



SECURITY BEST PRACTICES FOR *WINDOWS* AZURE SOLUTIONS

AUTHOR

Ben Ridgway (Security Program Manager, Windows Azure Security)

Acknowledgments

Gareth Bradshaw (Sr. Program Manager, Windows Azure)

Lori Clark (Sr. Technical Writer)

Kathy Davies (Sr. Technical Writer)

Karthick Jayaraman (Software Development Engineer, Windows Azure Security)

Duke Kamstra (Sr. Program Manager, Windows Azure)

Charlie Kaufman (Principal SDE, Windows Azure)

Michael Howard (Cybersecurity Architect, MCS)

Harris Majeed (Sr. Product Planner, Windows Azure)

Andrew Marshall (Principal Program Manager Lead, Skype)

Walter Myers III (Principal Consultant, MCS)

Ashwin Palekar (Principal Program Manager, Windows Azure)

Mark Russinovich (Technical Fellow, Windows Azure)

Joel Sloss (Program Manager, Windows Azure)

Stefan Schackow (Principal Program Manager Lead, Windows Azure)

Ganesh Srinivasan (Sr. Program Manager, Windows Azure)

Tania Tanaka (Security Program Manager, Trustworthy Computing Security)

Nick Torkington (Sr. Program Manager, Trustworthy Computing Security)

Selcin Turkarslan (Technical Writer)

Lori Woehler (Principal Program Manager, Windows Azure Compliance & Security)

FEBRUARY 2014 (REVISION 3¹)

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2014 Microsoft. All rights reserved.

Microsoft, Active Directory, Hyper-V, Visual Basic, Visual C++, Visual C#, Visual Studio, Windows, Windows Azure, Windows Live and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

¹ Previous versions of this document were titled Security Best Practices for Developing Windows Azure Applications

Table of Contents

Executive Summary.....	6
Intended Audience.....	7
Overview of Windows Azure Security.....	7
Threats in the Cloud.....	7
Securing Windows Azure IaaS Virtual Machines	9
Common Security Topics for Windows Azure IaaS Virtual Machines.....	9
Securing Windows Azure Web Sites	11
Common Security Topics for Windows Azure Web Sites.....	12
Securing Windows Azure PaaS Roles	12
Common Security Topics for Windows Azure PaaS Roles	13
Overview of Identity Management and Access Control	15
Windows Azure Active Directory	16
Windows Identity Foundation	19
Active Directory Federation Services 2.0.....	19
Active Directory Authentication Library	20
Graph API	21
More Information	22
Windows Azure Service Layer Security Considerations.....	22
Windows Azure Service-Layer Security Overview	22
Namespace Configuration.....	23
Data Access Security	24
Protecting Data in Transit	25
Fault Domains	25
Data Redundancy with Windows Azure Storage	25
Windows Azure SQL Database Security Considerations.....	26
Certificate Management in Windows Azure.....	27
Certificate Management for Windows Azure Cloud Services.....	27
Certificate Management for Windows Azure Virtual Machines.....	27
Encrypting Data in Windows Azure	27
Certificates and Data Encryption within Windows Azure.....	28
Securing Connection Strings and Storage Keys.....	29

Protecting Data at Rest within Windows Azure SQL Database.....	30
TDE with Microsoft SQL Server Deployed as a Virtual Machine.....	30
Additional Cryptography Information.....	30
Auditing and Logging for PaaS Cloud Service Roles.....	31
Windows Azure Network Layer Security Considerations	31
Windows Azure Networking Fundamentals	31
Networking in PaaS Roles	32
Networking in IaaS Virtual Machines.....	32
Spoofing	32
Windows Azure Virtual Networks.....	33
Secure Usage of DNS.....	33
Eavesdropping / Packet Sniffing	34
Web Application Firewall.....	34
Windows Azure Platform and Infrastructure Layer Protections.....	34
Multi-tenant hosting.....	34
Side-Channel Attacks	34
Verification and Certification.....	35
Conducting Penetration Tests.....	35
Designing Effective Cloud Penetration Tests	36
Securing Access to Roles and Virtual Machines.....	37
Securing Access to the Windows Azure Management Portal.....	37
Privileged Application-Level Configuration.....	38
IaaS Virtual Machine Access Security.....	38
PaaS Remote Access Security.....	40
Secure Cloud Application Design	40
Threat Modeling	40
Using Claims-Based Authentication.....	40
Gatekeeper Keymaster Design Pattern.....	41
Storage as a Relay Design Pattern	42
Hybrid Cloud/On-Premises Deployments.....	44
Applying SDL Practices to Windows Azure Applications.....	44
Security Education and Awareness.....	45
Secure Development Practices on the Windows Azure platform	45

Verification and Release	46
Conclusion.....	46
Appendix A. Glossary	47
Appendix B: Using SDL-approved Cryptography in Windows Azure Applications.....	48
Appendix C: Security Resources for Azure Developers.....	49
Security Resources for Windows Azure IaaS Virtual Machines	49
Security Resources for Windows Azure Websites	49
Security Resources for Windows Azure PaaS Roles.....	49
Security Resources for Identity and Access Management.....	50
Security Resources for Windows Azure Storage.....	50
Resources for Data Encryption	50
Windows Azure Platform and Infrastructure Resources	50
Security Resources for Windows Azure SQL Database and Microsoft SQL Server.....	51
Microsoft Security Development Lifecycle	51

Executive Summary

This paper focuses on the security challenges and recommended approaches to designing, developing, and deploying secure applications on Microsoft's Windows Azure platform. It also discusses some of the security considerations when migrating existing on-premises applications into the cloud when using Windows Azure Virtual Machines (VMs). The goal is to provide specific design guidance which customers or 'tenants' of Windows Azure can use to design, deploy, and manage security-enhanced solutions.

This paper is organized into three major topics:

Topic 1: An overview of common security tasks broken down by compute model and offering type:

- Securing Windows Azure Virtual Machines (Infrastructure as a Service)
- Security Windows Azure Web Sites (Software as a Service)
- Securing Windows Azure Roles (Platform as a Service)
- Overview of Identity Management and Access Control (Software as a Service)

Topic 2: Deeper technical details on developing secure Windows Azure solutions broken down by layer:

- Windows Azure Service Layer Security Considerations
- Windows Azure Network Layer Security Considerations
- Windows Azure Platform and Infrastructure Layer Security Considerations

Topic 3: Design and process guidance and additional resources:

- Secure Cloud Application Design
- Applying SDL Practices to Windows Azure Applications
- Appendices which provide links to additional resources

NOTE: For more information about the internal structure of Windows Azure, please refer to the companion document, [Windows Azure Security Overview](#). In that document, we discuss the security measures employed within Windows Azure in more detail. This best practices document is primarily focused on how customers can best utilize security features in Windows Azure, Windows Server, and other technologies to create secure applications.

Intended Audience

This paper is intended to be a resource for Information Technology (IT) Professionals. This could include designers, architects, developers and testers who build and deploy secure Windows Azure solutions. It is not a programming guide with detailed code samples, although many links to detailed implementation guidance are included throughout. Those links are aggregated in [Appendix C: Security Resources for Windows Azure Developers](#) for easier reference.

Overview of Windows Azure Security

Securing your Windows Azure deployment can follow different paths depending on which type of compute service is chosen. For example, the steps necessary to secure Windows Azure Cloud Service Web Roles and Worker Roles are slightly different than the steps required for a Windows Azure Virtual Machine. In practice, many customers choose to mix several compute types in their cloud environment, as certain models may apply better to different tasks; multiple cloud services, virtual machines, and Web Sites can all work in conjunction. The pros and cons of each should be weighed when making architectural decisions.

This section gives an overview of the different types of compute services and the steps necessary to secure them. Later sections will provide more specifics.

Threats in the Cloud

In 2010, the [Cloud Security Alliance](#) published a paper outlining the [top threats to cloud computing](#). In this paper, they note that cloud computing represents “one of the most significant shifts in information technology many of us are likely to see in our lifetimes.” There is great potential and promise for the cloud, but those looking to adopt cloud computing are understandably nervous and excited about the business prospects. Customers are excited about reducing capital costs, divesting themselves of infrastructure management, and taking advantage of the agility delivered by on-demand provisioning of cloud-based assets. However, IT architects are also concerned about the risks of cloud computing if the environment and applications are not properly secured, and also the loss of direct control over the environment for which they will still be held responsible. Thus, any cloud platform must mitigate risk to customers as much as possible, but it is also incumbent on the subscriber to work within the cloud platform to implement best practices as they would for on-premises solutions.

Generally, traditional threats will continue to exist in the cloud, such as cross-site scripting (XSS) or code injection attacks, Denial-of-Service (DOS) attacks, or credential guessing attacks. Some old threats are mitigated, since patching may be automated (for Platform-as-a-Service, or PaaS, only), and cloud resiliency improves failover across a service. Some threats are expanded, such as those concerning data privacy (location and segregation) and privileged access. New threats are introduced, such as new privilege escalation attacks (VM to host, or VM to VM), jail-breaking the VM boundary or hyper-jacking (a rootkit attack on the host or VM). Microsoft has taken extraordinary measures to protect Windows Azure against those classes of threats.

The diagram below demonstrates how management responsibilities are shifted from the customer to the cloud:

Cloud Services Shared Responsibility

On-Premises	IaaS	PaaS	SaaS
Customer-Owned	Windows Azure		
	Applications		
	Data		
	Runtime		
	Middleware		
	O/S		
	Virtualization		
	Servers		
	Storage		
	Networking		

Figure 1: Separation of Roles and Responsibilities in a Cloud Environment

Moving to a cloud platform is ultimately a question of trust vs. control. As seen above, with the Infrastructure-as-a-Service (IaaS) model, the customer places trust in the cloud provider for managing and maintaining hardware. The cloud provider secures the network, but the customer must secure the host and the applications. However, for PaaS, the customer gives further control of the host, the network, and runtime components. Thus, the cloud vendor would be responsible for ensuring that the host and runtime are properly secured from threats. In both cases the customer would be responsible for securing applications and data (e.g., authentication, authorization, configuration management, cryptography, exception management, input validation, session management, communication, audit and logging).

Software as a Service (SaaS) presents one further level of abstraction. In this case, the cloud provider manages all levels of the stack all the way up to the application. Customers provide configuration information and sometimes high level code, but that is the end of their responsibility. Two examples of SaaS on Windows Azure are Windows Azure Web Sites and Windows Azure Active Directory. In both of these cases, management of the underlying system is the responsibility of Windows Azure. Security is still an important consideration with SaaS, as an insecure configuration can be just as much of a risk as an insecure virtual machine. For example, a misconfiguration of Windows Azure Active Directory could result in an unauthorized user gaining access to something they shouldn't.

Securing Windows Azure IaaS Virtual Machines

Windows Azure Virtual Machines provide the ability to create and manage the entirety of a VM. These VMs can be deployed from a library of images including Windows Server 2008 R2 SP1, Windows Server 2012, Windows Server 2012 R2 and several Linux distributions provided by Microsoft partners. You can also create custom images using one of these operating systems as a starting point and upload them to Windows Azure.

The Windows Azure VM Depot, an offering of Microsoft Open Technologies, provides a community gallery of preconfigured Linux images for Windows Azure. Note, however, that Microsoft does not scan these images or otherwise evaluate their security posture. Customers should use these images only if they trust the suppliers.

Windows Azure Virtual Machines run on virtual hard disks (VHD) stored in Windows Azure Blob Storage. Unlike the other two compute service offerings (Windows Azure Cloud Service Roles and Windows Azure Web Sites), this provides the customer full and exclusive control of their VMs. Windows Azure does not take any ongoing role in managing or configuring the operating system running in the VM. For deployments using the Azure-provided Windows Server images, very limited initial configuration is done at the time of deployment. These changes are chiefly done to enable tighter integration with Windows Azure. For example, this includes an agent which allows keys to be deployed into VMs through the Portal or Service Management API.

This section will concentrate on Windows Server image deployments. Hardening information for Linux distributions can be obtained from the distribution's provider. Security for Windows Azure Virtual Machines shares many aspects with security for more traditional deployments. Securing the operating system of a VM requires roughly the same steps as securing a physical server. The security features provided by Windows Azure mainly reside at the network and infrastructure layer. Those features will be described in greater detail further below.

Common Security Topics for Windows Azure IaaS Virtual Machines

A Windows Virtual IaaS Virtual Machine is the compute offering which places the most security responsibility in the hands of the tenant. It is also the offering which provides the most flexibility and ability for customization.

Anti-Malware and Anti-Virus

Microsoft recommends that customers run some form of anti-malware or anti-virus on all virtual machines. In general, running anti-virus on a virtual machine should be no different than running on a conventional physical host, although centralized management may take a different form. The use of [Windows Azure Virtual Networks](#) could be used to facilitate a connection back to a central management solution such as [Microsoft System Center Configuration Manager](#).

Monitoring and Diagnostics

Windows Azure will maintain an external log of performance metrics for Virtual Machines. Events logged within a VM such as security events and audit events will remain on that virtual machine's VHD. It is recommended that

those logs be moved off of the VM for analysis and to thwart attempted tampering. Either moving to a separate location within Windows Azure or sending back on-premises are appropriate. Both [Microsoft System Center Operations Manager](#) and [Windows Event Forwarding](#) are tools which can be used to facilitate this.

Operating System Configuration

Stock Windows Server images in the Windows Azure gallery are very similar to out-of-the-box installations, and there are several varieties from which to choose. As of this document's publication date, these range from default Windows Server 2012 R2 Datacenter to Windows Server preconfigured with BizTalk Server, SQL Server, and various other products. Windows Azure keeps reconfiguration of these images to a minimum. Microsoft products follow a ['Secure by Default'](#) principal. Customers can perform additional security hardening on these images, using the same procedures for a conventional [physical deployment](#). Two important deviations from out-of-the-box installations are that Windows Azure images enable both Remote Desktop and PowerShell Remoting by default.

