

# Securing .NET and COM+ Applications

co-organized by  & 

Thomas Phan  
Project Manager, Entersys Limited  
Microsoft Most Valuable Professional for Visual C++

## Overview

- Web Authentication and Authorization
- COM+ Role-Based Security

### Requirement

- Handle user state that spans the web application

### Approaches

- Use ASP session, and check per page
- Use J2EE; by using web.xml (and ejb-jar.xml), the container(s) handles credentials, and makes callbacks
- What about ASP.NET?

## Anonymous Access

### Who runs Apache in UNIX?

- Processes started by root do not listen to ports under 1024

### Who runs IIS in Windows?

- Different per Windows version: ASPNET, IIS\_WPG, and IUSR\_MachineName
- Since the IIS process has security constraints inherited from the process owner, we need to gain accesses to the owner appropriately
- e.g. if our application needs to write a file, we need to enable the “Modify” permission for the folder

- Basic and Digest Authentication
- Integrated Windows Authentication
- Forms-Based Authentication



## Anonymous

- No authentication occurs

## Basic

- Client sends user name and password as clear text
- Can be encrypted by using SSL
- Part of the HTTP specification and supported by most browsers

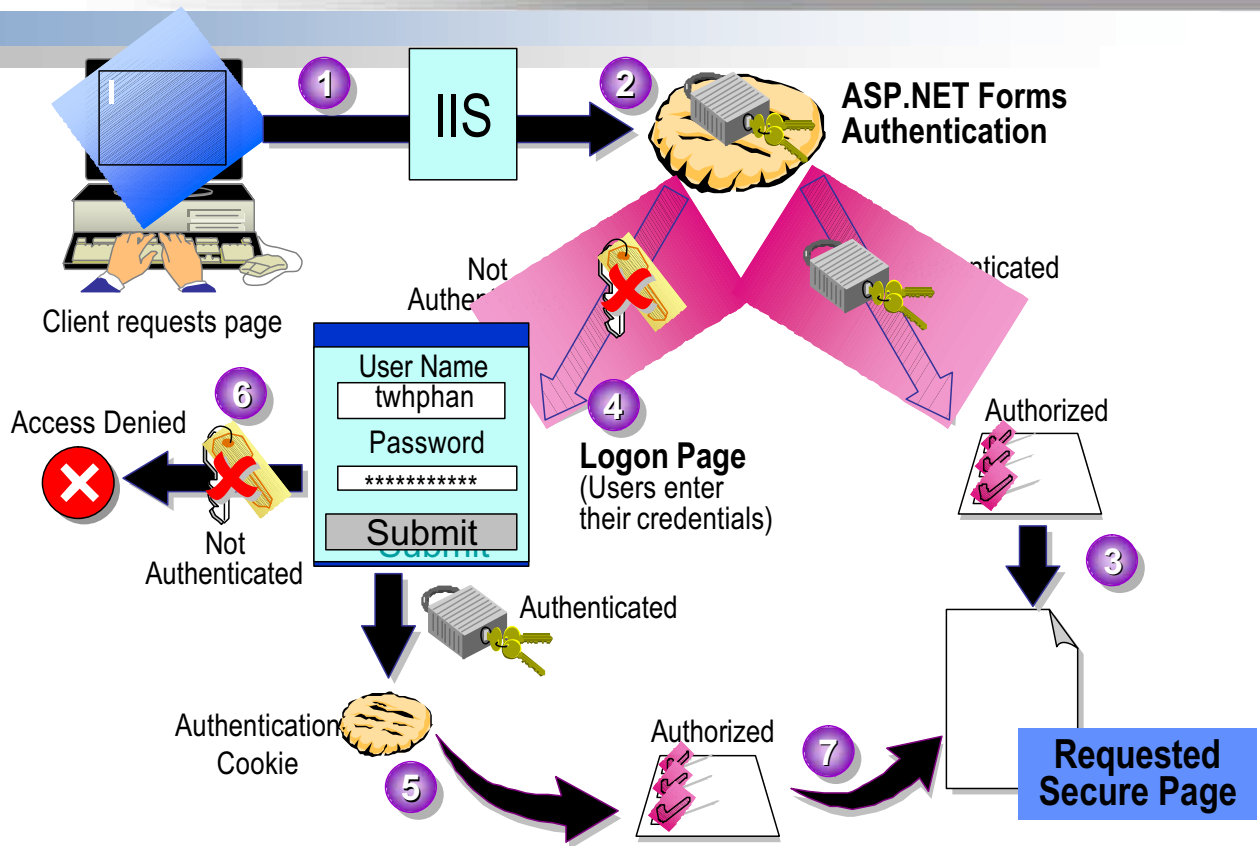
## Digest

- Sends information as encoded hash
- Supported by HTTP 1.1 or higher clients
- Requires Active Directory

## Windows

- Uses either NTLM or Kerberos
- Good for intranets, not Internet
- Does not work through firewalls

