Authentication

Authentication is the process by which an entity, also called a *principal*, verifies that another entity is who or what it claims to be. A principal can be a user, some executable code, or a computer. Authentication requires *evidence* in the form of *credentials*, and evidence can be in many forms, such as a password, a private key, or perhaps in the case of biometric authentication, a finger print.

Many authentication protocols are available to you in windows. Some are built into the product, and others require you to use building blocks in the operating system to create your own system. The schemes include the following:

- Basic authentication
- Digest authentication
- Forms-based authentication
- Passport authentication
- Windows authentication
- NT LAN Manager (NTLM) authentication
- Kerberos v5 authentication
- X.509 certificate authentication
- Internet Protocol Security (IPSec)

Note that some authentication schemes are more secure than others. In other words, as an application developer, you will be able to place more trust in the user's credentials when using some authentication schemes rather than others. For example, Basic authentication is much weaker than, say, Kerberos, and you should keep this in mind when determining which assets need protecting. Also, some schemes authenticate clients, and others authenticate servers. It's vitally important you understand this when considering the threats. For example, Basic authentication does not authenticate the server, only the client.

The following table shows which protocols authenticate clients, and which authenticate the server.

Protocol	Authenticates Client?	Authenticates Server?
Basic	Yes	No
Digest	Yes	No
Forms	Yes	No
Passport	Yes	No
NTLM	Yes	No
Kerberos	Yes	Yes
X.509 certificates	Yes	Yes
IPSec	Yes (computer)	Yes (computer)

Basic Authentication

Basic authentication is a simple authentication protocol defined as part of the HTTP 1.0 protocol defined in RFC 2617. Although virtually all Web servers and Web browsers support this protocol, it is extremely insecure because the password is not protected. The username and password are base64-encoded, which is trivial to decode! In short, the use of Basic Authentication in any Web based application is extremely discouraged, owing to its insecurity, unless the channel between the client and server is secured with SSL or IPSec.

Digest Authentication

Digest authentication, like Basic authentication, is defined in RFC 2617. Digest authentication offers advantages over Basic authentication; most notably, the password does not travel from the browser to the server in clear text. Also, Digest authentication is being considered for use by Internet protocols other than HTTP, such as LDAP for directory access and IMAP (Internet Message Access Protocol), POP3 (Post Office Protocol 3), and SMTP (Simple Mail Transfer Protocol) for e-mail.

Forms-Based Authentication

There is no standard implementation of forms-based authentication, and most sites create their own solutions. However, a version is built into Microsoft ASP.NET through the *FormsAuthenticationModule* class, which is an implementation of the *IHttpModule* interface.

Here's how forms-based authentication works. A web page is presented to the user, who enters a username and password and hits the Submit button.

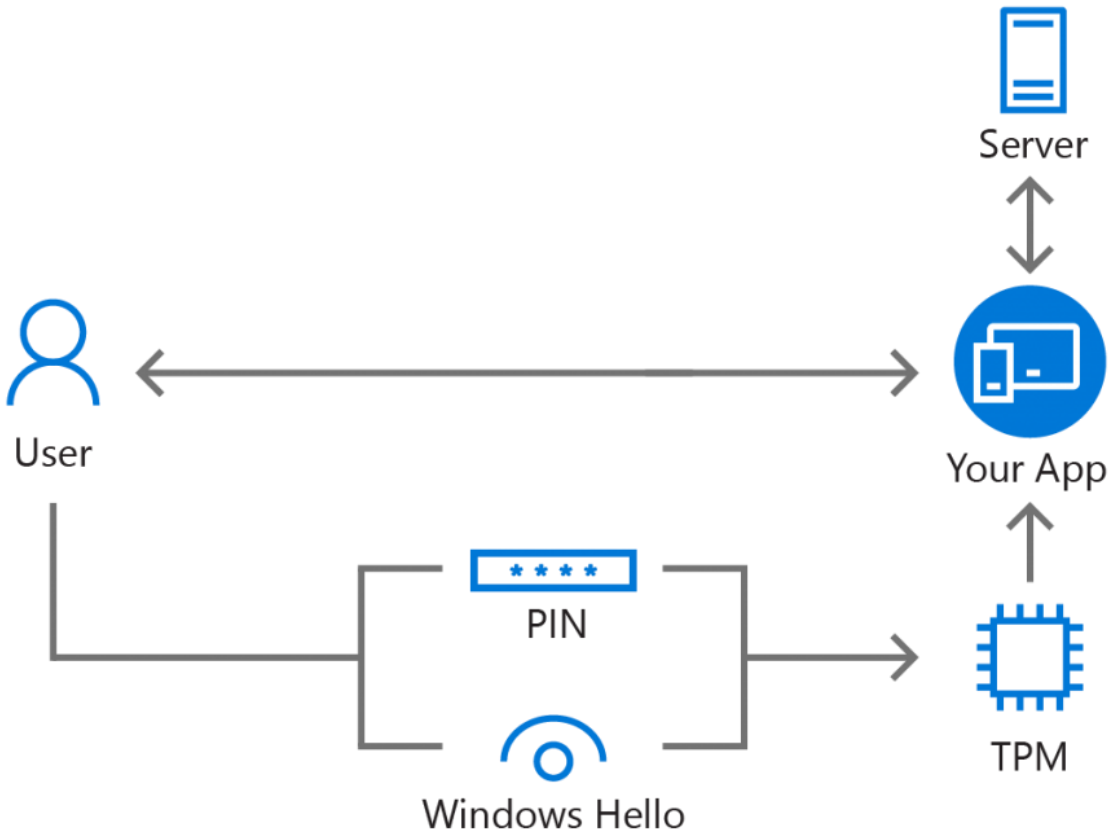
Next, the form information is posted to the web server, usually over an SSL connection, and the web server reads the form information. The web server then uses this information to make an authentication decision. For example, it might look up the username and password in a database or in an XML configuration file.

