

On Your Computer

In this lab, you will manage firewall rules with PowerShell.

**[Controller] Display Firewall Information**

Please switch into the C:\SANS\Day4\Firewall directory:

*Cd C:\SANS\Day4\Firewall*

**Note:** Since your VM is the only domain controller, sometimes the connection profile for your network interface gets stuck during boot up at “Public”, but disabling and enabling that interface will change it back to “Domain Authenticated” as expected.

Display the connection profile (Public, Private, Domain) for each live interface: *Get-NetConnectionProfile*

**Note:** If a setting is “NotConfigured”, it is not disabled; it is using the factory default, such as allow outbound.

Display the state of the firewall for each interface profile type: *Get-NetFirewallProfile*

**Note:** Tab completion in ISE is your friend!

Display the names only of the inbound, enabled, public, allowing firewall rules: *Get-NetFirewallRule -Enabled true -Direction inbound -Action allow | Where { $\_.profile -match ‘any|public’ } | Select-Object DisplayName*

**Configure Windows Firewall Rules**

Export your firewall and IPsec rules to a backup file: ***netsh.exe advfirewall export c:\temp\current.wfw***

Reset the firewall and IPsec rules back to their factory defaults: ***netsh.exe advfirewall reset***

Enable the firewall for public interfaces, block inbound connections by default, enable logging of blocked packets, and set the log size to 4096KB: *Set-NetFireWallProfile -Name Public -Enabled True -DefaultInboundAction Block -LogBlocked True -LogMxSizeKilobytes 4096*

Enable all firewall rules for the “File and Printer Sharing” group (in en-US culture): ***Enable-NetFirewallRule -Displaygroup “File and Printer Sharing”***

Open the Enable-FirewallRulesByGroup.ps1 script and examine it: *ise .\Enable-FirewallRulesByGroup.ps1*

Hold down the Ctrl key on your keyboard, highlight the “Remote Desktop” group and a couple other groups at random, click the OK button at the bottom, and the highlighted groups of rules will then be enabled in the firewall. This script and Out-GridView work on Server Core too.

**Note**: A “security filter” for a firewall rule is just all the details of that firewall rule collected into a single object to make these details easier to manage.

Configure the “Remote Desktop” group of firewall rules to require IPsec:

***Get-NetFireWallRule -Displaygroup “Remote Desktop” -Direction inbound | Get-NetFirewallSecurityFilter | Set-NetFirewallSecurityFilter -Authentication Required -Encryption Dynamic***

Using your mouse, drag out and highlight the above command you just typed in (inside the shell), right-click that highlighting, and select Copy. The text is now in your clipboard. You have the Enable-FirewallRulesByGroup.ps1 script open in the ISE editor. Modify the script so that it 1) enables firewall rules and 2) configures those rules to require IPsec. Save your changes and test the script by selecting other groups of rules. Confirm that the script is working by going to the graphical Windows Firewall snap-in, right-clicking Inbound Rules, and selecting Refresh.

**Hint:** ise .\Hints\Enable-FirewallRulesByGroupForIPsec.ps1

Create an inbound firewall rule to block access to TCP port 3666: ***New-NetFirewallRule -DisplayName “Drop APT Back Door” -Direction inbound -LocalPort 3666 -Protocol tcp -Action Block***

This new rule can now be seen in the graphical Windows Firewall snap-in (you may have to right-click Inbound Rules and select Refresh).

Delete a firewall rule by its display name: *Remove-NetFirewallRule -DisplayName “Drop APT Back Door”*

**Create Firewall Rules to Block IP Address Ranges**

View an example file of IP address ranges whose traffic we wish to block: *ise .\Country-Blocklist.txt*

View the help for a script that can create blocking rules from an input file: ***Get-Help -Full .\Import-FirewallBlocklist.ps1***

In the graphical Windows Firewall tool, refresh the lists of inbound and outbound rules. You will see new rules have been created with names like “Country-Blocklist-#XXX”. In the properties of any of these rules on the Scope tab, you will see the IP address ranges blocked by that rule.

Delete the firewall rules created by the script, but leave all other rules alone: ***.\Import-FirewallBlocklist.ps1 -InputFile .\Country-Blocklist.txt -DeleteOnly***

There are also free and commercial lists of country IP network blocks available. You can also obtain IP addresses for other unwanted hosts, such as ransomware command and control servers, from threat intelligence companies, usually for a fee. For examples of country IP allocations, see the following:

* <http://www.ipdeny.com/ipblocks/>
* <http://www.okean.com/thegoods.html>
* <http://www.countryipblocks.net/>
* <http://www.ipblocklist.com>
* <http://lite.ip2location.com>

**Starter Templates**

You have two starter scripts for managing firewall rules: one script for servers and another one for workstations. These scripts can be modified to suit your needs. The scripts are too long to examine line by line here, but feel free to examine the code yourself in the time remaining. There are lots of comments.

Run the starter script for servers: ***.\Server-Firewall-Template.ps1***

Refresh your list of rules in the Windows Firewall snap-in. The starter script first disables all rules, then re-enables just the groups of rules desired. Some of these rules require IPsec and limit access by the source IP address range set aside for jump servers and administrative workstations (on the Scope tab of the rule). The script also creates a couple new rules for OpenSSH and HTTP/HTTPS.

Return to the starter script for workstations: ***.\Workstation-Firewall-Template.ps1***

Refresh your list of rules in the Windows Firewall snap-in. The workstation starter script first deletes all rules, then creates just the rules needed. When troubleshooting, just run the script again to start from a clean state. Because outbound connections are allowed by default, the only outbound rules created by the script are blocking rules; however, if you don’t want to block any outbound packets, you don’t need any outbound rules at all.

Scripts such as these can be applied through PowerShell remoting, executed as Group Policy startup scripts, or run every night at 3:00 a.m. as scheduled tasks.

Reset the firewall and IPsec rules back to their factory defaults: ***netsh.exe advfirewall reset***

***[End of Lab]***

**Today’s Agenda**

1. **Scripting Windows Firewall Rules**
2. **Scripting IPsec for Role-Based Access Control**
3. **Server Hardening Automation**
4. **PowerShell and Windows Logging**

Today’s Agenda

Now that we’re comfortable with the Windows Firewall, let’s create IPsec rules that can encrypt packets. When a firewall rule only allows packets in/out that have been secured with IPsec, the firewall rule doesn’t automatically create the necessary IPsec rule to actually encrypt or sign those packets. Hence, the firewall rules and the IPsec rules work both with each other, but they do not create or manage each other.

**Overview of IPsec**

**IPsec Benefits:**

* Mutual Authentication
* Port Permissions
* Encryption (Optional)
* Digital Signatures
* Network Logon Right
* Compatible with NAT
* Transparent to Users!

| **OSI Layer** | **Protocols** |
| --- | --- |
| Application | HTTP, SMB, SSH |
| Presentation | SSL,TLS |
| Session | RPC, NetBIOS |
| Transport | TCP, UDP |
| Network | IP, ICMP, IPsec |
| Data Link | Ethernet, ARP |
| Physical | 802.11n |

Overview of IPsec

Internet Protocol Security (IPsec) is a suite of protocols used for the authentication, integrity checking, encryption, and encapsulation of TCP/IP packets.

IPsec is implemented at the Network layer in the seven-layer OSI protocol model, the same layer as IP and ICMP. This seemingly simple fact, that IPsec is at Layer 3 in the OSI model, has drastic benefits.

An important point to understand is this: as authentication/encryption features are implemented at a lower and lower level in the protocol stack, these features become more *transparent* to users and more *compatible* with a wider range of applications and services. Users do not have to be trained to use IPsec-compatible applications because all their applications are already IPsec-compatible, even if the original application developers have never even heard the word “IPsec” before. Services and daemons do not have to be replaced with IPsec-compatible versions because they are already compatible. In short, if a piece of software sends IP packets over the wire, you can use IPsec to invisible secure those packets. *(Including ping?* Yes, including ping packets.)

That is the real shortcoming of doing encryption/authentication at the Application and Transport layers. PGP, SSH, and Stunnel have to be installed separately, for example, and users have to be trained and reminded to use them. TLS only works with applications and services specifically designed to support TLS, but at least TLS is more transparent than PGP or GnuPG because it is at the Transport layer instead of the Application layer. On the other hand, IPsec is invisible to all users, does not have to be installed (it’s installed already), and is compatible with all applications and services that communicates via IP.

As the table in the slide shows, only hardware-based cryptographic devices provide security at a lower level than IPsec. But in this case, you have to purchase special crypto hardware! IPsec is compatible with any off-the-shelf hardware that is capable of using IP, including Ethernet, token ring, FDDI, wireless, modems, serial lines, and infrared. In sum, IPsec is both independent of the underlying hardware and invisible to the upper-level protocols and applications above it – *that* is why IPsec is the standard.

**Threats**

IPsec is needed because the standard Internet Protocols – IP, ICMP, UDP, TCP, etc., – do not provide security for themselves. In fact, these protocols are inherently secure and archaic. They were never designed for security in the first place. They are wonderfully designed fossils of the DARPA-net Cold War era that, through the accident of technological evolution, have been pressed into service to make an information economy.

IPsec can help to prevent attackers from causing harm when attackers try to:

* Use a port scanner to discovery active TCP and UDP ports (reconaissance).
* Spoof source IP (denial-of-service and other attacks).
* Capture, modify, and resend packets (replay attacks).
* Impersonate hosts (man-in-the-middle attacks).
* Extract confidential information from captured packets (attacks against privacy).
* Connect to any open TCP port if the computer is directly accessible from the Internet, even if you wish some ports were only available to your LAN users.

**Benefits of IPsec**

There are numerous benefits of IPsec for network security:

* **Mutual Authentication.** Unlike SSL, both IPsec peers usually must authenticate to each first before an IPsec session can be established between them. Windows supports Kerberos, certificate, NTLM, and pre-shared key authentication methods for IPsec. Kerberos is enabled automatically when a computer joins the domain.
* **Strong AES Encryption.** The payload of a packet (or the entire packet itself when tunnelled) can be strongly encrypted for privacy with 256-bit AES.
* **Integrity Checking.** Packets can be checked to verify that they have not been accidentally damaged or deliberately modified during transit, using hashing algorithms like SHA-384, not just CRC checksums.
* **Transparent to Applications and Services.** Because security is implemented at the Network layer, IPsec is transparent to applications and services. Applications and services do not have to be upgraded or patched to make them IPsec-compatible. Legacy applications can be used, and they too will benefit from IPsec because they will be completely unaware of IPsec’s operations or existence. You do not have to look for an *“IPsec Inside!”* sticker on your new software to verify that it will work with IPsec.
* **No User Training Required.** Users do not have to be trained to use IPsec. In fact, users don’t even need to be aware that IPsec has been enabled on their computers. This is one of the most important benefits of IPsec. Without this user transparency, IPsec would be undeployable on desktops because of its complexity.
* **Group Policy Management.** Users do not have to be trained because IPsec can be centrally managed through Group Policy. Each OU or site could have a different set of IPsec policies applied to the computers in it. This feature is what enables IPsec on Windows to relatively easily scale out to thousands of machines.
* **PowerShell Management.** IPsec can be managed with PowerShell on local and remote systems. Some IPsec settings can only be managed through PowerShell.
* **Windows Firewall Integration.** The Windows Firewall can do more than just allow or drop IPsec packets. A firewall rule could allow access to a TCP or UDP port, for example, only if that traffic is encrypted with IPsec and the connection was initiated by a user who is a member of a specific group in Active Directory.
* **AES CPU Acceleration.** IPsec encryption on Intel or AMD processors built in Q4’2010 or later will benefit from the AES-NI instruction set designed into these processors for AES hardware acceleration. This is also benefits TLS and BitLocker.
* **IPsec Hardware Acceleration NIC’s.** IPsec cryptographic operations can be offloaded to smart network adapter cards. These IPsec-enabled network cards possess cryptographic processors to perform the CPU-intensive work of authenticating, integrity checking, and encrypting packets on behalf of the operating system. When using Hyper-V on a Windows Server with such a NIC, guest VMs using IPsec can benefit from the hardware acceleration too.
* **Network Logon Right Integration.** When using either Kerberos or certificate-to-computer-account-mapping authentication, Windows Server will enforce the “Access This Computer From The Network” and “Deny Access To This Computer From The Network” user rights against remote computers and users when they initiate inbound IPsec connections to it. Hence, you can limit by group membership which computers or users are permitted to open inbound IPsec connections and then block all non-IPsec connections to the port or service you are trying to secure. Remember, computer accounts can be allowed/denied logon rights just like users (and must be, if rights restrictions are used for IPsec).
* **Firewall Compatibility.** When used by servers that must communicate through a firewall to the internal LAN, such as web servers and database servers in the DMZ, IPsec simplifies the firewall design because the vagaries of the protocols used do not have to be protected and managed. The servers can be required to communicate over IPsec.
* **Network IDS Compatibility.** IPsec packets do not have to be encrypted. Using either AH, ESP with encryption disabled, or “null encapsulation”, IPsec packets will be sent in plaintext. These cleartext packets can be examined by Network Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). If the IDS/IPS is confused by the AH or ESP header, IPsec null encapsulation means that no IPsec headers are added to packets whatsoever after the initial IKE negotiation.
* **NAT Compatibility (Usually).** The NAT-T extension to IPsec can permit sessions through firewalls, wireless access points, and other devices that perform Network Address Translation (NAT), but only if the devices performing NAT are well-behaved (see RFC3948). There can be problems with double-NAT scenarios, though, but, single-NAT network paths are usually no problem.
* **Interoperable.** Because IPsec is an IETF standard, it is not owned by any single vendor. Hence, different vendors can create interoperable IPsec products; for example, a Windows client can establish an IPsec connection to a Sophos UTM gateway running strongSwan IPsec on top of Linux ([www.strongswan.org](http://www.strongswan.org)).
* **Extensible.** IPsec was designed to be extensible and flexible. For example, as new ciphers and key lengths are desired, IPsec can be extended to support them while maintaining backward compatibility.
* **IPv6 Support.** IPsec was designed with IPv6 in mind. The version of IP in widespread use today is version 4. IPv6 is the next generation of IP, with 128-bit addresses, extensible headers, and a number of new features for a heavily networked world. Windows supports IPv6 natively as well.
* **Virtual Private Networking Support.** Virtual Private Networking is the ability of a remote user or router on the internet to connect to the corporate LAN through an encrypted secure channel. This encrypted channel will encapsulate or “tunnel” entire packets during transit. VPNs reduce long distance charges while providing very secure remote access.
* **Network Load Balancing (NLB)**. IPsec and NLB are compatible and can be used together to create a load-balanced and fault-tolerant farm of VPN gateways (maximum of 32 gateways in the farm). With DNS round robin, multiple farms can be used for even greater scalability. For configuration steps, see KB820752 and KB323437, as well as the help menu in the Network Load Balancing Manager in the Administrative Tools folder.

Drawbacks to Using IPsec

There are a few important drawbacks to using IPsec on Windows:

* **NAT.** While the IPsec implementation on Windows supports NAT-T for Network Address Translation (NAT) compatibility, that does not mean all the other networking devices through which one’s IPsec connections are routed will be well-behaved or configured correctly (see RFC 3948). This is especially a problem when both IPsec peers are behind their own separate firewalls that perform NAT (this scenario is called “double NAT”). There is a registry value named “AssumeUDPEncapsulationContextOnSendRule” that may be set to help the two Windows IPsec peers, not the firewalls or routers along the network path accessing VMs hosted by cloud providers. Often, the only solution to the problem is to either 1) establish a non-IPsec VPN tunnel between the two networks and use IPsec within the VPN, or 2) move one of the IPsec peers from behind its NAT-ing device and give that peer a public IP address. Once IPv6 is standard, NAT should finally go away forever.
* **Throughput.** A Windows IPsec/VPN will not even have the same throughput as a hardware-only IPsec/VPN solution, even if IPsec-enabled network adapter cards are used. If your organization is not already using Windows, or if you must support a very large number of simultaneous VPN clients, then a hardware solution may be more cost-effective.
* **IKEv2 and Tunnel Mode.** Windows only supports enough of IPsec tunnel mode for bare RFC compliance with IKEv2. Microsoft doesn’t really want you to use tunnel mode for anything other than gateway-to-gateway connections. There are some reasons for this, but the tunnel mode limitations and other IKEv2 issues in Windows can be disappointing to IPsec pros.
* **IKEv2 Fragmentation.** IKEv1 and AuthIP support the fragmentation of messages at the IKE level so that packets are not fragmented at the IP level. Some firewalls drop IP-layer fragmented packets. However, IKEv2 does not support IKE-level message fragmentation; hence, if UDP packets for IKE\_AUTH messages are too large, they will be fragmented by IP, causing IKEv2 negotiation timeouts and failures when perimeter firewalls drop the fragments. To avoid this situation, either use IKEv1 or AuthIP, make sure your own firewall allows IP-fragmented packets to UDP ports 500 and 4500, or use an EAP authentication method with your IKEv2 VPN that does not require packet fragmentation.
* **Interoperability.** With the possible exception of Cisco devices, Microsoft is not particularly concerned about interoperability with non-Windows IPsec peers. Some degree of interoperability is currently possible with various flavors of UNIX/Linux, but the next update could easily change all this. The only way you can know if it it’ll work is to set it up and see what happens.
* **Pre-Shared Keys.** Microsoft stores “pre-shared keys” for IPsec authentication in cleartext in the registry. If the key can be read by an adversary, the adversary can open her own IPsec sessions with one’s systems. However, capture of this key does not enable an adversary to decrypt anyone else’s sessions because the pre-shared key is only used for authentication, not data encryption. When using Group Policy to manage IPsec, any pre-shared keys will be in plaintext in the GPOs as well, which adversaries could sniff or extract from the SYSVOL share.
* **Transparency.** Transparency to application-layer software is not always ideal. Unlike SSL/TLS, the details of an IPsec session are usually not (easily) available to applications if those applications wish to confirm the existence or details of the IPsec session, such as cipher suite or peer credentials, before proceeding with critical actions. Services and client applications are at the mercy of the operating system and the administrators who manage it.
* **DoS Attacks.** Like other implementations of IPsec, flooding of UDP port 500 or 4500 can prevent a victim host from establishing new IPsec sessions with other hosts. UDP 500 and 4500 are the ports used for IPsec Internet Key Exchange (IKE) negotiations; hence, it is important to quickly apply new IPsec-related patches.
* **Checksum Offload.** In rare cases, with some network adapters, especially when using UDP port 4500 encapsulation and dealing with double-NAT issues, it may be necessary to use the Device Manager tool to edit the properties of the adapter to disable “Checksum Offload” for IPv4, IPv6, UDP, and TCP. This imposes a tiny performance penalty on the mainboard CPU.

IPsec = IKE + ESP + AH

There is no one protocol named “IPsec”. IPsec is a suite of protocols that work together as a team. Three main protocols are Internet Key Exchange (IKE), IP Authentication Header (AH) and IP Encapsulating Security Payload (ESP).

The following table summarizes the purposes of IPsec’s core protocols.

| Protocol | Purpose |
| --- | --- |
| IKE | IKE securely negotiates IPsec communication parameters and cryptographic keys between peers. Parameters include authentication methods, ciphers, key lengths, sequence numbers, time-to-live counters, session ID numbers, and so on. |
| AH | AH provides authentication of packet source, data integrity, and protection against replay attacks, but does **not** provide encryption. It’s also incompatible with Network Address Translation (NAT). |
| ESP | ESP provides data encryption, authentication of packet source, data integrity, protection against replay attacks, and unlike AH, is compatible with Network Address Translation (NAT). |

**Key Management and Authentication Come First**

IKE is used first, then ESP and/or AH come afterwards. IKE handles the up-front work of authenticating users and computers, negotiating encryption keys, and doing all the other housekeeping. ESP and AH, on the other hand, are the main workhorse protocols. Perhaps 99% of the data secured by IPsec are encrypted or signed by ESP/AH. When sniffing IP packets with Wireshark or tcpdump, you’ll likely see about a dozen IKE packets at the beginning of a connection, followed by thousands of ESP packets. Why not AH?

**AH Never Provides Packet Payload Encryption**

The most important distinction between AH and ESP is that ESP provides data encryption, while AH does not. You must use ESP when you want privacy. However, note that it is possible to disable the encryption, if desired, when using ESP.

AH also authenticates the *IP layer* of the packet and higher, while ESP only authenticates the *transport layer* and higher. This means that changes to the IP layer of the packet are detected by AH but not by ESP.

Both AH and ESP can be used simultaneously, though this is very rare. Hence, a single packet can use ESP to encrypt its transport layer and higher, while using AH to authenticate and integrity-check the entire packet (except the physical layer).

**NAT Incompatibility: Just Never Use AH**

Very importantly, ESP can traverse devices performing Network Address Translation (NAT) without errors, while AH is always incompatible with NAT.

Because ESP can sign plaintext packets without necessarily encrypting them, and because ESP is compatible with NAT, just never use AH. If you want to just sign packets, not encrypt them, use ESP, and simply turn off the encryption.

**RFC Standards**

IPsec standards are defined by Internet Engineering Task Force (IETF) Requests for Comments. IPsec itself is not owned by any one vendor, such as Microsoft or Cisco.

When security is the number one concern, IPsec is the gold standard for securing TCP/IP traffic and doing Virtual Private Networking over the internet. Other protocols might be simpler than IPsec, but they really can’t complete for security, extensibility, and being future-proof.

There are many RFCs related to IPsec. The following are the core RFCs and should be read in this order:

* RFC 4301: Security Architecture for the Internet Protocol
* RFC 2409: Internet Key Exchange (IKEv1)
* RFC 7296: Internet Key Exchange (IKEv2)
* RFC 4303: IP Encapsulating Security Payload (ESP)
* RFC 4302: IP Authentication Header (AH)

**Example IPsec Scenarios and Uses**

* **Dangerous protocols and ports on endpoints:** Wireless traffic, SMB, RPC, FTP, DNS, RDP, VNC, etc.
* What global groups should be allowed to access these ports?
* ***Prefer* IPsec on high-value endpoints:** Allow plaintext whenever necessary, but IPsec is preferred.
* ***Require* IPsec to make an encrypted VLAN:** Different inbound vs. outbound rules, easily make exceptions.
* **Secure servers in the cloud or in your DMZ:** Permit secure administration only to necessary groups. Combine with firewall rules for host-based segmentation.

Example IPsec Scenarios and Uses

IPsec is a powerful and versatile protocol. A few examples of its uses can illustrate.

**Making “Dangerous” Ports, Protocols,and Tools Safer On Endpoints**

There are many tools and services one would like to use – especially for remote administration – but cannot because of their security weaknesses, e.g., they transmit cleartext passwords. But if servers were configured to require and authenticated IPsec session before allowing connections to these dangerous ports, and if 3DES encryption were required for all the traffic, then these dangerous tools and services might be made secure enough to use, perhaps even on bastion hosts in the firewall’s DMZ. The following is a sampling of the applications many administrators would like to use, or services they would like to enable on their servers, but cannot because of security worries, yet IPsec might make them secure enough (even the ancient ones):

* FTP
* TELNET
* SNMP
* SYSLOG
* RADIUS and LDAP authentication
* RPC-based applications (like most MMC snap-ins)
* Microsoft File and Print Sharing (SMB/CIFS)
* Remote Desktop Services (remote administration mode)
* VNC Remote Control ([www.realvnc.com](http://www.realvnc.com))
* PSEXEC -U (otherwise sends passwords in cleartext)

And the list goes on for all those protocols, services, and tools that would have been nice to run on servers in the DM but were too dangerous even when protected by the firewall.

**Wireless Networking**

Wireless networks can be securely bound together with IPsec. Authentication will limit communication channels to just those systems participating in the domain or PKI, and encryption can provide privacy. And you don’t have to require IPsec for all wireless communications either; you could rely upon the security enhancements of WPA2 for the most part, then require IPsec in addition whenever connecting to critical servers through the wireless link. If you do wish to use IPsec for all wireless traffic, install a wireless card that can run in Access Point mode in a Windows router, then configure the clients to use IPsec tunnel mode to forward all packets destined to the LAN to the wireless router instead. The Windows box will decrypt the IPsec packets and route/bridge the original packets onto the LAN. If your hardware Access Point vendor’s box supports IPsec natively, then all the better! (And if one needs to push out IPsec digital certificates, one might as well push out certificates for 802.1X EAP-TLS authentication too.)

**Servers in the Cloud or DMZ**

Many websites are composed of a farm of web servers that act as front-ends to middleware servers that themselves communicate with multiple backend databases (in a “three-tier” design). IPsec could be used to authenticate and integrity-check (but not encrypt) all communications among the servers and databases, then block everything else from the internet and DMZ. A farm of web servers is also often a part of an isolated domain in the DMZ to provide for administration and content management. The domain controllers for this domain would be on a separate DMZ segment, and IPsec would be used to armor the communications between DCs and the web servers. Authentication protects the vital communications links between these servers, but without the overhead of encryption. Encryption isn’t needed here because the threat of packet sniffing is relatively small.

**Critical Data Flows and Sensitive Machines (Preferring IPsec)**

“Defense-in-depth” means providing multiple redundant layers of security. At many sites, if an attacker can penetrate the firewall, there are no other barriers between the attacker and critical internal servers. Exclusive reliance upon the firewall creates a single point of failure. IPsec can help to provide in-depth security for sensitive internal servers and workstations, for example:

* Domain controllers could replicate using IPsec.
* Databases could synchronize securely over IPsec.
* The workstations of the security administrators could require IPsec for all communications because the boxes they manage would too.
* The file server with the R&D source code might always require IPsec.
* The computers of a certain OU might be isolated from all other machines by their secret authentication keys and packet filtering rules.
* The computer OUs for corporate executives, HR, and legal staff might always attempt to negotiate IPsec encryption, but fall back down to cleartext when required.
* And, in general, IPsec can help to defend a network against its own “trusted insiders” who are, in fact, according to the FBI, behind the majority of network security breaches that cause measurable financial harm.

For example, at the time of this writing, Microsoft’s corporate network comprises 18 domains in six forests with cross-forest trusts and over 200,000 managed and unmanaged hosts. Yet approximately 70% of the internal traffic at Microsoft uses IPsec ESP (with no encryption enabled) to help protect the managed boxes from the unmanaged ones. (ESP is used instead of AH since ESP supports NAT-T and AH doesn’t.) ESP with encryption is enabled on an as-needed basis through Group Policy on critical servers.