Operating System Updates

Windows Azure does not push Windows Updates to Windows Azure Virtual Machines since these machines are intended to be managed by the user. This is just like any on-premises machine. Customers are, however, encouraged to leave the automatic Windows Update setting enabled. Another option is to deploy a [Windows Server Update Services \(WSUS\)](#) server or another suitable update management product either on another Windows Azure Virtual Machine or on-premises. Both WSUS and Windows Update keep VMs up to date. It is also advisable to make use of a scanning product to verify that all IaaS Virtual Machines are up to date.

Stock images provided by Windows Azure are routinely updated to include the most recent round of Windows Updates. However, there is no guarantee that images will be current at deployment time. It is possible there may be a slight lag (no more than a few weeks) from public releases. Checking for and installing all Windows Updates should be the first step of every deployment. This is especially important to remember when deploying images you provide or from your own library. Images provided as part of the Windows Azure gallery always have automatic Windows Updates enabled by default.

Resiliency and Fault Tolerance

Windows Azure periodically has to reboot or shut down the systems hosting customer Virtual Machines. This can happen when the host operating system is updated or when there is planned hardware maintenance. By configuring [Availability Sets](#) with [Load Balancing](#), Windows Azure will ensure that these Virtual Machines are assigned to different update and fault domains. The infrastructure will take this into account and honor update domains during planned maintenance. If properly configured with at least two instances, Windows Azure guarantees uptime of IaaS per the Service Level Agreement (SLA). More information about data resilience is provided in the [Data Redundancy with Windows Azure Storage](#) section. A separate section details

[Fault Domains.](#)

Secrets Management

Certificates on IaaS Virtual Machines should be stored in the standard [Windows Certificate Store](#). This way they are protected by the operating system. Certificates can be distributed to the machine at initial deployment [via PowerShell cmdlets](#) utilizing the Service Management API and through the management portal. These processes work for both Linux and Windows Virtual Machines. Distribution can also be accomplished by more conventional means such as logging into the Virtual Machine or by using Active Directory group policies on domain-joined server images.

More information on [Certificate Management is available through TechNet](#).

Securing Windows Azure Web Sites

Windows Azure Web Sites enables developers to quickly build, deploy and run Web Sites in the scalable cloud environment. Developers can author Web Sites using ASP.NET, PHP, node.js, Python or classic ASP and then publish their Web Sites using one of a number of standard publishing technologies.

Windows Azure Web Sites is a multi-tenant Software as a Service (SaaS) offering that is designed specifically for hosting Web pages. Developers who are familiar working with Web applications on IIS should find Windows Azure Web Sites to be a very similar experience. In the *Free* and *Shared* [scaling modes](#), a customer's Web Site runs in its own application pool (worker process) that is isolated from all other processes running on the same server. In other words, multiple tenants potentially share the same Virtual Machine. In the *Standard* scaling mode, a customer's Web Sites runs on a set of dedicated Virtual Machines reserved exclusively for a single customer's use. This provides an additional isolation layer not provided by the other modes.

From a security perspective, most commonly-held best practices and [security guidelines](#) for Web developers building on Windows Server IIS continue to apply in the cloud. Since Windows Azure Web Sites is a Web hosting offering built on Windows Azure's PaaS capabilities, the one major difference security-wise is that developers don't have to manage machine-level or network-level security configurations. Unlike PaaS, however, operating system configuration for Windows Azure Websites is completely abstracted from the end user. Security configuration of the surrounding network environment, as well as individual Web servers, is handled by the Windows Azure Web Sites service. As a result, developers can instead concentrate on writing scalable and secure Web applications following the best security practices appropriate for their chosen development environment (ASP.NET, PHP, node.js, Python, classic ASP, and others).

Common Security Topics for Windows Azure Web Sites

A Windows Azure Web Site is a Software as a Service compute offering. This means that many common security topics which apply to PaaS and IaaS do not apply.

<i>Anti-Malware</i>	Windows Azure Web Sites tenants are not permitted to install software on the underlying operating system. Those needing to do so should choose from either Windows Azure Cloud Service Web Roles or Windows Azure Virtual Machines. Windows Azure monitors the state of the Virtual Machines hosting Windows Azure Web sites.
<i>Monitoring and Diagnostics</i>	Windows Azure Web Sites offer the ability to monitor a number of metrics relating to Web Site usage. Tenants do not have access to the underlying operating system's event logs.
<i>Operating System Configuration</i>	Windows Azure Web Sites tenants are not permitted to reconfigure the underlying operating system.
<i>Operating System Updates</i>	Windows Azure manages and deploys all applicable operating system updates to the Virtual Machines running Windows Azure Web Sites. The Windows Azure security team meets regularly with the Microsoft Security Response Center to prioritize and deploy Windows Updates such that they will not adversely affect tenant workload availability.
<i>Resiliency and Fault Tolerance</i>	The Windows Azure infrastructure automatically manages Windows Azure Web Sites deployments to ensure uptime and recovery from faults. More information about data resilience is provided in the Data Redundancy with Windows Azure Storage section.
<i>Secrets Management</i>	Web Sites running in Free mode only support a single, shared SSL certificate for sites addressed via https://sitename.windowsazurewebsites.net . Custom hostnames and SNI SSL certificates are supported for Web Sites running in Shared mode. Web sites running in Standard mode have the additional option to purchase a static IP address and configure IP-based SSL.

Securing Windows Azure PaaS Roles

The primary factor which makes a Windows Azure PaaS Role different from a standard Virtual Machine is that the underlying operating system used to host your software is temporary and transient. This is a compute model known as Platform as a Service (PaaS). To avoid confusion with Windows Azure Virtual Machines, the Virtual Machine used to run a Windows Azure Cloud Service is known as a Role. A complete Cloud Service can also include IaaS Virtual Machines alongside PaaS Roles. This section is only concerned with PaaS Roles.

Roles are created and deleted dynamically, can scale themselves based on load, and handle redundancy by creating multiple identical clones and automatically balancing work.

Unlike Windows Azure Virtual Machines (which are entirely the tenant's responsibility), the Windows Azure infrastructure actively configures and manages Cloud Service Roles. For example, Windows Azure manages tasks such as installing updates and restarting or migrating Roles upon fault. The effect of this is that data written to the local hard disk in a Windows Azure Cloud Service Role shouldn't be treated as persistent. Roles are transient and developers must use other options such as Windows Azure Storage to persist state.

The fact that Roles can be thought of as temporary disposable artifacts affects much of how security must be implemented.

Common Security Topics for Windows Azure PaaS Roles

A Windows Azure PaaS Role fills the middle ground between an IaaS Virtual Machine and a SaaS Windows Azure Web Site. Some security management is in the hands of the tenant, while some is managed by the platform.

Anti-Malware

Malware has a difficult time living on a Cloud Service Role. Roles are temporary and a single VM doesn't run for long before being recycled by the fabric. Nevertheless, Anti-malware protection for Cloud Service Roles is a recurring ask from customers. Windows Azure is working on an integrated version of Microsoft Endpoint Protection for Windows Azure Roles. As of the publication date of this document, an official release date has not been confirmed. Check the [Windows Azure Security](#) blog for future updates.

Third party anti-malware products which support command-line installation on Windows Server may work as well. Installation and configuration could be performed using [startup tasks](#) specified in the Role's service definition file. Applications provided by third parties are not covered by Microsoft Support.

Monitoring and Diagnostics

[Windows Azure Diagnostics](#) can be configured to collect application and security logs from Roles and transfer them to a storage account. Tenants may wish to offload this data to a Security Information and Event Management suite. The method which best suits your organization's log management processes should be chosen.

1. A security log aggregation agent can be installed on each Role through the use of startup tasks and be used to push data back to the central aggregator.
2. A security log aggregation agent can be written to pull data from [Azure Table Storage](#) using the REST API. PaaS Roles are configured to write events there.

Operating System

In general, very little reconfiguration of the underlying OS of a PaaS Role should need to happen. Most hardening steps for traditional deployments

Configuration

usually lock down access controls, networking controls, and software controls.

In the case of PaaS, access is already severely limited. The only way to log into a Role is through a single Remote Desktop Protocol (RDP) user account which is enabled on demand; this act is discouraged on production Roles. Most file-level Access Control List (ACL) considerations are replaced by [storage keys and SAS tokens](#). Networking [is automatically configured](#) for least privilege by the service definition file. The only additional software deployed to a role is defined within your application package. It is also important to note that Roles are disposable and run for short periods of time before being recycled.

If there are specific aspects of the OS which need to be changed, it is possible to reconfigure the underlying Windows Service instance. Developers can use [startup tasks](#) to make targeted configuration changes. Various command line scripts can be added to a Role's service definition file which are executed before the application runs. This is the preferred method for making OS-level changes as opposed to Active Directory (AD) joining and Group Policy Objects (GPOs) which present an entire set of difficulties in PaaS deployments.

Operating System Updates

Windows Azure manages and deploys all operating system updates. The Windows Azure Security Team meets regularly with the Microsoft Security Response Center to prioritize and deploy Windows Updates while not adversely affecting tenant availability. Updates will be automatically bundled into new versions of the Guest Operating System. Once Microsoft releases that new version of the Guest OS into production, the upgrade process begins.

If a Cloud Service has been configured to use "automatic" guest OS versions, Windows Azure will migrate the service to Roles running the newest OS version. It stages the upgrade so that availability will not be adversely impacted. The existing Roles running your cloud service's software are not patched in place. Entirely new, clean virtual machines are created from the Guest OS image and then configured to run your software. It is highly recommended that Guest OS version retain the default setting of "automatic" in the service configuration.

More information about the Guest OS Update process is available in the documentation for [Managing Upgrades to the Windows Azure Guest OS](#).

Resiliency and Fault Tolerance

Windows Azure will automatically configure update domains for Role instances. Update domains allow the Azure infrastructure to maintain availability of a service through planned maintenance. Some examples of planned maintenance include installation of Windows Updates or planned hardware outages and upgrades. An explanation of the PaaS Role update process is provided above.

More information about storage resiliency is provided in the [Data Redundancy with Windows Azure Storage](#) section. As a reminder, the VMs hosting PaaS

Roles are temporary and important data must be saved in Windows Azure Storage and not on the Role's local disk.

*Secrets
Management*

Certificates should be [uploaded to the Windows Azure Management portal](#). Once uploaded to the portal, these certificates are automatically securely provisioned to the appropriate Roles. It is not recommended to store certificates with private keys in Windows Azure Storage. Additional secrets management steps will be detailed in [the Encrypting Data in Windows Azure](#) section.

Overview of Identity Management and Access Control

Identity management has emerged as an increasingly complex issue for developers, and remains a priority, even as business networks change. Identity management is as much about preventing unauthorized third-party access to data as it is about controlling the authorized use of data. Identity management helps systems control the amount and type of data that users can access, and it helps ensure that users are performing necessary functions at the lowest-possible privilege levels. Identity management is also critical for maintaining separation of roles and duties, which may be required by specific regulatory and compliance standards.

Today, developers face a common challenge: to provide authorized users, clients and systems with access to the data they require at any time, from any technology, in any location, while meeting basic security requirements for confidentiality, availability and integrity. Furthermore, developers of business software increasingly need to securely access user and organizational data from applications outside of the company's perimeter.

At first glance, cloud computing technology might seem inappropriate or poorly suited to meet such rigorous standards. Traditionally, developers seek to restrict access and keep a close hold on data. In the cloud, data lives in an environment which is owned and managed by another party. In addition, cloud-based resources are generally exposed to the Internet. These concerns, while legitimate, can be partly addressed by effectively using claims-based identity and cloud-based directory technologies.

Claims-based identity, an approach to authentication and access management based on open protocols, is an access control strategy that is consistently applied across the full range of Microsoft products and services. One of the key properties of claims-based identity is that it reduces infrastructure dependencies because applications protected with claims-based identity can be hosted on-premises or in the cloud without changes. Applications targeting Windows Azure can take advantage of the same developer tools, identity management features and services that are available to their on-premises counterparts.

The most relevant identity technologies and services that work with Windows Azure to protect applications and resources are:

- Windows Azure Active Directory (AAD)
- Windows Identity Foundation

- Active Directory Federation Services 2.0

Windows Azure Active Directory

Windows Azure Active Directory enables cloud-based applications to perform essential identity management tasks, such as Web sign-on and directory queries, without burdening developers with significant infrastructure requirements such as VPNs and domain membership. Windows Azure Active Directory includes two functional areas, detailed below.

Directory

Windows Azure Active Directory is a cloud-hosted, Software as a Service (SaaS) offering analogous to Windows Server Active Directory. Windows Azure AD is the directory used by some of Microsoft's other SaaS products such as Office 365, Windows Intune, and even Windows Azure's Management Portal itself. It handles user authentication, directory queries and extensibility. Windows Azure AD can extend an existing on-premises AD deployment, providing a single point of management for all identity assets, allowing users to federate their credentials with Windows Azure and other cloud services. Windows Azure AD also supports cloud-only directories, where user accounts are maintained in cloud-only tenants with no dependency on—or need for—on-premises infrastructure. Windows Azure AD can provide the following capabilities for cloud applications.

- Adding sign-on capabilities to a Web application, accepting users from cloud directories using an ever-growing range of open protocols, from Security Assertion Markup Language (SAML) to Web Services- Federation (WS-Federation). With the use of open protocols, developers can take advantage of Windows Azure AD's sign-on capabilities from any modern development platform, including those from third parties such as Java or Python.
- Enablement of cloud native applications to query the directory for users and organization information, without having to pierce holes in a firewall or having to deal with the latency associated with reaching back to on-premises assets. Windows Azure AD offers a REST-based API, the Graph API, which can be easily consumed from any application and platform where an HTTP stack is available. The Graph API is protected via OAuth2 that empowers administrators to control what applications have access to the directory data.
- For SaaS applications, developers can easily onboard organizations as new customers by offering an easy consent flow. Doing so allows Windows Azure AD tenant administrators to grant to an application access to their cloud directories for sign-on and query privileges. For example, all Office365 customers can now be enabled to use applications with as little as a couple of clicks in their administration portal.