# Forms-Based Authentication

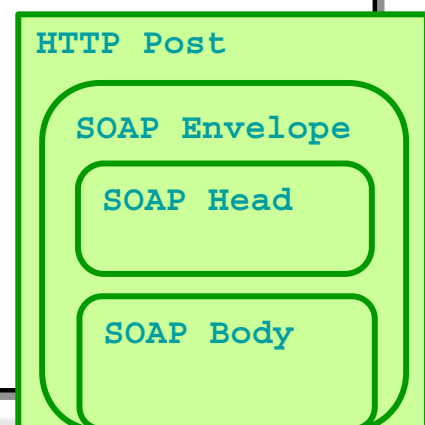


# Custom Authentication: SOAP Headers

- A SOAP Header is a part of an XML Web Service
- WSDL describes SOAP headers
- Web service proxies contain SOAP header classes
- Passing header objects when calling Web services

```

public class AuthHeader : SoapHeader { public string UserName; public string Password; }
public class AccountService : WebService
{
    public AuthHeader sHeader;
    ...
    [WebMethod]
    [SoapHeader("sHeader", Required=false)]
    public decimal GetAcctBalance(string acctID)
    ...
}
    
```



- In .NET, there are interfaces, and generic concrete classes to generalize storing credential
- Identity and IPrincipal Interfaces
- WindowsIdentity and WindowsPrincipal Classes
- GenericIdentity and GenericPrincipal Classes
- Authentication and Authorization with HttpModules

```
application.Context.User = Thread.CurrentPrincipal = new  
    GenericPrincipal(new GenericIdentity(userName),  
        roleNames);
```

## Authentication and Authorization with HttpModules

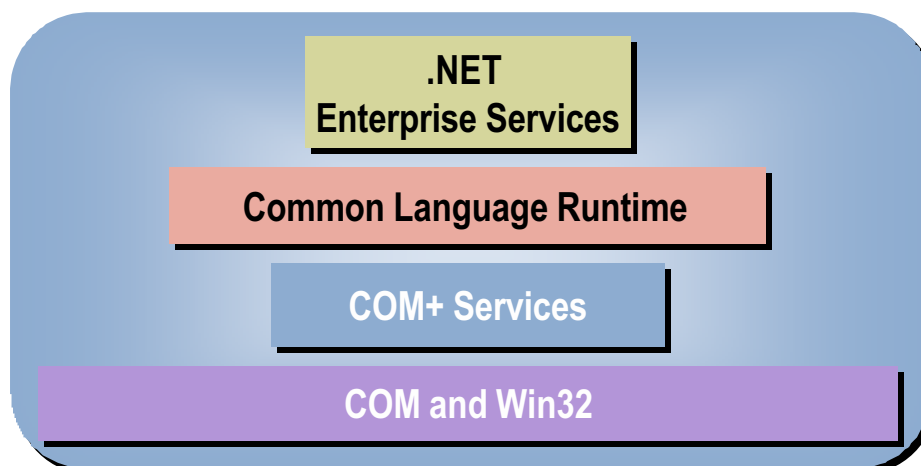
- HttpApplication events and HttpModules
- Authentication using HttpModules
- Authorization after authentication

```
public class SecurityModule : IHttpModule  
{  
    public void Init(HttpApplication application)  
    { application.AuthenticateRequest += new EventHandler(OnAuthenticate); }  
    public void OnAuthenticate(object sender, EventArgs args)  
    { //gets user info from HTTP/SOAP header... }  
}
```