Very importantly, notice that in many of the examples above that IPsec is not required from the other computer; IPsec is merely preferred. This means a system can be configured to attempt to negotiate IPsec whenever a new inbound or outbound connection is being established, but if the other computer cannot or will not authenticate with IPsec, the system will fall back to unsecured communications automatically, thus permitting the connection like normal. Hence, you do not always have to require IPsec when configuring a system; you can configure that system to merely prefer IPsec when configuring a system you can configure that system to merely prefer IPsec but be willing to talk to non-IPsec-capable machines, as necessary. And you can either require or prefer IPsec on a case-by-case basis, depending on the IP addresses, protocols, or ports being used.

**Getting Aggressive with Domain Isolation (Requiring IPsec)**

Imagine if all your workstations and servers (or a large subset of them) required IPsec mutual authentication prior to any communication whatsoever. This would be similar to a VLAN, but instead of using switches, you’re using IPsec. Only domain-joined computers would have the necessary Kerberos tickets or certificates to be able to authenticate to other domain members, and all (most) systems would be configured through Group Policy to require mutual IPsec authentication. If an unauthorized computer attempted to open a TCP or UDP connection with a domain member, the connection would fail because of the inability of the rogue computer to authenticate with IPsec first.

**Partner Networks (Outside Your Active Directory)**

You may have SMTP relays, websites, file servers, etc. to which you wish to give a partner network limited access. However, the partner company is not a part of one’s forest and/or does not use Windows systems. Windows IPsec connections can be authenticated with digital certificates. These certificates can be distributed to hosts in other forests or to non-Windows systems. In short, the use of IPsec is not restricted to those within one’s own organization or Active Directory forest.

**Virtual Private Networking**

When roaming users need to gain access to the LAN while on the road, IPsec can be used with L2TP to provide encrypted and authenticated client-to-router VPN tunnels. The client can now communicate with other hosts on the LAN just as though she were physically connected. If all the users in a branch office need access to the main office, the two offices can be connected over the internet with router-to-router IPsec. The routers perform all the encryption and authentication transparently for the users. The solution is secure, relatively easy to set up, and saves on long distance leased lines. From the perspective of the two LANs, there just appears to be another “segment” connecting them. But that “segment” is actually an encrypted VPN tunnel through the Internet.

**Internet Key Exchange (IKE)**

**Security Association:**

* **It’s a contract negotiated between two IPsec peers.**

**Phase I (Main Mode)**

* **Mutual Authentication**
* **Diffie-Hellman Exchange**

**Phase II (Quick Mode)**

* **Session details like cipher**

Internet Key Exchange (IKE)

An “**IPSec peer”** is any device – host, server, router, gateway, phone, etc,. – which can be a party to an IPsec-secured communication. IPsec peers always begin their communications with an Internet Key Exchange (IKE) negotiation.

**I like IKE, IKE Is a Good Negotiator**

IKEv1 (RFC2409) and IKEv2 (RFC7296) are protocols for negotiating communication parameters and cryptographic keys between IPsec peers. IKE also negotiates what method(s) will be used to authenticate those peers to each other. Authenticated IP (AuthIP) is a Microsoft proprietary extension of IKEv1. IKEv2 is not backward compatible with IKEv1 or AuthIP.

**Security Association (SA)**

The end result of an IKE negotiation is an in-memory data structure called a **“Security Association (SA)**.**”** An SA is a set of parameters and cryptographic keys used by a peer to manage an IPsec session with another IPsec peer. Think of an SA as a legal contract negotiated between the IPsec peers, specifying how they promise to communicate with each other while the contract is still alive, but either peer can terminate the contract at any time. Every bidirectional IPsec session will correspond to two SAs on each peer: one for securing outgoing packets to the other peer and one for securing incoming packets from that other peer. An SA is always one-way; hence, two are required on each peer for bidirectional communications.

An SA is what maintains the context or “state” of an IPsec session. The information and keys in an SA enable the IPsec driver on a peer to successfully encrypt, decrypt, and verify the integrity of IPsec packets from the other peer. An IPsec peer will keep all of its SAs in its own **“Security Association Database (SADB)”** in memory. Each SA in a peer’s SA database is identified by a unique number called a **“Security Parameters Index”** number. One SPI number for each SA in the SADB. When peers send IPsec packets to each other, they will include the SPI numbers of the SAs controlling their IPsec communications in the headers of the IPsec-secured packets they exchange. This fact is important for understanding how IPsec operates. Including an SPI number in the header of every IPsec packet explains how a host can maintain many IPsec associations with multiple peers simultaneously. The SPI number in an IPsec packet received by a peer allows that packet to be matched up to an SA in the SADB in the memory of that peer.

**IKE Speaks ISAKMP (“Eye-Sa-Camp”)**

IKE negotiations are carried out using a more abstract negotiation language allied to the **Internet Security Association and Key Management Protocol (ISAKMP)**. IKE *speaks* a dialect of ISAKMP used for IPsec.

For IKEv1, a special document called a “Domain of Interpretation (DOI)” is used to lay out the details of how IKE negotiates for IPsec, as defined in RFC 4306. (IKEv2 doesn’t use DOI documents anymore though.) And for key negotiation, IKEv1 uses the OAKLEY Key Determination Protocol, as defined in RFC 2412. (IKEv2 does not use OAKLEY anymore either.) Hence, IKE uses DOI, OAKLEY, and ISAKMP together.

All this can be confusing, but just keep in mind that IKE uses ISAKMP to do speedy and extensible SA negotiations, part of which is OAKLEY to share a key. Some authors and protocol analyzers will mention IKE, and others ISAKMP, when referring to the same thing. By analogy, lawyers in the United States follow strict rules while in the courtroom (IKE), and they speak English while following these rules to plead their cases (ISAKMP), but it’s just one activity described in two ways.

**UDP Port Numbers 500 and 4500**

IKE negotiations occur over UDP port 500 by default, then switches over to UDP 4500 when going through a device performing Network Address Translation (NAT). NAT-T and NAT-D are extensions to the IKE protocol to allow the use of IPsec through one or more NAT devices like routers and firewalls. NAT-D is for the **d**etection of NAT, while NAT-T is for the **t**raversal through NAT devices.

**Diffie-Hellman-Merkle**

One of the most important cryptographic techniques used by IKE is the **Diffie-Hellman-Merkle (DH)** key exchange. DH is a method for two parties to negotiate a shared secret key over an insecure channel like the internet, but without ever sending that secret key itself over that channel, not even in an encrypted form.

The parties must first agree which “group” to use. A **group** is a set of numbers used to control the DH exchange, such as a large prime number and second smaller number related to that prime, or elliptic curve information. The numbers can be transmitted between the peers in the clear without risk of compromising the secret key generated with them. (Search Wikipedia for nice explanations of the mathematics behind the different flavors of DH.)

The important thing to know when configuring DH is that different “groups” have different security strengths. In general, choose the strongest DH group available to your peers that does not impact performance too much. Later operating systems support better DH groups than older ones. In Windows Vista and later, for example, the DH exchange can be performed using Elliptic Curve Cryptography (ECC) for ECDH, which is faster and more secure than traditional DH. In some cases, you only get to choose the DH group you want in PowerShell, since the Windows Firewall snap-in is somewhat feature-locked by Microsoft’s sprawling documentation (and laziness).

**IKE Negotiation Phases**

IKE negotiations occur in two Phases: Phase I and Phase II. The end result of each Phase is a separate SA; hence we make a distinction between Phase I SAs and Phase II SAs. An IPsec session with a remote peer will require two SAs on each peer.

The SA negotiated in Phase I is separate from and prior to the Phase II SA. Importantly, the Phase II SA relies upon the SA negotiated in Phase I. The most important encryption key exchange occurs in Phase I, and this encryption key is used to secure the Phase II negotiations. The SA contract agreed upon in Phase II is only secure and trustworthy to the peers because of the authentication and key exchange that occurred in the Phase I.

Unfortunately, now things get a little complicated. Strictly speaking, these Phases only exist in the RFC standards for IKEv1, but something very similar happens with IKEv2. So similar, in fact, that it’s OK to talk about these Phases for IKEv2 also, as long as we remember that it’s more of an analogy. Let’s talk about the Phases for IKEv1, then what is similar to IKEv2 afterwards.

**IKEv1 Phase I Negotiation Steps**

In outline, the steps of a Phase I negotiation in IKEv1 are:

1. **Negotiate Policy:** Cipher to be used (like AES), hashing algorithm to be used (like SHA-1 or SHA-384), authentication method (Kerberos, certificate, pre-shared key), and Diffie-Hellman settings. This is for IKE itself, not the bulk encryption of user data later on.
2. **Perform a Diffie-Hellman-Merkle Exchange:** The DH technique is used to derive an identical, secret, encryption key on each peer. Adversaries sniffing all IKE packets cannot derive this key; only the two IPsec peers can.
3. **Authenticate Peer**: The DH-derived key from the prior step is used to secure an authentication sequence, such as Kerberos or a passphrase, to authenticate the peers to each other.
4. **Create the IKE SA**: With the above authenticated credentials and key, the end product IKE needs – a phase I SA – is created in the SA database in-memory.

Note that even when there are no active Phase II SAs between two peers, there may still be a Phase I SA between them. If a new Phase II SA is needed, the existing Phase I SA will be used to carry out a Phase II negotiation to produce the needed SA.

**IKEv1 Phase II Negotiation Steps**

In outline, the steps of a phase II negotiation for IKEv1 are:

1. **Negotiate Policy**: IPsec protocol (AH or ESP), cipher (such as 3DES or AES), and hashing algorithm (like SHA-256). These details are for securing the user’s data now, not any IKE traffic.
2. **Key Generation**: The DH-derived key from Phase I is used to derive a new key for this Phase II SA. Rekeying timers are defined too.
3. **Create Phase II SA**: The above parameters and keys are combined to form a new SA in the SADB with a unique SPI number.

When *an* SA needs to be renegotiated, perhaps because its lifetime has expired or an error has occurred, usually this means a new Phase II negotiation is required, but it can also mean that a new Phase I negotiation is required too. Because each Phase II SA relies on some other Phase I SA, when a Phase I SA is deleted, all the Phase II SAs that depend on it are deleted as well from the SADB.

**Main Mode and Quick Mode IKEv1 Negotiations**

The IKEv1 Phase I negotiation can occur either in “main mode” or “aggressive mode”. The end result of the modes is the same (an IKE SA) but their details differ (main mode is more secure, but slower).

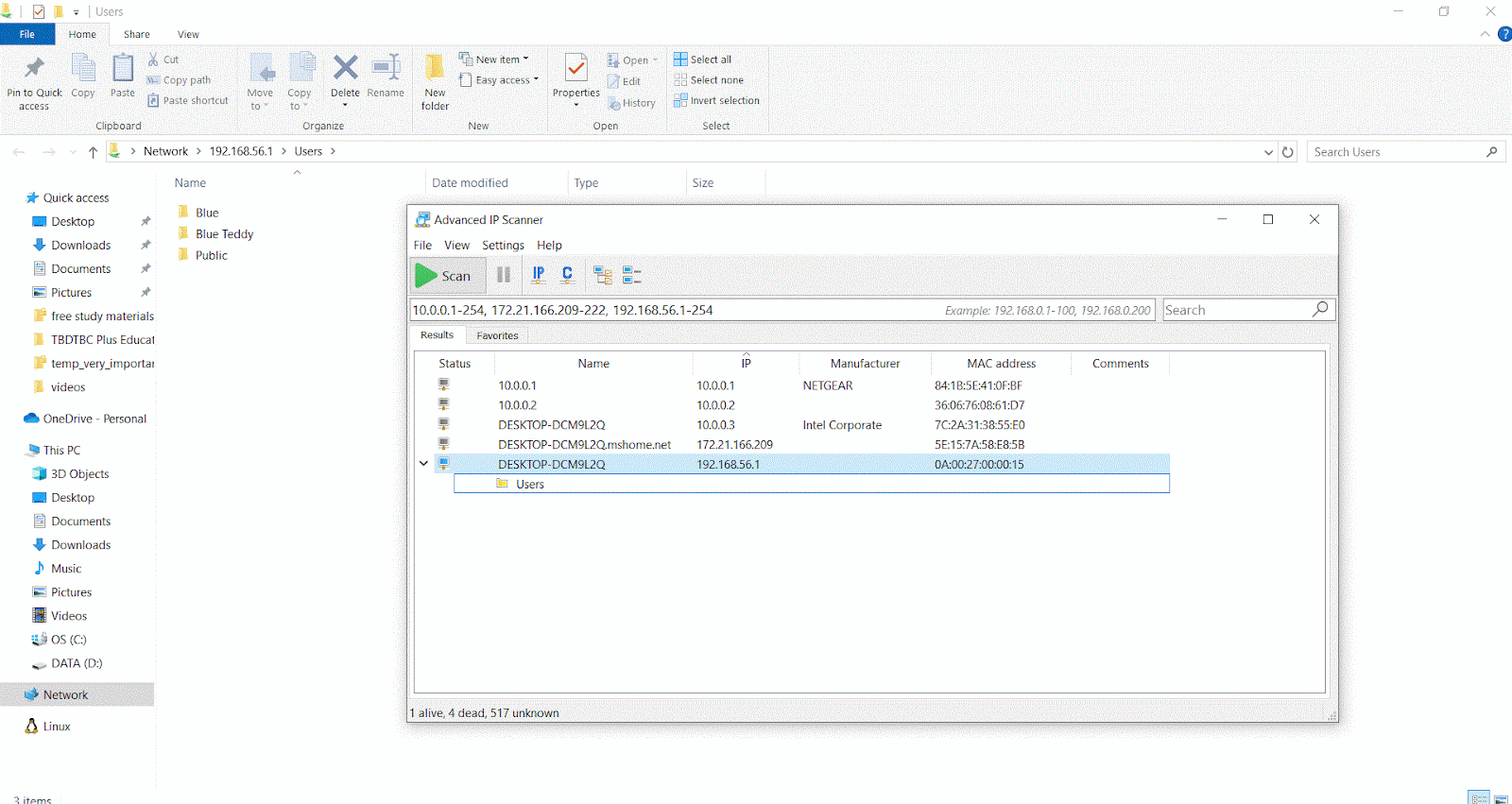
Windows does not support Phase I in aggressive mode, only main mode. Non-Windows implementations of IPsec, such as strongSwan, may support aggressive mode.

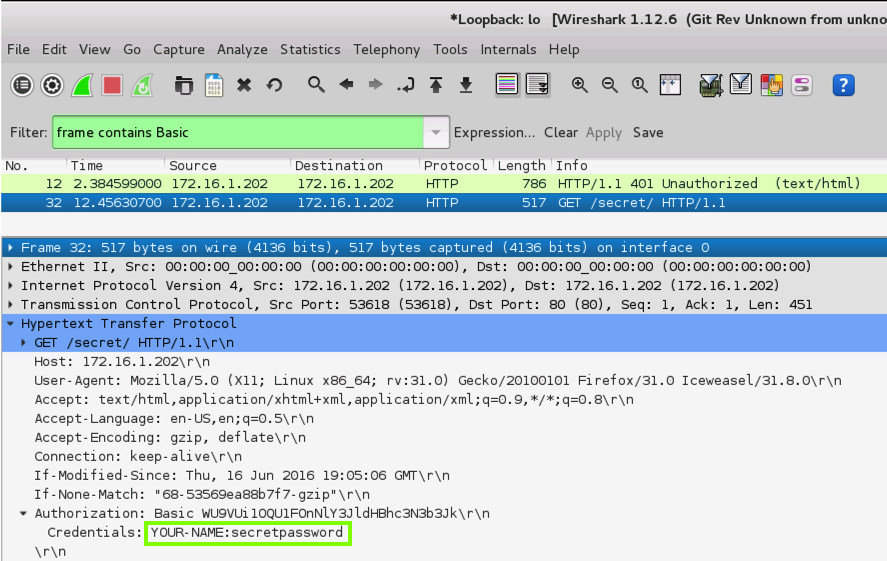
Phase II negotiations always occur in a third mode named “quick mode”, whether it is Windows or not.

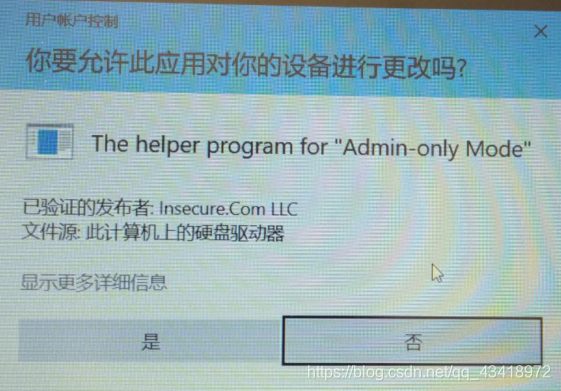
There is only quick mode for Phase II.

The important thing to remember is that “main mode” is often used synonymously with “IKEv1 Phase 1”, while “quick mode” is often used synonymously with “IKEv1 Phase II”, even though these terms are not exactly synonymous. (Note that IKEv2 does not have these modes of operation at all.)

IKEv1 borrows these modes from the OAKLEY Key Determination Protocol (RFC 2412), so you will sometimes also see them referred to as “OAKLEY modes” or “OAKLEY negotiations”. IKEv1 also borrows from the SKEME protocol for public key authentication, but that’s another story…I mean, another RFC. (And IKEv2 uses neither OAKLEY nor SKEME.)







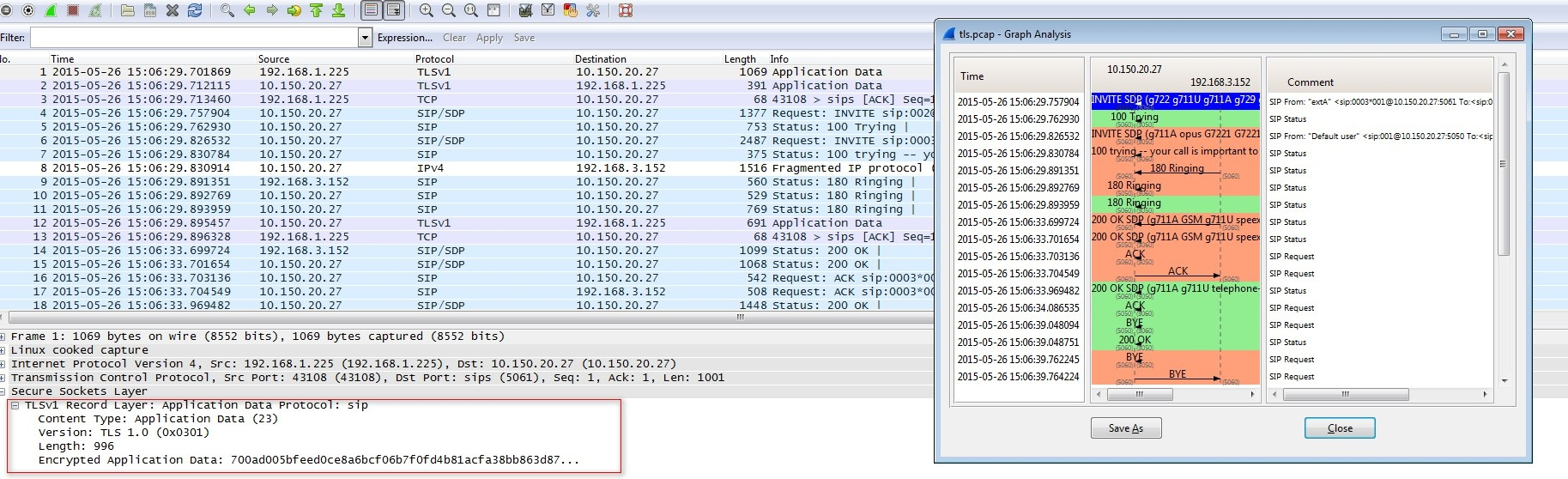
**IKEv1 On Steroids: Microsoft Authenticated IP (AuthIP)**

IKEv1 has some frustrating limitations, especially regarding peer authentication. In response, Microsoft extended and enhanced IKEv1 in a proprietary way that is only supported on Windows. “Authenticated IP (AuthIP)” is the name of Microsoft’s set of enhancements to IKEv1. AuthIP is like IKEv1++, but it’s not IKEv2. AuthIP is not the same thing as IKEv2, even though they solve some problems in similar ways. AuthIP and IKEv2 are not compatible. In fact, AuthIP is not even compatible with IKEv1!

AuthIP is compliant with the RFC standards for IKEv1 because those standards allow vendor-specific extensions to be added – and boy did Microsoft add some.

AuthIP is enabled by default on Windows Vista, Server 2008, and later. AuthIP will be used instead of IKEv1 whenever possible. Windows will fall back to IKEv1 when necessary for backward compatibility and non-Windows compatibility. (When IKEv2 is used, AuthIP is turned off completely for that IPsec session.) The use of AuthIP, or the fallback to IKEv1 for that matter, is automatic and completely transparent to the user. How?

Recall the Phase I main mode negotiations mentioned earlier. When Windows sends out its very first IKE/ISAKMP packets to start negotiating IPsec with another user, Windows will send both an AuthIP request and a regular IKEv1 request. If the other peer is Windows, it will respond to the AuthIP request and ignore the IKEv1 request. If the other peer is non-Windows or older than Vista, it will ignore the AuthIP request and respond only to the IKEv1 request. However, the responder replies will determine how the rest of the negotiation process proceeds, namely, with vanilla IKEv1 or with Microsoft’s AuthIP enhancements.

****

In packet captures of AuthIP sessions, find the Exchange Type field in the ISAKMP header. AuthIP uses exchange types of 243 (Main Mode), 244 (Quick Mode), 245 (Extended Mode), and 246 (Notify) in the Exchange Type field.

So what are these AuthIP enhancements? What’s the big deal? Actually, it’s very big.

AuthIP adds significant enhancements on top of vanilla IKEv1:

* AuthIP supports peer authentication of the remote *user*, not just the remote computer. User authentication is necessary if you wish to limit access to a TCP/UDP port with IPsec based on the user’s group memberships of Active Directory. AuthIP supports Kerberos, NTLMv2, and certificate-based *user* authentication. IKEv1 only authenticates devices, not human users.
* AuthIP permits multiple authentication attempts using different authentication protocols. IKEv1 only allows one authentication attempt with one authentication protocol; if that one attempt fails, it’s possible to have a list of supported authentication protocols, each protocol will be attempted in turn, and the IPsec connection will fail only if every one of the authentication attempts on the list fails too.
* AuthIP can either require or just prefer IPsec; for example, a file server might require authentication with a certificate for the server as a whole, but only prefer user authentication too, so that an IPsec SA can still be established even if authentication of the initiating user fails or is not attempted at all. Indeed, IPsec as such could be preferred but not required; hence, when two peers begin to communicate in plaintext, they will switch over to IPsec if negotiations and authentication succeeds, but if IPsec fails for any reason, the peers will continue to communicate in plaintext like normal.
* Because AuthIP supports NTLMv2 authentication, it is much easier to use IPsec between a domain controller and a member computer during initial domain join. Standalone computers cannot use Kerberos, and certificate auto-enrollment requires Group Policy and domain membership too. NTLMv2 helps to solve this chicken-and-egg problem of getting computers joined to the domain when domain controllers require IPsec.
* AuthIP uses Generic Security Services API (GSS-API) as defined in RFC2743 and <https://tools.ietf.org/html/draft-ietf-ipsec-iskmp-gss-auth-07>. This means IKE Phase I authentication can tap into and leverage the underlying authentication services of the Windows operating system and the larger Active Directory environment for Kerberos and NTLMv2. Architecturally, this is the main reason Microsoft invented AuthIP and is how many of the above benefits are obtained for IPsec under the hood. Because this is so Microsoft-centric, integrating GSS-API never became popular for IPsec outside of Windows. If you happen to know about such things, you might wonder if AuthIP also uses Simple and Protected GSS-API Negotiation Mechanism (SPNEGO). It does not; AuthIP uses GSS-API, but not SPNEGO (You don’t have to know what this means).

By the way, when AuthIP is using additional rounds of authentication, Microsoft calls this behavior “Extended Mode (EM)” for AuthIP. Extended Mode has nothing to do with IKEv1 main mode, aggressive mode, quick mode, or anything in IKEv2. You don’t have to know about Extended Mode when configuring IPsec on Windows. Technically, Extended Mode begins, if requested, after Phase II negotiations are complete. Whenever a Microsoft document talks about “Extended Mode IPsec”, it’s really just talking about AuthIP attempting more than one authentication method. When creating an IPsec rule in PowerShell, you can explicitly choose the “KeyModule” to control how Phase I and Phase II negotiations are carried out. Here is a summary:

| Command to Create an IPsec Rule | Behavior |
| --- | --- |
| New-NetIPsecRule -KeyModule **IKEv1** | Only use vanilla IKEv1, nothing else |
| New-NetIPsecRule -KeyModule **AuthIP** | Only use AuthIP, no fallback to IKEv1 |
| New-NetIPsecRule -KeyModule **IKEv2** | Only use IKEv2, nothing else |
| New-NetIPsecRule -KeyModule **Default** | Offer both AuthIP and IKEv1, prefer AuthIP, but fallback to IKEv1 as needed |

AuthIP-with-IKEv1-fallback is not only the default when configuring IPsec in PowerShell, it’s also the default when using the Windows Firewall snap-in or Group Policy to manage IPsec. Currently, the only way to create IPsec rules that use IKEv2 is PowerShell. So how does IKEv2 differ from IKEv1 and AuthIP?

**IKEv2**

IKEv2 is supported on Windows 7, Server 2008 R2, and later.