Forms-based authentication is extremely popular in the Internet. However, when implemented incorrectly, it can be insecure.

Windows Hello (Formerly Windows Live ID (Formerly Microsoft Passport))

With Windows 10, Microsoft introduced two security features called Windows Hello and Microsoft Passport. Windows Hello is the biometrics system built into Windows—it is part of the end-user's authentication experience. Microsoft Passport is a two-factor authentication (2FA) system that combines a PIN or biometrics (via Windows Hello) with encrypted keys from a user's device to provide two-factor authentication.

How does it work?



Microsoft Passport lets users authenticate to a Microsoft account, an Active Directory account, a Microsoft Azure Active Directory (AD) account, or non-Microsoft service that supports Fast ID Online (FIDO) authentication. After an initial two-step verification during Microsoft Passport enrollment, a Microsoft Passport is set up on the user's device and the user sets a gesture, which can be Windows Hello or a PIN. The user provides the gesture to verify identity; Windows then uses Microsoft Passport to authenticate users and help them to access protected resources and services. Microsoft Passport also enables Windows 10 Mobile devices to be used as a remote credential when signing into Windows 10 PCs.

Microsoft Passport and Windows Hello together represented Microsoft's FIDO 2.0 aligned end to end multi-factor authentication solution. Microsoft today announced that Windows Hello will be their brand for FIDO aligned end to end multi-factor authentication solution. So, they are once again killing Microsoft Passport brand. Microsoft Passport for Work will now become Windows Hello for Business. The credentials part along with Factors part is now considered part of Windows Hello.

Does it affect end users?

No. From a customer's perspective, this is simply a naming change and there are no changes from a configuration or security perspective.

Also, with the Windows 10 Anniversary Update, Microsoft has designed Windows Hello's architecture to be more flexible, enabling it to support devices, PINs, and biometrics as factor options for authentication. The architecture has also been made flexible enough to support the addition of new factor types which may be added in the future. Since Windows Hello now supports devices as factors, it enables scenarios where a user can unlock his laptop using Microsoft Band. This new system is called the Companion Device Framework.

Some history on Microsoft Passport brand:

In 1990s, Microsoft started a service called Microsoft Passport which was positioned as a single sign-on service for all web commerce. In early 2000s, Microsoft used the same brand to become Internet-wide unified-login system. Both these attempts failed. Later on, it was rebranded as Windows Live ID and now it has become the Microsoft Account which we use to login into all of the Microsoft's services. The Windows Hello Companion Device Framework created in June of 2016 has been marked for depraction just one year later, see: https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/windows-hello-companion-device-framework. Windows Hello and its Biometric Framework appears to be live and well, allowing Fingerprint and Video login on specialized devices.

Windows Authentication

Windows supports two major authentication protocols: NTLM and Kerberos. Authentication in Windows is supported through the Security Support Provider Interface (SSPI).

NTLM authentication

The NTLM protocol is supported by all current versions of Windows. It is a challenge-response protocol used by the major Windows services, including File and Print, IIS, and SQL Server. Two versions of NTLM exist, Version 1, and Version 2.