To be clear, a Windows Azure AD domain is not the same as one for on-premises Windows Server AD Domain Services. In Windows Server AD Domain Services, a domain is a security, storage, and trust boundary. In Windows Azure AD, a domain is more synonymous with a DNS domain. By using the Windows Azure Active Directory Sync tool, your company's administrators can keep your on-premises Active Directory continuously synchronized with Windows Azure AD. Directory synchronization is

intended as an ongoing relationship between your on-premises environment and Windows Azure AD. You can also federate Windows Azure AD with your on-premises directory to enable single sign-on scenarios, as seen in the figure below.

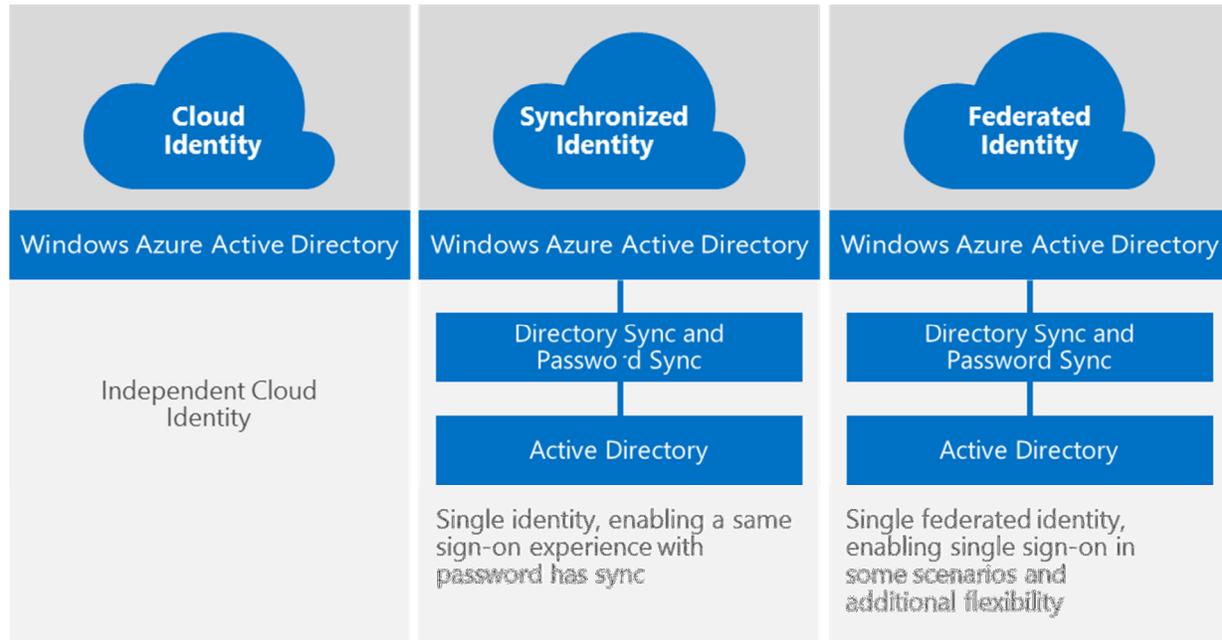


Figure 2: Tiers of Identity Synchronization

In Windows Azure AD, cloud identities are created with domains in the <tenant>.onmicrosoft.com form. Custom public domains may subsequently be registered with Windows Azure AD, and must be registered with DNS. Multiple public domains may be registered with a single tenant.

Access Control

Part of the Windows Azure AD offering, the Access Control Service (ACS) is a hosted service that can be used to offload from your apps authentication, access control and management of relationships with external identity providers. Web applications, rich clients, server to server flows and REST services are all eligible to take advantage of ACS.

ACS provides developers centralized authentication and authorization for applications in Windows Azure using either popular Web identity providers (such as Microsoft Account, Google, Yahoo!, Facebook and any OpenID2.0 provider) or your on-premises Windows Server Active Directory. Furthermore, ACS provides a sophisticated rule engine to transform and enrich incoming claims to normalize authentication information for your application portfolio.

Enabling Sign-On to Web Applications

A Web single sign-on solution typically entails configuring a Web application to outsource its authentication functions to one external entity, commonly referred to as Identity Provider (IdP). In concrete terms, this means that the application will be configured to redirect unauthenticated requests to the IdP, according to some sign-on protocol such as SAML-P or WS-Federation.

The authority (or IdP) handles the authentication experience, and it usually requires the Web application to be known via some provisioning process. Upon successful authentication, the browser gets redirected back to the Web app along with a security token carrying information about the incoming user; the application validates the token, typically via some identity-aware middleware, and if the check succeeds the user is considered authenticated and is signed in. That high-level description applies to Windows Azure AD, as well. For example, you would register your Web application with a Windows Azure AD tenant. The according passive (browser-based) flow would work as follows:

1. The user browses to the Web application
 - The Web application detects that the user is not authenticated and redirects him to the Federation Service
2. The user automatically browses to the sign-on page
3. The user authenticates with Windows Azure AD
 - Windows Azure AD builds the security token and passes it back to the user
4. The user POSTs the token to the Web application
 - The Web application uses the token to gain access to a resource, such as a Web service

A notional flow is illustrated in the figure below:

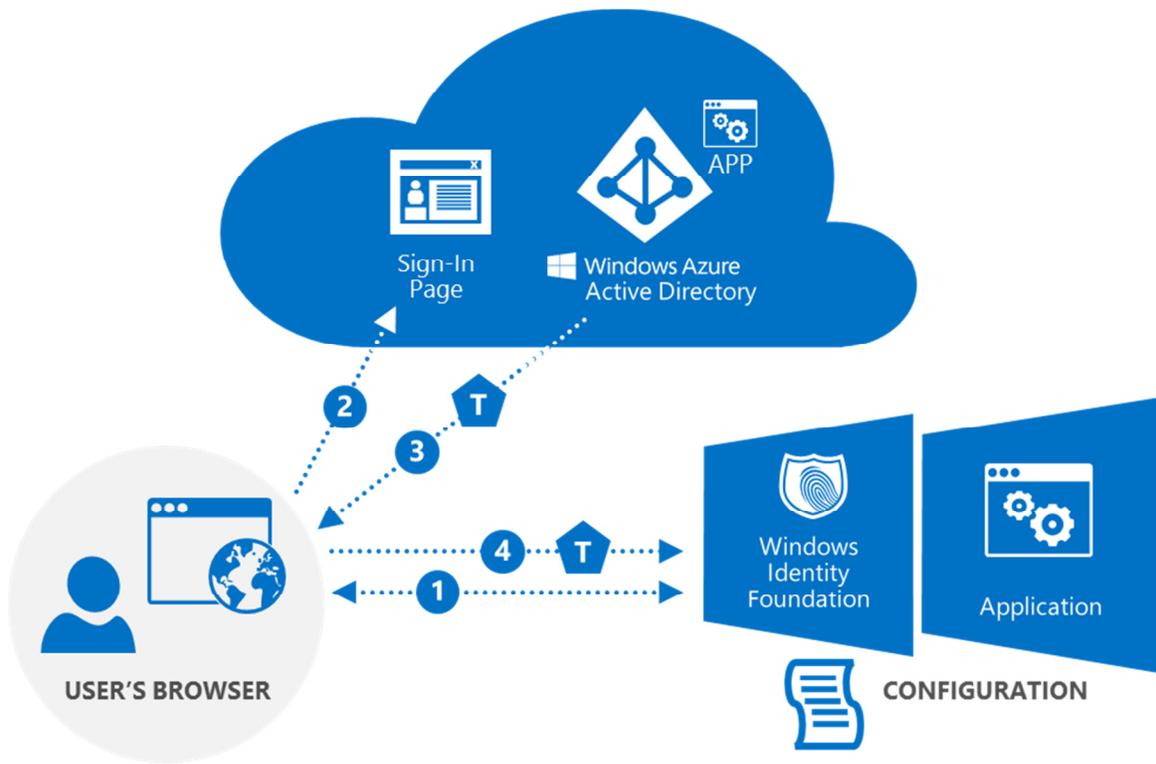


Figure 3: Browser Application Authentication against Windows Azure AD

In the .NET platform, the first step boils down to configuring the classes that .NET offers out of the box for working with claims-based identity and federated authentication. Those classes are collectively known as Windows Identity Foundation (WIF), and they include HTTP modules and configuration settings which can be used to add the interception layer performing the redirection and authentication tasks introduced in Steps 1 through 3. Visual Studio 2012 offers tools to help automatically configure Web applications to use WIF to outsource authentication to external authorities which support specific Web Single Sign-on (SSO) protocols such as WS-Federation.

Windows Identity Foundation

Windows Identity Foundation (WIF) is a foundational technology in the .NET Framework 4.5. It enables .NET developers to offload the identity logic from their application, providing a solid development model based on separation of concerns. Non-experts can easily secure their applications without being exposed to the underlying complexity of cryptography and protocols. They can leverage Visual Studio integration features such as point-and-click wizards which result in applications protected using open, interoperable standards such as WS-Federation and, via extensibility points, OAuth 2.0.

When using Windows Identity Foundation the mechanics of authentication are provided by external services, using platform-independent protocols. The application receives information about authenticated users in forms of claims, which can be used for simple or traditional role-based access control (RBAC) to sophisticated access control policies.

Because open standards are used, the authentication can take place regardless of where the user accounts are maintained or where the application is hosted. As a result, single sign-on across on-premises and Windows Azure hosted resources is easily achieved.

Although the authentication services can be provided from any platform complying with the open protocols used by Windows Identity Foundation, the best way to leverage existing investments in the Windows infrastructure is to outsource authentication to Windows Azure Active Directory or Active Directory Federation Services 2.0.

Active Directory Federation Services 2.0

Although Active Directory Federation Services 2.0 (AD FS 2.0) is a technology that can be deployed on-premises, it can play a key role in enabling authentication for Windows Azure applications.

AD FS 2.0 is a Windows Server role that extends Active Directory with claims-based identity capabilities. AD FS 2.0 provides AD with a Security Token Service (STS) which is a single interface enabling existing users to authenticate using applications regardless of whether they are hosted in a data center, at one partner's site or in the cloud. Users are no longer constrained by the boundaries of their local network. If an application hosted in Windows Azure has been developed using Windows Identity Foundation (or an equivalent stack complying with the same open standards), AD FS 2.0 allows instantly granting anyone with an account in the on-premises Active Directory access to this application. All of this can occur without requiring any form of synchronization, new account provisioning or duplication.

AD FS 2.0 facilitates the establishment and maintenance of trust relationships with federated partners, simplifying access to resources and distributed single sign-on. AD FS 2.0 implements standards such as WS-Trust, WS-Federation and the SAML protocol, and successfully passed the latest public Liberty Alliance SAML 2.0 interoperability testing which proved out-of-the-box interoperability with products from IBM, Novell, Ping Identity, SAP, Siemens and many others.

Active Directory Authentication Library

The [Windows Azure Authentication Library \(ADAL\)](#) is a library meant to help developers take advantage of Active Directory for enabling client apps to access protected resources. ADAL enables client application developers to easily authenticate users to Windows Azure Active Directory or other identity providers (whether on-premises or cloud authorities alike), and then obtain access tokens for securing API calls. To duplicate this functionality without ADAL, developers must write non-trivial authentication code and understand protocol details. With ADAL, a developer can focus on business logic in their application, ignore most protocol details and easily secure resources without being an expert on security. ADAL leverages the JSON Web Token (JWT) format as well as OAuth 2.0 application authorization flows, as described below.

JWT Token Format

A relatively new token format in addition to SAML 2.0 tokens is the lightweight JWT format. This token format uses JSON to represent security claims, and is designed for compactness. It is easily transportable over HTTP using Query String parameters and HTTP headers. Though it is not a replacement for SAML, it is less expensive in size and complexity. Signing and encryption are defined in the header, though not required (and so would have to be secured by the transport).

The JSON Web Token Handler for Microsoft .Net Framework (JWT handler) is a library which adds the JSON Web token format as a first-class citizen in the .NET programming model. The JWT handler can be used both within the WIF pipeline to secure existing Web sites and services with JWT tokens in addition to the formats supported out of the box (such as SAML1.1 and SAML2). The JWT handler can also be used standalone, with no direct dependencies on WIF configuration.

OAuth 2.0

In Microsoft's ongoing efforts to build the world's most open Identity Management service, Microsoft has introduced the new OAuth 2.0 code grant type authorization flow. This builds on top of already strong support for SAML, WS-Federation and the client credentials grant type in OAuth 2.0 for server to server flows. OAuth 2.0 is the next evolution of the OAuth protocol which was originally created in late 2006. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for Web applications, desktop applications, and mobile phones.

The authorization code grant enables user authentication flows from native applications, and is optimized for confidential clients (i.e., a client that maintains the confidentiality of their credentials). Moreover, it offers important features, such as refresh tokens, which can help maintain long running sessions while minimizing the need to prompt users for their credentials. The authorization code

provides a few important security benefits, such as the ability to authenticate the client, as well as the transmission of the access token directly to the client without passing it through the resource owner's browser and potentially exposing it to others, including the resource owner.

Native/Modern Application Support

Windows Azure AD supports native client applications allowing users to authorize client applications to access a Web Service on their behalf. This enables a mobile workforce to use applications that are registered in a Windows Azure AD tenant. Administrators are able to configure allowed applications in Windows Azure AD as well as the permissions granted to the applications. The OAuth 2.0 authorization grant supports the issuance of refresh tokens that provide native applications the ability to maintain long running sessions while minimizing the number of times a user is prompted for their credentials. In addition, administrators are able to control the relationships between client and service applications by controlling the permissions of individual client's permissions to access specific services.