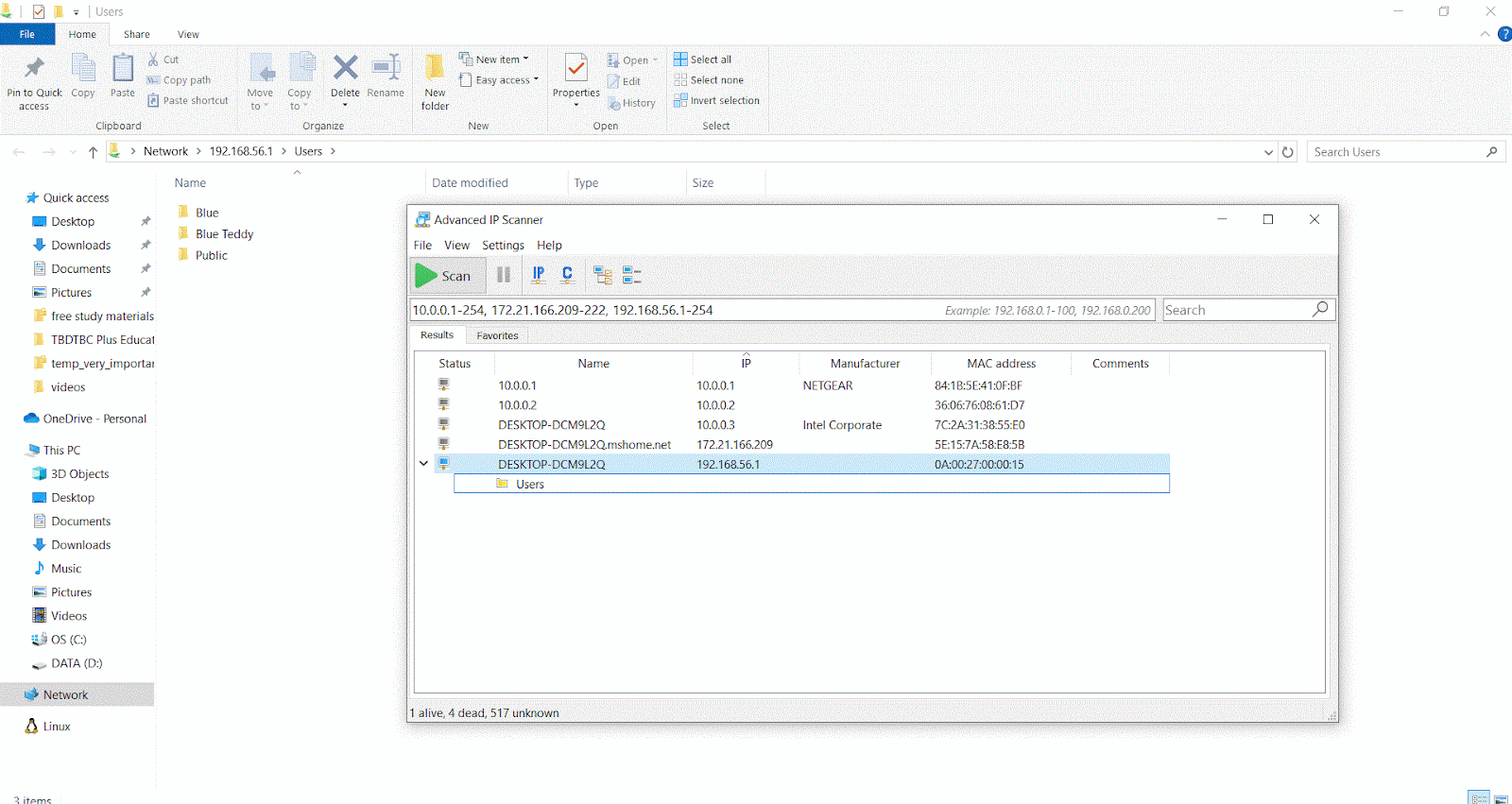
IKEv2 is not compatible with IKEv1 or AuthIP. IKEv2 is actually simpler and a bit faster than IKEv1. IKEv2 does not use GSS-API authentication services in the operating system, but it does support multiple authentication attempts using Extensible Authentication Protocol (EAP) as defined in RFC 3748. EAP is not mandatory for IKEv2, just available. On Windows, however, using IKEv2 means using either machine certificate authentication or EAP. When authenticating users in IKEv2 on Windows, it virtually always means using EAP for a VPN.

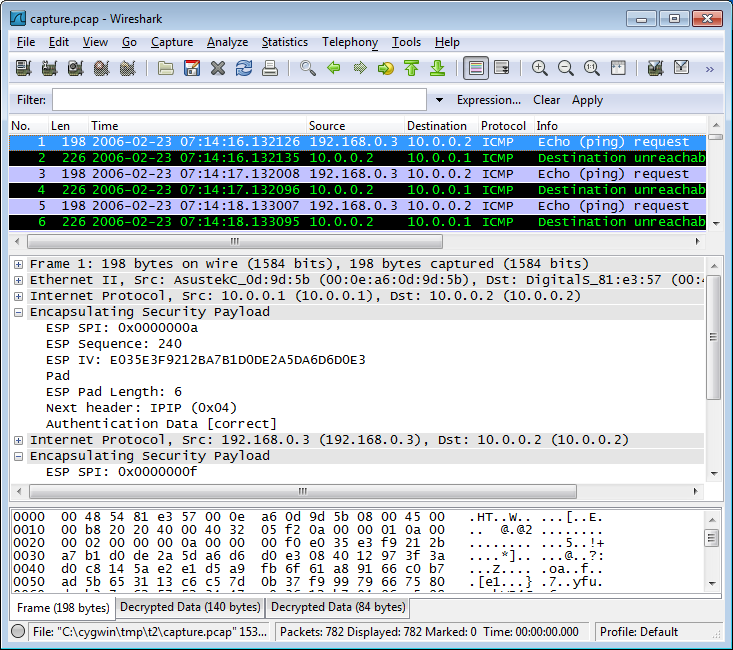
RFC 7296 for IKEv2 defines sets of message exchanges that roughly correspond to Phase I and Phase II negotiations mentioned for IKEv1. Strictly speaking, there are no “Phases” in IKEv2, but there are similarities and it aids in understanding. Communication using IKEv2 always begins with exchanges named “IKE\_SA\_INIT”, which comes first, and “IKE\_AUTH”, which comes second, assuming that the first IKE\_SA\_INIT exchange was successful. An “exchange” is a request-response pair of two or more packets. These two exchanges are *roughly* equivalent to Phase I negotiations in IKEv1, but this is really more like an analogy — there are no “Phases” in IKEv2.

IKEv2 Phase 1 is approximately equivalent to IKE\_SA\_INIT and IKE\_AUTH exchanges together.

Normally, there is a single IKE\_SA\_INIT exchange and a single IKE\_AUTH exchange (a total of four messages) to establish both the Phase I SA and an initial Phase II SA. But in IKEv2, there is no “Phase I SA”, it’s called an “IKE SA”, and there is no “Phase II SA”; this is called a “Child SA” or sometimes called an “IPsec SA.”

IKEv2 Phase 1 SA is approximately equal to an IKE SA and IKEv2 Phase II SA is approximately equivalent to a Child SA (sometimes called an “IPsec SA”)





The first pair of messages (**IKE\_SA\_INIT**) negotiate cryptographic algorithms and do a Diffie-Hellman (DH) exchange. DH is always used with IKEv2, unlike AuthIP. The second exchange (**IKE\_AUTH**) authenticates the previous messages, exchanges identities, and establishes the first Child SA. The IKE\_SA\_INIT exchange must come first and must be successful before the IKE\_AUTH exchange can begin. The DH-derived key material is used to encrypt some of the data in the IKE\_AUTH exchange. Again, this is somewhat similar to Phase 1 with IKEv1. (Note an attacker sniffing packets can still see the identity of the initiator; this is sent in plaintext. The identity of the responder, though, is encrypted.)

There is a third type of IKEv2 exchange named “**CREATE\_CHILD\_SA**”. This exchange is used to create new Child SAs and to generate new encryption and signing keys for both IKE SAs and Child SAs. The CREATE\_CHILD\_SA exchange consists of a single request/response pair. Some of what CREATE\_CHILD\_SA exchanges do roughly match what were Phase II negotiations in IKEv1, but they are not really the same.

IKEv2 Phase II is approximately equal to CREATE\_CHILD\_SA exchanges (kind of)

Finally, there is a fourth kind of IKEv2 exchange called the “**INFORMATIONAL**” exchange. INFORMATIONAL exchanges delete IKE SAs and Child SAs when they are no longer needed, deal with errors, and do other IPsec housekeeping chores.

Overall, an IKEv2 IPsec session might look like this in a packet sniffer:

1. IKE\_SA\_INIT exchange to do Diffie-Hellman and set encryption details.
2. IKE\_AUTH exchange to authenticate the peers, create the IKE SA, and create the first Child SA using ESP or AH to protect user traffic.
3. Send and receive packets secured with ESP and/or AH.
4. CREATE\_CHILD\_SA exchanges as needed to create new Child SAs and generate new keys for both IKE SAs and Child SAs. For a very short IPsec session, there might be no CREATE\_CHILD exchanges at all.
5. Send and receive more packets secured with ESP and/or AH.
6. CREATE\_CHILD\_SA exchanges as needed to create new Child SAs and generate new keys for both IKE SAs and Child SAs. For a very short IPsec session, there might be no CREATE\_CHILD exchanges at all.
7. Send and receive more packets secured with ESP and/or AH.
8. INFORMATIONAL exchanges to do a graceful teardown of the SAs, similar to FIN-ACKs with TCP, to tidy up the in-memory SA databases of the peers.

Part of the IKE\_SA\_INIT exchange is when the initiating peer sends one or more proposals to the other peer, which responds by selecting either zero or just one of these proposals. A proposal usually includes things like cipher, key size, hashing algorithm, and other similar details. If no proposals from the initiator are acceptable, the responder selects zero of them and the SA fails. If multiple proposals from the initiator are acceptable, the responder must choose just one of them as is, no modifications allowed, and continue with the rest of the exchange(s) using that proposal.

Where does EAP authentication fit in? EAP authentication is not mandatory but can be made mandatory in IPsec policy rules. When additional authentication using EAP is required, it’s done by simply requiring more IKE\_AUTH exchanges than the one that is mandatory by default. Before the IKE SA may be created, the additional IKE\_AUTH exchanges must complete successfully too. And without a fully baked SA, the IPsec session cannot continue. Hence, there is always one IKE\_AUTH exchange, but there may be multiple. When EAP is not used, the IKE SA is authenticated with either a shared secret or public/private key pair. A shared secret is typically just a passphrase. A public/private key pair normally comes from a public key certificate and its associated private key.

**IKEv2 Machine Certificates**

For an IKEv2 certificate with the widest possible compatibility on Windows, like for both regular IPsec peers and for VPN clients and gateways, the certificate in the Local Computer certificate store should have the following characteristics:

**Application Policies:**

Client Authentication (1.3.6.1.5.5.7.3.2)

Server Authentication (1.3.6.1.5.5.7.3.1)

IP security IKE intermediate (1.3.6.1.5.5.8.2.2)

**Key Usage:**

Digital signature

Allow key exchange only with key encryption (key encipherment)

**Subject Name:**

Subject name format: Common Name (with FQDN)

Alternate Subject Name, DNS name (with FQDN)

These characteristics are defined either in the certificate template in Active Directory used by your Enterprise CA or manually defined in the original certificate request itself.

Strictly speaking, the Client Authentication EKU application policy is not required for IKEv2, but it is sometimes used for AuthIP too, so might as well add it. Overall, it’s best to create a new certificate template specifically for IPsec in AD. This template can also be used to issue certificates compatible with SSTP VPNs and other protocols too if you wish. In general, try to avoid having multiple IPsec-compatible certificates installed on each device makes troubleshooting more difficult because of the complexity of how Windows selects which certificate to use. Finally, note that client-side validation of the server certificate cannot be disabled; hence, make sure your OCSP web servers and domain controllers are accessible (controllers offer CRLs over LDAP).

If you make a change to your next IPsec certificate used for IKEv2 and new connections fail, try restarting the IKEEXT service. The IKEEXT service implements both AuthIP and IKEv2.

**Troubleshooting**

IPsec activity can be written to the Security event log on Windows Vista, Server 2008, and later operating systems. To enable the necessary audit policies, run this command:

*#Script: IPsec\_Firewall\_Audit\_Policies.ps1*

*Auditpol.exe /set /subcategory: “IPsec Main Mode, IPsec Quick Mode, IPsec Extended Mode, IPsec Driver” /success: enable /failure: enable*

Then restart the Windows Firewall service:

***Restart-Service -Name MPSSVC -Force***

You can also create trace files in PowerShell.exe (not ISE) like this:

*Netsh.exe trace start scenario = WFP-IPsec*

*#Peform the IPsec action which is failing or causing errors…*

*Netsh.exe trace stop*

The above commands will output the path to a NetTrace.cab file. Using 7-Zip or Windows File Explorer, extract all the files from the CAB to a new folder. These files contain information that can help troubleshoot the IPsec issue. Start with the report.html file. The CAB can also be sent to Microsoft for technical support. If the error is certificate-related, also try enabling CAPI2 Diagnostics logging in Event Viewer (open Event Viewer, then go to Applications and Services Logs > Microsoft > Windows > CAPI2 > Operational).

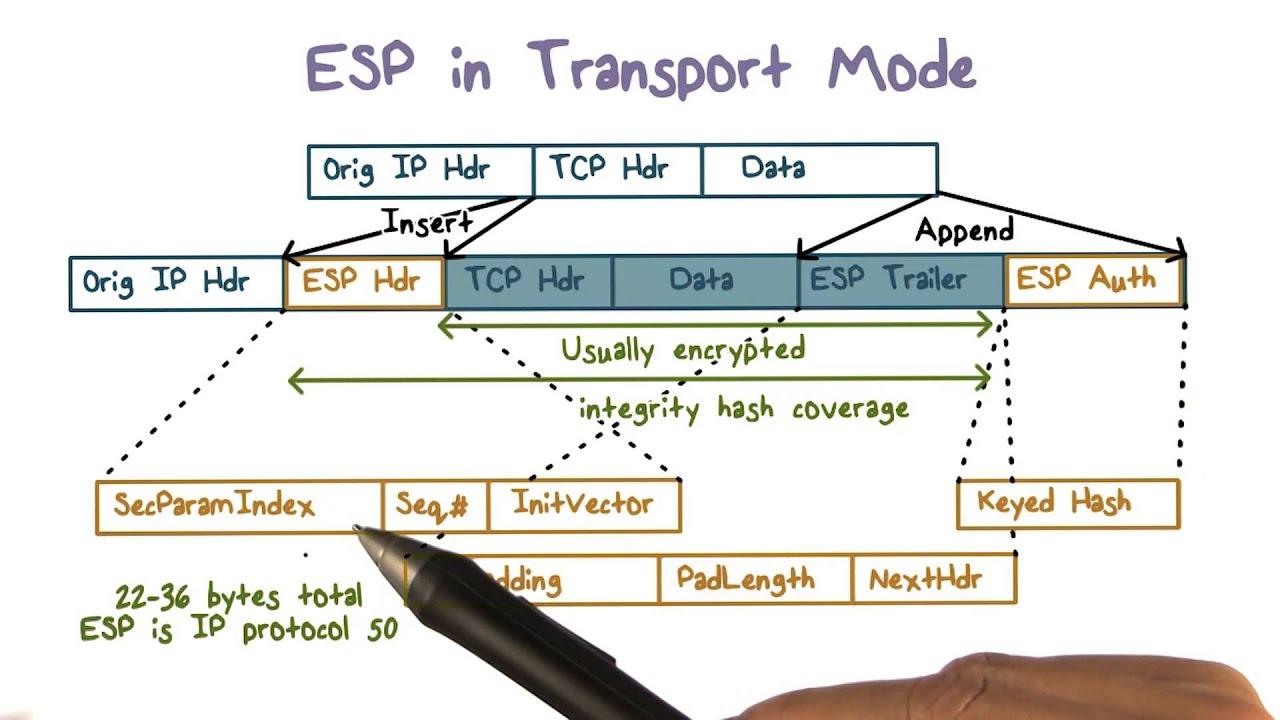
Finally, you can always sniff IPsec packets with Wireshark or Microsoft Message analyzer to hopefully uncover the culprit.

Encapsulating Security Payload (ESP)

Encapsulating Security Payload (ESP) is the core IPsec protocol that provides integrity, authentication, and encryption. It can operate in either “transport mode” or “tunnel mode”. The use of ESP is negotiated in IKE v1 Phase II Quick Mode, AuthIP Phase II Quick Mode, or in Child SAs after IKE\_AUTH and CREATE\_CHILD\_SA exchanges in IKEv2.

**ESP Transport Mode**

ESP uses both a header and a trailer. In transport mode, the ESP header is inserted just after the IP layer and before any transport layer protocols such as TCP or UDP. The ESP trailer has two parts, and both are appended to the very end.



The layers in gray are encrypted, perhaps with 168-bit 3DES in CBC mode or 256-bit AES. Encrypted layers include everything in between the ESP header and the second ESP trailer at the very end. The encryption key was generated when the Phase II SA or Child SA was established. The encryption key was generated when the Phase II SA or Child SA was established.

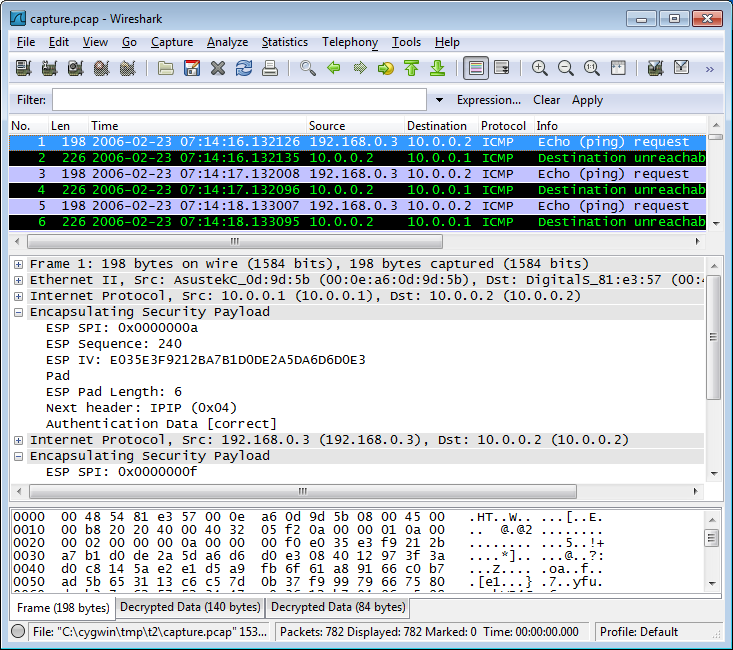
The second ESP trailer at the very end contains the Authentication Data field for the HMAC hash of the packet. Importantly, however, the scope of the authentication is smaller with ESP than in AH. In AH, the entire packet is authenticated and integrity-checked, including the front IP layer (but not the datalink layer). In ESP, by contrast, the IP header and the last ESP trailer are not authenticated or integrity-checked.

**ESP Tunnel Mode**

In tunnel mode, the entire original packet is placed after the ESP header and a new IP header is fabricated and placed in front of the ESP header. In tunnel mode, the entire original packet is both encrypted and authenticated. However, the new fabricated IP header is still not authenticated. AH can be used simultaneously with ESP in order to authenticate the front IP header.



ESP tunnel mode is commonly used for Virtual Private Networking (VPN). Windows uses L2TP with ESP in transport mode, not tunnel mode, but an IKEv2 VPN does use tunnel mode. ESP is protocol number 50. There are no port numbers associated with ESP or AH.



**Authentication Header (AH)**

Authentication Header (AH) is the IPsec protocol that provides integrity and authentication, but not encryption (RFC 2402). AH can operate in either “transport mode” or “tunnel mode.” But just ignore AH; it’s incompatible with Network Address Translation and unnecessary anyway; if you want to send plaintext packets that are digitally signed, just use ESP, and turn off the encryption.

**AH Transport Mode**

In *transport mode*, AH authenticates the entire packet at and above the IP layer, except for a few fields in the IP layer that change during transit, e.g., time to live, type of service (these fields are zeroed out during signature construction and verification). The AH layer itself comes directly after the IP layer and just before the TCP/UDP transport layer. Despite the AH header’s location in the middle of the packet, the entire packet is still authenticated.

| IP | AH | TCP/UDP | Payload Data |
| --- | --- | --- | --- |

AH can be combined with ESP. In this case, the ESP data is treated no differently than any other type of data, and the AH header again comes after IP and before ESP.

| IP | AH | ESP |
| --- | --- | --- |

The AH header contains the “Authentication Data” field. This field contains the authentication hash of the packet. The HMAC hash might be based on MD5, SHA-1, SHA-256, or some other algorithm, depending on the OS. The encryption key used in the Hashed Message Authentication Code (HMAC) is negotiated when the Phase II SA or Child SA is constructed. An *HMAC hash* can only be calculated and checked if both parties possess a shared secret key; this key is appended to the packet data before the packet + key is hashed.

Interestingly, the hash value produced, whether with HMAC-MD5 or HMAC-SHA1, is truncated to 96 bits (down from 128 bits with MD5 and down from 160 bits with SHA1). This weakens the authentication a bit, but also helps to thwart brute force searches for the shared secret key.

**AH Tunnel Mode**

In *tunnel mode*, the entire original packet is placed behind the AH header and a new IP header is fabricated and placed in front of the AH header. In this case, the entire original packet is authenticated as well as the non-changing fields of the new IP header. The new IP address is called the “tunnel endpoint”, but it is not necessarily the final destination of the original packet. The tunnel endpoint is likely to be an IPsec-enabled router.

| IP (new) | AH | IP (original) | TCP/UDP | Payload Data |
| --- | --- | --- | --- | --- |

Because of the lack of encryption, AH tunnel mode is rarely used. AH is protocol number 51. There is no port number associated with either AH or ESP.

**Summary of Differences Between AH and ESP**

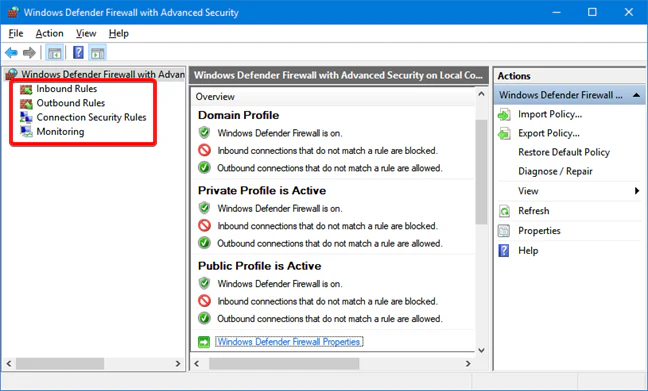
* AH cannot encrypt data, but ESP can.
* ESP packet encryption is optional.
* AH authenticates the entire packet, including the IP layer, while ESP only authenticates data above the IP layer.
* Both AH and ESP can be used in either transport or tunnel mode.
* AH and ESP can be combined in the same packet.
* AH cannot traverse NAT devices, but ESP can.

In general, avoid using AH; just use ESP all the time, either with or without encryption.

Connection Security Rules = IPsec Rules

**Define which traffic types trigger IPsec negotiations:**

* **Your IPsec driver sniffs all the packets you send or receive in the protocol stack.**
* **Packets are matched against the patterns in the IPsec rules.**
* **A matching rule will trigger an IKE negotiation process.**

****

The firewall rules allow/block packets, and sometimes packets must be secured with IPsec or else they’ll be blocked. But it’s the Connection Security Rules that specify which IP addresses, protocols and port numbers should trigger the negotiation of IPsec. In order to use IPsec, each peer must have a Connection Security Rule that indicates that certain packets exchanged with the other peer must use IPsec.

Remember, though, that a Connection Security Rule does not by itself allow traffic through the Windows Firewall. There must still be a firewall rule enabled to allow the traffic through *after* it has been decrypted or otherwise processed by IPsec. If a firewall rule permits a certain type of traffic but does not require IPsec for those packets, then that traffic is permitted whether or not it is secured with IPsec. If a firewall rule permits a certain type of traffic but only if it is secured with IPsec, then there must be a Connection Security Rule that successfully negotiates IPsec before the firewall examines that traffic.

An inbound or outbound firewall rule that requires a secure connection will not by itself, in the absence of Connection Security Rule, trigger an IPsec negotiation between the peers. At least one Connection Security Rule must be configured on each peer before IPsec will function on that peer.

**Creating Connection Security Rules**

There is a built-in wizard to help you create Connection Security Rules. The wizard will ask you why you are creating the rule and then will prompt you for the necessary information. However, once a rule is created, you can always go back to its properties to edit its settings, and these property sheet tabs are the same no matter what type of rule you create with the wizard; hence, let’s focus on editing the rule after it has been created.

***Try It Now!***

To create a Connection Security Rule, open the Windows Firewall snap-in > right click Connection Security Rules > New Rule > follow the wizard’s questions.

The default IPsec options in the properties of the Windows Firewall snap-in will be used by Connection Security Rules when those rules do not specify those options explicitly, i.e., when an option in a Connection Security Rule is set to “Default”, the value is taken from the options configured on the IPsec tab of the properties of the Windows Firewall snap-in. To see what options are actually being used, open Windows Firewall Monitoring > Connection Security Rules. (Note: Sometimes when you click “Default” in the IPsec tab of the Windows Firewall properties, the change doesn’t stick after clicking OK, so you’ll need to choose a particular option in that tab instead.)

Let’s discuss each of the tabs in the property sheet of a Connection Security Rule. Simply right-click the rule and select Properties.

**Default Exemptions**

Some exemptions are built into the IPsec driver by default, which causes the driver to simply ignore certain types of traffic, such as broadcast and multicast; however, a registry value can be set (NoDefaultExempt) that changes this behavior (KB811832, KB810207). Windows 2000 requires SP1 or alter to enable this registry value and note that SP4 and later *changes* this to a value of 1. Windows Server 2003 and later support the value by default and it is set to 3 by default (create the value to change it to something else).

Hive: HKEY\_LOCAL\_MACHINE

Key: \SYSTEM\CurrentControlSet\Services\IPSEC

Value Name: NoDefaultExempt

Value Type: REG\_DWORD

Value Data: 0,1,2, or 3 (depending on operating system)

Windows 2000: 0 or 1 possible with SP1, 0 by default, 1 by default with SP4.

Windows XP: 0 or 1 possible, 0 by default, 1 by default with SP2.

2003 and Later: 0, 1, 2, or 3 possible, 3 by default.

Value 0: Multicast, broadcast, RSVP, Kerberos, and IKE are exempt.

Value 1: Multicast, broadcast, and IKE are exempt.

Value 2: RSVP, Kerberos,and IKE are exempt.

Value 3: Only IKE is exempt.

Note that Windows 2000/XP cannot be configured to *not* exempt broadcast or multicast traffic. And IKE can never be made not exempt too. If you wish to block IKE, it must be done with a firewall or some other filtering capability.

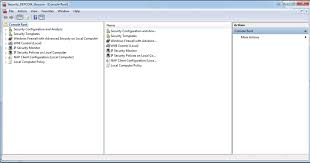
It is mandatory to set this value to 1 on 2000/XP and either 1 or 3 on Windows Server 2003 and later. This importance is highlighted by Microsoft’s decision to set these values automatically with Service Packs despite the interoperability problems that may arise. The problem is that attackers can send malicious packets with any source port they wish, including the well-known port for Kerberos (UDP/TCP 86). Any defender relying on IPsec packet filtering could be compromised with this trick (perhaps using FPIPE.EXE from Foundstone). It is also possible to send malicious broadcast packets; hence, the value should be set to 3 on Windows Server 2003 and later.