Version 2, introduced with Windows NT 4 Service Pack 4 offers one major security benefit over Version 1, it mitigates "man-in-the-middle" attacks.

Note that NTLM authenticates the client to the server; it does not verify the servers authenticity to the client.

Kerberos v5 authentication

Kerberos v5 authentication was designed at MIT and defined in RFC 1510. Windows 2000 and later implement Kerberos when Active Directory is deployed. One of the major advantages Kerberos offers is mutual authentication. In other words, the client's and the server's authenticity are both verified. Kerberos is generally considered a more secure protocol than NTLM, and in many cases it can be quicker.

X.509 Certificate Authentication

The most pragmatic use of X.509 certificates today is Secure Socket Layers (SSL). When you connect to a web server with SSL using HTTPS rather than HTTP your application verifies the authenticity of the server. This is achieved by looking at the common name in their server's certificate and comparing this name with the host name your application is connecting to. If the two are different, the application will warn you that you might not be communicating with the correct server.

SSL, by default authenticates the server. However, there is an optional stage in the SSL handshake to determine whether the client is who it says it is. This functionality is supported through client authentication certificates.

One of the most promising implementations of client certificates is smartcards. Smartcards store one or more certificates on a device the size of a credit card. Windows 2000 and later natively support smartcards.

We will examine SSL in greater detail in a future topic.

IPSec

IPSec is a little different from the protocols mentioned previously in that it authenticates servers only. Kerberos can also authenticate servers to other servers, but IPSec cannot authenticate users. IPSec offers more features than simply authenticating servers; it also offers data integrity and privacy. IPSec is supported natively in Windows 2000 and later.

Authentication Mechanism Comparison

The following table presents a comparison of the available authentication mechanisms.

Table 3.4: Available authentication methods

	Basic	Digest	NTLM	Kerberos	Certificates	Forms	Passport
Users need Windows accounts in server's domain	Yes	Yes	Yes	Yes	No	No	No
Supports delegation	Yes	No	No	Yes	Can do	Yes	Yes
Requires Win2K clients and servers	No	Yes	No	Yes	No	No	No
Credentials passed as clear text (requires SSL)	Yes	No	No	No	No	Yes	No
Supports non-IE browsers	Yes	No	No	No	Yes	Yes	Yes

** Note: Firefox and Chrome can be configured to "sort of work" with Windows Authentication via add-ins or extra settings

Who Are You Running As, Really??

Identity becomes a major issue with web programming. Developing websites using Visual Studio's built-in Web Server or IIS Express insulates you from requiring/understanding identities as your code executes under the context of your Login ID, so your code executes as you, so to speak.

As long as your application lives on your local workstation life is very easy. Soon as you deploy your app to a full blown IIS Server, things change very quickly. IIS (the full blown one, not Express) executes your web application using its own account.

AND IT'S DIFFERENT FOR EVERY OPERATING SYSTEM AND VERSION (Pretty Much)

Microsoft loves to tinker with IIS's service account. While I find it interesting and fun to review each and every version I will spare you the details of looking at all of them.

The following link can show you a little of the complexity throughout the years, read it at your own peril. Note that it only covers the latter half of Microsoft's OSes, there are many more.

http://blogs.iis.net/davcox/archive/2009/08/12/what-is-my-iis-code-running-as.aspx

The current identify (For IIS 7.5 and IIS 8) is a local user named DefaultAppPool, or more specifically 'IIS APPPOOL\DefaultAppPool'. Note that 'IIS AppPool' is local to the server.

So our new IIS Development Server, WebBaist.Nait.ca, will run your app as DefaultAppPool locally.

Any requests leaving the machine will operate under the NetworkService account which uses a format of DOMAIN\MACHINENAME$ or

---

3rd Party Authentication

Using a 3rd party Authentication System such as OpenID or OAuth allows the developer to farm out identifying the User to another service. Users can utilize an existing account on another service to Authenticate their identities, so they don't have to remember yet another account/password and the developer does not have to store and manage credentials, a win/win for both parties.

A list of OAuth providers can be found at: https://en.wikipedia.org/wiki/List_of_OAuth_providers

Biometric Devices

Solve all of your problems with Bio!!! Sorry, not really.

Biometrics such as fingerprint readers, Eye Scanners, or facial recognition eliminate the need for passwords but creates a new set of problems.

Hackers are constantly trying to use FaceBook pictures or recreate fingerprints you are leaving around and I still don't sleep well at night thinking someone might be after my eyeball or finger, which is why I use my pinky finger as I would mind losing that one the least.