Windows Azure AD and ADAL enable the key scenarios below:

- **Authenticating Users of a Client Application to a Remote Resource:** In this scenario, a developer has a client application that needs to access a remote resource, such as a REST service. He has a Windows Azure subscription, and therefore can leverage Windows Azure AD to manage user authentication. He also knows how to invoke the downstream service, and knows the Windows Azure AD tenant the service uses. As a result, he can use ADAL to facilitate authentication with Windows Azure AD, either by fully delegating the authentication experience to ADAL and Windows Azure AD, or by explicitly handling user credentials. Authentication with Windows Azure AD results in a token, and ADAL makes it easy to obtain this token and use it to make requests to the REST service.
- **Authenticating a Server Application to a Remote Resource:** In this scenario, a developer has an application running on a server that needs to access a remote resource, such as a REST service. She has a Windows Azure subscription, and therefore, can leverage Windows Azure AD to manage the application's identity. She also knows how to invoke the downstream service, and knows the Windows Azure AD tenant the service uses. As a result, she can use ADAL to facilitate authentication with Windows Azure AD, by explicitly handling the application's credentials. Authentication with Windows Azure AD results in a token, and ADAL makes it easy to obtain this token and use it to make requests to the REST service.

Graph API

Windows Azure AD Graph provides programmatic access to Windows Azure AD through REST API endpoints. Using Windows Azure AD Graph, developers can execute create, read, update and delete (CRUD) operations on Windows Azure AD objects such as users and groups. In the on-premises world, you would usually programmatically access Windows Server Active Directory by using ADSI or ADO.NET libraries. In the cloud, you programmatically access Windows Azure AD using Windows Azure AD Graph. Windows Azure AD Graph enables the following key scenarios:

- **Multi-Tenant Applications That Require Windows Azure AD Access.** Building a multi-tenant application that requires access to a tenant's directory data (very similar to an on-premises application that use LDAP to query the local directory). Directory access for reading or writing data is done by calling the Graph API. Typical use cases include people pickers, validating a user's security group membership, updating group membership, provisioning new users and groups, resetting users' passwords, and validating a tenant or users' licensing information.
- **Creating Reusable Features That Require Windows Azure AD Access.** An independent software vendor (ISV) that specializes in creating and selling reusable features that extend the functionality of cloud applications. An ISV can offer its customers reusable features that require access to Windows Azure AD objects.

More Information

Detailed implementation guidance is available through the [Windows Azure Identity Fundamentals](#) article.

Windows Azure Service Layer Security Considerations

This section highlights some specific threats that developers are responsible for mitigating in the cloud and describes the protections that Windows Azure provides against an array of service-, platform- and infrastructure-layer security threats.

Windows Azure Service-Layer Security Overview

The threat landscape for a cloud-based Web Service is substantially different from traditional hosted Web services in terms of mitigating tools and technologies. Threats also vary dramatically among cloud providers.

The key points are as follows:

- Developers must map the traditional on-premises enterprise security requirements of their application or service to Windows Azure platform services that provide comparable functionality. Any remaining threats must be mitigated by the application or service.
- Understand the security requirements of the service being designed or migrated, especially in the context of authentication, authorization and auditing. The platform services which provide these features (Windows Identity Foundation, Windows Azure Active Directory, Windows Azure Monitoring and Diagnostic APIs) and the methods developers use to invoke them are substantially different from those provided in an on-premises enterprise deployment (Kerberos, Active Directory, and Windows Event Logs).
- Leverage Windows Azure platform services to build more secure applications.

Despite some of the protections that moving to the cloud may offer, developers are still responsible for writing inherently secure code -- and they are still responsible for the security of their applications when dealing with threats to the Web service code itself. Input field constraint, sanitization and validation are still some of the most important ways to help protect a Web service application, whether it is a cloud-based application or not. Windows Azure runs Web Roles in Internet Information Services (IIS). Because of IIS's secure default configuration, developers inherit some basic IIS protections, such as the `ValidateRequest` configuration setting.

Developers must mitigate cross-site scripting attacks by encoding and validating inputs to their services with the [Microsoft AntiXSS](#) library or other similar libraries, especially if that input is displayed back to the user in a Web page. Cross-site request forgery (CSRF) attacks must also be mitigated by setting a hidden session token in client requests and setting [Page.ViewStateUserKey](#) to the current session ID.

Namespace Configuration

Here are a few more configuration issues to be aware of in the cloud:

- Avoid using the `<sitename>.cloudapp.net` or `<sitename>.azurewebsites.net` domains as the externally facing name for production deployments - use a custom domain name instead. An important distinction between the enterprise and the cloud is that the `cloudapp.net` and `azurewebsites.net` namespaces are shared among all Azure customers. This means that the `cloudapp.net` and `azurewebsites.net` namespaces are inherently less trusted than the domain namespace of a single enterprise because one customer of Windows Azure does not automatically trust all other customers in that domain namespace. Don't create code that requires users to place `cloudapp.net` or `azurewebsites.net` in the trusted sites list in their Web browser.

Note: Windows Azure does not currently support custom domains using DNS "A" records. Rather, a custom domain name is facilitated by pointing a "CNAME" record at the appropriate `cloudapp.net` or `azurewebsites.net` name.

- Note: communication between different roles within the same subscription should resolve IPs using the `cloudapp.net` namespace. For example, internal name resolution should be used between a front end Web site which calls into a back end Web service. Doing so routes DNS traffic through Azure's internal DNS system and mitigate the threat of DNS poisoning. It also improves performance by keeping traffic internal. For more information, see [Secure Usage of DNS](#).
- Never scope cookies or [document.domain](#) to `cloudapp.net`. Instead, scope to the service subdomain (such as `contoso.cloudapp.net`, for example or, better, `www.contoso.com`).

For most content, only interactions with content from the same domain are allowed. For example, a typical page on `www.microsoft.com` can freely script content on any other page on `www.microsoft.com`, but it cannot script to pages that are located on a different Web domain. The DHTML Object Model uses the `document.domain` property to enforce this restriction. Only

pages with identical domain properties are allowed free interaction. The protocol of the URL must also match. For example, an HTTP page cannot access HTTPS content.

Important: Any attempt to broaden document.domain access can leave a service open to scripting attacks from the entire cloudapp.net domain namespace. IIS7 sets document.domain to the full subdomain by default (like contoso.cloudapp.net or www.contoso.com), but the scoping is changed often by Web developers that it warrants mention.

Data Access Security

When designing Cloud Service interactions with Windows Azure Storage, Shared Access Signatures can be a powerful tool. Shared Access Signatures are effectively access tokens that bestow a set of rights against an Azure Storage container that normally would not be set public. Since Web applications are responsible for generating these tokens, they are also responsible for securely distributing them. In the event that one of these tokens is compromised, developers can either update the container metadata to invalidate the token, or simply create a new container for the data. However, this is still a reactive measure taken after damage could have already been done. Outlined below, are guidelines for minimizing risk while using Shared Access Signatures.

- Generate Shared Access Signatures with the most restrictive set of ACLs possible that still grant the access required by the trusted party.
- Use the shortest lifetime possible.
- Use HTTPS in the request URL so the token cannot be snooped on the wire if the request comes from customer on-premises applications. For more information, see [Protecting Data in Transit](#).
- Remember that these tokens are used to grant access to Windows Azure Storage. As with passwords, it's a bad idea to use the same ones over and over.
- When a storage key is changed, all Shared Access Signatures associated with it are revoked. It is important to add a procedure for regenerating SAS keys to any storage key rotation policy.

Different types of storage resources can receive different permission attributes. The documentation for creating [Shared Access Signatures](#) details all of these permissions.

Windows Azure Table Storage provides a simple structured storage environment and is not SQL-based, so common [SQL injection vulnerabilities](#) do not apply to an application that uses it. However, applications using the Azure SQL Database or other relational database services still need to mitigate traditional SQL injection threats.

Windows Azure Table Storage requests are either made directly, using HTTP GET and POST requests, or they are translated into these requests by the SDK and LINQ. Since HTTP requests are text, if this request string were constructed based on data gathered from users, then the user's strings could possibly change the semantics of the request (e.g. by inserting carriage returns or & characters.) One could think of this attack as being roughly analogous to SQL Injection. To avoid potential 'storage injection' attacks,

do not base any container names, blob names, blocks, block IDs, or table names on data gathered from users. When constructing a REST query that involves user data, help ensure the data's safety by URL encoding before concatenating it into the query.

Protecting Data in Transit

Whenever sensitive data leaves the Azure perimeter, it should be secured using HTTPS or similar means. Data traveling internally between different Windows Azure services (e.g. between a Cloud Service and Azure Storage) never leaves a Microsoft-owned network. Data traveling between compute machines in the same subscription doesn't leave the physical network segment (cluster). For those cases, HTTPS should be considered additional defense in depth.

More information about how Azure internally protects data in transit is provided in the [Eavesdropping / Packet Sniffing](#) section.

Note: HTTPS can also be used with [client certificate mutual authentication](#). This provides authentication and potentially authorization. These functions are above and beyond traditional one-sided SSL which provides confidentiality.

More information about network security which can be used to help decide if intra-role HTTPS is a necessary precaution is contained within the [Windows Azure Network Layer Security Considerations section](#).

Fault Domains

When Windows Azure instances are deployed, Windows Azure can automatically spread them among different fault domains. A fault domain is a physical unit of failure that roughly (i.e., not 1:1) corresponds to racks within the physical infrastructure of a data center. In other words, Windows Azure is responsible for deploying application instances to different physical servers or racks so that a single hardware failure does not bring down all instances. This occurs automatically when multiple Role instances are deployed in PaaS Applications. It is manually configured through [availability sets](#) with IaaS.

Data Redundancy with Windows Azure Storage

All data written to Windows Azure Storage is [automatically replicated](#) to three physical machines within the same datacenter. These machines share physical proximity (different racks in the same datacenter) but are identified with separate fault domains.

Customers who opt into the [Geo-Replication](#) service will see that data further replicated to another three places at a second physical location.

Note: Inter-datacenter geo-replication is encrypted in transit.

Windows Azure SQL Database Security Considerations

There are two different ways to add a SQL server to a Windows Azure application. The first, and simplest, is to utilize the Windows Azure SQL Database. The second is to [deploy Microsoft SQL Server on an IaaS Virtual Machine](#).

No matter which approach is taken, there are some common security themes when securing an application using SQL Database on Windows Azure.

- Connection strings should be [encrypted](#) (if applicable)
- The application must be hardened against [SQL Injection attacks](#)
 - Stored procedures should be used wherever possible
 - Parameterized SQL is discouraged
- Least privilege accounts should be used to access the database

There are pros and cons to consider when deciding whether to use the Windows Azure SQL Database or deploy SQL Server on an IaaS Virtual Machine, detailed in the table below.

Windows Azure SQL Database

- Simple setup and deployment
- Software as a Service (no configuration of underlying platform)
- Inherent fault tolerance
- Transparent Data Encryption (TDE) is not supported
- Windows Authentication is not supported (username/password only)

All data written to Windows Azure SQL Database is automatically replicated to what equates to three physical machines within the same datacenter. These machines share physical proximity (different racks in the same datacenter) but are identified with separate fault domains. Windows Azure SQL Database uses this model for high availability of the database and can trigger an automatic failover from the primary to the secondary node if a disruption should occur (hardware failure, scheduled maintenance, etc.). This will trigger a transient error on any open connection to the database. The consumer of the database should always implement transient error handling to trigger the reconnection to the currently active node.

SQL Server Deployed on an IaaS Virtual Machine

- Windows Authentication is supported
- TDE is supported
- Tenant has full responsibility for OS/platform configuration and hardening

MSDN provides documentation about [Connectivity Considerations for SQL Server in Windows Azure Virtual Machines](#).

There is a specific guide discussing [High Availability and Disaster Recovery for SQL Server in Windows Azure Virtual Machines](#).

Certificate Management in Windows Azure

This section describes the methods for managing certificates in Windows Azure. Properly managing certificates is a pivotal step in the process of creating and managing a secure Windows Azure application. Just like the traditional on-premises world, it is important to periodically rotate certificates.

Certificate Management for Windows Azure Cloud Services

Windows Azure provides a method to manage and securely provision certificates to Cloud Service roles. All certificates destined for Windows Azure Cloud Services – especially those containing private keys -- should be uploaded either through the Windows Azure Management Portal or through the Service Management API. Once these certificates are uploaded, they cannot be retrieved back through the Portal. Private keys are specially delivered in an encrypted state to the role during the service deployment process and marked as non-exportable as they are installed. Developers may access them through the standard Windows Certificate Store or via thumbprint references in the Service Configuration.

Storing private keys in Windows Azure Storage is not recommended. To do so is to fail to utilize a fundamental platform security measure. Public keys aren't as concerning as they are by definition, public.

Certificate Management for Windows Azure Virtual Machines

Certificates can be provisioned into the Virtual Machine images certificate store using the [standard process for Windows](#).

The process of pushing certificates and Secure Shell (SSH) keys into new Windows Azure Virtual Machines can be scripted with the Windows Azure PowerShell module. This is an especially important detail if your organization runs its own internal certificate authority or needs to deploy a large number of Virtual Machines. This is an excellent opportunity, if applicable, to add internal CA-signed certificates for RDP and remote Windows PowerShell administration.

Encrypting Data in Windows Azure

Encrypting data at rest in the cloud faces some interesting challenges when compared against traditional on-premises deployments. This is particularly true when it comes to Windows Azure Cloud Services and Windows Azure Web Sites. In both of those cases, local storage on the Virtual Machine hosting an application is transient and non-persistent. This means that standard Windows platform methods such as Data Protection Application Programming Interface (DPAPI) are unsupported in Cloud Services and Web Sites.