Once these more-secure values are set, keep in mind that your IPsec policies will have to take into account these once-exempted protocols when you wish to maintain functionality (KB254949, KB253169).

**Order of Filter Precedence**

The order of the Connection Security Rules in the dialog box does not matter. If any of the rules match a packet, then that rule will be triggered. However, a single IPsec policy can contain multiple Connection Security Rules, and each rule will have its own defined endpoints, profiles, and interface types. An inbound or outbound packet may match two different rules, and these rules may have conflicting actions. The ambiguity is resolved by the way the IPsec driver matches packets against rules. All the filters from all the rules in the policy are arranged in order from most specific to least specific when processed by the IPsec driver. (This is similar to the behavior of a router matching packets against its route table.) Each filter is assigned a weight value, and of the filters that match a given packet, the filter with the highest weight value (i.e., the most specific filter) is the one that will be matched.

On Windows 2000 through Server 2003, you can see all the filters being enforced by the IPsec driver using the IP Security Monitor MMC snap-in. In the snap-in, navigate to the Quick Mode > Specific Filters container and sort the lines on the Weight column in descending order. From top to bottom, this is the order in which packets are compared against loaded filters; the first filter that matches a packet is the one that wins,and the action associated with that filter is what determines what the IPsec driver will do with that packet.



Note that if a packet matches two filters because they both specify a range of IP addresses in which the packet falls, the range defined with the greater number of subnet mask bits is the filter that will win. Also, tunnel mode filters always take precedence over transport mode filters. However, it is still possible for two filters in two different rules to be equally specific. When this occurs, the filter associated with the more restrictive action is the filter (and rule) that takes precedence; for example, two rules with equally specific filters might select the same packet, but the rule that blocks that packet takes precedence over the rule that only encrypts it because blocking is more restrictive than encrypting.

In the course media files, the IPsec folder will contain a file named IPsec\_Order\_of\_Specificity.csv, which lists the relative weightings of the various criteria that a packet can match (highest weight at the top, smallest weight at the bottom). Double-click this file to open it in Excel.

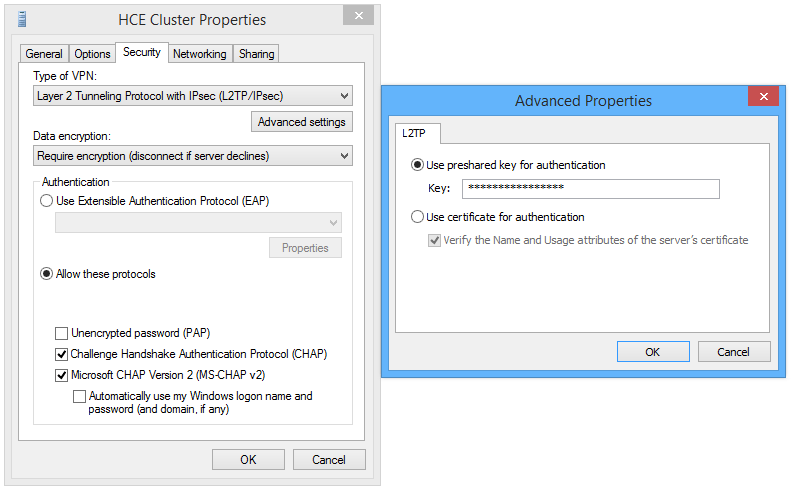
**IPSec Rule: Computers Tab**

**To which source and destination IP addresses will the IPsec rule apply?**

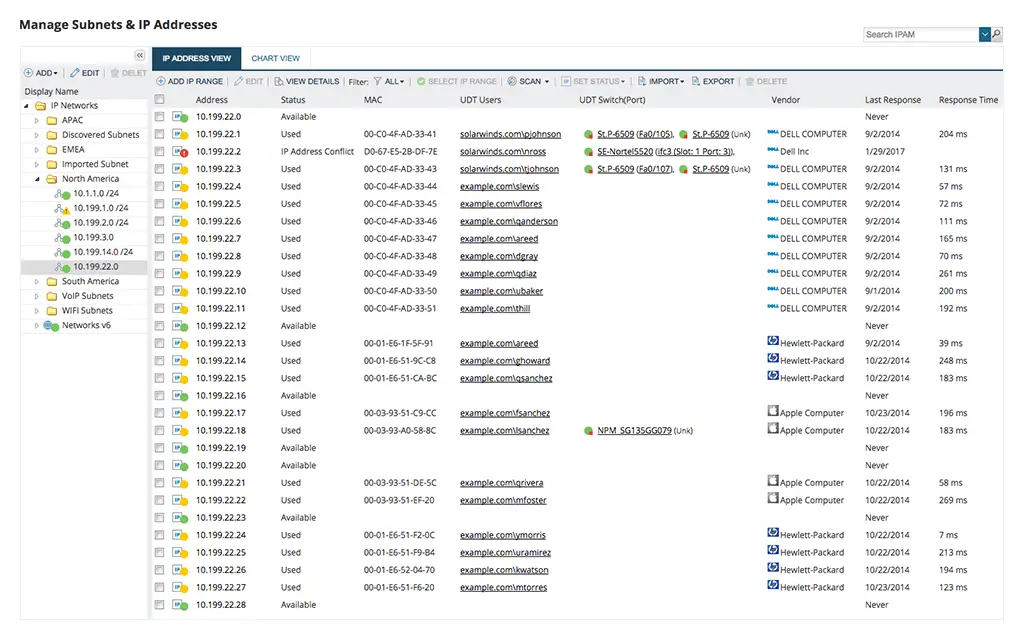
You can limit IPsec use to only the internal LAN, or to a list of individual IP addresses, or to just one single IP address.

IPsec Rule: Computers Tab

The Computers tab is for defining the endpoints of peers of the IPsec connection to which this rule applies. If a connection attempt matches the information on the Computers tab, then the Connection Security Rule is triggered for IPsec negotiations. The information on the other tabs does not apply or have any effect if the two endpoints under scrutiny do not match the endpoints defined on the Computers tab.



When you click Add or Edit, you can specify a single IP address or a set of IP addresses by range, subnet mask, or CIDR notation (IPv4 or IPv6). You can also specify IP addresses that may change dynamically and do not need to be hard-coded into the IPsec policy. In this case, you may specify endpoints being your DNS, WINS, or DHCP Servers, whatever they are at the moment, or your current local subnet, as defined by your current IP address and subnet mask. These dynamic options are especially nice for roaming tablets and laptops.



Why does the dialog box refer to “Endpoint 1” and “Endpoint 2” instead of source and destination? It’s because the IP addresses entered for the two endpoints are automatically duplicated and the duplicate copy is reversed for the sake of the IPsec driver. Hence, if the following IP addresses are configured in the dialog box:

**Endpoint 1:** Any

**Endpoint 2:** 10.1.0.0/16

Then what the IPsec driver actually enforces is the following:

**Source:** Any

**Destination:** 10.1.0.0/16

**-Or-**

**Source:** 10.1.0.0/16

**Destination:** Any

Network connections are usually bidirectional. Instead of requiring the IT administrator to enter every source and destination pairing twice, with the IP addresses and UDP/TCP port numbers reversed in the second copy, the IPsec driver does this automatically as a convenience.

IPsec Rule: Protocols and Ports Tab

The Protocols and Ports tab exists only on Windows 7, Server 2008-R2, and later. This tab is missing on Windows Vista, Server 2008, and earlier. On this tab, you can refine the IPsec rule so that it is triggered only by a specific protocol and port number combination. For example, the Server Message Block (SMB) protocol, also known as “File and Printer Sharing” protocol, uses TCP ports 139 and 445. An IPsec rule could apply to just SMB traffic and nothing else.

An IPsec rule is not like a light switch, in that, when turned on, everything must be secured with IPsec, and when turned off, nothing is secured with IPsec. We get to choose exactly which combination of source and destination IP address, protocol, and port number(s) will be secured with IPsec, and then everything else will flow like normal.

If multiple port numbers are entered, they must be in a comma-delimited list of individual numbers (such as “20,21,23,3389”). Defining a range of port numbers with a hyphen (such as “3000-9000”) is permitted, but only if the authentication mode is set to “Do not authenticate” on the rule’s Authentication tab. In other words, a range of port numbers is only accepted if the purpose of *this* IPsec rule is to exempt certain traffic from *another* IPsec rule. In real life, you will have multiple IPsec rules. Each rule will normally apply to different traffic flows, but if one rule is too broad or general, then another rule could carve out a subset of that rule’s covered traffic in order to treat it differently, perhaps to simply exempt that subset of traffic from IPsec consideration entirely.

**IPSec Rule: Authentication Tab**

**Kerberos**

* **Best for inside LAN**
* **Domain members**

**NTLMv2**

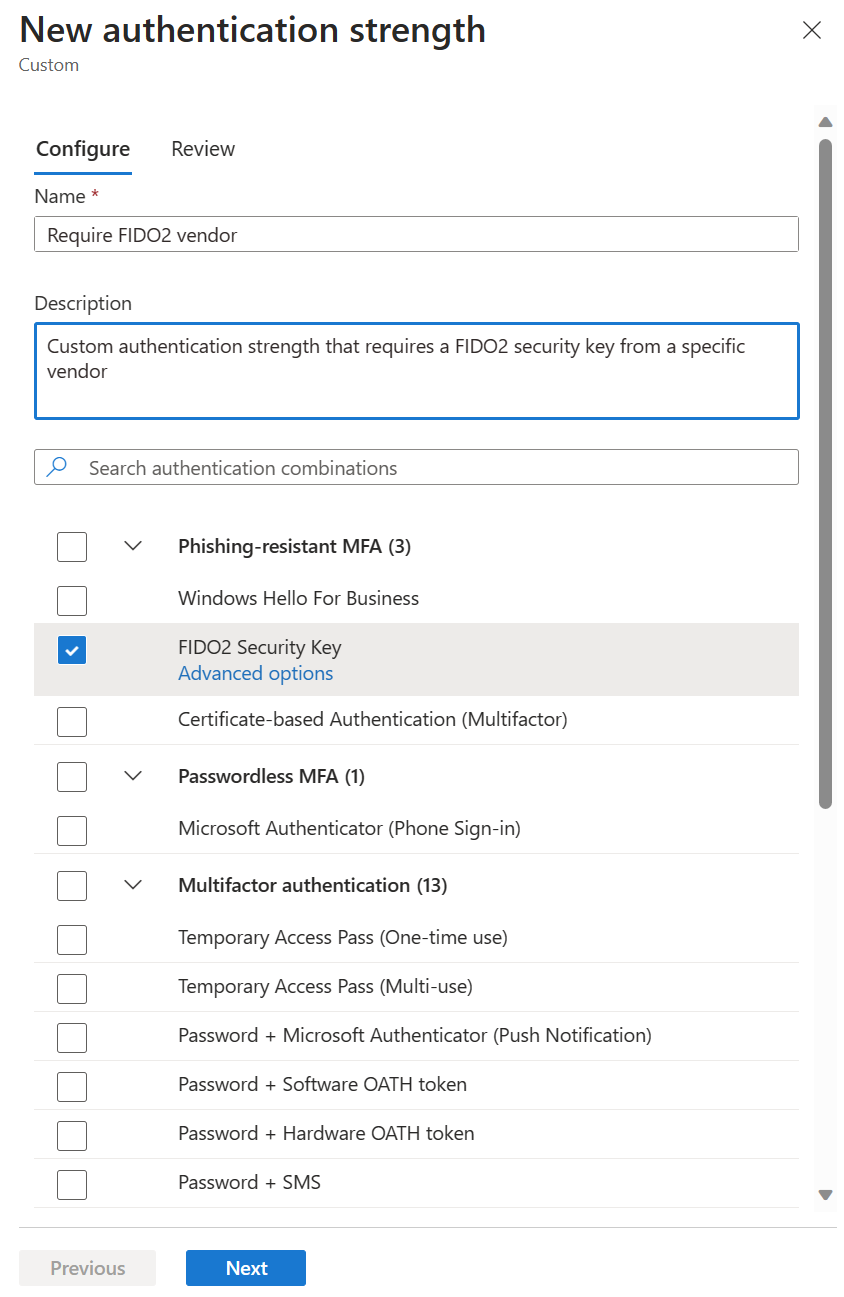
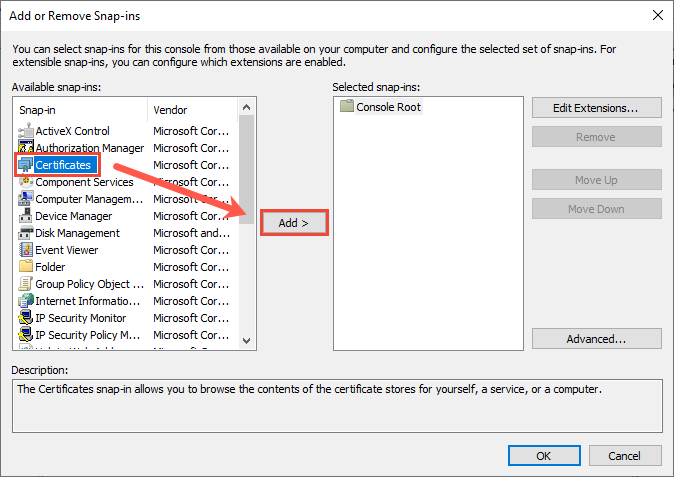
* **Standalone**

**Certificate**

* **PKI + Group Policy**

**Pre-shared Key**

* **Plaintext in the registry and GPOs!**

****

IPSec Rule: Authentication Tab

The authentication methods you choose controls how the computer and/or user authenticates to the other peer during Phase I negotiation using IKEv1 or AuthIP. Note that in Windows 2000/XP/2003, only the computer can be authenticated, not the user, but in Windows Vista and later, the user can authenticate via IPsec too because of AuthIP.



Multiple authentication methods can be enabled simultaneously. There are three authentication methods available in Windows 2000/XP/2003:

* Kerberos (Computer)
* Certificate (Computer)
* Preshared Key

In Windows Vista and later, you can also authenticate using:

* Kerberos (User)
* Certificate (User)
* Health Certificate (Computer – deprecated)
* NTLMv2 (User)
* NTLMv2 (Computer)

The authentication methods are attempted in the order shown on the property sheet, from top to bottom. The first method acceptable to both peers is used. If two peers do not have at least one common authentication method, the authentication will fail. Once a mutually agreed-upon method fails, the next method in the list is *not* attempted in times using the different authentication protocols listed. This is because Vista and later support the AuthIP enhancements to IKEv1.

In Windows 2000/XP/2003 with IKEv1, for example, if a remote peer is configured with both Kerberos and certificate authentication, *in that order*, and the server is configured with both Kerberos and mutual authentication *in that order*, and both sides have mutually acceptable certificates, then authentication will still fail because the remote peer using AuthIP, on the other hand, these systems can be configured to try a second time, and, in fact, if both computers are Vista or later, each peer can use a different authentication method (as long as it is supported by the other peer, even though it is not that peer’s first choice).

Note the option named “Do Not Authenticate”. Why is this an option? This is how you exempt certain endpoints from the necessity of using IPsec at all.

**Kerberos Authentication**

Every Windows computer in the domain has a computer account in Active Directory. The computer uses this account to authenticate to the domain when it starts up. This same account can be used with IPsec Kerberos authentication. In Windows Vista and later, the user can also use his/her account in AD for Kerberos authentication of IPsec.

Keep in mind that for Kerberos authentication to work, the IPsec peers must be in the same or trusted domains, and each peer must be able to contact a domain controller.

Hence, Kerberos authentication is not really appropriate for most remote access scenarios, unless IPsec is being used *through* the VPN, not *for* the VPN itself.

**Certificate Authentication**

Peers can be mutually authenticated with their machine certificates. To be successfully authenticated, the issuer of each computer’s certificate must be trusted by the other IPsec peer. It is not the case that an IPsec peer must have a copy of the other peer’s certificate beforehand or know any of the details of the certificate’s credentials. No one-to-one or many-to-one mapping occurs. Computer certificates are not mapped to computer accounts in Active Directory by default. The purpose of certificate authentication is, in part, to authenticate with peers in foreign domains or that can’t be domain members.

A Windows peer must have the certificate of the issuing Certification Authority (CA) of its own IPsec certificate and the other peer’s certificate in its personal Trusted Root CA Store. This is the “personal store” of the computer itself, not the human sitting at it. Use Group Policy to distribute trusted CA certificates automatically (as discussed in the PKI seminar).

A Windows computer can obtain an IPsec certificate through auto-enrollment from a Windows Enterprise CA. An alternative is to manually request and install the certificate with the Certificates snap-in or the Certificate Services Website. Manual installation requires administrative rights at the system.

Certificate authentication is appropriate when the other peer does not support Kerberos, domain controllers cannot be accessed for Kerberos authentication, the other host is not in a trusted domain, or L2TP is in use.

For the widest compatibility on Windows, for any IKE flavor (IKEv1, AuthIP, or IKEv2), for both regular IPsec peers and for VPN clients and gateways, the certificate in the Local Computer certificate store should have the following characteristics:

**Application Policies:**

Client Authentication (1.3.6.1.5.5.7.3.2)

Server Authentication (1.3.6.1.5.5.7.3.1)

IP security IKE intermediate (1.3.6.1.5.5.8.2.2)

**Key Usage:**

Digital usage

Allow key exchange only with key encryption (key encipherment)

**Subject Name:**

Subject name format: Common Name (with FQDN)

Alternate Subject Name: DNS name (with FQDN)

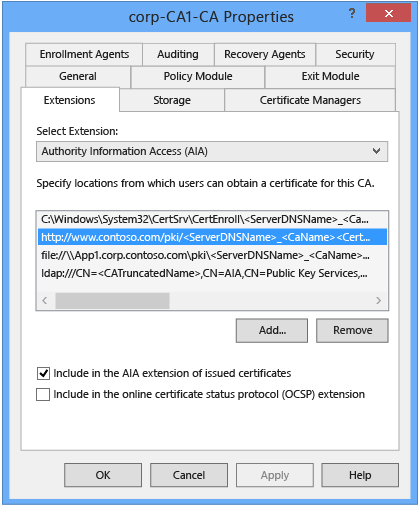
These characteristics are defined either in the certificate template in Active Directory used by your Enterprise CA, or manually defined in the original certificate request itself.

Strictly speaking, not all of the above certificate characteristics are required in every scenario, but save yourself the troubleshooting pain. Overall, it’s best to create a new certificate template specifically for IPsec in AD. This template can also be used to issue certificates for use with SSTP VPNs and other protocols too if you wish, if they happen to be compatible. In general, try to avoid having multiple IPsec-compatible certificates installed on each device, it makes troubleshooting more difficult because of the complexity of how Windows selects which certificate to use. Finally, note that client-side validation of the server certificate cannot be disabled; hence, make sure your OCSP web servers and domain controllers are accessible (controllers offer CRLs over LDAP).

**Tip:** If you make a change to your IPsec certificate used for IKEv2 and new connections fail, try restarting the IKEEXT service. The IKEEXT service implements both AuthIP and IKEv2.

**Advanced Certificate Criteria**

On Windows 8, Server 2012, and later, there is an Advanced button next to the certificate authentication options when configuring IPsec authentication. The following is the dialog box shown when clicking the Advanced button:



This allows us to reject all certificate authentications that do not meet the criteria in the dialog box, such as for Enhanced Key Usage (EKU) settings in the certificates, subject name fields, or an exact certificate hash value.

Using these filtering criteria, for example, would permit rejecting all IPsec connections from any computer or user not in a particular Organizational Unit (OU); hence, group memberships are not the only way to isolate IPsec groups. These criteria might also be used with a custom certificate template used only for IPsec, and perhaps even a subordinate CA that issues only IPsec certificates.

**Certificate Revocation Checking**

When using IKEv1 or AuthIP, the IPsec driver does not perform certificate revocation checking by default. A certificate is “revoked” if it is on the list of revoked certificates its issuing CA publishes. A certificate might be revoked if its private key (or the private key of the CA) have been compromised. Hence, for maximum security, you should enable certificate revocation checking by setting the following registry value, but beware of performance penalties and hassles caused by configuration mistakes:

Hive: HKEY\_LOCAL\_MACHINE

Key: \SYSTEM\CurrentControlSet\Services\PolicyAgent\Oakley

Value Name: StrongCrlCheck

Value Data: 1 or 2 (select “Hex” as the Radix)

1: Normal CRL checking in which the check fails only if a CRL is successfully obtained and the certificate is on it.

2: Strong CRL checking in which any error whatsoever in the download or processing of the CRL returns a failure, including finding that the certificate is on the CRL.

A certificate will include a CRL Distribution Point (CDP) field, which lists the URLs where the certificate’s CRL can be obtained. The IPsec peer performing the revocation checking must be able to access one or more of these URLs.

When using IKEv2, on the other hand, revocation checking is enabled by default.

**Health Certificates and NAP: Deprecated**

IPsec supports certificate-based authentication. A special type of certificate is called a “health certificate.” A health certificate is issued to a computer only after that computer has passed a variety of security health tests, e.g., confirmation of a running virus scanner, a personal firewall is engaged, recent patches applied, etc. Health certificates are a part of the much larger system of servers, protocols, and procedures that comprise Microsoft’s Network Access Protection (NAP) scheme, but NAP was deprecated; hence, just ignore any references to health certificates for IPsec.

**Preshared Key Authentication**

The “pre-shared key” is actually just a passphrase. The passphrase bits are mixed with and protected by the master key derived in Phase 1. However, the passphrase itself is stored unencrypted in the IPsec Policy in both Active Directory and in the local registry.

Prior to Vista, the registry stores the passphrase in a value named “ipsecData” under HKLM\SOFTWARE\Policies\Microsoft\Windows\IPsec\Policy\Local\ipsecNFA{*xxxxxx-xx-xxxx-xxxx-xxxxxxxx*}, where the *“xxx”* number is the GUID for the IPsec policy object itself. The permissions on these keys allow the local Users group read access.

With Vista and later, the pre-shared keys are found under HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\Phase1AuthenticationSets

**Important**: If an adversary obtains a certificate or knows the pre-shared key, this only permits the adversary to open her own IPsec channel to your peer. The adversary *cannot* decrypt other captured sessions with this information because the Diffie-Hellman-Merkle exchange with each session is unique.

The following PowerShell command will extract all local pre-shared keys:

*Get-NetIPsecPhase1AuthSet | Select-Object -ExpandProperty Proposal | Where { $\_.AuthenticationMethod -eq ‘PreSharedKey’ } | Select-Object -ExpandProperty PreSharedKey*

Support for pre-shared key authentication is mainly intended for RFC compliance, lab testing, or when security is not an issue (KB240262). Kerberos or certificate authentication should be used on production servers instead.

The exception, ironically, is when responding to the threat of quantum computing. A random pre-shared key that is over 100 characters in length can help to withstand brute force attacks against IKEv1 by adversaries using quantum computers. However, that being the case, if an adversary can simply steal the pre-shared key from a GPO or the registry of a previously compromised machine, then the quantum computing threat is irrelevant. For most organizations, avoiding pre-shared keys is still the better overall policy, at least for the time being.

**Authenticated IP (AuthIP) in Windows Vista and Later**

Authenticated IP (AuthIP) is an enhanced version of Internet Key Exchange (IKEv1) for IPsec. AuthIP has the following benefits over regular IKEv1:

* Authentication of the user, instead of the user’s computer, using Kerberos, NTLMv2, a user certificate, or a Network Access Protection health certificate.
* The user authentication can be performed alone or after the user’s computer first authenticates itself to the target (dual authentication).
* Multiple authentication attempts with different authentication protocols.
* A different authentication protocol can be used by each IPsec peer when negotiating a session, e.g., a client might use Kerberos and the server might use a certificate.

Only Windows Vista and later supports AuthIP. When communicating with Windows Server 2003 and earlier peers, IPsec on Windows Vista and later will automatically downgrade to IKEv1 for backward compatibility.

Optional vs. Mandatory Authentication

If you mark both authentication choices as “optional”, authentication will no longer be required. This would permit untrusted components to open their own IPsec channels to your machines and makes man-in-the-middle attacks more likely. Never choose this option unless you are forced to do so by your planning constraints.

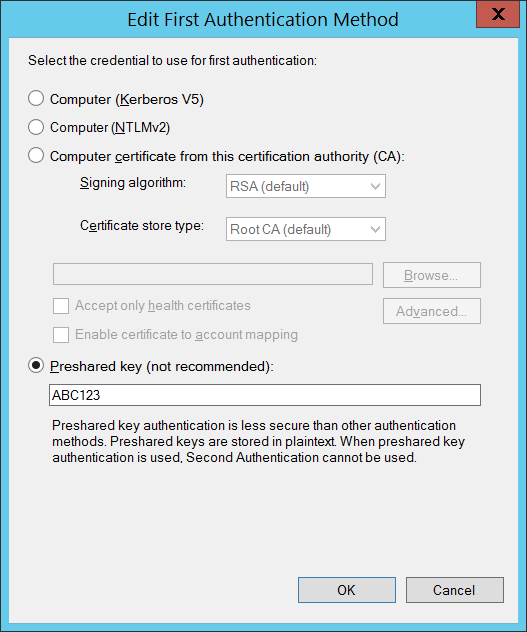
If neither authentication choice is marked as “optional”, then both authentication events are mandatory. You can use this feature, for example, to first force authentication of the peer’s computer using a certificate, then force the user sitting at that computer to authenticate with Kerberos. If either the remote computer or user lacks the Log On Over The Network right, the IPsec channel can be blocked.

The first authentication method might safely be marked as “optional” if you intend to rely solely on the second method. This would be done, for example, when you wish to only authenticate users, not computers, because IPsec user authentication can only be enabled in the second authentication attempt, not the first one.

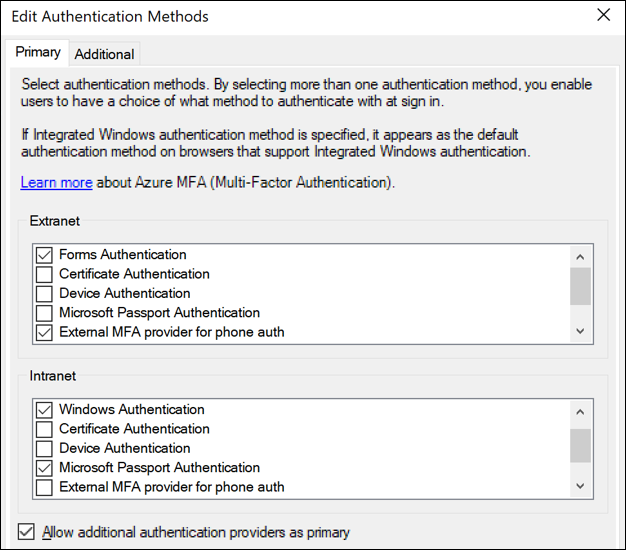
If you select a user-based authentication method in the second choice, any other methods on the second-choice list must be user-based as well. If you choose a computer-based method in the second-choice list, all the other methods in that list must be computer-based too. In short, you cannot mix user- and computer- based authentication methods in the second-choice list — it is all of one or the other.