Your Biometric needs to be associated with a Windows Account and a Password. There is no standard for this. 5 years ago keyboards with built in fingerprint readers were going to solve all of our problems until it was discovered that they stored with Account info in the Registry, in cleartext at that, easily readable.

HKEY_LOCAL_MACHINE\SOFTWARE\DigitalPersona\DB\Data\Users was the key used, at least they eventually started encrypting it, but it is reversible given enough effort.

https://arstechnica.com/information-technology/2012/09/windows-passwords-exposed/

2-Factor Authentication

As discussed in the Password Lab I strongly suggest you use 2-Factor Authentication such as a physical key generation device, SMS Text, or an App on your phone. This adds another layer of security and the hassle of using it is well worth the extra security it provides.

Authorization

Once a principal's identity is determined through authentication, the principal will usually want to access resources, such as printers and files. Authorization is determined by performing an access check to see whether the authenticated principal has access to the resource being requested. Some principals will have more access rights to a resource than other principals do.

Windows offers many authorization mechanisms, including these:

- Access control lists (ACLs)
- Privileges
- IP restrictions
- Server-specific permissions

Access Control Lists

All objects in Windows NT and later can be protected by using ACLs. An ACL is a series of access control entries (ACEs). Each ACE determines what a principal can do to a resource. For example, Blake might have read and write access to an object, and Cheryl might have read, write, and create access.

Privileges

A privilege is a right attributed to a user that has systemwide implications. Some operations are considered privileged and should be possible only for trusted individuals. Examples include the ability to debug applications, back up files, and remotely shut down a computer.

IP Restrictions

IP restrictions are a feature of IIS. You can limit part of a Web site, such as a virtual directory or a directory, or an entire Web site so that it can be accessed only from specific IP addresses, subnets, and DNS names.

Server-Specific Permissions

Many servers offer their own form of access control to protect their own specific object types. For example, Microsoft SQL Server includes permissions that allow the administrator to determine who has access to which tables, stored procedures, and views. COM+ applications support roles that define a class of users for a set of components. Each role defines which users are allowed to invoke interfaces on a component.

Tamper-Resistant and Privacy-Enhanced Technologies

Numerous networking protocols support tamper resistance and data privacy. Tamper resistance refers to the ability to protect data from being deleted or changed either maliciously or accidentally. If Blake orders 10 dump trucks from Luke, he doesn't want an attacker to modify the order en route to Luke to 20 dump trucks. Privacy means that no one else can read the order Blake has placed with Luke; only the two parties can read the message. The most common tamper-resistant and privacy-enhanced protocols and technologies in Windows are the following:

- SSL/TLS
- IPSec
- DCOM and RPC
- EFS

SSL/TLS

SSL was invented by Netscape in the mid-1990s. It encrypts the data as it travels between the client and the server (and vice versa) and uses message authentication codes (MACs) to provide data integrity. TLS is the version of SSL ratified by the Internet Engineering Task Force (IETF).

IPSec

As I've mentioned, IPSec supports authentication, encryption for data privacy, and MACs for data integrity. All traffic traveling between the IPSec-secured servers is encrypted and integrity-checked. There's no need to make any adjustments to applications to take advantage of IPSec because IPSec is implemented at the IP layer in the TCP/IP network stack.

DCOM and RPCs

Distributed COM and remote procedure calls support authentication, privacy, and integrity. The performance impact is minimal unless you're transferring masses of data.

Encrypting File System

Included with Windows 2000 and later, the Encrypting File System (EFS) is a file-based encryption technology that is a feature of the NT File System (NTFS). While SSL, TLS, IPSec, and DCOM/RPC security concerns protecting data on the wire, EFS encrypts and provides tamper detection for files.

Auditing

The aim of auditing, also called logging, is to collect information about successful and failed access to objects, use of privileges, and other important security actions and to log them in persistent storage for later analysis. Windows offers logging capabilities in the Windows event logs, the IIS Web logs, and numerous other application-specific log files, including the SQL Server and Exchange log files.

Filtering, Throttling, and Quality of Service

Filtering means inspecting data as it's received and making a decision to accept or reject the packet. This is how packet-filtering firewalls work. Many IP-level denial of service threats can be mitigated through the use of a packet-filtering firewall.

Throttling means limiting the number of requests to your system. For example, you might allow only a small number of anonymous requests but allow more authenticated requests. You would do this because an attacker might not attempt to attack you if she needs to be identified first. It's important that you limit anonymous connections.