Conventional physical deployments often rely on [BitLocker Drive Encryption](#) to protect data at rest. BitLocker is not supported for boot volumes in a Windows Azure deployment. This is partially because guest Virtual Machines don't have the Trusted Platform Modules (TPMs) which normally store

encryption keys. This is combined with the fact that it isn't possible to input a decryption key during the Virtual Machine's boot process.

Windows Azure employs several measures to protect customer data. It may be useful to weigh those measures when deciding whether to implement data encryption. Those measures are the subject of the [Windows Azure Security Overview](#) document. A brief overview of those measures is provided below:

- Physical access to Windows Azure datacenters is severely restricted.
- The Windows Azure infrastructure is segmented off from Microsoft's internal corporate network.
- Microsoft does not maintain credentials into customer Virtual Machines.
- Access to infrastructure components (e.g. the machines which host customer Virtual Machines) is severely limited and heavily audited.
- All data traveling within Windows Azure datacenters or globally between Windows Azure datacenters travels over tightly controlled, Microsoft-owned links.
 - It is important to note that these links are not line-encrypted and may cross national borders.
- Numerous platform features prevent network eavesdropping and spoofing.
- All disk drives which housed customer data are securely wiped before disposal.

Those measures notwithstanding, many customers choose to implement data at rest encryption as an additional security measure. This is often a compliance requirement. It is also a best practice when handling extremely sensitive data. There are several methods for encrypting data within Windows Azure.

Certificates and Data Encryption within Windows Azure

Key management is one of the fundamental engineering challenges when implementing an encryption solution. In Windows Azure, this means that there is a question about how to get encryption keys into your application. Storing plain text encryption keys in either Windows Azure Storage or in an application configuration file is not recommended.

As mentioned in the Certificate Management in Windows Azure section, Windows Azure provides a method to securely provision private keys to Roles via the Windows Azure Portal or Service Management API. This method is limited to public/private key pairs and not symmetric keys. However, there are methods where securely provisioned private keys can be used in conjunction with symmetric cryptography.

There are two recommended methods to implement symmetric cryptography in Windows Azure using securely provisioned private keys, as follows:

<i>Method 1:</i>	A data string encrypted using PKCS-7 has a new symmetric key generated every time.
<i>PKCS-7</i>	The symmetric key is encrypted using a provided certificate's public key. The data string is encrypted using the symmetric key. Both the encrypted symmetric key and the encrypted data are concatenated together and stored.

private key can decrypt the symmetric key and in turn the data.

Pros: this mechanism is very simple to implement. Methods within the `System.Security.Cryptography.Pkcs` namespace will do the encryption and decryption with very little additional development.

Cons: asymmetric cryptography is slower than symmetric cryptography. This can have a performance impact when a large number of operations are performed. This method requires that every encryption/decryption include both asymmetric and symmetric operations. In addition, every encrypted element will increase in size due to the added encrypted symmetric key.

See

Appendix B: Using SDL-approved Cryptography in Windows Azure Applications for information on recommended algorithms.

*Method 2:
Shared
Symmetric
Key* A symmetric key can be generated at development time. This key is then encrypted using the public half of a key pair. That encrypted value can be safely stored in a configuration file, Windows Azure Storage, or a SQL database. The associated private key is deployed to Windows Azure through one of the prescribed methods as described in the previous section.

When the application loads, it pulls the encrypted symmetric key from storage or configuration, and then uses the private key to decrypt it. From then on, every encryption and decryption operation can utilize that symmetric key.

Pros: there is little additional processing or storage overhead. The bulk of all calculations are done using symmetric algorithms. The symmetric key is stored once.

Cons: due to the complexity of this mechanism, there are no readily callable methods which will do this in entirety. Implementation is largely manual.

[See Appendix B: Using SDL-approved cryptography in Windows Azure](#) for information on recommended algorithms.

Securing Connection Strings and Storage Keys

Unless additional security measures are employed, all developers working on a Windows Azure application have access to a plain text version of the connection strings and storage keys. Standard methods for encrypting a connection string using DPAPI or avoiding the problem entirely by using Windows Integrated Authentication are not supported within Windows Azure.

This vulnerability can be mitigated by encrypting the connection string within a PKCS-7 envelope and placing that envelope into the configuration file. This can be done by using the `RSAProtectedConfigurationProvider`.

SQL Connection String Encryption Resources

The following two guides provide steps which can be used within Windows Azure to protect connection strings.

- How to Encrypt Configuration Settings using ASP 2.0: <http://msdn.microsoft.com/en-us/library/ff650304.aspx>
- The following article describes a method for encrypting SQL connection strings. The same methods can easily be modified to encrypt other sensitive configuration items such as Storage Keys. [Windows Azure SQL Database Connection Security](#)

Note: `DPAPIProtectedConfigurationProvider` is not supported within Windows Azure PaaS or Windows Azure Web Sites

Protecting Data at Rest within Windows Azure SQL Database

Windows Azure SQL Database does not support SQL Server Transparent Data Encryption (TDE). If possible, sensitive data should be encrypted before being entered into Windows Azure SQL Database using one of the two previously mentioned methods.

TDE with Microsoft SQL Server Deployed as a Virtual Machine

Transparent Data Encryption (TDE) is supported when running Microsoft SQL Server on a Windows Azure Virtual Machine. If encryption at rest for a majority of data is a design requirement, adding a Microsoft SQL Server Virtual Machine to your cloud deployment might make sense.

More information is contained within the guide [Security Considerations for SQL Server in Windows Azure Virtual Machines](#).

Additional Cryptography Information

Appendix B: Using SDL-approved Cryptography in Windows Azure Applications contains more security best practices for cryptography usage.

A more detailed look at Crypto Services within Windows Azure is available from the MSDN article: [Crypto Services and Data Security in Windows Azure](#).

Auditing and Logging for PaaS Cloud Service Roles

Local Virtual Machine disk access by Windows Azure Cloud Service and Web Site Roles should be considered temporary and not reliable for anything more than temp files and caching. Event data can be logged to Windows Azure Storage through the Diagnostics Agent via trace listeners, system event logs, or performance counter samples in Web application code. The collected data is then stored long-term in Windows Azure Storage via scheduled transfers performed by the Diagnostics Agent.

The Diagnostics Agent does not currently support encryption-at-rest in Windows Azure Storage, so developers should not write unencrypted sensitive information to any events sent to the Diagnostics Agent. Refer to “[Encrypting Data within Windows Azure](#)” for more information about protecting sensitive data.

Windows Azure Network Layer Security Considerations

Network security is one of the pivotal building blocks in a secure application. It is also a topic which is highly differentiated in the cloud from traditional on premises networks. This section provides an overview and fundamental information about network security within Windows Azure. Detailed network security guidance is provided in a companion to this paper, [Windows Azure Security Overview](#).

Windows Azure Networking Fundamentals

There are two types of IP addresses assigned to Windows Azure deployments. Virtual IP Addresses (VIPs) are the external, Internet-routable addresses which can be used to contact a cloud service. Dynamic IP Addresses (DIPs) are the internal, non-routable addresses used inside the datacenter. A special device known as the Software Load Balancer (SLB) performs IP address translation between VIPs and DIPs. It can also perform port translation and acts as a load balancer when configured to do so.

Enforcement of all IP policies is performed by a network filter running in the Host OS. The objective of these policies is to protect our infrastructure from potentially malicious customers, and isolate one customer from another. This means that nothing done on the VM can override Windows Azure’s configuration to make the firewall less restrictive. It is possible, however, to enable additional, more restrictive rules in the VM using [Windows Firewall](#) or a Linux equivalent.

Networking in PaaS Roles

Access from the Internet to Platform as a Service Roles occurs through **input endpoints** which are specified in the service definition file. The entire service will receive a single VIP and incoming connections will be balanced between one of the running Role instances.

Access between Roles internally occurs through **internal endpoints**. Complex rules can be set up in the service definition specifying which Roles can talk to which other Roles.

More information, including how to set up endpoints and traffic rules is available through the guide [Enable Communication for Role Instances in Windows Azure](#).

Networking in IaaS Virtual Machines

Just like Roles, Virtual Machines receive traffic from the Internet through **input endpoints**. An input endpoint tells the SLB to route communications from the external VIP to the internal DIP. There can be a 1:1 relationship or load balancing can be set up where one VIP can be routed to multiple DIPs. Again, this process can include port translation.

Because Windows Azure Virtual Machines don't have a detailed "map" like Roles have a service definition, there is no way to define internal endpoints. Instead, the host machine's packet filter checks incoming connections and makes sure they come from within the same subscription. Thereby traffic originating from other customers is blocked.

The exception to this occurs when an input endpoint is set up. That allows inbound traffic to your Virtual Machine on the specified internal port. Normally, this traffic originates from the Internet by way of one of the SLBs. It is possible, however, that a VM within the same network segment (cluster) can send traffic directly to your VM's DIP. This is because the host packet filter sees all traffic originating from outside your subscription as equivalent, whether it comes from a load balancer or from another environment.

The pivotal factor is that this can only occur if an input endpoint has been set up, otherwise it is blocked. This is normally not a concern due to the fact that this can only occur when a port is already open to the hostile Internet. It is a problem, however, if your application is written to "trust" all IPs on your un-routable DIP subnet.

This behavior can be prevented by attaching your Virtual Machines to a Windows Azure Virtual Network. All VMs attached to the virtual network can only talk to other VMs attached to the same virtual network. Input endpoints can be set up to allow inbound communication from the Internet; however, in this case, only inbound traffic from the Internet is allowed. The ancillary benefit of Windows Azure Virtual Networks is that you can create a secure tunnel back to your on-premises network.

Spoofing

The hypervisor performs checks on outgoing network traffic which prevent spoofing attacks. A compute node is disallowed from sending traffic from any other IP than its own. Additional hypervisor VMSwitch

filters are in place to block broadcast and multicast traffic, with the exception of Address Resolution Protocol (ARP) and what is needed to maintain Dynamic Host Configuration Protocol (DHCP) leases.

Windows Azure Virtual Networks

Windows Azure virtual networks provide three primary capabilities:

1. The ability to create a virtual private network within Windows Azure

Virtual network provides the customer with a network realized on shared infrastructure that behaves as if dedicated to the customer. The first step is connectivity virtualization, wherein the customer can extend into Windows Azure its private address space (typically RFC 1918 – 10.x.x.x space), without overlap with other customers or with Windows Azure itself.

2. Connect into Windows Azure via VPN using a Site-to-Site (On-Premises-to-Azure) IPsec VPN tunnel

It is possible to securely span an enterprise network into Windows Azure, leveraging the same address space, and cryptographically secure transit across the Internet.

3. Connect into Windows Azure via VPN using a Point-to-Site (PC-to-Azure) VPN tunnel.

It is possible to connect into a Windows Azure deployment from a remote PC over VPN. This allows things such as administering Virtual Machines without opening an external endpoint for RDP. Today, routing rules prevent connection “back out” to the internet through via a P2S network.

Additional information on virtual networks is available through the [Windows Azure Virtual Network Overview](#).

Secure Usage of DNS

To allow Virtual Machines within a cloud service to be addressed by name, Windows Azure provides an internal DNS service. This allows Virtual Machine names to be resolved to DIPs within a cloud service while maintaining privacy across cloud services, even within the same subscription.

The DIPs assigned to both PaaS Roles and IaaS Virtual Machines can change during repair. Due to this communication between Roles within a Windows Azure hosted service must be resolved via DNS name and not by IP address. The one exception to this rule is when virtual networks are being used for custom IP spaces. In those cases, IP addresses can be assumed to be static. Also, as DIPs can change, the DNS TTL (time-to-live) values of the DNS responses should be honored in the client.

Windows Azure Virtual Machines always have a hostname, this is used to address the Virtual Machine through DNS. To give Cloud Service Role instances a name, use the [vmName parameter in the service definition](#).

More information about Windows Azure DNS and choosing the right DNS architecture can be found in the [Windows Azure Name Resolution](#) overview.

Eavesdropping / Packet Sniffing

Virtual network interfaces within Windows Azure are disallowed from entering promiscuous mode. This prevents them from receiving traffic destined for any other Virtual Machine. In addition, protections at lower levels prevent packets from ending up at the wrong VM.

Web Application Firewall

Customers wishing for greater protection against threats to Web applications can now add Barracuda Network's Web Application Firewall to their Windows Azure deployments. More information is available on [Barracuda's Azure program overview](#). Existing Barracuda customers can deploy a virtualized Web Application Firewall Vx using Barracuda's [step-by-step instructions](#).

As of the publication date of this document, Barracuda Web Application Firewall Vx is the only tested and supported solution in this product space.

Windows Azure Platform and Infrastructure Layer Protections

This section outlines security and availability threats that are mitigated on developers' behalf by the Windows Azure platform and underlying network infrastructure.

Multi-tenant hosting

Windows Azure is a multi-tenant system where many different groups may share the same piece of physical hardware. Microsoft has gone to great lengths to provide security isolation for tenants. There are many defense-in-depth security measures employed throughout the system that help enforce this.

Windows Azure Cloud Services and Windows Azure Virtual Machines rely on a virtualized Host/Guest boundary to enforce security separation. A Windows Azure Web site can share a virtual machine with other Windows Azure Web Sites tenants. Isolation in that case is provided within IIS.

Side-Channel Attacks

When Virtual Machines belonging to different customers are running on the same server, it is the Hypervisor's job to assure that they cannot learn anything important about what the other customer's Virtual Machines are doing. While it is straightforward to block unauthorized direct communications (e.g., the Virtual Machines do not share read/write memory or disk sectors and communications over the network are only allowed if the same communications would be allowed over the Internet), there are more subtle effects whereby a customer can characterize the work being done by another customer.