If you select pre-shared key authentication as the first method, you cannot have a second authentication attempt of any type. This is, in part, to maintain backward compatibility with IKEv1. Using a pre-shared key disables the use of AuthIP.

*The following screenshot shows the first authentication method options.*

**

The next screenshot shows the second authentication method choices, but remember that if you choose pre-shared key authentication as the first choice, you cannot add a second choice. Note that only the second authentication choices include user-based authentication methods.



In Windows 7/2008-R2 and later, you can specify the signing algorithm used with certificate-based authentication. RSA is the default, very widely used, and backward compatible; while the Elliptic Curve Digital Signature Algorithm (ECDSA) is newer, faster, more secure, but not as widely used or backward compatible. Only Windows 7/2008-R2 and later support ECDSA for IPsec. Because the foregoing pages describe the default IPsec settings these will be the settings used whenever a firewall rule is marked to require a secure connection. But what if you need to use something other than the defaults on a particular server or OU of laptops? This is what Connection Security Rules are for, namely, to use different settings than the defaults for particular IP addresses, protocols or ports.

**IPsec Chicken-and-Egg Problems**

Configuring the authentication settings is the most difficult part of managing IPsec. One reason this is so is because if you require IPsec to access the TCP/UDP ports necessary to perform authentication, you’ll never get to the ports, which means you’ll never establish an IPsec connection, and so on an in infinite loop. If you require IPsec to access the Kerberos ports (TCP/UDP 88) and IPsec itself requires Kerberos authentication, how will you ever get to the Kerberos ports? Hence, you either cannot require IPsec to access the Kerberos ports or you’ll have to use a different authentication method (pre-shared key, NTLM or certificates) to access the Kerberos ports with IPsec.

There is another IPsec authentication issue too. When you use a Windows Firewall rule to restrict access to a listening port based on the group memberships of the user or computer accessing that port, how is the group memberships information conveyed to the target computer? It turns out that the target computer needs to query a domain controller for this information using various protocols; hence, the domain controllers must be configured to allow these inbound protocols and the target computer needs to be configured to allow these outbound protocols to the controllers. If you want to use IPsec with these protocols to/from the controllers, then we have to worry about the chicken-and-egg problem again; for example, you cannot restrict by group memberships access to the ports on the domain controllers for querying group memberships. This is driven by AuthIP and its reliance on the GSS-API authentication services provided by Windows.

Here are the ports that must be accessible on domain controllers in order to allow the querying of group memberships when using IPsec and firewall rules to restrict access:

DNS: UDP and TCP 53

Kerberos: UDP and TCP 88

RPC: TCP 135, 49152-65535

LDAP: UDP and TCP 389

Remote Procedure Call (RPC) networking uses upper-level port numbers, which are dynamically assigned. You can see what range of ports are used for RPC on your computer by running this command:

***Netsh.exe int ipv4 show dynamicportrange tcp***

And don’t forget the ports or protocols used by IPsec itself:

IKE: UDP 500 and 4500

AH: Protocol ID 51

ESP: Protocol ID 50

Finally, at your perimeter firewall, do not block fragmented IP packets destined for UDP 500/4500 when using IKEv2 on Windows. Windows IKEv2 often fragments UDP packets carrying certificates. If your firewall blocks fragmented IKEv2 traffic, the IPsec session will fail.

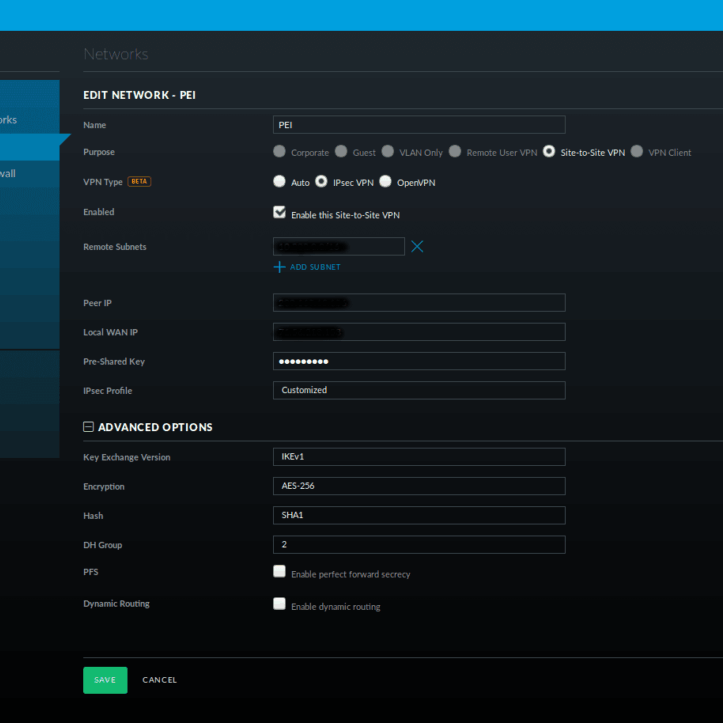
IPsec Rule: Advanced Tab

Similar to a firewall rule, the Advanced tab has options for restricting the IPsec rule to only certain network profile types (Domain, Private, Public) or restricting the IPsec rule to only certain interface types (LAN, VPN, Wireless). For example, an IPsec rule might only apply to SMB traffic traversing Domain Wireless interfaces for the company’s internal IP address range; in this case, you do not want the IPsec rule to apply to SMB traffic going over the Ethernet LAN or when the user is at home.

**IPSec Tunneling**

IPsec tunneling is very similar to Virtual Private Networking (VPN). In fact, many organizations use IPsec tunnels as their VPNs and there is a widespread misconception that “IPsec tunnel” and “IPsec VPN” are synonymous terms. Strictly speaking, though, an IPsec VPN does not have to use IPsec in tunnel mode (think of L2TP IPsec VPNs) and you can use IPsec in tunnel mode between two endpoints in a way that few people would consider to be a VPN. If you do wish to set up a VPN with Windows, either use IKEv2, L2TP with IPsec in transport mode, or SSTP (avoid using PPTP). As discussed earlier, IPsec can either operate in transport mode or tunnel mode. In tunnel mode, the original packet is encapsulated behind a new IP header (among other changes) and the new front IP header does not have to have the same source and destination IP addresses as the original header (now encapsulated inside the new IPsec packet). In fact, the IP addresses are usually different. Usually the tunnel endpoints are gateway devices, and these gateways might be using IPv6 internally and IPv4 on their internet interfaces.

Since the tunnel endpoint IP addresses are normally different from the IP addresses of the hosts communicating, what are the IP addresses of the endpoints? On the Advanced tab, click the Tunneling button and enter these IP addresses. These will be the IP addresses of the outermost IP header created for the sake of IPsec.



When would you use such a feature instead of a full VPN? Rarely. You might have a non-Windows IPsec gateway or server that doesn’t support L2TP, but normally you’ll either use regular transport mode IPsec or a true VPN like L2TP, SSTP, or IKEv2 with an EAP authentication of the user.

The purpose of the “Apply authorization” checkbox is to limit which computers and users are permitted to tunnel through the present device. These computers and users are configured by going to the properties of the Windows Firewall snap-in > IPsec Settings tab > IPsec Tunnel Authorization section > Advanced > Customize button.

The “Exempt IPsec protected connections” checkbox will cause the computer to not pump any packets through the tunnel that have already been secured with IPsec by virtue of a different Connection Security Rule. If you want all matching packets to go through the tunnel, don’t check this box.

**What Is Being Negotiated? What Are the Default IKE Settings?**

**Right-click the Firewall snap-in > Properties > IPsec Settings > Customize button.**

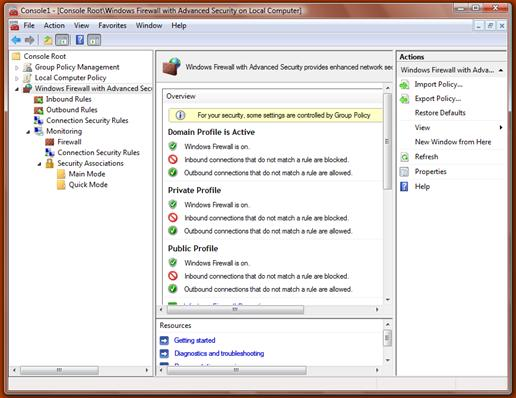
**You do not have to change the default IPsec settings!**

**Default settings are documented in today’s manual for reference.**

**IPsec defaults can be overridden with Connection Security Rules.**

What Is Being Negotiated? What Are the Defaults?

When a firewall rule or connection security rule uses IPsec, but an IPsec option is set to “Use Default (Recommended)”, what is this default and how can the defaults be changed?



The default settings come from the IPsec Settings tab on the property sheet of the WFAS snap-in itself (right-click WFAS snap-in > Properties > IPsec Setting tab > Customize button). The settings tab also allows you to exempt ICMP traffic from IPsec rules. It doesn’t say so, but the default settings are all for IKEv1 and AuthIP, not IKEv2.

When you click the Customize button, you’ll see the dialog box shown in the slide. The defaults are as follows:

**Phase I (Main Mode) Defaults:**

Diffie-Hellman-Merkle: Group 2 (1024-bit prime)

Encryption: 128-bit AES is tried first (primary), then 168-bit 3DES (secondary)

Hashing: SHA-1

Key Lifetime (Minutes): 480 Minutes

Key Lifetime (Sessions): 0 Sessions (hence, only the minutes decides).

**Phase II (Quick Mode) Defaults for “Data Integrity” Only:**

Protocol: ESP is tried first (primary), then AH is tried (secondary)

Encryption: Disabled for ESP

Hashing: SHA-1

Key Lifetimes: 60 Minutes or 1000000KB, whichever comes first.

Authentication: Kerberos (Computer Only)

**Phase II (Quick Mode) Defaults for “Data Integrity with Encryption”:**

Protocol: ESP

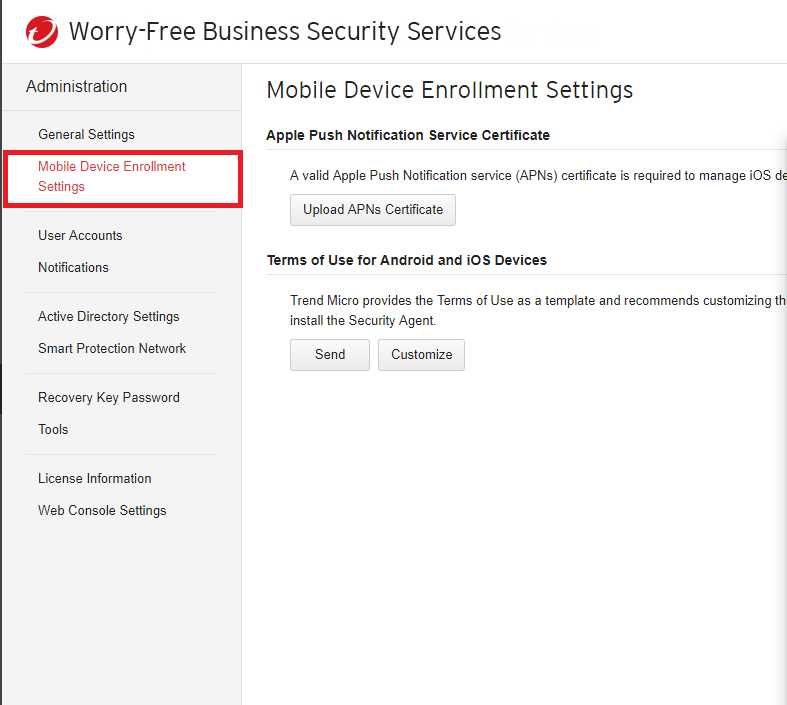
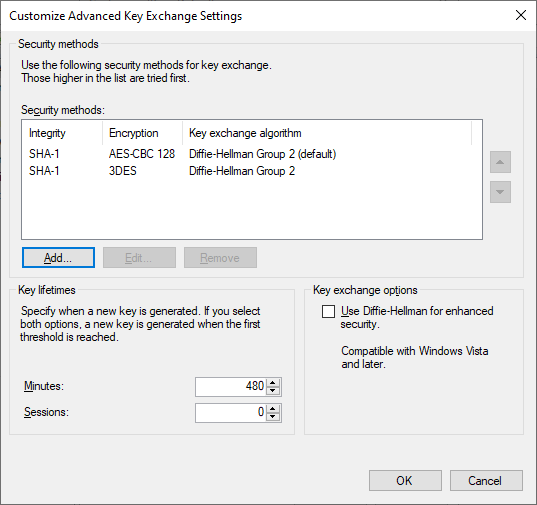
Encryption: 128-bit AES is tried first (primary), then 168-bit 3DES (secondary)

Hashing: SHA-1

Key Lifetimes: 60 Minutes or 1000000KB, whichever comes first.

Authentication: Kerberos (Computer Only)

Note that SHA-1 is not recommended when an SHA-2 algorithm can be used instead. SHA-2 is supported on Windows XP-SP3, Server 2003-SP2, Vista, Server 2008, and later operating systems. SHA-1 is supported for backward compatibility only.



Phase I: Main Mode Advanced Options

The Phase I master key secures the exchange of keys negotiated in Phase II for IKEv1 and AuthIP. The master key is ultimately protected using the Diffie-Hellman exchange that occurs in Phase I.

**DH Group**

The most important option here is the Diffie-Hellman key exchange method. A DH “group” determines the algorithm and values used by the IPsec peers to exchange encryption keys. Each group is assigned a number by IANA. There are several Modular Exponential (MODP) and Elliptic Curve (EC) group members.

The DH group used determines how resistant to attack the exchange will be. Some DH groups are obsolete, but are available for RFC compliance or backward compatibility.

| Description | Group Number | Recommendation |
| --- | --- | --- |
| Diffie-Hellman Group 1 | 1 | Avoid |
| Diffie-Hellman Group 2 | 2 | Avoid |
| Diffie-Hellman Group 14 | 14 | Prefer |
| Diffie-Hellman Group 24 | 24 | Avoid |
| Elliptic Curve DH P-256 | 19 | Acceptable |
| Elliptic Curve DH P-384 | 20 | Acceptable |

The above recommendations come from RFC 8247 and <https://safecurves.cr.yp.to>.

Group I uses a 768-bit MODP prime number and is totally obsolete. It can be broken by off-the-shelf hardware and provides no meaningful security. Always avoid it.

Group II uses a 1024-bit MODP prime and is obsolete. It can be broken by even moderately funded adversaries with their own hardware or by renting a cluster of VMs from a cloud computing vendor.

Group 14 uses a 2048-bit MODP prime and is still considered secure, except perhaps against very large nation states and near-future quantum computing attacks. It is widely implemented across platforms. When in doubt, use Group 14 for both security and compatibility reasons.

Group 24 uses a 2048-bit MODP prime with a 256-bit prime order subgroup, but it should be avoided because the cryptographic seed for this group has not been published and academic research indicates that a similar group (Group 22) is flawed. Even though the DH group number is larger, that does not make it more secure. Group numbers are just IANA reference numbers, not key-bit sizes.

Elliptic Curve DH is newer and can be more secure than the original DH even though the listed bit sizes are smaller. However, the details are important and not all the existing Elliptic Curve DH flavors are available in Windows. Note that Elliptic Curve DH is only supported on Windows Vista, Server 2008, and later.

Group 19 is Elliptic Curve Group Modulo a Prime (ECP) using a 256-bit field and is considered secure, except against future possible quantum computing attacks. However, it is not on the list of “safe” curves at <https://safecurves.cr.yp.to>. Nonetheless, it is considered acceptable in RFC8247.

Group 20 is Elliptic Curve Modulo a Prime (ECP) using a 384-bit field and is considered secure, except against future possible quantum computing attacks. However, it is not on the list of “safe” curves at <https://safecurves.cr.yp.to>. RFC 8247 does not mention this group at all.

**DH Choices in Windows IPsec**

However, the above list is limited in comparison to what is available with other IPsec implementations, such as strongSwan ([strongswan.org](http://strongswan.org)) and Libreswan ([libreswan.org](http://libreswan.org)). Microsoft has simply not kept up with developments in cryptography, current attack trends, and near-future potential attacks, such as from quantum computing. For example, Microsoft’s EC curve options, though they should be adequate for 99% of one’s attackers, are not the most secure or efficient available. Other EC options have been available for years, so why hasn’t Microsoft implemented them? Or if they can only be implemented with PowerShell, why are they not exposed in the graphical tools?

**Use Diffie-Hellman for Enhanced Security**

Windows 7/2008-R2 and later include a checkbox to “Use Diffie-Hellman for enhanced security”. If this box is checked, DH is always used to derive keys instead of any other alternative method, such as when using AuthIP. AuthIP is an enhanced version of IKEv1 first introduced with Vista. Its use is negotiated automatically, with fallback to IKEv1 as necessary. When AuthIP is used, then a DH key exchange is usually not performed. Avoiding DH when possible is done to maximize performance. When AuthIP uses Kerberos, NTLM, or certificate authentication, then the authentication protocol is used to derive the keying material normally provided by DH. Kerberos, in particular, is faster than DH, and we’re totally dependent on the security of Kerberos anyway in an Active Directory environment.

Because computer account passphrases are random, 100+ characters in length, and changed automatically every 30 days, and because Kerberos can be configured to only accept AES-256-bit encryption or better, then using *computer* Kerberos instead of DH is a reasonable trade-off. Indeed, because DH and public key ciphers may become vulnerable to quantum computing attacks in the near future, an argument can be made that using a Kerberos-derived key for IPsec is potentially more secure. (With *user* Kerberos authentication, on the other hand, the story is likely to be different because users often choose short, weak passwords instead of long, random passphrases; the major exception being when a user is required to have a smart card for interactive logons.)

However, if your organization is under regulatory restrictions to always use DH, such as in a classified military network, then you may need to check the box to always use DH. If you want to maximize performance inside the LAN, use Kerberos with AuthIP to exchange IPsec keys; if you want to maximize security, check the box to always require Diffie-Hellman (and configure the peers to use Elliptic Curve DH too).

**Key TTL**

One of the golden rules of cryptography is to avoid using the same key for too long or to encrypt too much data. How long is too long? How much is too much? That depends on your adversaries, but in this dialog box you can specify your key lifetime in minutes and/or sessions. If you configure both, whichever maximum is hit first (minutes or sessions) will cause a rekeying and then both counters will be reset. Keep in mind that the keys referenced here are for securing the IKE channel itself, not all the gigabytes of data to follow. Those keys are negotiated in Phase II have their own lifetime parameters. What kind of key has a time to live here? If you edit one of the security methods on the left-hand side you can choose the cipher/key type as well as the hashing algorithm.

**Cipher and Key Size**

Never use 56-bit DES. DES is totally obsolete.

For backward compatibility, 3DES may still be used in a pinch, even though it will be significantly slower than AES. AES is supported in Windows Vista and later. AES is faster and more secure than 3DES. 256-bit AES is the modern standard.

“CBC” stands for Cipher Block Chaining, which is an implementation mode of AES and other ciphers. AES-CBC is considered secure, even if it is slower than AES used in Galois/Counter Mode (GCM). AES-GCM is also considered to be secure.

**Hashing Algorithm**

For the hashing algorithm, never choose MD5. MD5 is totally obsolete. Choose SHA-1 if you must for backward compatibility, but SHA-1 is obsolete.

SHA-2 class algorithms include SHA-256, SHA-384, and others. SHA-2 class algorithms are supported on Windows XP-SP3, Server 2003-SP2, and later operating systems. Use SHA-256 by default to balance performance and security. NSA recommends SHA-384 for high security environments.

**Perfect Forward Secrecy (PFS)**

Perfect Forward Secrecy (PFS) for IPsec means that a new DH exchange is performed when a new symmetric key needs to be created. PFS is disabled by default. PFS can either be enabled for either Phase I or Phase II keys independently using PowerShell or the NETSH.EXE command line tool. In Windows 2000/XP/2003, there was a GUI checkbox for PFS, but in Windows Vista and later this GUI element was removed for some unknown reason.

**Quantum Computing Resistance**

Sometime in the next 50 years there will be a “quantum apocalypse” for cryptography when quantum computing becomes widely available. The threat is real enough that in August of 2015 the National Security Agency (NSA) in the United States provided public guidance concerning quantum-resistant countermeasures.

The most important points from the announcement were 1) new quantum-resistant algorithms must be developed and deployed in the next several years, 2) increase the size of keys used today, and 3) pre-shared keys can provide quantum resistance if they are very large and truly random, such as for IPsec IKEv1. How large is “large” though?

Here are the recommended minimum key sizes from the NSA announcement while we wait for the better algorithms:

AES: 256 bits

RSA: 3072 bits

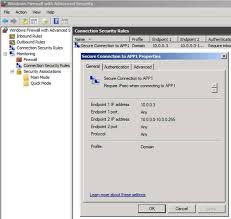
Hashing: SHA-384

Diffie-Hellman (DH) key exchange: 3072 bits

Elliptic Curve (ECDH) key exchange: curve P-384

Elliptic Curve (ECDSA) key signatures: curve P-384

Other IPsec implementations, such as strongSwan ([www.strongswan.org](http://www.strongswan.org)), already have a larger variety of ciphers and algorithms available for use. Hopefully Microsoft will catch up soon. Pre-shared keys with IPsec will be discussed in a few pages (below).



Phase II: Quick Mode Advanced Options

The left-hand side of this dialog box is for AH by itself or for ESP with the payload encryption disabled. The right-hand side is only for ESP with encryption enabled. These are the settings used by firewall rules and connection security rules that specify a “secure connection” (left side) or “secure connection with encryption” (right side). Again, all this is only for IKEv1 and AuthIP. IPsec connection rules that don’t specify their own Phase II settings will pull these settings from here. These are the system-wide defaults.

**Require Encryption Checkbox**

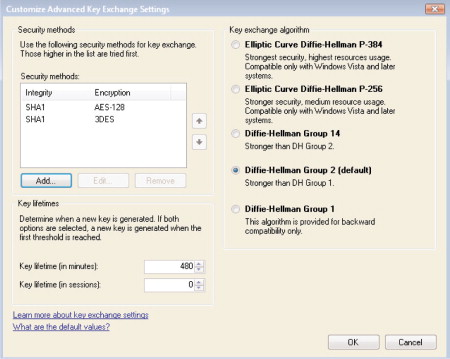
If you check the box at the top labeled “Require encryption for all connection security rules that use these settings”, then the left-hand side of the dialog box becomes greyed out: you can no longer use AH or ESP with the encryption disabled. For the sake of troubleshooting and avoiding unpleasant surprises, it’s probably best to check this box and ignore the plaintext IPsec options.

Remember, if an inbound firewall rule is configured to require IPsec encryption (General tab of the firewall rule > Customize button), but the other computer is configured to use IPsec with no encryption (either AH or with the ESP encryption disabled), then the connection will fail. Forget security, the troubleshooting advantages alone suggest always checking the “Require encryption…” checkbox on all computers.

**Plaintext Proposals (Left Side)**

If you edit one of the integrity-only methods on the left-hand side, you can choose the algorithm (AH or ESP with encryption disabled), the hashing algorithm and the key lifetimes.

Very importantly, remember that ESP can traverse devices like routers and firewalls that perform Network Address Translation (NAT) without the IPsec integrity checks failing. Because NAT is so widely used, ESP is preferred over AH in all cases, even when encryption for ESP is disabled. Hence, unless you have a specific reason for doing so, don’t use AH.



What about null encapsulation? That topic has a slide by itself.

**Encryption Proposals (Right Side)**

If you edit one of the methods on the right-hand side that use ESP payload encryption, you can choose whether to use ESP by itself or ESP + AH, the encryption algorithm, hashing method, and key lifetimes.

56-bit DES encryption is totally obsolete and should never be used.

3DES encryption may still be used for backward compatibility, but it is slower than AES. AES is the modern default standard and widely compatible.

AES Cipher Block Chaining (AES-CBC) is a mode of AES encryption in which each block of plaintext is XOR-ed with the previous block of ciphertext. AES-CBC is considered secure, but is slower than AES-GCM.

AES Galois Counter Mode (AES-GCM) is a mode of AES that is faster than CBC mode. AES-GCM is considered secure. Prefer GCM over GCBC mode when possible.

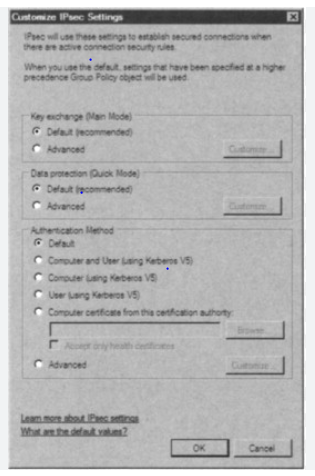
The fastest minimally acceptable proposal is 128-bit AES-GCM.

MD5 is a totally obsolete hashing algorithm and should never be used.

SHA-1 is obsolete and should only be used for backward compatibility.

AES Galois Message Authentication Code (AES-GMAC) is the use of AES-GCM for signing and integrity checking only, not bulk data encryption. AES-GMAC is more secure than SHA-1 and faster too. (Galois is pronounced “GOW-wah.”)

Note that AES-CBC requires Vista or later. AES-GCM and AES-GMAC each requires Vista+SP1 or later.



**Null Encapsulation**

**After the IKE negotiations, no further packets are encrypted or authenticated at all.**

* But the remote Windows Firewall knows who you are and can enforce TCP/UDP port permissions.
* Compatible with IDS/IPS sensors that are still confused by plaintext packets signed with IPsec headers.
* Requires Server 2008-R2, Windows 7, or later.

Null Encapsulation

Null encapsulation does not add an AH or ESP header to packets at all. Other than the IKE negotiations and some ESP heartbeat packets, the rest of the packets are sent normally without any modification whatsoever. This means that null encapsulation provides no encryption or integrity checking of data. So what’s the point then?

