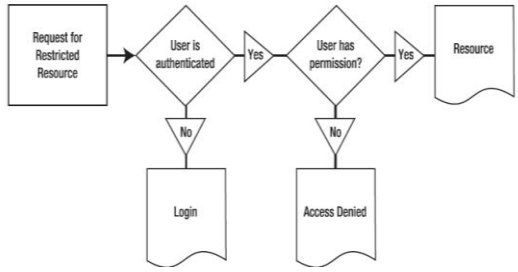
CON..

- *Encryption* is the process of scrambling data so that it's unreadable by other users.
- Reason to use encryption:
  - **To protect communication (data over the wire).**
  - **To protect permanent information (data in a database or in a file)**.



Con..

1. The request is sent to the web server. Since the user identity is not known at this time, the user is asked to log in .
2. The user provides his or her credentials, which are then verified, either by your application or automatically by IIS (in the case of Windows authentication).
3. If the user credentials are valid, the user is granted access to the web page. If his or her credentials are not valid, then the user is prompted to log in again, or the user is redirected to a web page with an “access denied” message.



Con..

1. The request is sent to the web server. Since the user identity is not known at this time, the user is asked to log in .
2. The user provides his or her credentials, which are verified with the application. This is the authentication stage.
3. The authenticated user’s credentials or roles are compared to the list of allowed users or roles. If the user is in the list, then the user is granted access to the resource; otherwise, access is denied.
4. Users who have access denied are either prompted to log in again, or they are redirected to a web page with an “access denied” message.

Understanding Certificates

- Before sending sensitive data, a client must decide whether to trust a website.
- Certificates can be installed on any type of computer, but they are most often found on web servers.
- With certificates, an organization purchases a certificate from a known certificate authority (CA) and installs it on its web server. The client implicitly trusts the CA and is therefore willing to trust certificate information signed by the CA.

Con..

- The certificate itself contains certain identifying information. It is signed with the CA’s private key to guarantee that it is authentic and has not been modified.
- The industry-standard certificate type, known as x.509v3, contains the following basic information.
  - The holder’s name, organization, and address
  - The holder’s public key, which will be used to negotiate an SSL session key for encrypting communication
  - The certificate’s validation dates
  - The certificate’s serial number

Con..

- The two biggest CAs are as follows:
  - Thawte: http://www.thawte.com
  - VeriSign: http://www.verisign.com

Understanding SSL

- Every certificate includes a public key.
  - A public key is part of an *asymmetric key pair*.
  - The basic idea is that the public key is freely provided to anyone who is interested.
  - The corresponding private key is kept carefully locked away and is available only to the server.
  - The interesting twist is that anything that's encrypted with one of the keys is decipherable with the other. That means a client can retrieve the public key and use it to encode a secret message that can be decrypted only with the corresponding private key.

Con..

- This process is called *asymmetric encryption*, and it's a basic building block of SSL.
- An important principle of asymmetric encryption is that you can't determine a private key by analyzing the corresponding public key.
- Asymmetric encryption limitations are :
  - it's much slower
  - generates much larger messages than symmetric encryption.

Con..

- *Symmetric encryption* is the type of encryption that most people are intuitively familiar with. It uses the same secret key to encrypt a message as to decrypt it.
- The drawback with symmetric encryption is that both parties need to know the secret value in order to have a conversation.

Con..

- The great trick of SSL is to combine asymmetric and symmetric encryption.
  - Asymmetric encryption manages the initial key exchange—in other words, agrees on a secret value.
  - Then, this secret value symmetrically encrypts all subsequent messages, which ensures the best possible performance.

Con..

- The whole process works like this, where the *client* refers to the web browser running on the end user's machine and the *server* refers to the web server hosting the websites the user wants to get access to :

1. The client sends a request to connect to the server.
2. The server signs its certificate and sends it to the client. This concludes the handshake portion of the exchange.

Con..

3. The client checks whether the certificate was issued by a CA it trusts. If so, it proceeds to the next step. In a web browser scenario, the client may warn the user with an ominous-sounding message if it does not recognize the CA, and allows the user to decide whether to proceed. The client recognizes CAs when their certificate is stored in the Trusted Root Certification Authorities store of the operating system. You can find certificates stored in this store through the Internet Explorer options by clicking the Certificates button on the Content tab.
4. The client compares the information in the certificate with the information received from the site (including its domain name and its public key). The client also verifies that the server-side certificate is valid, has not been revoked, and is issued by a trusted CA. Then the client accepts the connection.
5. The client tells the server what encryption keys it supports for communication.