- Transactions
- Compensating Resource Managers
- Resource management
  - JIT
  - Object pooling
- Synchronization
- Security
- Loosely coupled events
- Queued Components

## How the .NET Framework Integrates with COM+ Services

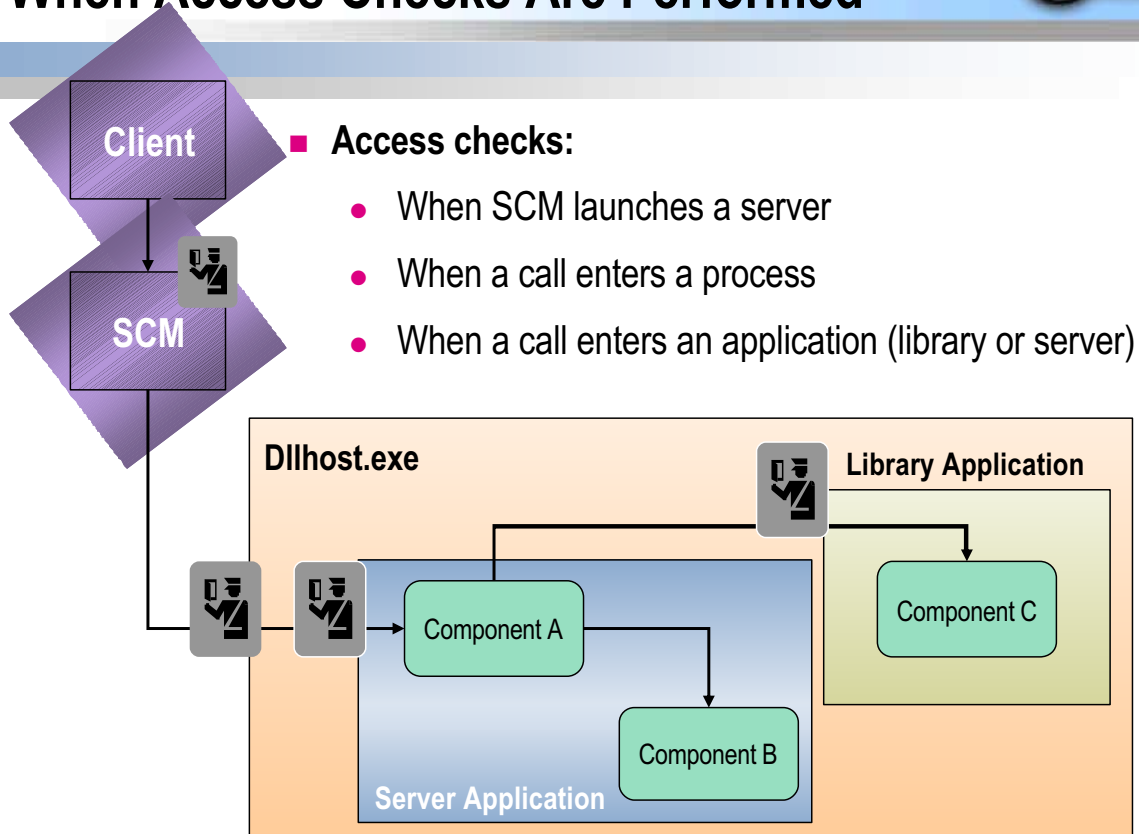
- The `System.EnterpriseServices` namespace provides necessary programming types
- The `ServiceComponent` class handles object life cycles and runtime attributes automatically