**Advantages**

The Windows Firewall can be configured to allow packets to a particular listening port from a particular source IP address, but only if these packers were sent by a peer that had first been authenticated with IKE. The successful IKE authentication is what triggers the firewall to allow access to the listening port, but only from the source IP address of the host that authenticated with IKE first (more specifically, using the AuthIP extension to IKE).

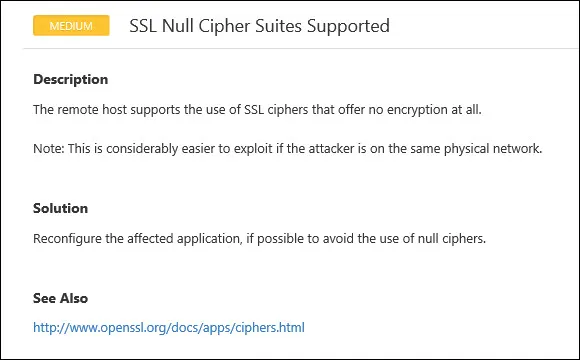
Another advantage is that null encapsulation is compatible with IDS/IPS sensors that are confused by AH or ESP headers. Even when packet payloads are in the clear, some IDS/IPS sensors just can’t handle looking past the AH/ESP header to examine the plaintext payload. While this is clearly the fault of these vendors’ IDS/IPS products, the issue is moot if your favorite IDS/IPS vendor won’t update their code. Null encapsulation provides a partial stopgap solution while we wait for our IDS/IPS vendors to fix their products.

**Disadvantages**

IPsec null encapsulation does not encrypt, authenticate, or integrity check any packets (other than IKE negotiation packets). Remember, “IPsec” and “encryption” are not synonyms, even when using ESP. After the IKE authentication, an attacker could spoof or modify packets from the host’s IP address (which is why ESP should be used), but at least the IKE authentication requirement places another hurdle in the way of the attacker.

After the IKE authentication, an attacker could spoof or modify packets from the host’s IP address (which is why ESP should be used), but at least the IKE authentication requirement places another hurdle in the way of the attacker. If the client host’s traffic is already encrypted and integrity checked, such as with SSL, then the spoofing and man-in-the-middle attack risks are reduced. Again, use ESP instead whenever possible.

Enabling Null Encapsulation comes with a pleasant warning:



**Requirements**

IPsec null encapsulation requires Windows Server 2008-R2, Windows 7, or later operating systems. Null encapsulation is incompatible with pre-shared key authentication. Null encapsulation must be used with transport mode IPsec; it is incompatible with tunnel mode. Transport mode is the default.

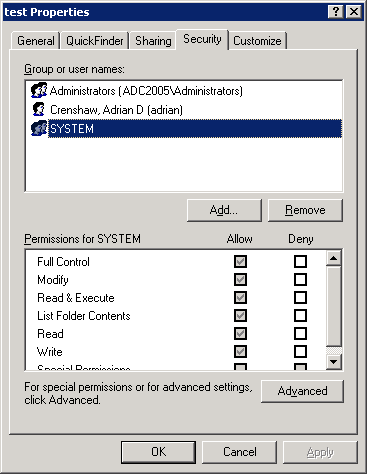
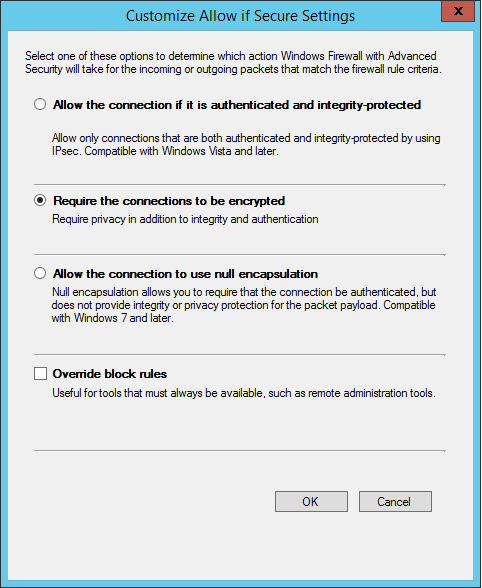
**Firewall Rule: General Tab > Customize Button**

**Recommendation:**

* Either always check this box
* Or configure the the default IPsec policy to never allow plaintext security associations

Firewall Rule: General Tab > Customize Button

How do firewall rules and IPsec rules relate to each other? In the properties of an inbound firewall rule, go to the General tab and click on the Customize button. This property sheet is very important for security and troubleshooting.





On the General tab, the option to “Allow the connection if it is secure” does not by itself require ESP payload encryption of packets. Digitally signed, *plaintext* payloads are also considered “secure” here, such as when using AH or when the ESP payload encryption is turned off. This might not be what you expected.

**Require Encryption**

To require ESP encryption on an inbound firewall rule, click the Customize button and choose the option named “Require the connections to be encrypted”. Now this firewall rule will drop any matching packets that are not using ESP encryption.

But then why is there a strange checkbox to “Allow the computers to dynamically allow encryption” when encryption is already required? Does this sometimes allow plaintext packets despite the rule? No, but here is the problem: Does the other computer know to *begin* the IPsec session with encryption enabled? What if the other computer, by default, would rather use IPsec with payload encryption turned off? How does the IPsec conversation get started then? We want to avoid any chicken-and-egg problems.

Here is what happens when the “Allow the computers to dynamically negotiate encryption” box is checked or unchecked:

**Unchecked:** The other computer must *include* the IPsec association with a request for ESP encryption from the very start. This means that the other computer must either 1) have an *outbound* rule that requires ESP encryption for this connection, or 2) it must have the box checked to “Require encryption for all connection security rules that use these settings” in its IPsec configuration settings.

**Checked**: The other computer is allowed to establish a *first* IPsec security association with no encryption (AH or ESP with encryption disabled), but then it must immediately establish a *second* IPsec security association with ESP encryption, then use that second one going forward. This means that the other computer 1) does not have to have its outbound rules configured in any particular way with regard to IPsec, and 2) it does not have to have the box checked to “Require encryption for all connection security rules that use these settings” in its IPsec configuration settings, i.e., it can sometimes use IPsec with no payload encryption, but just not for this connection being considered.

If a computer’s firewall rule is allowed to “dynamically negotiate encryption”, it does not mean that the rule might accept plaintext traffic for all matching packets; it only means that the IPsec connection might be plaintext *at first*, for the first few packets, before the peers switch over to a fully encrypted IPsec security association.

By analogy, if I only spoke Greek, and you spoke dozens of languages including Greek, then how would you know to switch to Greek if I refused to respond to any sentence that was already not in Greek? If you said, “Hello”, and I responded with “Let’s speak Greek please”, then you would know that I heard you and that I prefer to use Greek. We could then continue communicating. So if a server requires ESP encryption to access one of its TCP ports, it would be nice if the server would respond when necessary with “Let’s switch to encryption please, even if that meant that these first few packets were not themselves encrypted. Think of it as the IPsec version of being courteous to others. Yes, it would be more secure to configure the client with an outbound rule that requires ESP encryption to correspond to the server’s inbound rule that also requires encryption (so no need to “dynamically negotiate”), but this situation would require a specifically configured outbound rule on the client. This would be a hassle, it would be another source of problems to worry about, it would be another barrier to deploying IPsec across the enterprise. (Note that *outbound* firewall rules lack the option named “Allow the computers to dynamically negotiate encryption” since it is always up to a target machine to decide whether to respond with an IKE negotiation or not. If an outbound firewall rule requires IPsec encryption, then the encryption is required – period, end of story.)

**Recommendations**

Especially when first deploying IPsec, the recommendation is to either 1) always check toe box to “Allow the computers to dynamically negotiate encryption” in all firewall rules that require IPsec encryption or 2) check the box to “Require encryption for all connection security rules that use these settings” in the IPsec configuration settings on all computers IPsec. This will simplify troubleshooting.

The second option, namely, to just never allow AH or ESP with the encryption turned off, is preferable to avoid any misunderstandings with IT managers who have preconceptions about IPsec. It is also preferable on machines directly exposed to the internet and on high-value target systems prone to attack.

**Allow Null Encapsulation**

This option allows a “secure” connection to be both unsigned and unencrypted, using the null encapsulation technique discussed previously. The connection is “secure”, in this case, only because of the initial IKE mutual authentication and periodic heartbeat ESP packets.

**Override Block Rules (For Jump Servers and Admin Workstations)**

This option exists to reassure administrators who are worried about IPsec becoming a self-imposed denial-of-service (DoS) attack. When two firewall rules with equal specificity apply to the same connection attempt, where one rule blocks and the other rule allows with IPsec, the allow rule with IPsec will always win.

However, this IPsec-override feature mandates that the Remote Computers tab in the firewall rule must specify at least one computer. Only the computer(s) listed on this tab will be allowed to exercise this override feature. In real life, specify a group of computer accounts that includes the jump servers and workstations of the administrators. If the lab testing of a firewall rule change does not reveal the problem hidden in the change, hopefully the IPsec-override feature will still allow IT people to get their jobs done.

**Deployment Automation Options**

**Group Policy**

* Best for hosts inside the LAN or with VPNs.
* Easiest agility with OUs, GPO permissions, and WMI filtering.

**PowerShell Remoting**

* Great for quick deployment and centralized control.
* Not as scalable as GPOs and not good for roaming endpoints.

**Scheduled Scripts**

* Scripts can be cached locally and updated from a shared folder.

Deployment Automation Options

There are many options for deploying and managing firewall and IPsec rules across thousands of endpoints and servers. Each option has advantages and disadvantages.

**Group Policy**

Group Policy is best for servers and endpoints inside the LAN or that established permanent or regular VPN tunnels back into the LAN. Servers that are hosted on cloud provider networks can also access their domain controllers. Group Policy provides the most flexibility for the management of IPsec and firewall rules. Different GPOs can be assigned to different OUs, and GPOs can have permissions or WMI Filters to control which machines receive those GPOs. But Group Policy only works with domain-joined machines. If you use a tool like LGPO.EXE to apply a GPO to a standalone computer, then it might be easier to just run a configuration script instead.

**PowerShell Remoting**

Remoting works on both standalone and domain-joined targets, but those targets must be accessible over the network, which makes this option less ideal for roaming devices on the internet. Using remoting this way is similar to push mode DSC, with all the pluses and minuses for scalability this entails. Nonetheless, remoting is still a great way to manage internal or cloud-hosted machines, such as servers.

**Scheduled Scripts**

Scheduled scripts can run other scripts that manage IPsec and firewall rules. Scheduled jobs can be managed through Group Policy, PowerShell remoting, SCHTASKS.EXE, Desired State Configuration (DSC), and many other enterprise configuration management solutions. Your own scheduled scripts allow for potentially great scalability and flexibility. If you can think it up, and you’ve got an enterprise-scale method of managing scheduled jobs, then you can probably do it! But with all this roll-your-own agility comes complexity. So with scheduled jobs and PowerShell scripts, you can get whatever you want, but you’ll have to design, deploy, and maintain that solution yourself too.

**Security Zone IP Addressing Scheme**

Firewalling internal traffic is greatly simplified if different IP ranges are assigned to different internal security zones. Not all computers are equally important or valuable from a security point of view. A “security zone” is a range of IP addresses allocated to a set of computers that have similar security requirements, usually because these servers and/or workstations have similar roles in the organization. Keep in mind that the following are not hard rules to be blindly followed, but general strategies to implement.

**Typical Security Zones**

Your environment will be unique, so your security zones may need to be customized, but the following security zones are typical:

* Security
* DMZ (or Extranet)
* Servers
* Clients
* Internet

The **Security Zone** includes everything that enforces or manages security for the internal network, such as domain controllers, RADIUS servers, SIEM consoles, IDS sensors, IDS consoles, administrator workstations, jump servers, log consolidation servers, and so on. These are the highest of your high-value targets. The Security Zone is often associated with special “shadow” segments or VLANs as well and might even correspond to an entire separate AD forest, such as for administrative workstations. Instead of a single Security Zone, you might subdivide this into special purpose zones for monitoring, penetration testing, SIEM, domain controllers, etc.

The **DMZ** (**or Extranet Zone**) includes the servers exposed to the internet, plus any associated servers or workstations that are not exclusively part of another Zone. If you have no perimeter firewalls, such as at a university, then this zone doesn’t exist.

The **Servers Zone** includes all internal servers, though there will likely be some overlap with some of the DMZ and Security servers, which is fine; just allocate IP addresses in a way that makes sense for your environment. Servers include your internet cloud-based VMs that are directly accessible from the LAN through a site-to-site VPN or other private tunnel into the internet cloud provider’s network.

The **Clients Zone** includes all internal workstations, laptops, tablets, and smartphones, plus any remote clients with a VPN or DirectAccess connection into the LAN.

The **Internet Zone** is the rest of the world, including your own servers that are accessed over the public internet, such as your VMs hosted by cloud providers but without a site-to-site VPN or other private tunnel.

**Per-Zone IP Addressing**

At each site, each zone should be assigned one or more IP address ranges. It is nice if the same subnet range for each zone is used at every site, but this isn’t required; it’s just easier to manage and understand. If you also wish to implement a segmentation, VLAN, firewalling, or tunneling scheme on top of the IP addressing scheme, that can be handy too, but from the perspective of host-based firewalls, such details are usually invisible. The machines in the Security, DMZ, and Servers Zones are typically assigned static IP addresses (or at least DHCP reservation), and the other devices in the Clients Zone will use DHCP and/or IPv6 network IDs advertised by your routers.

As an example, let’s assume your organization uses 10.0.0.0/8 as the enterprise-wide network ID (we’ll ignore IPv6 for now, but with its much larger addressing space, implementing the various per-site zones is actually even easier than with IPv4).

**Note:** These examples are not optimized for CIDR route aggregation or conservation of scarce IP addresses; they are intended to be easy to understand for attendees who don’t work at Layer 3 and to make the management of firewall and IPsec rules simpler.

Each of your 200 sites might be assigned a different number in the second octet:

* Dallas LAN: 10.**1**.0.0/16
* DC LAN: 10.**2**.0.0/16
* San Diego LAN: 10.**3**.0.0/16
* Amsterdam LAN: 10.**4**.0.0/16

Within a site, the Security Zone might always have 1 through 5 as the third octet:

* Dallas Security: 10.1.**1-5**.0/24
* DC Security: 10.2.**1-5**.0/24
* San Diego Security: 10.3.**1-5**.0/24
* Amsterdam Security: 10.4.**1-5**.0/24

Within a site, the DMZ or Extranet Zone might always get 6-10 as the third octet:

* Dallas DMZ: 10.1.**6-10**.0/24
* DC DMZ: 10.2.**6-10.**0/24
* San Diego DMZ: 10.3.**6-10**.0/24
* Amsterdam DMZ: 10.4.**6-10**.0/24

Within a site, the Servers Zone might get 11-20 as the third octet:

* Dallas Servers: 10.1.**11-20**.0/24
* DC Servers: 10.2.**11-20**.0/24
* San Diego Servers: 10.3.**11-20**.0/24
* Amsterdam Servers: 10.4.**11-20**.0/24

The important thing is not the number of bits in the CIDR mask, the names of the zones, the number of special purpose zones, etc. The important thing is the overall strategy of associating IP address ranges with different types of devices based on their roles and their relative importance for security. You might begin with a Client Zone and the Everything-Else Zone and then see what makes sense from there. Again, it’s the strategy we’re interested in here, not the particular example details, because every attendee’s network will be different.

**Per-Zone Firewall Rules and IPsec Policies**

The benefit of using per-zone IP addressing is that it will greatly simplify the management of firewall rules and IPsec policies. And not just host-based firewalls, your perimeter and internal firewall devices can also leverage these zones. Your proxy servers, IDS sensors, and other network devices can benefit as well. Using Group Policy preferences with item-level targeting, different GPO policies can be applied to different preferences with item-level targeting, different GPO policies can be applied to different Zones based just on IP address (though using GPO permissions is probably the better approach). Investing in a rational IP addressing scheme will pay off in many ways.

Consider this: Do your workstations, laptops, and tablets typically need to communicate directly with each other? Other than VoIP, they typically do not. So a very effective host-based firewall strategy is to have every computer in the Clients Zone block all the unnecessary inbound and outbound packets to every other IP address in the Clients Zone. Now when malware and hackers take over an initial soft target in the Clients Zone, it will be more difficult for them to directly attack other client computers.

**Mapping Zones to Organizational Units**

Ideally, the roles a computer plays will determine its security zone and IP address, and these roles might also determine that computer’s organizational unit (OU) in Active Directory. The same logic applies to both design strategies: we need to apply different security policies based on the role(s) of the computer, so some security policies affect the physical placement, Ethernet switch, VLAN, host-based firewall rules, and IPsec settings of the computer (OSI Layers 1-4), while other policies affect the computer’s operating system, applications, and users (OSI Layers 5+).

Security zones also affect where we allow administrators and other high-value accounts to authenticate. Domain Admin users might be allowed to authenticate interactively or over the network to servers in the Security Zone, but not to the Servers Zone and certainly not the malware-infested Clients Zone. Each zone, then, might have separate administrative accounts in order to contain the harm from stolen credentials and token abuse attacks. How do we control such authentication traffic? Through Group Policy, and these GPOs are typically assigned to OUs, hence we see the possible mapping between zones and OUs again.

None of these recommendations are absolute; you will need to customize and fine-tune them for your environment, but we do want to scale up our security work to the entire enterprise, while keeping a lid on the growth of complexity this entails.

Group Policy Management

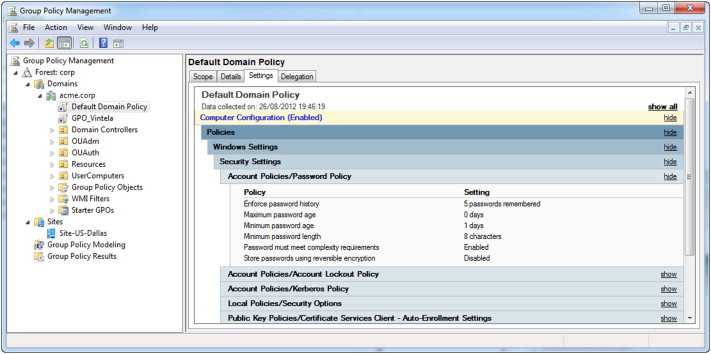
How do you scale out management of IPsec and firewall policies to thousands of computers? Group Policy of course! And for your standalone systems, you can write a batch script to manage IPsec and firewall policies on them too. Not only Windows Vista and later but also Windows XP/2003 can have their IPsec and firewall settings managed through Group Policy and/or custom scripts.

**Group Policy Management**

* Windows Firewall Rules
* IPsec Policies

**Overall Group Policy Design**

* You do not have to block by default!
* Rules from multiple GPOs are merged together following LSDOU.
* Set generic rules at upper-level OUs, then override and expand at sub-OUs.



**Export and Import Policies**

IPsec policies can be exported and imported. On Vista and later, the IPsec and firewall policies are exported/imported together in one file. In Windows 2000/XP/2003, right-click the “IP Security Policies” MMC snap-in > All Tasks > Import/Export Policies. In Windows Vista/2008 and later, right-click the “Windows Defender Firewall” snap-in > Import/Export Policy. In either case, you are saving or restoring settings from a file. (The word “Defender” was added to the tool in Windows Server 2019 because the original name was too short.)

The ability to export and import policy files is important because you’ll want to test your IPsec and firewall policies in an isolated lab first, then you’ll later import them into a GPO for pilot testing and then into another GPO for production deployment. It is also a good idea to keep snapshots of your various IPsec and firewall policies for recovery and audit reference.

**Group Policy Object Paths**

Windows Firewall settings for Server 2008/Vista and later are located in a GPO under Computer Configuration > Policies > Windows Settings > Security Settings > Windows Defender Firewall with Advanced Security.

Windows Firewall settings for Windows XP/2003 are located in a GPO under Computer Configuration > Policies > Administrative Templates > Networks > Network Connections > Windows Firewall.

IPsec policies for Windows 2000/XP/2003 computers are located in a GPO under Computer Configuration > Policies > Windows Settings > Security Settings > IP Security Policies on Active Directory.

Note that all IPsec policies created in Active Directory for Windows 2000/XP/2003 will be visible in the GPO, but only one of these IPsec policies can be activated or “assigned” at a time (notice the Assigned column). It is not the case that all visible policies will be assigned like in Vista/2008 and later.

**Assigning Multiple Policies via GPO**

Recall that GPOs are applied in the following order: local, site, domain, and OUs from outermost to innermost (mnemonic: LSD-OU). On Windows 2000/XP/2003, the last GPO to be applied that assigns an IPsec Policy will replace and override any other IPsec policies earlier in the GPO precedence order. So an IPsec policy assigned at the OU level will supplant any IPsec policy assigned at the domain, which will replace any IPsec policy assigned at the site, which replaces the local, and so on. IPsec policies assigned to the site, domain, or OU are not merged on Windows 2000/XP/2003: the last IPsec policy assigned while GPOs are being processed is the one and only IPsec policy that becomes effective. There are no conflicts because only one IPsec policy wins in the end, i.e., the last one applied wins. And if a local IPsec policy was assigned to a machine, this local policy will be overwritten by the IPsec policy assigned from Group Policy too.

If you use Group Policy to push out an IPsec policy to Windows 2000/XP/2003 and also Windows Vista or later, that policy is accepted and enforced on all operating systems. Windows Vista and later are backward compatible with the older IPsec policies. But on Windows Vista and later, any additional IPsec policies configured through WFAS are also accepted and enforced at the same time. If there is a conflict, the older Windows 2000/XP/2003 IPsec policies are evaluated first and then the WFAS IPsec Connection Security Rules are evaluated afterwards.

**Note:** One way or another, though, if a Windows Firewall rule requires certain traffic to be secured with IPsec, it still has to be secured, but it doesn’t matter what triggered the IPsec negotiation process (an older Windows 2000/XP/2003 IPsec policy, a WFAS Connection Security Rule, or the firewall itself). And as a kicker, remember that dynamic mode IPsec settings are evaluated before any of the static mode settings!

Another difference to note is that WFAS IPsec policies inherited from multiple GPOs are *all* accepted, combined, and enforced. In Windows 2000/XP/2003, the last IPsec policy assigned through Group Policy is the only IPsec policy that gets enforced. This is no longer the case in Vista or later. If multiple GPOs apply to a Windows Vista box, for example, and each GPO has different WFAS IPsec quick mode settings, then all the applicable quick mode IPsec settings from all the inherited GPOs are combined and enforced. If there are conflicts, then GPOs processed later will override conflicting settings applied earlier. However, this merging of IPsec quick mode (Phase II) settings does not apply to the main mode (Phase I) settings. You can only have one main mode policy, and the last GPO to assign a main mode policy is the winner (just like in Windows 2000/XP/2003). Therefore, standardize on one set of main mode configuration settings and use them everywhere on every operating system.

**PowerShell Management**

**Overall Script Design**

1. Scrub the slate clean; delete every IPsec rule.
2. Restart the IKEEXT service for IPsec.
3. Set the machine-wide defaults you want for authentication, encryption, hashing, and so on.
4. Create one or more rules which secure just the traffic flows you want (based on IP subnets, protocols, and port numbers).

PowerShell Management

Both IPsec and firewall settings can be scripted from the command line. These tools work on local or remote systems, as long as you are a member of the local Administrators group.

There are different tools for different operating systems, but the operating systems that don’t support PowerShell are mostly dead. For managing IPsec settings, use PowerShell whenever possible:

* PowerShell 3.0 or later: Over 200 cmdlets (get-help \*-net\*)
* Windows 2000: IPSECPOL.EXE
* Windows XP: IPSECCMD.EXE
* Windows 2003/Vista/2008/7 and later: NETSH.EXE (deprecated)

To use PowerShell, you must have PowerShell 3.0 or later. There are over 200 cmdlets related to networking, IPsec, and the firewall, so they can’t be discussed here.

To see a listing of the cmdlets for IPsec, firewall rules, network interfaces, TCP/IP, etc.:

***Get-Help \*-net-\****

***Get-Help \*-net\*ipsec\****

***Get-Help \*-net\*firewall\****

***Get-Help \*-net\*ip\****

Overall Script Design

The complexity of IPsec scares away many administrators. But once you have a script that creates the rules you want, edit the script so that the first thing the script does is to scrub the slate clean: create a function that deletes every existing rule and then restarts the IKEEXT service. After your scrubber function runs, your script would then add back just the rules you want, rules that have been tested are known to work correctly. With a script like this, how do you troubleshoot a machine with IPsec problems? Don’t “troubleshoot” it – just run the script again! The script is now like a template you can reapply at any time.

**Tip:** Scrub the slate clean first, then create the IPsec rules you want.

The IKEEXT service handles IKEv1 negotiations with other machines. Technically, you don’t have to restart this service after modifying your IPsec rules. The service is designed to read any rule changes after you make them. But in the spirit of “stress reduction through wiping the slate clean”, it’s nice to do. When troubleshooting, restarting the IKEEXT service is a useful step. The only bad thing about restarting the IKEEXT service is that this terminates existing IPsec security associations. If you modify your existing IPsec rules in a way that is compatible with existing IPsec security associations, then these SAs are not (usually) broken by changing the rules.

It is possible for every IPsec rule to have its own separate authentication, encryption, and integrity settings. Avoid doing this. Configure one set of machine-wide or global defaults for IPsec and then use these defaults in every IPsec rule. These are the defaults you see when you right-click the Windows Firewall snap-in and go to the IPsec Settings tab. Whenever possible, use the same defaults for every rule on every device, including non-Windows devices.

**Tip:** Whenever possible, IPsec rules should use the machine’s global defaults.

It is possible for the machine to have global defaults that include many authentication methods, many ciphers, many different key sizes, many different hashing algorithms, and a variety of Diffie-Hellman flavors. Avoid doing this. Try to just have one proposal for each type of setting, and only add more proposals to the list when absolutely necessary. Just choose sane defaults that should be widely compatible and secure enough for the next several years, like 256-bit AES, SHA-256 hashing, and Diffie-Hellman group 14. It’s the KISS principle applied to IPsec (Keep It Simple and Sane).

**Tip:** Choose one sane offer for authentication, encryption, and hashing, and stick to it whenever possible; only add more offers to the list when forced to do so.

Later, when your team is more comfortable with IPsec, you can customize your rules to benefit from IPsec’s “cryptographic agility”, but we don’t have to start out by diving into the deep end.

**Dynamic Mode vs. Static Mode**

The IPsec command line tools operate in two modes: dynamic mode and static mode. When used in dynamic mode, they inject rules directly into the IPsec driver’s in-memory database of IPsec rules. Dynamic rules take effect immediately, but they do not show up in any graphical tool; hence, they can only be managed from the command line. When used to create static mode policies, these tools can create visible named policies that are permanently stored in the registry or in Active Directory. Which mode is being used depends on the command line switches used.