Con..

6. The server chooses the strongest shared key length and informs the client.
7. Using the indicated key length, the client randomly generates a symmetric encryption key. This will be used for the duration of the transaction between the server and the client. It ensures optimum performance, because symmetric encryption is much faster than asymmetric encryption.

Con..

8. The client encrypts the session key using the server's public key (from the certificate), and then it sends the encrypted session key to the server.
9. The server receives the encrypted session key and decrypts it using its private key. Both the client and server now have the shared secret key, and they can use it to encrypt all communication for the duration of the session.

3.2 Form Authentication

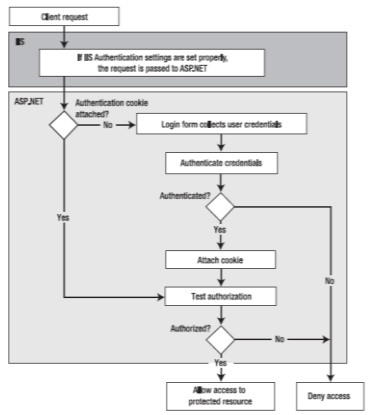
- Forms authentication is an all-purpose authentication system that's based around two concepts.
  - First is a login page that can validate users (usually, by comparing a user name and password combination against a database or some other data store).
  - Second, is a mechanism for preserving and reestablishing the security context on each request (usually, using a cookie). This way, the user needs log in only once.

Con..

- Forms authentication is a *ticket-based* (also called *token-based*) system. This means when users log in, they receive a ticket with basic user information. This information is stored in an encrypted cookie that's attached to the response so it's automatically submitted on each subsequent request

Con..

- When a user requests an ASP.NET page that is not available for anonymous users, the ASP.NET runtime verifies whether the forms authentication ticket is available.
- If it's not available, ASP.NET automatically redirects the user to a login page. At that moment, it's your turn.
- You have to create this login page and validate the credentials within this login page.
- If the user is successfully validated, you just tell the ASP.NET infrastructure about the success (by calling a method of the FormsAuthentication class), and the runtime automatically sets the authentication cookie (which actually contains the ticket) and redirects the user to the originally requested page.
- With this request, the runtime detects that the authentication cookie with the ticket is available and grants access to the page.



Why Use Forms Authentication?

- Forms authentication is an attractive option for developers for a number of reasons:
  - You have full control over the authentication code.
  - You have full control over the appearance of the login form.
  - It works with any browser.
  - It allows you to decide how to store user information.

Why Would You Not Use Forms Authentication?

- Forms authentication has downsides:
  - You have to create the user interface for users for page. You can either create the page completely on your own or use the ASP.NET security controls.
  - You have to maintain a catalog with user credentials.
  - You have to take additional precautions against the interception of network traffic.

3.3 Form Authentication and Role

- The Forms Authentication Classes:-
  - FormsAuthenticationModule is an HttpModule class that detects existing forms authentication tickets in the request.
    - If the ticket is not available and the user requests a protected resource, it automatically redirects the request to the particular page configured in your web.config file before this protected resource is even touched by the runtime.
    - If the ticket is present, the module automatically creates the security context by initializing the HttpContext.Current.User property with a default instance of GenericPrincipal, which contains a FormsIdentity instance with the name of the currently logged-in user.
    - You don't work with the module directly. Your interface to the module consists of the classes, which are part of the System.Web.Security namespace.

Con..

- *The Forms Authentication Framework Classes:*
  - FormsAuthentication
  - FormsAuthenticationEventArgs
  - FormsAuthenticationTicket
  - FormsIdentity
  - FormsAuthenticationModule

Con..

- The FormsAuthentication Options

Options	Default Value
Name	.ASPXAUTH
loginUrl	login.aspx
Timeout	30 minutes.
slidingExpiration	True
Cookieless	UseDeviceProfile
Protection	All
requireSSL	False
enableCrossAppRedirects	False
defaultUrl	default.aspx
Domain	<empty string>
Path	/