The most important of these are timing effects when different Virtual Machines are competing for the same resources. Because of memory bus contention, CPUs will in general be able to execute fewer instructions per second if other CPUs on the same chip or sharing the same memory are running than if they are idle. By carefully comparing operations counts on CPUs with elapsed time, a Virtual Machine can therefore, learn something about what other Virtual Machines on the same server are doing. These are referred to as side-channel attacks. Windows Azure does not attempt to prevent such gross level inferences. Similarly, the timing of accesses to the disk and the network may reveal some information about the disk and network usage of other Virtual Machines on the same server.

There have been a number of academic efforts to learn much more specific information about what is going on in a peer Virtual Machine. Of particular interest are efforts to learn the cryptographic keys of peer Virtual Machines by measuring the timing of certain memory accesses and inferring which cache lines the victim Virtual Machine is reading and updating. Cryptographic implementations that use large tables and access different parts of those tables based on the key being used for an operation are particularly susceptible to such attacks. Under controlled conditions with Virtual Machines using hyper-threading, successful attacks have been demonstrated against commercially available implementations of cryptographic algorithms.

There are a number of mitigations in Windows Azure that make it unlikely that such an attack would be successful:

- The standard cryptographic libraries in Microsoft Windows have been designed to resist such attacks by not having cache access patterns depend on the cryptographic keys being used.
- All Windows Azure servers have at least eight (8) physical cores and some have substantially more. Increasing the number of cores that share the load placed by the various Virtual Machines adds noise to an already weak signal.

Verification and Certification

Microsoft conducts regular penetration testing on targeted infrastructure components. These tests are carried out by both Microsoft's experienced internal security teams as well as a number of external vendors. Security is of the utmost priority during the development of Azure's infrastructure components. Windows Azure employs a rigorous Common Engineering Criteria which goes above and beyond the bar for most online services.

Windows Azure has obtained a number of external certifications, including ISO 27001, SSAE 16, PCI DSS, FedRAMP, and others. More information regarding certifications and security practices is available from the [Windows Azure Trust Center](#).

Conducting Penetration Tests

Microsoft recognizes the value of conducting penetration tests to assess the security of your cloud service deployments. Customers of Windows Azure are permitted and encouraged to simulate attack scenarios in their efforts to build more secure applications. But special care must be taken during a penetration test of a cloud provider such as Windows Azure. Microsoft continuously monitors all

Windows Azure systems for hostile or abusive activities to make the platform a safer environment for all customers.

Customers are allowed to test their own applications in order to determine the strength of their developers' hardening tactics. However, attempts to attack the Windows Azure infrastructure or other tenants is a direct violation of the [Terms of Service](#). For example, attacks against Windows Azure Storage, Service Bus or the Windows Azure Service Management API are disallowed; you should direct the test only at your own applications.

In order to prevent having a legitimate test inadvertently flagged as an attack, Windows Azure asks that customers fill out a [Penetration Testing Approval Form](#) and submit it to Windows Azure Customer Support a minimum of seven (7) days before the test commences. This alerts the Azure Security Team to your planned activities and can help prevent unintended disruptions of your service. It is also important to note that aggressive use of bandwidth can result in increased service charges.

Low-aggression scans such as ones aiming to determine the patch level and OS configuration of Virtual Machines do not require penetration testing pre-approval. However, more aggressive network-level tests such as NMAP-style scans do. It is especially important to target those scans only at your own Virtual Machines, as any attempt to port-scan the Windows Azure infrastructure or another tenant is a violation of the Windows Azure [Terms of Service](#).

Designing Effective Cloud Penetration Tests

Penetration tests can be expensive undertakings, but a properly scoped penetration test results in the highest ROI while preventing the pitfalls mentioned above.

A test of a PaaS application should concentrate on the application layer. The runtime, OS and everything that supports them are common structures provided by the Windows Azure platform. That isn't to say that your PaaS Role's OS is monitored by Microsoft—that is your responsibility. The infrastructure provided by Windows Azure PaaS is designed to minimize attack surface while still providing capabilities such as automatic OS updates (if enabled). Windows Azure Web Sites should be treated in a similar fashion where effort should be concentrated at the application layer.

Assessments against PaaS Applications

An effective security assessment of a PaaS application should evaluate the following:

- Coding practices
- Application design
- Management of connection strings and storage keys
- Controls of privileged access management

In general, there is little to no value in executing OS or network-level tests against PaaS applications. Aspects such as patch level and network configuration are controlled by the infrastructure. It is also difficult and sometimes inherently less secure to deploy these types of scanners in a PaaS application. A better solution is to perform an audit of the application's configuration, ensuring that automatic updates are enabled, all role-to-role

communication is properly secured, and that no unnecessary endpoints have been opened (such as RDP).

Assessments against IaaS Applications

A thorough security assessment of an IaaS-based applications is broader and should include the Application *and* OS layers. Because the entirety of the contents of an IaaS VM are under the tenant's control, everything running within should be equally tested. In many ways, an IaaS penetration test should mimic a traditional on-premises test. The primary difference is that certain aspects of network configuration and the physical environment are off limits. Examples of common issues to test in IaaS applications are:

- Missing OS patches
- OS configuration
- Issues caused by a gold image being replicated many times
- Shared or weak machine credentials
- Unnecessary attack surface
- All of the issues mentioned above for PaaS applications

Assessments against Hybrid Applications

Hybrid PaaS, IaaS and/or on-premises deployments require a more complicated scope. On-premises components should be tested as thoroughly as possible while IaaS and PaaS components should follow the guidance outlined above.

Securing Access to Roles and Virtual Machines

Access control to a Windows Azure application can be broken into three distinct levels.

0. Access to the Windows Azure Portal and via Remote Desktop
1. Access to privileged application-level configuration
2. Access for general users

In the on-premises world, level 0 is roughly analogous to physical access to the datacenter and interactive console access to machines. Within Windows Azure, access to the Management Portal provides a very similar level of control. Day to day administration of a cloud-based application should be treated separately from administration of the Windows Azure subscription.

Securing Access to the Windows Azure Management Portal

The most highly privileged account within Windows Azure is the Account Owner. This account has the ability to create one or more subscriptions and assign or modify the subscription owner. It also has the ability to add co-administrators to the portal. In addition, it grants access to storage keys. These keys provide full read/write access into Windows Azure Storage.

Due to this, the Account Owner should be specially protected. Ideally, the Account Owner should be a service account – that is, an account not associated to a real human who could leave the company or change roles. The credentials for that account need to be protected and changed regularly.

Subscriptions should be used to separate duties whenever possible. For example, a business may want to create a finance subscription for finance tasks and a Web presence subscription for a customer facing Web site.

If possible, corporate identities should be federated with a Microsoft Account. This allows Azure account access to synchronize with internal account status. This eases the problems which occur when an authorized user leaves the company. Their account only needs to be deleted in one place.

For more information, read [Manage Subscriptions and Storage Accounts in the Windows Azure Management Portal](#) on MSDN.

Privileged Application-Level Configuration

Administrators whose duty is to update Web page content through a Content Management System (CMS) generally shouldn't be given access to the physical servers. In the same way, Windows Azure customers may wish to implement their own privileged application-level configuration. Access control to this system should be separate from access to the Management Portal. The [Identity Management and Access Control](#) section provides methods which could be used to implement such features in the cloud.

IaaS Virtual Machine Access Security

It is critical to secure access to IaaS Virtual Machines. Historically, the majority of security incidents involving customers if Windows Azure can be attributed to RDP or SSH credential brute forcing. It is pivotal to secure remote access to Virtual Machines in Windows Azure.

Remote Desktop

The Microsoft Windows Remote Desktop Protocol (RDP) is secured against spoofing and eavesdropping via digital certificates. The RDP certificate's thumbprint is displayed within the Windows Azure Portal in order to enable out of band verification. When a connection is established from Remote Desktop, the details of the certificate can be examined. By comparing these two thumbprints, you can verify that the communication channel is with the correct endpoint.

Another solution which doesn't require out of band verification is to swap out the Azure-generated, self-signed certificate with one signed by your own trusted Certificate Authority (CA). Deployment of new certificates can happen via the Windows Azure Developer Portal, via PowerShell cmdlets, or through the standard Windows Certificate Store (only available in Windows Azure Virtual Machines). By using PowerShell, deployments can be scripted to create instances and set up their certificates all at once.

The single greatest threat to RDP is that of password guessing. Using strong passwords is of the utmost importance. Long, high entropy passwords are a good defense against brute force attacks. Account names should also be changed to something [not easily guessed](#). Passwords should be chosen which are stronger than the default Windows policy. Generic variants of common words such as replacing vowels with numbers or punctuation do not add much security and are often already available in exploit tool dictionaries. A good rule of thumb is to use a pass phrase which consists of multiple words, upper and

lower case, numbers, and punctuation. There are also many applications which will generate high entropy, random passwords. These passwords provide great security but are difficult to remember.

By default, all Windows Virtual Machines are configured with an external endpoint for RDP. This means that RDP is accessible from the Internet. The endpoint for every Virtual Machine is configured with a random port. That makes it slightly more difficult to an attacker to find. If a Windows Azure Virtual Network has been configured with a site-to-site connection, the RDP endpoint can be deleted by the Account Owner to prevent access to it via the Internet. This forces all administration to occur through the tunneled network.

More information about securing Remote Desktop (IAAS only) is provided through [this guide](#).

Remote PowerShell

Just like RDP, Remote PowerShell is secured using digital certificates. By default, Azure generates a self-signed certificates for administration. As soon as possible, this certificate should be swapped out for a CA signed certificate. Ideally, this certificate would be deployed via PowerShell cmdlets when the Virtual Machine is created.

SSH

Linux clients use SSH as the default method for administration. Like RDP, the SSH thumbprint is displayed in the Windows Azure Developer Portal. Upon the first connection to a Virtual Machine, SSH clients display this thumbprint. Comparing these thumbprints and ensuring they are identical helps you verify that the communication channel is secure. SSH clients will usually cache the thumbprint and won't prompt again in the future unless the thumbprint changes.

By default, Windows Azure configures SSH on the default port, TCP22. By default, this port is open to the Internet. Additionally, the Linux machines created through the Quick Create process always have the user `azureuser` configured. Because this is a widely known account name, it is a best practice to disable or rename that account. It is also advisable to reconfigure the Virtual Machine's [endpoint](#) so that SSH is running on a nonstandard port. If a Windows Azure Virtual Network has been configured with a site-to-site connection, the SSH endpoint can be deleted to prevent access to it via the Internet. This forces all administration to occur through the tunneled network.

For an extra level of convenience and potentially security, many customers choose to forgo SSH password authentication in favor of public key authentication. When this method is chosen, a primary concern is protecting the associated private keys. Much like Remote PowerShell, SSH keys can be pushed to Virtual Machines using [PowerShell cmdlets](#). This provides a good way to combine the Virtual Machine creation process with the key management process.

More specific information about hardening SSH and other Linux hardening tasks is available from the distribution's publisher. Windows Azure does not provide support or guidance for Linux distributions.

PaaS Remote Access Security

Unlike IaaS, RDP isn't generally required for the day-to-day operation of a PaaS service. Access to PaaS RDP is secured with certificates; due to this, password guessing attacks are ineffective. Nevertheless, RDP can be an unnecessary attack surface and should be enabled only when absolutely necessary. It is important to note that RDP can only [enabled or disabled with an application publish operation](#) (as all roles need to be rebuilt with the correct configuration).

Secure Cloud Application Design

An application employing the most advanced security measures can still be undone by simple design errors. A security feature doesn't need to be compromised if it can be avoided. This is just as true for applications in the cloud as it is for conventional deployments.

Threat Modeling

Microsoft's [Security Development Lifecycle](#) specifies that teams should engage in a process called Threat Modeling during the [design phase](#). In order to help facilitate this process Microsoft has created the [SDL Threat Modeling Tool](#). Modeling the application design and enumerating [STRIDE](#) threats across all trust boundaries has proven an effective way to catch design errors early on.

The following table lists the STRIDE threats and gives some example mitigations using features provided by Windows Azure. These mitigations won't work in every situation.

Threat	Security Property	Potential Azure Platform Mitigation
Spoofing	Authentication	Require HTTPS connections
Tampering	Integrity	Validate SSL Certificates
Repudiation	Non-repudiation	Enable Windows Azure Monitoring and Diagnostics
Information Disclosure	Confidentiality	Encrypt sensitive data at rest using Service Certificates
Denial of Service	Availability	Monitor performance metrics for potential denial of service conditions. Implement connection filters.
Elevation of Privilege	Authorization	Windows Azure Active Directory Access Control Services (AAD-ACS)

Using Claims-Based Authentication

In Microsoft's ongoing efforts to build the world's most open Identity Management service, Microsoft has introduced the new OAuth 2.0 grant type authorization flow. This builds on top of already strong support for SAML, WS-Federation and the client credentials grant type in OAuth 2.0 for server to server flows. OAuth 2.0 is the next evolution of the OAuth protocol which was originally created in late 2006.

OAuth 2.0 focuses on simplicity while providing specific authorization flows for Web applications, desktop applications, and mobile phones.

The authorization code grant enables you to drive user authentication flows from native applications; moreover, it offers important features (such as refresh tokens) which can help you to maintain long running sessions while minimizing the need to prompt users for their credentials.