**Scheduled Scripts, Per-User Rules, and Other Creative Uses**

A script with the desired IPsec or firewall rules could be scheduled to run every hour. Because these tools can take a command line argument of a remote computer, domain member, or standalone, this single script could configure hundreds of systems from a central location. IPsec policies are assigned to computers, not users. However, Group Policy can be used to define logon/logoff scripts for users. These scripts could create dynamic IPsec rules when the user logs on, then delete these dynamic rules when the user logs off again. Hence, you can implement per-user IPsec settings with logon scripts for users who are local Administrators wherever they log on.

**Group Policy and Windows Firewall Snap-In Limitations**

The graphical Windows Firewall snap-in (WF.MSC) can manage most IPsec settings, but there are some settings that can only be managed through PowerShell, NETSH.EXE, or direct registry edits.

The following is a list of IPsec settings or rules that can only be configured from the command line or with scripts, not with Group Policy or the WF.MSC console:

* Use customized authentication options for main mode negotiations based on the IP address of the source or destination peer, or based on the network profile type (Domain, Public, or Private) of the interface through which the IPsec connection is being established. When using WF.MSC, only one machine-wide, global set of authentication options may be configured; these options can only be customized for separate connections from the command line.
* Similarly, per-IP or per-profile quick mode negotiation settings can only be created using PowerShell or NETSH.EXE. Using WF.MSC, you can only create one machine-wide, global set of quick mode negotiation options, such as for encrypted ESP versus null encapsulation for different connections.
* Quick mode rules that require Perfect Forward Secrecy (PFS) may only be configured using PowerShell or NETSH.EXE.
* Certificate revocation checking can only be configured using PowerShell, NETSH.EXE, or direct registry edits.
* Exemptions or requirements related to DHCP, IPv6 Neighbor Discovery protocol, or Teredo are managed only from the command line.

**NETSH.EXE (Deprecated)**

NETSH.EXE can be used on Windows XP and later to manage firewall settings and on Windows Server 2003 and later to manage IPsec settings. Until every machine is running PowerShell 3.0 or later, you might prefer this binary for a while, even though it has been deprecated by Microsoft. Here are commands with which to experiment.

For managing the firewall on Windows XP and later, use only PowerShell or NETSH.EXE. In Windows XP/2003, the NETSH.EXE context is “firewall”, while in Windows Vista and later, the context is named “advfirewall” (execute “netsh.exe /?” to see these contexts). The NETSH.EXE program is built into Windows 2000 and later by default, but it has different capabilities on different OS versions.

To see a summary of your profile options: *netsh.exe advfirewall show allprofiles*

To dump the details of every connection security rule (IPsec): *netsh.exe advfirewall consec show rule name = all*

To see how to create a connection security rule (IPsec): *netsh.exe advfirewall firewall show rule name = all*

To see how to create a firewall rule from the command line: *netsh.exe advfirewall firewall set rule /?*

The following is an example of a batch file that could be used to create a static IPsec policy object on Windows Server 2003 or later. This policy would block all packets except for TCP 80/443, and it would require AH for all traffic to/from network ID 10.0.0.0 (note that many of the lines must be wrapped).

***REM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***REM Create The Ipsec Policy Object.***

***REM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***Netsh.exe ipsec static add policy name = “IIS\_Server\_Policy” assign = no***

***REM Create Filter Lists and Add Filters To Them.***

***Netsh.exe ipsec static add filterlist name = “HTTP\_Traffic” srcaddr=any dstaddr=me description=”HTTP” protocol = TCP srcport =0 dstport = 80***

***Netsh.exe ipsec static add filter filterlist = “HTTP\_Traffic” srcaddr=any dstaddr=me description = “HTTPS” protocol = TCP srcport= 0 dstport=443***

***Netsh.exe ipsec static add filterlist name = “All\_Traffic”***

***Netsh.exe ipsec static add filter filterlist = “All\_Traffic” srcaddr=any dstaddr=me description= “All Traffic” protocol = any srcport=0 dstport = 0***

***Netsh.exe ipsec static add filterlist name = “Internal\_Traffic”***

***Netsh.exe ipsec static add filter filterlist = “Internal\_Traffic” srcaddr=10.0.0.0 srcmask=255.0.0.0 dstaddr=me description = “Internal Traffic” protocol = any srcport=0 dstport=0***

***REM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***REM Define Filter Actions.***

***REM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***Netsh.exe ipsec static add filteraction name = “Allow” action=permit***

***Netsh.exe ipsec static add filteraction name = “Block” action=block***

***Netsh.exe ipsec static add filteraction name = “AH\_Only” qmpfs=yes soft=no inpass=yes action=negotiate qmsec = “AH[MD5]:100000k/1000s AH[SHA1]:100000k/1000s”***

***REM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***REM Now Create Rules In The Policy With The Actions and Filters.***

***REM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***Netsh.exe ipsec static add rule name = “Allow HTTP” policy = “IIS\_Server\_Policy” filterlist = “HTTP\_Traffic” kerberos = yes filteraction = Allow***

***Netsh.exe ipsec static add rule name = “Block All” policy = “IIS\_Server\_Policy” filterlist=”All\_Traffic” kerberos = yes filteraction = Block***

***Netsh.exe ipsec static add rule name “AH for LAN” policy = “IIS\_Server\_Policy” filterlist = “Internal\_Traffic” psk=”myPreSharedKey” filteraction = AH\_Only***

***REM Disable The Built-In Response Rule.***

***Netsh.exe ipsec static set defaultrule policy = “IIS\_Server\_Policy” activate = no***

***REM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***REM Now Assign The Policy.***

***REM\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\****

***Netsh.exe ipsec static set policy name = “IIS\_Server\_Policy” assign = yes***

**Please turn to the next exercise… Tab completion is your friend! F8 to *Run Selection***

On Your Computer

In this lab,you will create and use an IPsec rule to secure selected traffic flows.

**[Controller] Configure Basic IPsec Rule**

Please switch to the C:\SANS\Day4\IPsec folder: *cd C:\SANS\Day4\IPsec*

Use ISE to open the New-PreSharedKeyIPsecRule.ps1 script and then read its contents: *ise .\New-PreSharedKeyIPsecRule.ps1*

**Note**: If remoting into your member server VM fails, switch to that VM in your virtualization software instead to run powershell.exe commands.

Use PowerShell remoting to run the script on the member server: *Invoke-Command -ComputerName member -FilePath .\New-PreSharedKeyIPsecRule.ps1*

Run the same script on the local computer:

*.\New-PreSharedKeyIPsecRule.ps1*

Display the local IPsec rules: *Get-NetIPsecRule | Select-Object DisplayName, Enabled*

In the Windows Firewall snap-in, right-click the “Connection Security Rules” container and choose Refresh. You should see your new IPsec rule named “Dangerous TCP Ports”. This rule applies to TCP ports 445 and 139 for the SMB protocol used by the File and Printer Sharing Service (also known as the “Server” service or “LanmanServer” service), TCP port 3389 for Remote Desktop Protocol (RDP), and TCP port 21 for FTP passwords and commands.

List the contents of the C$ shared folder on the member server to trigger IPsec: *dir \\member\C$*

In the Windows Firewall snap-in, navigate down to Monitoring > Security Associations > Main Mode. You should see your main mode (Phase 1) contract negotiated with the member server VM. (If necessary, right-click and Refresh.)

You can see the same information in PowerShell: *Get-NetIPsecMainModeSA*

**[Controller] Configure Custom IPsec Rule**

Use ISE to open the New-KerberosIPsecRule.ps1 script and then read its contents:

*Ise .\New-KerberosIPsecRule.ps1*

This rule is more complex. The script is probably difficult to understand if you are new to IPsec. This rule uses Kerberos authentication for both the computer and the user, plus it explicitly sets the cryptographic settings for the main mode (Phase 1) and quick mode (Phase 2) negotiations instead of relying on Microsoft’s defaults. You don’t have to understand every setting in the script right now to use it as a template for your future scripts.

Use PowerShell remoting to run the script on the member server: *Invoke-Command -ComputerName member -FilePath .\New-KerberosIPsecRule.ps1*

Run the same script on the local computer: *.\New-KerberosIPsecRule.ps1*

In the Windows Firewall snap-in, right-click the “Connection Security Rules” container and choose Refresh. You should see your new IPsec rule and the TCP ports to which it applies, but this time the rule uses Kerberos, not a pre-shared key.

List the contents of the C$ shared folder on the member server to trigger IPsec:

*Dir \\member\C$*

In the Windows Firewall snap-in, navigate down to Monitoring > Security Associations > Main Mode / Quick Mode. Here you can confirm the use of Kerberos, 256-bit AES encryption, SHA-256 hashing for integrity checking, and Diffie-Hellman Group 14 for the key exchange. (If necessary, right-click and Refresh.) Double-click a security association to show a property sheet with more details about the connection.

You can see even more information in PowerShell: *Get-NetIPsecMainModeSA; Get-NetIPsecQuickModeSA*

**[Controller] Import into a Group Policy Object**

Export your firewall and IPsec rules to a backup file with NETSH.EXE: *netsh.exe advfirewall export c:\temp\boston.wfw*

Imagine that your lab testing and “volunteer user” testing has been going well. Now you want to roll out your IPsec and firewall rules through Group Policy. Please open the Group Policy Management tool and edit the Boston\_GPO linked to the Boston OU. (If that GPO has disappeared, just right-click the Boston OU > Create a GPO in this domain, and Link it here.)

Inside the Boston\_GPO, navigate down to Computer Configuration > Policies > Windows Settings > Security Settings > Windows Defender Firewall with Advanced Security.

Next, right-click on the Windows Defender Firewall itself in the Boston\_GPO > Import Policy > Yes button > select the C:\Temp\boston.wfw file > Open > OK.

Right-click on the Windows Defender Firewall tool itself again > Properties. Notice you can define the default firewall policies for all three network interface profile types: Domain, Private, and Public. The familiar IPsec Settings tab is here as well. Click Cancel.

More Windows Firewall settings can be found in a GPO under Computer Configuration > Policies > Administrative Templates > Network > Network Connections > Windows Defender Firewall.

**[End of Lab]**

**Today’s Agenda**

1. **Scripting Windows Firewall Rules**
2. **Scripting IPsec for Role-Based Access Control**
3. **Server Hardening Automation**
4. **PowerShell and Windows Logging**

Today’s Agenda

There are several protocols that Windows cannot live without, but these ports and packets are vulnerable to attack. Defensible networking means reducing the exploitability of these protocols as much as practical without disrupting user activities. In this section, we will talk about server hardening automation with PowerShell and see some examples. The aim is to have a set of scripts that, ideally, could build a new server VM from scratch and enforce all of one’s desired hardening changes, including firewall and IPsec rules.

**What Is Server Hardening? How to Get Started?**

**Recent**

* The latest or prior version of the operating system.

**Redeployable**

* Boot from VHD, VMs, containers, templates, backups, etc.

**Patched**

* Test and apply all security patches as quickly as practical.

**Minimal**

* Eliminate unneeded roles, features, permissions, rights, etc.

What Is Server Hardening? How To Get Started?

For server hardening, we want to start with a recent, patched, minimal redeployable operating system. This is the foundation; it’s what everything else is built on top of.

Recent

This course assumes you have a recent operating system, where “recent” means either the latest version or the prior version. This is an expensive requirement, but remember that we are focusing on internet-exposed servers, such as IIS web servers in the DMZ or in a cloud provider’s network. In general, prefer the latest version you can afford, even for internal servers, and, of course, don’t run dead operating systems at all, like Server 2003.

Redeployable

Assume breach. Assume failure. Assume there will be problems after making configuration changes, such as when we upgrade the OS, apply patches, and enforce the Principle of Least Privilege. Complexity, mixed with a high rate of change in a hostile environment filled with hackers and malware, means that there will be unavoidable problems. The aim is to reduce the risk to an acceptable level. Ideally, we want it to be easier to rebuild a server than to fix it or clean it. If we suspect that a server has been compromised, why live with the risk and the worry? Save the VM or container for analysis if desired, then deploy a replacement. This is the ideal; it won’t always be possible, but we can try to automate the redeployment process as much as possible to prepare for the inevitable. In fact, if redeploying a fresh server replacement is push-button, maybe it should be done preemptively on a nightly basis? Think of how Virtual Desktop Infrastructure (VDI) solutions create and destroy user desktops on the fly as needed; wouldn’t it be nice if it worked that way for servers too? Being able to redeploy a server is important for the prior hardening tasks also; in fact it’s like a meta-enabler, a foundational capability. If we can automate the deployment, then maybe upgrading the OS will be easier next time. If we can automate the deployment, then it should be easier to roll back updates and other configuration changes when they break things; hence, we don’t have to worry as much about them. If we can automate the process of stripping out the roles, features, and other components we don’t need from our servers, then our hardening checklists and compliance policies can be “baked in” from the beginning. These ideas aren’t new; this is straight out of the DevOps playbook.

There are many technologies related to having redeployable servers; virtualization, containers, boot from VHD, templates, scripting, backup systems, patch management systems, enterprise management systems, network attached storage, monitoring infrastructure, and so on. We can’t cover them all in this course, so let’s just focus on Windows Server and PowerShell.

Patched

After installing the latest OS version you can afford, apply the latest updates and patches and enroll the server in your patch management system to keep it updated. Quickly testing and applying new security patches is one of the most important duties in maintaining a hardened server, especially when the patch relates to a vulnerability on a listening TCP/UDP port that is exposed to the internet.

Minimal

None of us, not even Microsoft or the hackers themselves, can predict what will be found to be vulnerable tomorrow. By definition, a new vulnerability is a *discovery*. What we can do, though, is uninstall or disable the roles and features we don’t need today in case they are discovered to be vulnerable tomorrow. Patches fix problems we know about already; patching is *reactive* only. To *proactively* harden a server is to try to eliminate the vulnerabilities that we don’t know about and that perhaps no one knows about (yet).

Throughout the course, the security recommendation is always the same: ***If you don’t need it, get rid of it!***

There’s no need to ask “Is *X* vulnerable?”, where *X* is an application, service, listening port, feature, permission, logon right, privilege, bell, or whistle. If it’s running, it’s vulnerable to some degree. The goal is to reduce risk by eliminating vulnerabilities while at the same time satisfying the needs of our users, managers, and compliance auditors. A minimal server runs the least amount of code necessary, starting at the OS layer and working our way up through roles, features, and applications. Minimal also means following the Principle of Least Privilege in configuring logon rights, permissions, privileges, and other access controls.

**Server Core vs. Graphical Desktop Experience**

**Server Core (Server 2008 and Later):**

* No Start Menu and very little graphical application support
* Smaller hard drive footprint (around 4-7 GB).
* Nice for appliances with SSD or M.2 drives.

**App Compatibility FOD (Server 2019 and Later):**

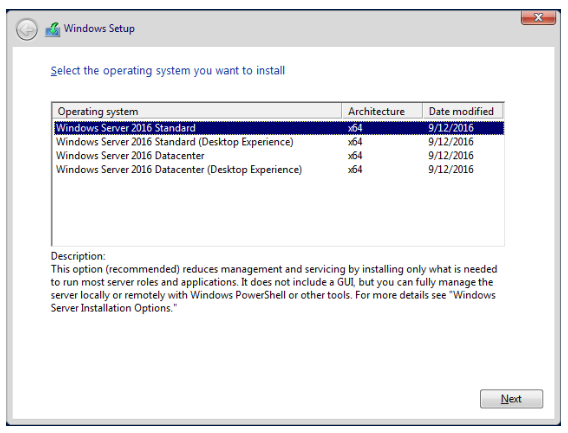
* FOD = Feature On Demand (no OS reinstall required).
* Includes explorer, powershell\_ise, regedit, notepad, and mmc.exe consoles like wf.msc, eventvwr.msc, and diskmgmt.msc.
* *Import-Module -Name DISM*

Server Core vs. Graphical Desktop Experience

Windows Server 2008 and later includes an installation option named “Server Core” that strips away services, applications, and other features that are typically only necessary on desktop computers. Most of the graphical desktop and user applications are removed, such as the Start Menu and the taskbar.

**Server Core Installation**

During the GUI install of any edition of Windows Server 2008 or Server 2012, you will see a screen similar to the one below, where you can select the “Server Core Installation”.



On Server 2016 and later, installing as Server Core is the default, so the word “Core” does not appear in the installation GUI; instead, if you do not want Core, if you want a graphical interface, you choose the edition with the words “Desktop Experience” in the name. In the past, installing Windows Server with a graphical desktop was the default, but now Server Core is the default.

**Switch Roles without a Reinstall? (Server 2012 R1/R2 Only)**

On Server 2008 and 2008 R2, there was no switching between the full GUI installation and the Core installation without a complete reinstall of the operating system. This was a major barrier to the deployment of Core around the world. But on Server 2012 and Server 2012 R2, when you’re done with the GUI components, the graphical desktop can be uninstalled without reinstalling the entire OS (you will need to reboot though). Similarly, if you have Core and need the GUI desktop, you can switch from Core to GUI desktop mode without reinstalling the entire OS. Very handy!

However, with Server 2016 and later, Microsoft changed its mind again, and it is no longer possible to switch between Core mode and the full graphical Desktop Experience mode without reinstalling. Once you install Windows Server 2016 or later, you are stuck forever with having or not having the graphical desktop. You’ll need to reinstall from scratch to switch.

**Supported Roles and Features**

The following roles are currently supported on Server Core, but more are planned:

* Active Directory Domain Services (ADDS)
* Active Directory Certificate Services (ADCS)
* Active Directory Rights Management Service (AD RMS)
* Active Directory Lightweight Directory Service (AD LDS)
* DHCP Server
* DNS Server
* File Services
* Web Server (IIS)
* Hyper-V
* RRAS
* Print and Document Services
* Streaming Media Services
* SQL Server
* Windows Server Update Services (WSUS)

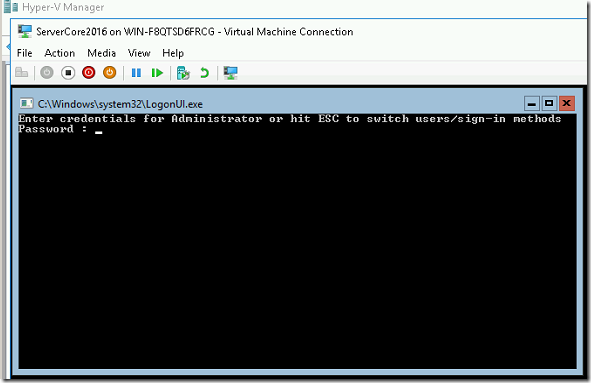
The following features can be installed on Server Core too:

* Failover Clustering (Enterprise Edition)
* Network Load Balancing
* Subsystem for UNIX-based applications
* Backup
* Multipath IO
* Removable Storage
* BitLocker Device Encryption
* Simple Network Management Protocol (SNMP)
* Windows Internet Name Service (WINS)
* Telnet client
* WoW64 support for 32-bit applications (2008-R2 and later)
* PowerShell (2008-R2 and later)

**No Graphical Interface Whatsoever?**

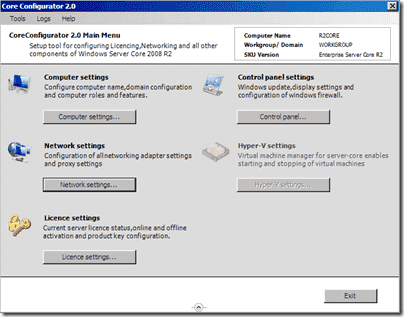
It is not entirely true that there is no graphical user interface with Server Core. There is a GUI desktop, but you can only run a CMD window, Task Manager, Notepad, and a few Control Panel applets by default (e.g., run “control.exe timedate.cpl” to change the time or “control.exe intl.cpl” for international settings). What you don’t get is the taskbar, Start menu, Administrative Tools, MMC consoles, desktop shortcuts, Control Panel, etc.





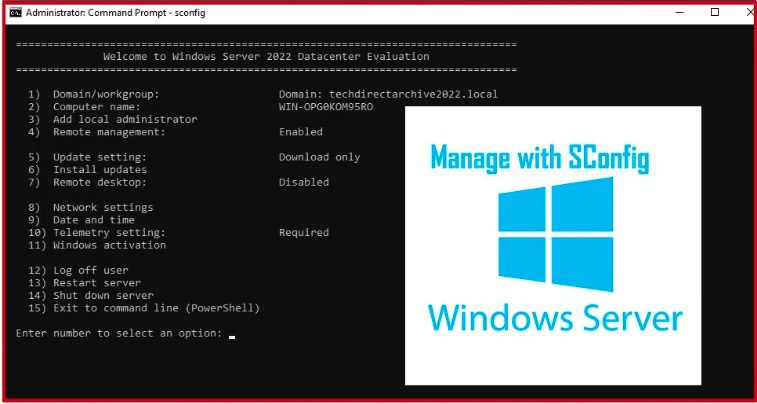
When you log on to a Server Core box, either at the local keyboard or via RDP, you get an empty desktop with a CMD shell window. If you close the CMD shell, you’ll have to hit Ctrl-Alt-Del to run Task Manager to relaunch CMD.EXE. From within CMD, you can run powershell.exe. However, because there is limited support for GUI applications on Server Core, others have been developing GUI tools for managing the box, such as:

* Server 2008 R1 and R2: Core Configurator (<http://coreconfig.codeplex.com>)
* Server 2012 and Later: Corefig (<https://github.com/ejsioron/Corefig>)



**SCONFIG.CMD**

On Server 2012 and later, there is also a configuration script (sconfig.cmd) that will prompt the administrator and perform several initial configuration tasks.



Once configured with an IP address and joined to a domain, you can connect over the network to a Server Core box with most MMC console snap-ins at your workstation. Hence, you can still use many of your favorite graphical tools to manage Server Core, just not while you are sitting at the box or while controlling it remotely through RDP.

**Minimal Server (Server 2012 and Server 2012 R2 only)**

On Server 2012 and Server 2012 R2, there is a third option: Minimal Server. This is actually not an official installation option, but rather it has the ability to add/remove certain graphical interface components on the fly without the necessity of reinstalling the OS (though you will need to reboot). You can start with a Full GUI install, then demote back down to Minimal or Core; or you can start with a Core install and add components to get back up to Minimal or Full. Minimal Server includes support for locally running Server Manager, the MMC.EXE console and its snap-ins, and most Control Panel applets. Internet Explorer, the desktop, and File Explorer are not included in Minimal mode, but they can be added back if the Full mode is restored (specifically, if the “Server-Gui-Shell” component is added back).

What about Server 2016? Officially, there is no such thing as “Server Minimal” for Server 2016, but, on the other hand, it is not true that you cannot run any graphical tools on Server Core at all, and the list of GUI tools you can run on Server 2016 and later is growing as Microsoft releases updates. There is no way to know for sure which GUI tools can and cannot be run other than doing internet searches and [testing].

To reduce from Full mode to Minimal on Server 2012: *Remove-WindowsFeature Server-Gui-Shell*

To reduce from Minimal to Core:

*Remove-WindowsFeature Server-Gui-Mgmt-Interface*

To reduce from Full GUI down to Core with a single command: *Remove-WindowsFeature Server-Gui-Shell, Server-Gui-Mgmt-Infra*

To go from Core back up to full is more difficult because you will likely need to provide the path to the install.wim file on the source DVD or ISO file. If the following command shows “Removed” for the Install State, the binaries are not present on the local drive will need to be copied from the DVD: ***Get-WindowsFeature Server-Gui\****

If the binaries are not present on the drive, they’ll have to be copied from the DVD or ISO file. You will need the index number of the correct image from the source DVD (drive d:\). If your source is an ISO file, just double-click or execute the name of the file in PowerShell in order to mount that ISO file as a drive letter.

To list the index numbers of the images in the source WIM file (replace d:\ with yours):

***Get-WindowsImage -ImagePath d:\sources\install.wim***

To list the index numbers of the images in the source WIM file using dism.exe instead:

***Dism.exe /get-wiminfo /wimfile:d:\sources\install.wim***

Here are the normal index numbers of the images in install.wim:

Index 1 = Server Core Standard

Index 3 = Server Core Datacenter

Index 2 = Full GUI Standard

Index 4 = Full GUI Datacenter

To go from Core back up to Full GUI Standard (Index 2): ***Install-WindowsFeature server-gui-mgmt-infra, server-gui-shell -source:wim:d:\sources\install.wim:2***

To go back from Core back up to Full GUI Datacenter (Index 4): ***Install-WindowsFeature server-gui-mgmt-infra, server-gui-shell -source:wim:d:\sources\install.wim:4***

**Server Core App Compatibility FOD (Server 2019 and Later)**

Starting with Windows Server 2019, there is another option for Core that is very similar to the Server Minimal concept, namely, you can install several graphical configuration and troubleshooting tools on Core without reinstalling the entire OS. A “Feature On Demand (FOD)” is not necessarily the same thing as a feature in Server Manager. Unfortunately, the word “feature” now has two slightly different meanings for Windows Server. There are FOD features that might not be listed in Server Manager, and there are roles and features in Server Manager that are not available as FOD features. Depending on the Windows Server OS version you are managing, you’ll have to research the best way to install the features you want in PowerShell.

**FOD Packages**

A FOD is a software package that can be installed at any time or added to a pre-installation OS image, such as to a WIM image or to a VHDX virtual machine image. FOD package is composed of one or more CAB files. A CAB file is similar to a ZIP archive. The CAB file for a FOD can be downloaded manually from Microsoft’s website (usually as part of an ISO archive with a bunch of other CAB files in the ISO too) or downloaded through Windows Update.

**DISM**

FOD features can be managed through the DISM PowerShell module or with the DISM.EXE command line tool. The DISM.EXE tool is available for Windows 7, Server 2008 R2, and later as part of the Windows Assessment and Deployment Kit (Windows ADK). The DISM.EXE tool is also built into Windows 10, Server 2019, and later by default. There are many PowerShell cmdlets from the DISM module and many command line arguments for the DISM.EXE tool. They are both very similar. Importantly, whenever there is an argument or parameter with the word “online” as part of it, “online” refers to the currently running operating system as opposed to an offline WIM, FFU, or VHDX image file. To show the DISM module cmdlets and/or command line arguments for DISM.EXE: ***Import-Module -Name DISM***

***Get-Command -Module DSM***

***Dism.exe /?***

To list the currently installed FOD features on your running (“online”) computer: ***Get-WindowsOptional Feature -Online; dism.exe /online /get-features***

**Server Core App Compatibility FOD**

How does this relate to Server Core? For Windows Server 2019 and later, there is a FOD named “Server Core App Compatibility” that includes at least the following tools:

* Windows PowerShell ISE (powershell\_ise.exe)
* File Explorer (explorer.exe)
* Microsoft Management Console (mmc.exe)
* Device Manager (devmgmt.msc)
* Disk Management (diskmgmt.msc)
* Event Viewer (eventvwr.msc)
* Failover Cluster Manager (cluadmin.msc)
* Local Users and Groups (lusrmgr.msc)
* Resource Monitor (resmon.exe)
* Performance Monitor (perfmon.exe)

After installing the above Server Core App Compatibility FOD, there is even an optional FOD that installs the Internet Explorer browser!