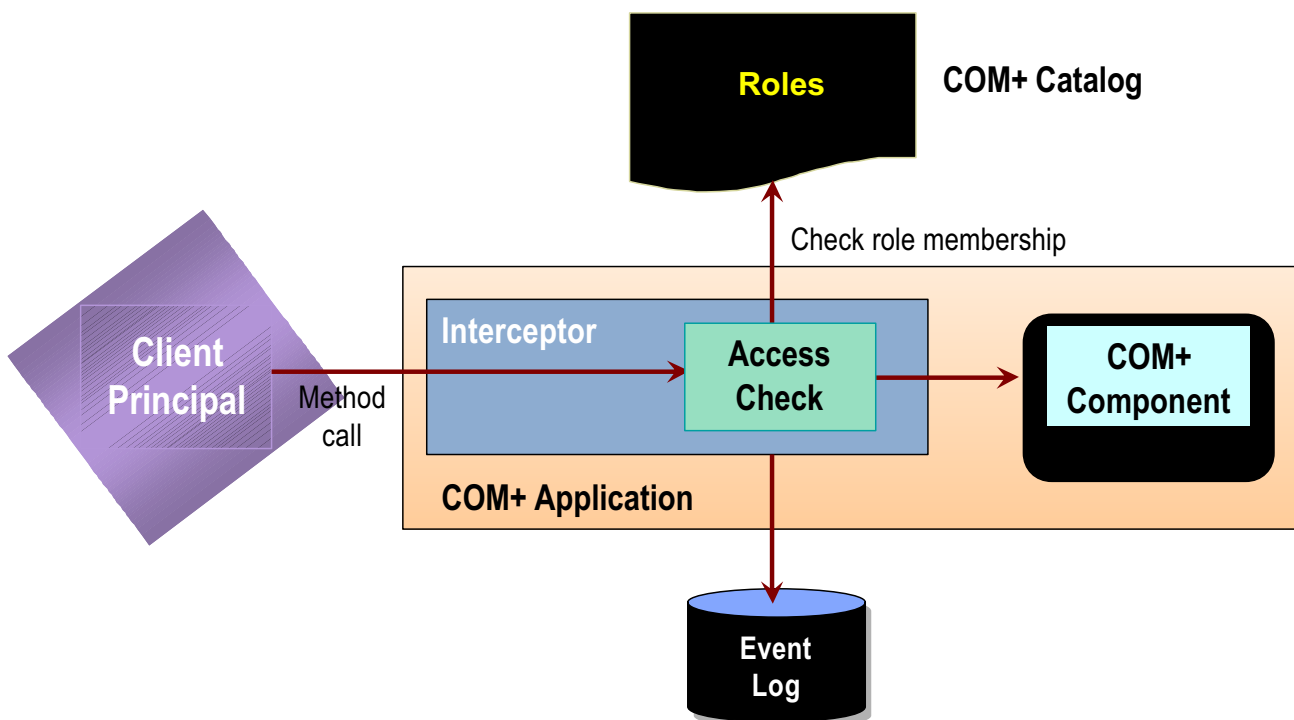
- **Where to store our resource manager password?**
  - Never hard code the connection sting in our code
  - Encrypt the password in Web.config (or app.config)
  - No password – use single sign on, a benefit of using pure Microsoft technology, i.e. Active Directory and SQL Server
- **We can borrow the approach from SQL authentication and impersonation for Enterprise Services**
- **With Enterprise Services, other than authentication, we can do authorization as well**

- **Authorization based on membership in an application-defined role**
  - Role examples: “Administrators”, “Architects”, “Developers”
  - Named set of principals that have the same privileges with respect to security
- **NET role-based security**
  - Uses Windows users or groups, custom, or generic scheme
- **COM+ role-based security**
  - Uses Windows users or groups
- **.NET and COM+ role-based security are mutually exclusive**