Gatekeeper Keymaster Design Pattern

A Gatekeeper is a design pattern in which access to storage is brokered so as to minimize an application’s attack surface. It separates unprivileged front end roles from privileged backend roles by limiting their interaction to communication over private internal channels and only to other roles. In the event of a successful attack on a front end role, privileged key material is not compromised. The pattern can best be illustrated by the following example which uses two roles:

- The GateKeeper – this is a Windows Azure Cloud Service hosting an ASP.NET Web site that services requests from the Internet. Since these requests are potentially malicious, the Gatekeeper is not trusted with any duties other than validating the input it receives. The GateKeeper is implemented in managed code. The service configuration settings for this role do not contain any Shared Key information for use with Windows Azure Storage.
- The KeyMaster – this is a privileged backend role hosting a WCF Web service that only takes inputs from the Gatekeeper and does so over a secured channel (an internal endpoint, or queue storage – either of which can be secured with HTTPS). The KeyMaster handles storage requests fed to it by the GateKeeper, and assumes that the requests have been sanitized to some degree. The KeyMaster, as the name implies, is configured with Windows Azure Storage account information from the service configuration to enable retrieval of data from blob or table storage. Data can then be relayed back to the requesting client.

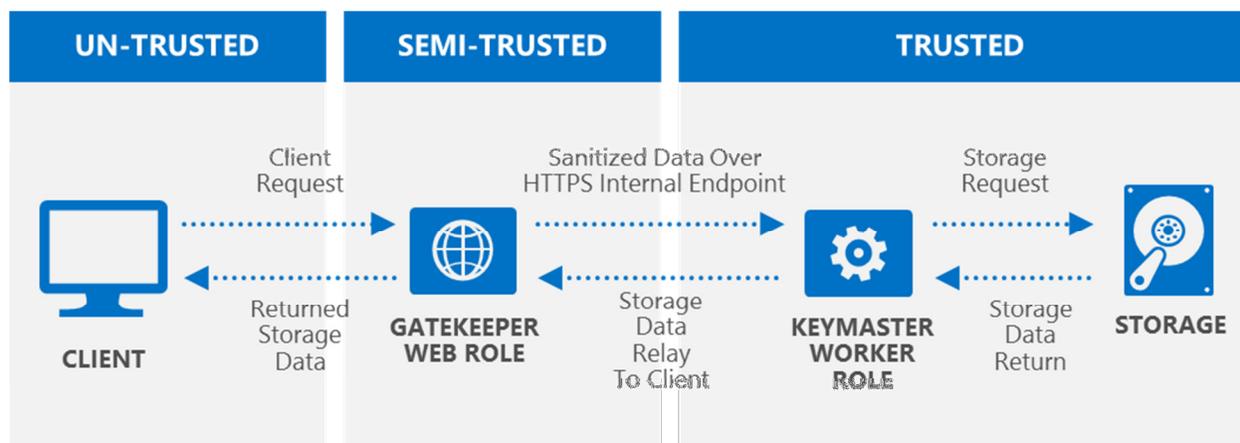


Figure 4. The Gatekeeper Design Pattern

This solution is not perfect, since the KeyMaster is relying on GateKeeper to tell it what content to serve out. However, it does provide for separation of duty and it can pose additional challenges for an attacker. The amount of trust the KeyMaster places in the GateKeeper can be custom-tailored, but it is advisable to minimize the types of access requests allowed from the GateKeeper -- allowing the KeyMaster to accept Storage Blob API read and list operations from GateKeeper, but not add or delete operations, for example.

This pattern can easily be adapted to work with more complex service architectures on Windows Azure. All that is needed is to put a less privileged "Gatekeeper" in front of more privileged (or native code) Roles to do the parsing of potentially malicious data from the network. The concept is analogous to running a front end service in a network Demilitarized Zone (DMZ) which interacts with untrusted payloads before pushing deeper in the network toward more critical components.

Storage as a Relay Design Pattern

In scenarios where a Gatekeeper cannot be placed in front of a more trusted back end Role, a multi-key design pattern can be used to protect trusted storage data. This design is illustrated by Figure 4. The Gatekeeper Design Pattern. An example case of this scenario might be where the majority of the processing has to happen on the front end -- perhaps for an AJAX Web site -- , and placing a Gatekeeper in front of it may be duplicative or degrade performance to an unacceptable level.

Sometimes the processing between front end and back end needs to happen asynchronously. Front end operations need to happen quickly while backend operations take significant processing time. Simple architectures can place a Windows Azure Storage Queue between the one or more front ends and one or more back ends. Queues can easily handle this sort of simple asynchronous transaction. More complicated architectures may wish to make use of [Windows Azure Service Bus](#) to manage inter-role communication.

An unlimited number of Shared Access Signatures can be assigned to one Windows Azure subscription. This diversity can be used to minimize exposure of a particular key to theft by placing lower-trust keys on lower-trust Roles and higher-trust keys on higher-trust Roles. Permissions can also be locked down by using Shared Access Signatures. Each Role should only receive the minimum required permission set.

"Untrusted Web Role A" in the figure below has access to only one storage account, and it is not trusted to do anything but parse incoming requests and log them to storage account 'A'. "Trusted Role B" (in the background) has the account keys for both storage 'A' and 'B', but is not exposed to potentially-malicious, malformed inputs. "Trusted Role B" processes and filters requests from the untrusted Role via storage 'A,' but expects a specific structure or format. 'B' never fully trusts 'A,' allowing for the "real" storage key of value (account 'B') to be protected even in the event of a fully compromised, externally-facing, untrusted Role. As with the Gatekeeper pattern, the trusted Role should not provide unfettered access to the trusted store on behalf of Web Role 'A.' Instead, only a subset of legitimate commands should be serviced (such as read, but not create/update/delete in certain cases).

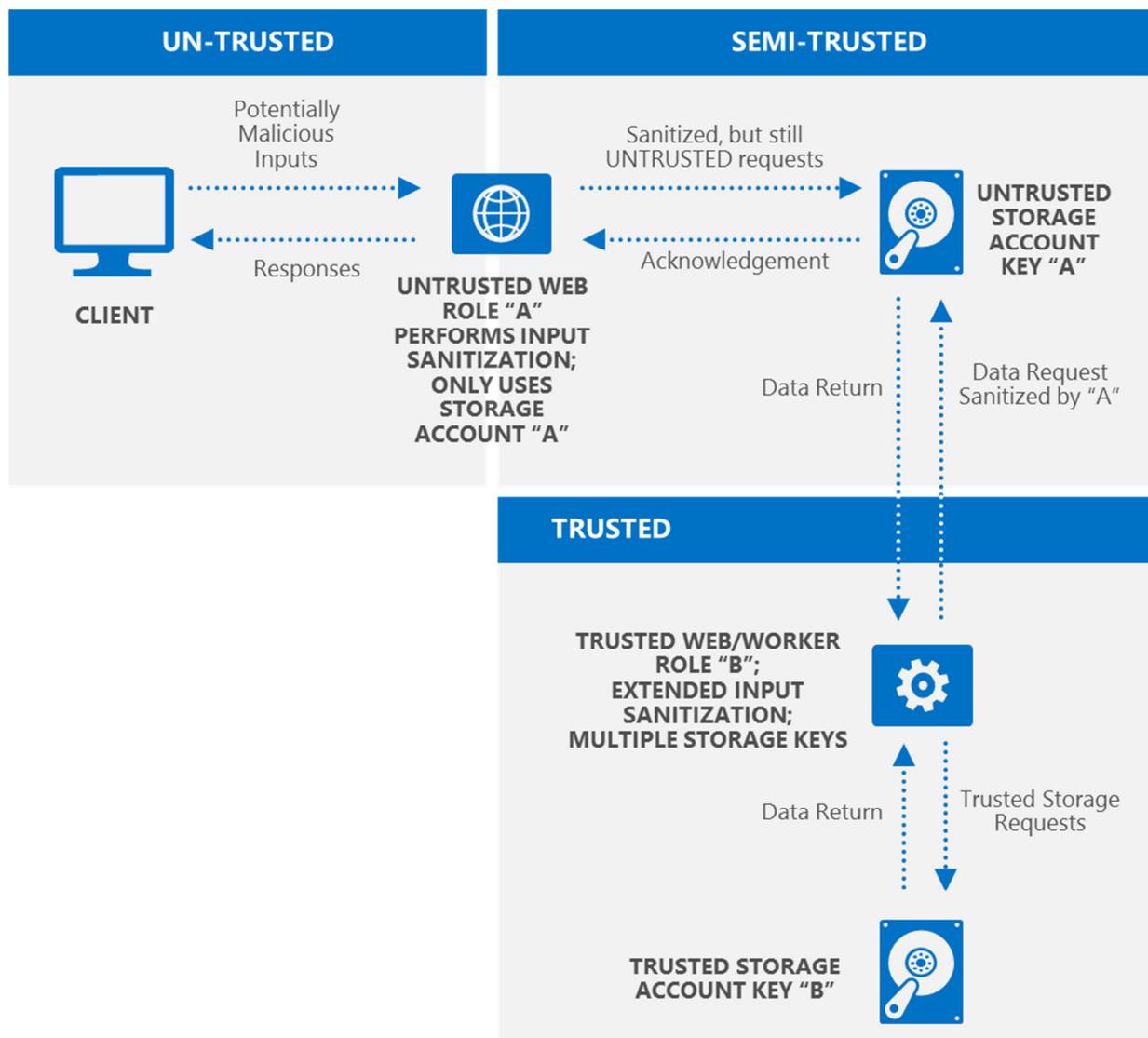


Figure 5. The multi-key design pattern

The multi-key design pattern has some advantages over the Gatekeeper/KeyMaster pattern:

- Providing separation of duty for storage accounts. In the event of Web Role A's compromise; only the untrusted storage account and associated key are lost.
- No internal service endpoints need to be specified. Multiple storage accounts are used instead.
- Shared Access Signatures can be used to lock down the permissions needed by Web Role 'A'.

The key disadvantage is that it is more complex to implement.

Hybrid Cloud/On-Premises Deployments

Through the use of [Windows Azure Virtual Networks](#) and site-to-site connections, it is possible to extend the boundary of an on-premises network into Windows Azure. A hybrid design can allow for the best of both worlds. The following list gives some examples of potential security-related use cases for hybrid deployments.

- Extend an Active Directory infrastructure into a Windows Azure deployment. Use Group Policies to manage Windows Azure IaaS Virtual Machines and Windows Azure PaaS Roles (not applicable to Web Sites).
- Use an on-premises encryption key management system for keys used by Windows Azure applications.
- Remove all remote management (RDP, PowerShell, SSH) external endpoints from Windows Azure and force administrative tasks to occur through the extended network.
- Offload data storage to an on-premises vault to satisfy encryption at rest or data warehousing compliance requirements.
- Couple a cloud-hosted application with on-premises backend functionality such as financial processing.

The scope of everything that is possible with a hybrid deployment is too large to explain in this paper. It is also impossible to categorically explain all potential threats and mitigations. The best tool to creating a secure hybrid deployment is thorough [threat modeling](#). The application should be decomposed into its components and analyzed for the threats to those components.

Applying SDL Practices to Windows Azure Applications

This section describes the security practices that should be considered when designing and building Windows Azure applications.

The Microsoft Security Development Lifecycle ([SDL](#)) applies equally to applications built on the Windows Azure platform and any other platform. Most Windows Azure applications built by Microsoft have been built, or will be built using agile methods. As a result, the SDL for agile process may be more applicable to applications hosted on Windows Azure than to the classic phase-based SDL.

Microsoft requires that the SDL be followed for any Microsoft-developed software deployed on Windows Azure. Microsoft also applies the SDL during development of Windows Azure core infrastructure components. The SDL addresses security threats throughout the development process by means that include:

- Threat modeling during the design process
- Following development best practices and code security standards during coding
- Requiring various tools for testing and verification before deployment

These proactive checks during development make software less vulnerable to potential threats after release, and the SDL provides a structured and consistent methodology with which to apply them. These

methodologies, which are supported by an executive commitment to security, have helped Microsoft develop more secure software. Anyone who develops software in Windows Azure can use these same methods to improve security.

Security Education and Awareness

If a development team does not understand the basics of secure design and development, or the risks of running Web-based software and services, then security training is imperative, and it should be completed before any Windows Azure application is designed, built, tested or deployed. All members of software development teams should be informed about security basics and recent trends in security and privacy, and they should attend at least one relevant security training class every year - at a minimum. Development team members should be encouraged to seek opportunities for additional security and privacy education. Developers who are well-versed and up-to-date on security issues are better able to design and develop software with security in mind first and foremost -- and not as an afterthought or “bolt-on” feature added at the end of the development process.

Given that Windows Azure applications are usually Web-based managed code (ASP.NET) applications, appropriate topics for security education include:

Secure design, including the following topics:

- [Attack surface reduction](#)
- Principle of least privilege
- [Threat modeling](#)

Secure coding, including the following topics:

- [Cross-site scripting](#)
- [SQL injection](#)
- [Managed code security](#)
(transparency, code access security, assembly strong naming, etc.)

The SDL process guidance [documentation](#) provides links to books and training materials that are useful for beginning an SDL training program.

Secure Development Practices on the Windows Azure platform

The SDL provides guidance for use of development tools and practices that are applicable on the Azure platform. For the current version of compilers, linkers and other tools mandated by the SDL, see the [SDL Process Appendix E](#).

- Use the SDL-required (or later) compiler versions to compile for the Win64 target platform:
 - C/C++ code: Microsoft Visual Studio .NET 2008
 - C# or Visual Basic .NET code: Microsoft Visual Studio .NET 2008
- Compile and link native C/C++ code with /GS, /SAFESEH, /DYNAMICBASE and /NXCOMPAT flags. These options are enabled by default in Visual C++ 2008 SP1 and later. You can verify these settings using the [BinScope](#) binary analyzer.