At the present time, the ISO archive that contains the Server Core App Compatibility FOD must be downloaded from Microsoft’s website manually. It will be listed on the same page that has the download link for the Windows Server installation ISO. If not, do an internet search on the phrase “Server Core App Compatibility FOD” to get the latest link. Once the downloaded ISO is mounted as a drive letter, perhaps as drive D:, run the following command to install version 1.0 of the App Compatibility FOD:

***Add-WindowsCapability -Online -Source D:\ -LimitAccess -Name “ServerCore.AppCompatibility~~~~0.0.1.0”***

As new versions of the FOD are released, change the above version number as instructed on Microsoft’s website. The -LimitAccess switch above, by the way, indicates that Windows Update should not be queried over the internet for the FOD files. A reboot will be required after installation.

After rebooting, launch any of the above graphical tools from within the command shell after logging on. Running EXPLORER.EXE provides File Explorer, but not the Start Menu or a taskbar. Learning about FOD and the DISM commands is important not just for Server Core but for Windows desktop computers too. Windows 10 and later includes the DISM.EXE command line tool and the DISM PowerShell module by default.

**Server Nano**

**Server Nano (Server 2016 and Later):**

* Can only be run as a Docker-style container OS image.
* Cannot run directly on hardware or as the only OS in a VM.
* No graphical desktop whatsoever (runs “headless”).
* Cannot be patched; it can only be replaced with a new image.
* Intended for fast DevOps deployments, like for web applications.
* Requires a Software Assurance agreement with Microsoft.
* Only supported if you have the current or prior update.

**What about the security benefits of Core or Nano?**

Server Nano

Windows Server 2016 introduced a new installation option: Server Nano. Originally, Server Nano could be installed directly onto hardware or as the only OS in a virtual machine. In 2017, however, Microsoft changed the distribution and licensing terms of Server Nano so that it could only legally run as a Docker-style OS image, preferably on top of Server Core. In this scenario, Server Core would be installed on the hardware or as the OS in a VM, then Server Nano would run as a container on top of Server Core.

**What Is a “Container?”**

Support for containers is built into Server 2016, Windows 10, and later operating systems. A “container” is similar to a very lightweight, fast-to-launch virtual machine checkpoint. Recall that a VM might have multiple checkpoints, with each checkpoint layered on top of zero, one or more earlier checkpoints of the VM. By analogy, what if there were only checkpoints and no VM to begin with? A container image is like a layering of one or more checkpoints. But layered on top of what?

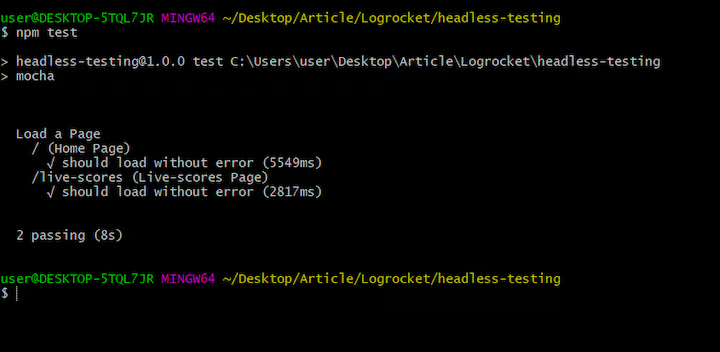
A container on Windows can either be a 1) shared kernel or 2) Hyper-V wrapped.

Traditional containers layer on top of the kernel of the host computer’s PS; hence, the host OS and the container share the binaries and other resources of the host computer’s OS. When the container attempts to make a change, that change is intercepted and written somewhere else in a way that maintains the illusion (to the container) that the change attempt succeeded. The container is like a sandbox. But because sandboxes can fail to trap malware and malicious commands inside of them, another option is to “wrap” the container using Hyper-V virtualization techniques. This is not the same thing as a whole Hyper-V virtual machine, but it does use isolation and management techniques built into the Type-I hypervisor of Hyper-V. What is a “Type-I hypervisor”? Well, that can’t be covered here. For more information about container support built into Windows Server, please see the tutorials and how-to guides here:

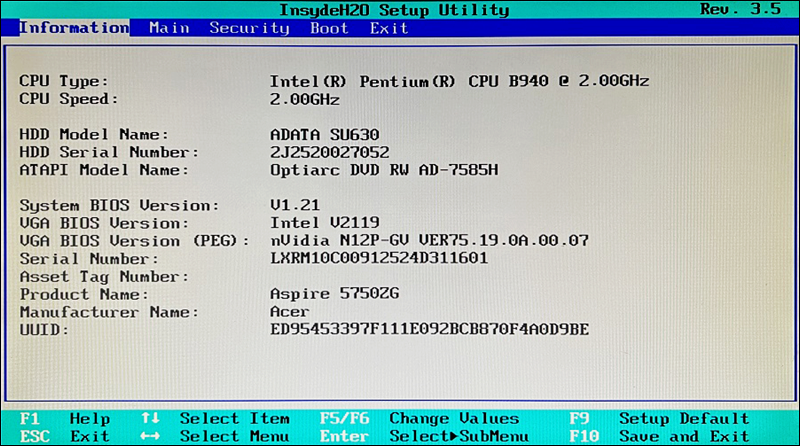
* <https://www.microsoft.com/en-us/cloud-platform/containers>
* <https://www.docker.com/partners/microsoft>

**Tiny Footprint and “Headless”**

At the time of this writing, the base container image of Server Nano is only about 80 MB in size! (That is not a typo.) To achieve this radical reduction in footprint size, Nano has no GUI support whatsoever, i.e., it runs “headless”, and it has extremely few services and features by default. If you need Powershell, it has to be added to the base container image, but ideally you won’t “manage” a Server Nano instance; you’ll update the template and PowerShell scripts used to deploy the image and then just create a whole new fresh one! Because the whole deployment process can be automated and because spawning fresh container instances is so fast, Server Nano is more like launching an application than running an OS. Welcome to the brave new world of DevOps!



The above screenshot is the all-text logon screen of a Nano image. And the following screenshot is what you see after logging on. Notice, there is no desktop, just some basic OS configuration information, similar to what we had with old BIOS firmware.



Using the textual menu system, you can configure IP networking settings, enable firewall rules, enable WinRM for PowerShell remoting, and not much else. Any necessary administration is done remotely over the network.

**Don’t Patch: Redeploy**

Server Nano cannot be patched. When Server Nano is updated with new features or fixes, Microsoft releases a whole new container image. After your in-house testing confirms success, the new image is incorporated into one’s deployment pipeline and a newly updated container is spun up to replace the image that is currently running.

**Server Nano Rules**

Server Nano is mainly intended for hosting web applications, such as [ASP.NET](http://asp.net) applications on the .NET Core Framework, though other roles are certainly possible. Originally, Microsoft stated that Server Nano was great for infrastructure roles too, like being an SMB file server or Hyper-V host, but they changed their minds in 2017.

**Licensing and Support Headaches**

Sound too good to be true? Be aware that there are some Nano headaches:

* You must either have a Software Assurance agreement with Microsoft or run your Server Nano containers in Azure (this is what they *really* want).
* You cannot legally install Server Nano directly onto hardware or as the only OS inside a VM. This might be possible, but it’s not supported. You must use containers.
* There are no long-term support options; hence, you must plan to regularly apply new container images, perhaps as often as twice per year (it depends on how quickly Microsoft releases new container images).
* Each new image will have all prior patches and fixes installed, just like a cumulative update; hence, each updated image “rolls up” all prior updates into one all-or-nothing container that must be used or skipped as a whole.
* There will be approximately two to three new container images per year, and Microsoft will only support you if you have either the latest or the prior image deployed; hence, if your last-deployed Nano image is more than two releases behind, Microsoft will not support you or help troubleshoot any Nano problems!
* Server Nano does not support Group Policy, and it is unknown if Microsoft will allow containers to be joined to an Active Directory domain at all. Nano is intended to be managed directly through PowerShell or similar automation tools.
* In the past, Windows Server was licensed per-processor, but with Server 2016 and later, it is now per-core (which may be good or bad for you – it depends). How will this impact the licensing of Server Nano images? Who knows?! Microsoft is fickle, so you’ll have to double-check your licensing at least once per year. Whatever the licensing scheme will be tomorrow, expect the deal to be sweeter if you move to Azure, which is really Microsoft’s ultimate aim now.
* Nano only supports 64-bit applications.

Clearly, Server Nano is installed for rapid DevOps-style management, especially for web applications, microservices, cloud-hosted VM appliances, and the like. Hence, use Nano if you can, but you’ll often need to use Server Core instead, especially inside the LAN for on-premises servers.

**Server Core / Nano Benefits**

These are the main benefits of using the Server Core or Nano installation option:

* Because a Server Core/Nano installation has fewer applications and services, it has fewer software components that can fail, lock up the system, waste memory, have bugs, listen on TCP/UDP ports, or otherwise be attacked and compromised, i.e., it has a smaller “attack surface”. Stripping away unnecessary roles, features, components, and bells and whistles is an essential part of making a hardened server.
* Fewer roles, features, and components means fewer components and reboots per year.
* A Server Core/Nano installation requires less hard drive space. When combined with data deduplication, a large number of VMs or containers can be crammed into one storage volume.

**Server Core / Nano Drawbacks**

However, there are some drawbacks and important issues to consider:

* Server Nano has restrictive licensing and support requirements.
* When you uninstall GUI components, third-party software that is installed and that requires these components may no longer function correctly.
* When moving from Core up to Full on Server 2012 and 2012 R2, be aware that several gigabytes of binaries will need to be installed either over the internet or from local media. Also, if updates are applied after moving to Full, several more gigabytes of patches will also need to be downloaded and applied.
* Most local administration is performed from the command line using PowerShell or tools like NETSH.EXE and SC.EXE, unless you install a third-party GUI tool on Server Core (no such option on Nano). Graphical tools can still be used remotely over the network, unless there are networking problems of course.
* If there are networking problems, you cannot use your favorite GUI tools to remotely manage Server Core/Nano. You are back to the console and whatever GUI tools that are compatible with Core (and none will be compatible with Nano).
* There are no graphical notifications of password expiration, new patch updates, or impending product activation deadlines when at the console on Server Core.
* To install the .NET Framework on Server Core, you must have Server 2008-R2 or later. Without the .NET Framework, no [ASP.NET](http://asp.net) and no PowerShell. Hence, Server 2008-R2 is the realistic minimum OS version.
* Only SQL Server 2012 and later is supported on Server Core, not earlier versions.

**Why Is Server Core/Nano More Secure?**

Server Core/Nano is nice for virtual machines, but the *security* benefits of Core or Nano is less important than other factors. Make sure your managers understand that simply using Core or Nano will not, by itself, be a magic silver bullet.

Here are some security implications regarding Server Core or Nano:

* A well-managed network is more secure than a poorly managed one. Windows administrators tend to be less proficient working at the command line than working with graphical tools (many hate the command line). Hence, as more Server Core/Nano boxes are added to the network, the quality of the management of the network as a whole may suffer.
  + Example: One reputed benefit of Server Core/Nano installations is that they are “like appliances, so you can deploy and forget them!” This is false, but the opinion will be very popular since Microsoft is one of the culprits spreading the notion. But these forgotten, unpatched, unmonitored, and unaudited Server Core/Nano boxes will now become more attractive targets for hackers and safe havens for malware.
  + Example: In the confusion and panic of a malware outbreak or successful penetration, there is no time to research which command line tools do the same things as familiar graphical tools; hence, some environments will respond less quickly and effectively to emergency incidents as more Server Core/Nano boxes are added to these environments. Plus, what if the security tool you must run to save the day only comes in GUI form and cannot be run over the network?
* Server Core boxes must also be patched, even if the patching is less frequent than on full install boxes. But the hard part of patch management is the setup and maintenance of the patching infrastructure, not the addition of new servers or the installation of a few more patches. Installing fewer patches on some of the servers doesn’t reduce the workload much, unless you are doing it all by hand to begin with. And Server Nano containers are not patched; they are replaced, but new container images must be tested before deployment too. A container image is like a roll-up or cumulative patch applied to a pre-built VM. Testing is still necessary.
* Like full installation machines, Server Core/Nano boxes must be hardened before deployment, even if there are fewer components on the system to harden. The assumption that you don’t have to put roughly the same amount of hardening effort into the Core boxes than the regular servers is often false. The base Nano container image is very minimal, but what about what’s layered up top?
* Whether a box has the full or Core/Nano installation of Windows Server, the more roles and features you add, the more listening ports and protocols the box will have. Is a Server Core box with five roles more secure than a full installation with only one role? Or consider it a different way. Both a full installation and a Server Core installation of IIS might listen on TCP ports 80/443 (HTTP/HTTPS), 21/20 (FTP), 139 (SMB), 445 (SMB/CIFS), 135 (RPC), and 3389 (RDP/Remote Desktop Services),and all the same services are bound to these ports on both the full and Core installations. To a hacker who is port scanning, the full and Core installation boxes will almost look identical if they have the same roles and features installed. Granted, the Core box has fewer components overall, and this is good for security, but the *dangerous* components are the ones that get installed as roles or features with listening ports; hence, it seems that reducing roles and features is more important than using the Core option *per se*. Nano has fewer listening ports, but that’s because it supports fewer features. For apples to apples comparisons, we need to compare systems running the same roles and features and if a role or feature is installed, we assume it is necessary and needed.
* You cannot install graphical browsers, email programs, peer-to-peer file sharing applications, instant messaging clients, or other such malware magnets on Server Core/Nano; hence, the average infection rate from these vectors should be lower on Core/Nano boxes. But through corporate security policies and other security controls, you can forbid, prevent and detect negligent administrators from doing stupid things than to embrace Server Core/Nano just to achieve the same end result.

Finally, this really isn’t a security issue, but Microsoft sometimes says that Server Core is great for remote branch offices that do not have local IT personnel since you can always get remote command line control of the box. But what if you can’t? What if you have to get some non-tech who can barely type out there on the phone and walk them through the entire troubleshooting process *using only a command shell?* The horror…

In short, while the Server Core/Nano option has gotten lots of great press, the reality is that it is more important to remove unnecessary roles, and to quickly apply new patches or new container images, than it is to just simply be running Core/Nano as such. Don’t let your managers believe that Server Core or Nano is automatically (or magically) secure because they don’t have graphical desktops. Not true for Linux, not true for Windows Server.

If so, then how do we strip away unnecessary roles and features?

**Remove Unnecessary Roles and Features**

**What are “roles” and “features”?**

* It’s Microsoft untangling decades of spaghetti code by organizing the OS into dependency layers and manageable units.

**With Server Manager:**

* (Un) install roles, features, and role services.
* Manage roles on local and remote servers.
* Works with offline Hyper-V images (VHD/VHDX).
* Manage groups of remote servers for bulk changes.

Remove Unnecessary Roles and Features

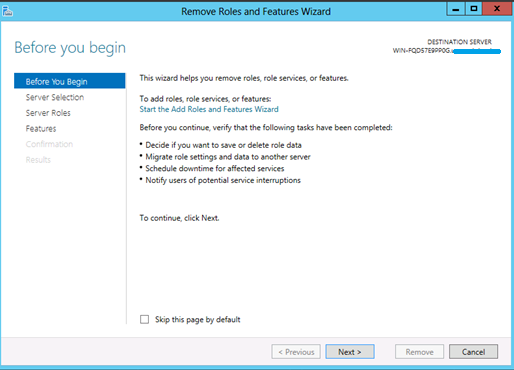
Windows Server 2008 and later versions are highly modular in the design, allowing very precise control over the features and components installed. In Server 2012 and later, the Server Manager tool can manage roles and features of remove machines over the network, including custom groups of servers simultaneously, and can install or uninstall roles and features from offline Hyper-V virtual machines that are not even running.

Server Manager can (un) install roles and features on remote Server 2008 and later operating systems, but there are special requirements for the older boxes. The older systems must have Server 2008 SP2, Server 2008-R2 SP1, .NET Framework 4.0 or later, PowerShell 3.0 or later, a special hotfix (KB2682011), and run a configuration script named Configure-SMRremoting.ps1 from Microsoft.

**Remove Unnecessary Roles and Features**

Many of the servers you will harden and expose to the internet will be IIS servers hosting applications like SharePoint, Outlook Web Access, OCSP, WSUS, FTP, and a variety of SOAP/REST applications for your tablets and smartphones. You will also have DNS, VPN, DirectAccess, Exchange, RDS, SMB, Active Directory, RADIUS, and other servers, but this manual will often focus on IIS because that’s what many organizations are most worried about.

So, as an example, if you needed to install IIS, you would open Server Manager > Manage menu, Add Roles and Features > follow the guidance of the wizard. When you select the Web Server (IIS) role, it includes many components or “role services”, but you will never need all of them at the same time on the server. As always, only install the components you need.



If IIS were already installed, but you wanted to remove unnecessary components, then go to Server Manager > Manage menu > Remove Roles and Features > select your server(s) > Web Server (IIS) section > uncheck the boxes for the unneeded components.

**What Is Minimal?**

But what are the minimal roles, features and role services for any given server? That depends on what you want that server to be able to do. Each server will have different purposes, so each server will be different. It all depends on the applications, services, and protocols needed. On an IIS web server, for example, if you just want to serve up static HTML and graphic files, then the following would be a minimal component set:

* Common HTTP Features: Static Content
* Common HTTP Features: Default Document
* Management Tools: IIS Management Console

But notice that the above list does not include support for logging, non-anonymous user authentication, authorization rules, request filtering, compression, or [ASP.NET](http://asp.net). These are things you might need on this or another server. Each server will be different. Remember, though, that you can start with a minimal set of components and then add more as needed very easily using the graphical Server Manager or the PowerShell cmdlets for the same. You don’t have to get that list exactly correct right from the beginning, you can always add more roles, features, and role services later on.

When building the server, do it on a protected subnet not directly accessible from the internet. Move the VM or physical server to the firewall’s DMZ only after the server has been fully configured and hardened. It is not unusual for new servers to be scanned by automated scripts within minutes of going live on the Internet (see the reports of such tests at <http://www.honeynet.org>).

**Server Manager Scripting with PowerShell**

**Your script as a server template:**

* Install or uninstall roles and features.
* Automate changes across many servers.
* Export XML list to install from Server Manager.
* Apply to local or remote machines.
* Apply to offline Hyper-V drive image files.
* Copy installation DVD/ISO to a shared folder.

**Remotely inventory all your servers:**

* Save as CSV, XML, or HTML report.

Server Manager Scripting with PowerShell

Instead of using the graphical Server Manager tool, on Server 2008 and later, you can view, install, and uninstall roles and features with PowerShell on both local and remote machines. This requires PowerShell 3.0 or later on both the local and remote computers. To see your currently installed roles and features:

***#Next command only required on 2008-R2 and earlier:***

***Import-Module ServerManager***

***Get-WindowsFeature***

To add or remove roles and features with PowerShell cmdlets, see the following: *Get-Help Install-WindowsFeature -Full; Get-Help Uninstall-WindowsFeature -Full*

When you run Get-WindowsFeature, a great deal of text will scroll by, but the layout is similar to what you see in the graphical Server Manager. The hyphenated name on the right is what would be used to (un)install components of an entire category of components for a role.

For example, to install IIS with all optional components and tools: ***Install-WindowsFeature -Name Web-Server -IncludeAllSubFeature -IncludeManagementTools -Restart***

Remember, these cmdlets work on offline Hyper-V image files and remote servers too: ***Install-WindowsFeature -Name BitLocker -Vhd .\ImageFilePath vhdx***

***Install-WindowsFeature -Name DNS -Computer Server47***

When a role is uninstalled, the -Remove switch deletes the unneeded binaries from the drive or Hyper-V image file, which reduces storage consumption and is slightly more secure because, if reinstalled again, a path to a presumably clean source can be given: *Uninstall-WindowsFeature -Name BranchCache -Remove*

If the server installation DVD is copied to a shared folder that grants Read permissions to Everyone (not Write!), then this UNC path can be given when installing roles: ***Install-WindowsFeature -Name DNS -Source \\server\share\Sources\SxS***

Server Manager XML Template

If there are many roles and features being installed at one time, you might prefer to use the graphical Server Manager to build a list of the additions, save the list to an XML file, then give the file as an argument to the Install-WindowsFeature cmdlet.

***Install-WindowsFeature -ConfigurationFilePath .\DeploymentConfg.xml***

To create this XML file with Server Manager, pull down the Manage menu, add roles and features like normal, but at the end of the wizard’s dialog boxes, don’t click the Install button; click the “Export configuration settings” link at the bottom instead to save the XML file, then cancel the wizard. Note that this XML file can only be used to install features, not remove them.

Group Policy: Set Default UNC Path or Allow Windows Update

Instead of providing the UNC path to the installation files as an argument, you can also set the default path through Group Policy. You can even have servers download the necessary files over the internet via Windows Update.

These options are controlled in the GPO setting named “Specify settings for optional computer installation and component repair”, which is located in a GPO under Computer Configuration > Policies > Administrative Templates > System.

**On Your Computer**

**Please turn to the next exercise… tab completion is your friend! F8 to *Run Selection***

On Your Computer

Display all roles and features on a local or remote server (scroll to the right also):

*Get-WindowsFeature -ComputerName $env:ComputerName | Select-Object -Property \* | Out-GridView*

Get only the roles and features currently installed:

*Get-WindowsFeature | Where { $\_,InstallState -eq “Installed” } | Select Name, DisplayName*

Get a particular role (for an Active Directory domain controller) and list the other roles it depends on and the status of the Windows services to which it is related:

*$role = Get-WindowsFeature -Name AD-Domain-Services*

*$role.DependsOn*

*$role.SystemService | Get-Service*

Query the roles and features from every server in the domain:

*Cd C:\SANS\Day4*

*.\Get-FeaturesInventory.ps1 | Export-Clixml -Path .\Servers.xml*

*$Servers = Import-Clixml -Path .\Servers.xml*

*$Servers*

For example, is the BitLocker feature installed on the first machine in the array?

*$Servers[0].Features.BitLocker*

**Add and Remove Roles**

Roles and features can be queried, but they can also be installed or uninstalled.

Install the role for the File Server Resource Manager (FSRM), plus any other necessary roles or features (called “subfeatures”), plus any management tools for it, and only if necessary, reboot the system (this role does not require a reboot to install):

*Install-WindowsFeature -Name “FS-Resource-Manager” -IncludeAllSubFeature -IncludeManagementTools -Restart*

Remove the XPS Viewer feature, which is installed by default: *Uninstall-WindowsFeature -Name “XPS-Viewer’*

You can see in the output of the command that it was successful and no reboot is needed. Attempt to remove the PowerShell version 2.0 engine (specifically, the DLL for it): *Uninstall-WindowsFeature -Name “PowerShell-V2”*

This also succeeded, but it was because no change was needed. PowerShell 2.0 was already not installed. It’s OK to try to (un) install a role already (not) installed; it won’t cause problems.

**Note:** Hackers love PowerShell 2.0 because of its lack of security and logging features in comparison to later versions. It’s OK to try to (un) install a role already (not) installed; it won’t cause problems.

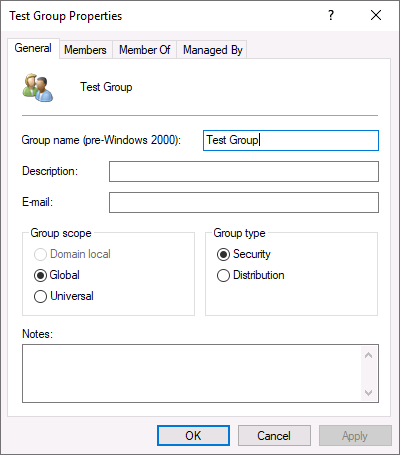
Delegation to Non-Admins

A user who is not in the local Administrators group on a server can be permitted to query role and service information from that server over the network. This delegation can be done with PowerShell Just Enough Admin (JEA), but it can also be done with WMI namespace permissions and other features used by Server Manager.

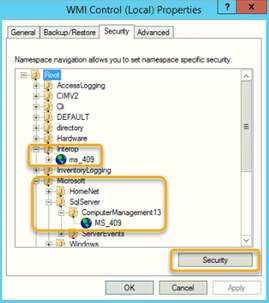
Briefly read the description of this command (use tab completion): ***Get-Help -Full Enable-ServerManagerStandardUserRemoting***

Use this command to grant a user (Billy Corgan) these remote audit permissions: ***Enable-ServerManagerStandardUserRemoting -User “testing\billy” -Force***

Billy has now been added to a few more built-in groups:



Billy has also been granted permissions on the CIMV2 namespace for WMI. This can be seen with the WMI Control snap-in in an MMC.EXE console. (Feel free to add that snap-in if you wish, but it’s not required for this lab.) Here is a screenshot from the WMI Control snap-in when viewing the Security tab properties of the Root\CIMV2 namespace:



This means that we do not need to add Billy to the Administrators group on this server in order for Billy to use the Get-WindowsFeature cmdlet over the network to query this server (it’s read-only access). Billy can also use Event Viewer and Server Manager to query this server over the network. We could have made these the same group membership and permissions changes with our own scripts instead of using Microsoft’s built-in command, but it works OK. (You can’t grant permissions to a group with it, only to individual users.)

Remove Billy from the above groups: *Disable-ServerManagerStandardUserRemoting -User “testing\billy” -Force*

**[End of Lab]**

**INF File Security Templates**

* **Just use a text file with security settings.**
* **Used to automate reconfiguration.**

1. **Run MMC.EXE > 2. File menu > Add/Remove > 4. Security Templates**

**What’s in a template?**

* **Password Policy**
* **Account Lockout Policy**
* **Kerberos Policy**
* **Audit Policy**
* **User Rights Assignments**
* **Security Options**
* **Event Log Settings**
* **Restricted Groups**
* **System Services**
* **Registry Key Permissions**
* **Filesystem Permissions**

INF File Security Templates