Quality of service is a set of components that allow you to provide preferential treatment for specific types of traffic. For example, you can allow favored treatment to streaming media traffic

Least Privilege in the Real World

You can bury your head in the sand, but the Internet is full of bad guys out to get your users as your users employ applications created by you, and many of the attacks in the past would have failed if the programs were not running as elevated accounts. Presently, two of the more popular kinds of attacks on the Internet are viruses/Trojans and Web server defacements. I want to spend some time on each of these categories and explain how some common attacks could have been mitigated if the users had run their applications as plain users.

Viruses and Trojans

Viruses and Trojans both include malicious code unintentionally executed by users. Let's look at some well-known malicious code; we'll see how the code would have been foiled if the user executing the code were not an administrator.

Back Orifice

Back Orifice is a tool that, when installed on a computer, allows a remote attacker to, among other things, restart the computer, execute applications, and view file contents on the infected computer, all unbeknownst to the user. On installation, Back Orifice attempts to write to the Windows system directory and to a number of registry keys, including HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run. Only administrators can perform either of these tasks. If the user were not an administrator on the computer, Back Orifice would fail to install.

SubSeven

Similar to Back Orifice, SubSeven enables unauthorized attackers to access your computer over the Internet without your knowledge. To run, SubSeven creates a copy of itself in the Windows system directory, updates Win.ini and System.ini, and modifies registry service keys located in HKEY_LOCAL_MACHINE and HKEY_CLASSES_ROOT. Only administrators can perform these tasks. Once again, if the user were not an administrator, SubSeven would fail.

FunLove Virus

The FunLove virus, also called W32.FunLove.4099 by Symantec, uses a technique that was first used in the W32.Bolzano virus. When the virus is executed, it grants users access to all files by modifying the kernel access checking code on the infected computer. It does so by writing a file to the system directory and patching the Windows NT kernel, Ntoskrnl.exe. Unless the user is an administrator, FunLove cannot write to these files and fails.

ILoveYou Virus

Possibly the most famous of the viruses and Trojans, ILoveYou, also called VBS.Loveletter or The Love Bug, propagates itself using Microsoft Outlook. It operates by writing itself to the system directory and then attempts to update portions of HKEY_LOCAL_MACHINE in the registry. Once again, this malware will fail unless the user is an administrator.

Web Server Defacements

Web server defacing is a common pastime for script kiddies, especially defacing high-profile Web sites. A buffer overrun in the Internet Printing Protocol (IPP) functionality included in Microsoft Windows 2000 and exposed through Internet Information Services (IIS) allowed such delinquents to attack many IIS servers.

The real danger is the IPP handler, which is implemented as an Internet Server Application Programming Interface (ISAPI) extension, running as the SYSTEM account. The following text from the security bulletin issued by Microsoft, available at http://www.microsoft.com/technet/security/bulletin/MS01-023.asp, outlines the gravity of the vulnerability:

A security vulnerability results because the ISAPI extension contains an unchecked buffer in a section of code that handles input parameters. This could enable a remote attacker to conduct a buffer overrun attack and cause code of her choice to run on the server. Such code would run in the local system security context. This would give the attacker complete control of the server and would enable her to take virtually any action she chose.

If IPP were not running as the local system account, fewer Web sites would have been defaced. The local system account has full control of the computer, including the ability to write new Web pages.

IMPORTANT
Running applications with elevated privileges and forcing your users to require such privileges is potentially dangerous at best and catastrophic at worst. Don't force your application to run with dangerous privileges unless doing so is absolutely required.

Programmatic Authorization

You can work with certain security objects in asp.net that will give you access to User permissions and properties.

Here are some useful objects and methods:
The ***User*** property of the ***Page*** object provides an instance of the ***IPrincipal*** object which provides a single method and a single property.

Identity: This property provides an instance of the System.Security.Principal.IIDentity object for you to get at specific properties of the authenticated user.

IsInRole: This method takes a single parameter, a string representation of the system role. It returns a Boolean value that indicates whether the user is in the role specified.

User.Identity

Attribute	***Description***
AuthenticationType	Basic, NTLM, Forms, Passport, etc
IsAuthenticated	Boolean specifying if the user is authenticated
Name	Username of the user + Domain

Usage

```
User.Identity.Name;
User.Identity.IsAuthenticated;
User.Identity.AuthenticationType;

If (User.IsInRole(“Admin”)  )
     {
     // Do something
Else
     // Error message
     }
```