- Starting with Visual Studio 2012, the compiler now includes the [/SDL](#) switch. This automatically enables many compiler warnings and defense mechanisms which previously needed to be enabled individually.
- Native C and C++ code must not use banned versions of buffer handling functions. For more information, see [Security Development Lifecycle \(SDL\) Banned Function Calls](#).
- Use the currently required (or later) versions of code analysis tools for native C and C++ (the [/analyze](#) compiler option).
- Run the FxCop code analysis tool against all managed code and fix all violations of the “Security” rules for the version of [FxCop](#) used.
- Follow data input validation and output encoding requirements to address potential cross-site scripting vulnerabilities.
- If using SQL database storage, only use parameterized queries or LINQ when accessing SQL-based data stores. Never dynamically construct a SQL statement from strings.
- Use an approved XML parser, such as .NET’s System.Xml classes or, for native C++ code, use MSXML6 or XmlLite.
- Make sure your application is compliant with [SDL cryptographic retirements](#).

Verification and Release

Testing tools covered under [SDL Process Guidance](#) should be selected according to the specific technologies a Windows Azure application uses. Many of the tools used for COM/DCOM, RPC, or ActiveX testing, as well as for File Fuzzing, can be used on Windows Azure applications.

Conclusion

Computing solutions that use Windows Azure are compelling to companies wishing to trim capital expenditure. However, security remains an important consideration. Software architects and developers must understand the threats to software developed for the cloud and use appropriate secure design and implementation practices to counter threats in the cloud environment. The progression from classic, client-server computing, to Web-enabled applications, to applications hosted in the cloud, has changed the boundaries of applications, and these boundary shifts make understanding the threats to Windows Azure-based software all the more important. The work required to develop secure Windows Azure applications isn’t new, revolutionary, or technically challenging; it simply requires that designers and developers consider the potential threats to their applications and apply the practices described in this paper.

Appendix A. Glossary

Management Portal	The Windows Azure Management Portal is an administrative portal for managing, deploying and monitoring Windows Azure services. The Developer Portal can be accessed at http://manage.windowsazure.com
Fabric	The logical cluster of machines making up Windows Azure Backend. The Windows Azure Fabric is responsible for managing deployments running on Azure
DIP	Dynamic IP address. This is the internal IP address used by a Windows Azure instance. It is similar to an internal address which hasn't gone through Network Address Translation (NAT)
REST	REpresentational State Transfer; a software design that uses a stateless client-server architecture in which the Web services are viewed as resources and can be identified by their URLs.
SAML	Security Assertion Markup Language. An XML-based industry standard for exchanging authentication and authorization information.
SDL	The Microsoft Security Development Lifecycle (SDL) is a security assurance process that is focused on software development. It is a collection of mandatory security activities, grouped by the phases of a software development life cycle (SDLC). You can learn more about the Microsoft SDL at http://www.microsoft.com/security/sdl/default.aspx
Svchost	Svchost is a process for hosting Windows services. The services are implemented as DLLs.
VHD	Virtual Hard Disk. A file-based representation of a hard disk that can be used as a storage medium by virtual machines. Also may refer to the .vhd file format, which is the format supported by Windows Azure.
VIP	Virtual IP Address. This is the external IP address represented on the external side of Azure's load balancer. This is the public IP address which is used to contact a machine over the Internet.
VM	A software emulation of a computer that runs in an isolated partition of a real computer.
Web Role	A PaaS Role primarily designed to host Web content.
Worker Role	A PaaS Role primarily designed to host a Web service endpoint

Appendix B: Using SDL-approved Cryptography in Windows Azure Applications

Many applications require the use of cryptographic technologies for encryption, tampering detection, signatures etc., and the SDL is prescriptive about which cryptographic technologies should be used. The following is a list of some of the more common requirements you will encounter when building a Windows Azure application.

- Where possible, use SSL/TLS (by using HTTPS) to transfer data between all parties, such as:
 - Clients <-> Web and Worker Roles
 - Clients <-> Storage endpoints
 - Web Roles <-> Web and Worker Roles over internal endpoints
 - Web and Worker Roles <-> Storage endpoints
 - IaaS Virtual Machines <-> Web or Worker Roles
 - IaaS Virtual Machines <-> Storage endpoints
- Endpoints supporting HTTPS should be provisioned with certificates supporting 2048-bit RSA keys.
- Certificates should be renewed annually by refreshing the service configuration with the thumbprint of any new certificate uploaded via the developer portal.
- Sensitive data stored in Windows Azure Storage should be stored encrypted with a symmetric key, and this symmetric key then encrypted with the provisioned certificate.
- Use AES for symmetric cryptographic operations.
- Use 256-bit symmetric keys. This is especially important when a role needs to store encrypted data in Windows Azure Storage.
- Use RSA or Elliptic Curve Cryptography (ECC) for asymmetric cryptographic operations.
- Use RSA keys that are 2048-bit or longer.
- Use a SHA-2 algorithm (SHA-256, SHA-384 or SHA-512) for hashing and message-authentication codes.
- Use the strong entropy provisioned into the virtual machine if keys are derived in role code. Windows Azure virtual machines are specially provisioned with strong entropy at boot.
- For more guidance on use of cryptography, see <http://msdn.microsoft.com/en-us/security/sdl-process-guidance.aspx>
- Store all certificates through the [certificate management feature](#) not in Windows Azure Storage or on local disk.

Appendix C: Security Resources for Azure Developers

Security Resources for Windows Azure IaaS Virtual Machines

- Windows Server Update Services (WSUS): <http://technet.microsoft.com/en-us/windowsserver/bb332157.aspx>
- Windows Event Collector: [http://msdn.microsoft.com/en-us/library/windows/desktop/bb427443\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/bb427443(v=vs.85).aspx)
- How to Load Balance Virtual Machines: <http://www.windowsazure.com/en-us/manage/windows/common-tasks/how-to-load-balance-virtual-machines>
- Manage the Availability of Virtual Machines: <http://www.windowsazure.com/en-us/manage/windows/common-tasks/manage-vm-availability/>
- Deploying Certificates with Windows Azure Virtual Machines and PowerShell: <http://michaelwasham.com/2012/08/23/deploying-certificates-with-windows-azure-virtual-machines-and-powershell/>
- How to Set Up Communication with a Virtual Machine: <http://www.windowsazure.com/en-us/manage/windows/how-to-guides/setup-endpoints/>
- Windows Firewall with Advanced Security Overview: <http://technet.microsoft.com/en-us/library/hh831365.aspx>
- Configure Security Settings for Remote Desktop Services Connections: <http://technet.microsoft.com/en-us/library/cc753488.aspx>

Security Resources for Windows Azure Websites

- How to Scale Web Sites: <http://www.windowsazure.com/en-us/documentation/articles/web-sites-scale/>
- How to Monitor Web Sites: <http://www.windowsazure.com/en-us/manage/services/web-sites/how-to-monitor-websites/>
- Web App Security with the Microsoft Simplified SDL: <http://technet.microsoft.com/en-us/security/gg622918.aspx>

Security Resources for Windows Azure PaaS Roles

- Managing the Windows Azure Guest Operating System -- <http://msdn.microsoft.com/en-us/library/windowsazure/ff729422.aspx>
- Running Startup Tasks in Windows Azure: <http://msdn.microsoft.com/en-us/library/windowsazure/hh180155.aspx>
- Manage Certificates: <http://msdn.microsoft.com/en-us/library/windowsazure/gg981929.aspx>
- How to Monitor Cloud Services: <http://www.windowsazure.com/en-us/manage/services/cloud-services/how-to-monitor-a-cloud-service/>
- Domain Joining Windows Azure Roles: http://blogs.msdn.com/b/windows_azure_connect_team_blog/archive/2010/12/10/domain-joining-windows-azure-roles.aspx
- Enable Communication for Role Instances in Windows Azure: <http://msdn.microsoft.com/en-us/library/windowsazure/hh180158.aspx>

- Enable Communication for Role Instances in Windows Azure: <http://msdn.microsoft.com/en-us/library/windowsazure/hh180158.aspx>
- Using Remote Desktop with Windows Azure Roles: <http://msdn.microsoft.com/en-us/library/windowsazure/gg443832.aspx>

Security Resources for Identity and Access Management

- Using Windows Azure AD for Authorization: <http://msdn.microsoft.com/en-US/library/windowsazure/dn385718.aspx>
- Windows Azure AD Authentication Library for .NET: <http://msdn.microsoft.com/en-us/library/windowsazure/jj573266.aspx>
- Windows Azure Identity: <http://www.windowsazure.com/en-us/documentation/articles/fundamentals-identity/>

Security Resources for Windows Azure Storage

- Creating and Use a Shared Access Signature: <http://msdn.microsoft.com/en-us/library/windowsazure/jj721951.aspx>

Resources for Data Encryption

- Certificate Management: <http://technet.microsoft.com/en-us/library/cc962026.aspx>
- Certificate Stores: [http://technet.microsoft.com/en-us/library/cc757138\(v=WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc757138(v=WS.10).aspx)
- System.Security.Cryptography.Pkcs Documentation: [http://msdn.microsoft.com/en-us/library/6see7k14\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/6see7k14(v=vs.85).aspx)
- Manage Certificates: <http://msdn.microsoft.com/en-us/library/windowsazure/gg981929.aspx>
- Installing Certificates in Windows Azure VMs: <http://blogs.msdn.com/b/jnak/archive/2010/01/29/installing-certificates-in-windows-azure-vm.aspx>
- Crypto Services and Data Security in Windows Azure: <http://msdn.microsoft.com/en-us/magazine/ee291586.aspx>

Windows Azure Platform and Infrastructure Resources

- Windows Azure Security Blog: <http://blogs.msdn.com/b/azuresecurity/>
- Windows Azure Identity and Access Scenarios: <http://www.windowsazure.com/en-us/manage/windows/best-practices/security/#header-5>
- Windows Azure Deployments and the Virtual IP: <http://blogs.msdn.com/b/windowsazure/archive/2011/07/06/windows-azure-deployments-and-the-virtual-ip-address.aspx>
- Windows Azure Name Resolution: <http://msdn.microsoft.com/en-us/library/windowsazure/jj156088.aspx>
- Microsoft AntiXSS Library: <http://www.microsoft.com/en-us/download/details.aspx?id=28589>
- Geo-Replication and new Storage features for Windows Azure: <http://blogs.msdn.com/b/sudhir/archive/2011/09/24/geo-replication-and-new-storage-features-for-windows-azure.aspx>
- Introducing Locally Redundant Storage for Windows Azure Storage: <http://blogs.msdn.com/b/windowsazurestorage/archive/2012/06/08/introducing-locally-redundant-storage-for-windows-azure-storage.aspx>

- Windows Azure Service Bus: <http://msdn.microsoft.com/en-us/library/ee732537.aspx>
- Manage Deployments in Windows Azure: <http://msdn.microsoft.com/en-us/library/windowsazure/gg433027.aspx>
- Manage Subscriptions and Storage Accounts in the Windows Azure Management Portal: <http://msdn.microsoft.com/en-us/library/windowsazure/hh531793.aspx>
- Windows Azure Virtual Network Overview: <http://msdn.microsoft.com/en-us/library/windowsazure/jj156007.aspx>
- Windows Azure Trust Center: <http://www.windowsazure.com/en-us/support/trust-center/>
- Windows Azure SQL Database Connection Security: <http://social.technet.microsoft.com/wiki/contents/articles/2951.windows-azure-sql-database-connection-security.aspx>
- Windows Azure Network Security Whitepaper: <http://download.microsoft.com/download/4/3/9/43902EC9-410E-4875-8800-0788BE146A3D/Windows%20Azure%20Network%20Security%20Whitepaper%20-%20FINAL.docx>
- Penetration Test Approval Form: <http://download.microsoft.com/download/C/A/1/CA1E438E-CE2F-4659-B1C9-CB14917136B3/Penetration%20Test%20Questionnaire.docx>
- Create Strong Passwords: <http://www.microsoft.com/security/online-privacy/passwords-create.aspx>

Security Resources for Windows Azure SQL Database and Microsoft SQL Server

- SQL Injection: [http://technet.microsoft.com/en-us/library/ms161953\(v=SQL.105\).aspx](http://technet.microsoft.com/en-us/library/ms161953(v=SQL.105).aspx)
- Security Considerations for SQL Server in Windows Azure Virtual Machines: <http://msdn.microsoft.com/en-us/library/windowsazure/dn133147.aspx>
- Managing Databases and Logins with Windows Azure SQL Database: <http://msdn.microsoft.com/en-us/library/windowsazure/ee336235.aspx>
- Overview of Security in Windows Azure SQL Database: <http://social.technet.microsoft.com/wiki/contents/articles/1573.overview-of-security-in-windows-azure-sql-database.aspx>
- Introducing Locally Redundant Storage for SQL Azure: <http://blogs.msdn.com/b/windowsazurestorage/archive/2012/06/08/introducing-locally-redundant-storage-for-windows-azure-storage.aspx>
- Provisioning a SQL Server Virtual Machine on Windows Azure: <http://www.windowsazure.com/en-us/documentation/articles/virtual-machines-provision-sql-server/>
- High Availability and Disaster Recovery for SQL Server in Windows Azure Virtual Machines: <http://msdn.microsoft.com/en-us/library/windowsazure/jj870962.aspx>
- Connectivity Considerations for SQL Server in Windows Azure Virtual Machines: <http://msdn.microsoft.com/library/dn133152.aspx>

Microsoft Security Development Lifecycle

- What is the Security Development Lifecycle: <http://www.microsoft.com/security/sdl/default.aspx>

- SDL Threat Modeling Tool 3.1.8: <http://www.microsoft.com/en-us/download/details.aspx?id=2955>
- Uncover Security Design Flaws Using the STRIDE Approach: <http://msdn.microsoft.com/en-us/magazine/cc163519.aspx>
- Required and Recommended Compilers, Tools, and Options for All Platforms: <http://msdn.microsoft.com/en-us/library/cc307395.aspx>
- Warnings, /sdl, and improving uninitialized variable detection: <http://blogs.msdn.com/b/sdl/archive/2012/06/06/warnings-sdl-and-improving-uninitialized-variable-detection.aspx>