# When Access Checks Are Performed

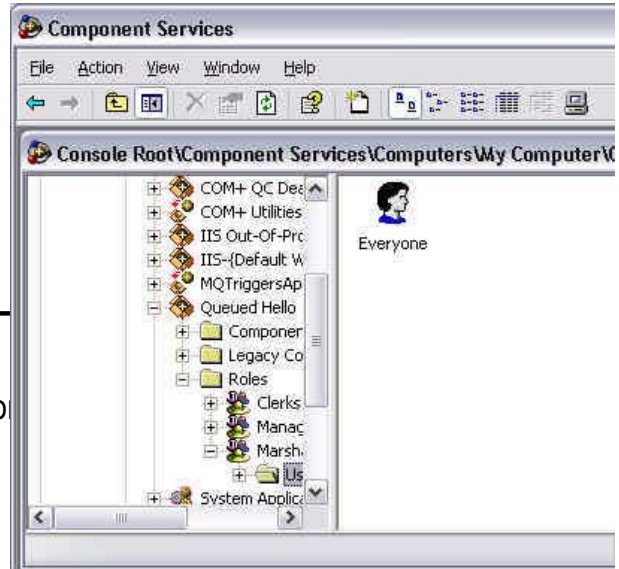


# How COM+ Performs Interapplication Role Checking



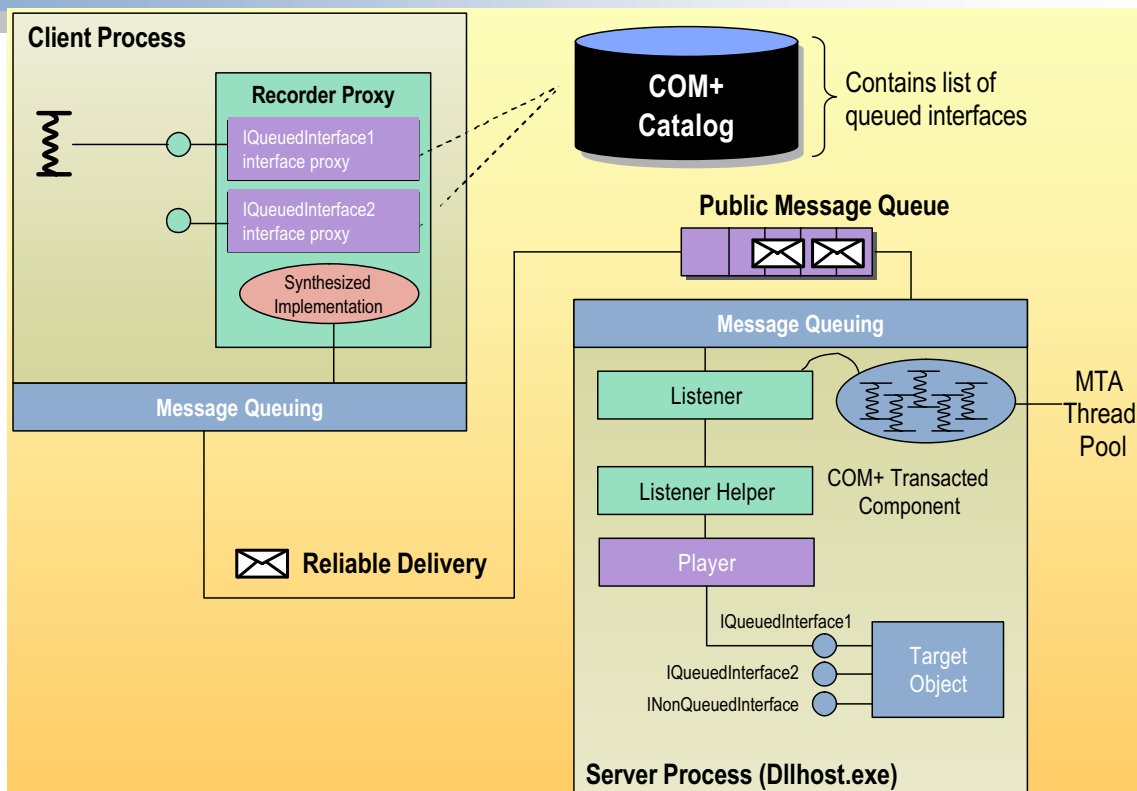
■ Roles can be assigned to

- Components
- Interfaces
- Methods



```
[ComponentAccessControl(true)]
public class HelloWorld : ServicedComponent
{
    [SecurityRole("Managers")]
    public void SayHello(string name)
    ...
}
```

# Queued Components Architecture



- Role-based security for queued components works as if it is called by its caller
- Player uses the security call context of the recorder client
- Client's security context marshaled in message payload
- Queued Component authentication uses Message Queuing authentication that requires a signature from Active Directory

## Summary

- .NET provides UIs, XML config files, callbacks, and libraries for managing web authentication and authorization
- Abstract interfaces, and generic concrete classes to generalize storing custom credentials in .NET
- Enterprise Services is a COM+ wrapper, and COM+ is for Windows only, Enterprise services uses Windows authentication and impersonation
- Enterprise Services (Queued Components) allows role-based security (with Active Directory)