

Securing Windows 2000 Communications with IP Security Filters, Part 2

Joe Klemencic 2002-04-10

Securing Windows 2000 Communications with IP Filters: Part Two

by *Joe Klemencic*

last updated April 10, 2002

This is the second part of a two-part series on implementing Windows 2000 IP Security filters. In the [first article](#), we offered an overview of IP security policies, including defining, testing, and expanding IP security policies. In this installment, we will be discussing encryption of Windows systems and implementing IP security filters.

Authentication Methods

The following discussion will pertain to the native Windows 2000 Kerberos for authentication. To enable Kerberos authentication, your machine must be part of a domain. If it is not part of a domain, you have the option of either using a Certificate from a Certificate Authority or a Preshared Key. If you are not in a domain and do not have a local Certificate Authority, some minor changes will need to be made before proceeding. Open up the Group Policy editor snap-in in the Microsoft Management Console (MMC) or simply launch GPEDIT.MSC from the Start menu and double-click on the IP Security policy you created and activated. Highlight the "Allow Terminal Services IP Security" rule and click the "Edit" button. You will be presented with the Properties window for this filter. Click the "Authentication Methods" tab. You should see the default authentication method of Kerberos. Click the "Add" button and select the "Use This String To Protect The Key Exchange" radio button. Enter a string text that you wish to use as the preshared key then press "OK". Please note that this preshared key is stored and viewed in clear text and must be configured on every host that will be communicating over the encrypted connection to your host. When you are back at the "Authentication Methods" list screen, highlight the "Preshare Key" entry and click the "Move Up" button to move this entry to the first available authentication method. This will reduce the connectivity time and will eliminate any errors due to Kerberos authentication failing.

Adding Encryption to Select Services

Building on the IP Security filters we previously created, we will add encryption to our Terminal

Services filters to ensure secure communications. Open up the Group Policy editor snap-in in the Microsoft Management Console (MMC) or simply launch GPEDIT.MSC from the Start menu and double-click on the IP Security policy you created and activated. Highlight the "Allow Terminal Services" IP Security rule and click the "Edit" button. You will be presented with the properties window for this filter. Click the "Filter Action" tab. You should see the "Permit" radio button selected, as built in the last section. In this example, we are going to require all allowed Terminal Server sessions to be encrypted. Select the "Require Security" radio button and click the "Edit" button. You are presented with options to change the "Security Methods" for this filter. Since we want to only accept encrypted connections, and not allow unencrypted connections, uncheck the "Accept Unsecured Communication" box. This properties box also allows you to change the encryption algorithms; but for this activity, we will retain the default settings, as shown in figure 10.

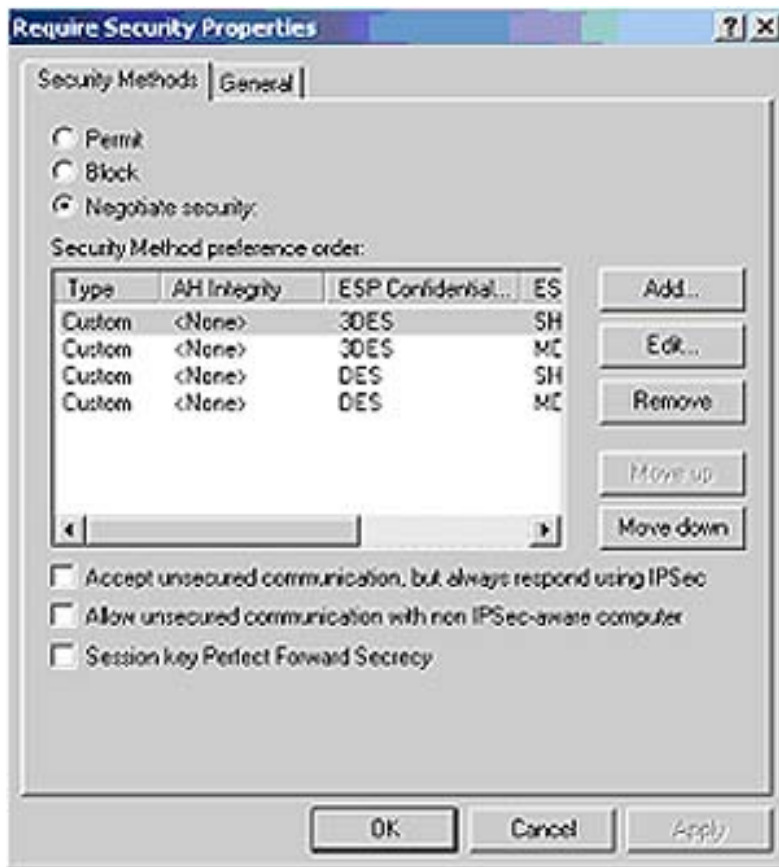


Fig 10: Security Methods

Click "OK" and close out the "Property" windows. You have now required all allowed Terminal Service connections to be encrypted with IPSEC. Since "Require Security" is enabled on your host, a special set-up will also be required on all clients that wish to connect to your host.

Configuring the Clients

The following simple change will be required on each host from which you wish to access your host's Terminal Service. Since you are requiring security, you are limiting the connections to Windows 2000 hosts only, as this is the only version of the Windows operating system that supports the IPSEC features. Open up the Group Policy Editor on an allowed host and click on the "IP Security Policies" on "Local Machine" under the Computer Configuration's Security Settings.

Preshared Key Authentication Method

If you previously set up a preshared key, double-click on the "Client (Respond Only)" policy, highlight the "Default Response" rule and click the "Edit" button. Select the "Authentication" tab and add your preshared key to the key you defined previously. Remember to use the "Move Up" button to move the preshared key to the top of the "Authentication Method Preference Order" list.

Default Kerberos Method

If your client is in a domain and you are using the default Kerberos authentication method on your target host, no special set-up is necessary.

Enabling the Client Policy

To enable the client to utilize encryption to your host, simply right-click on the "Client (Respond Only)" policy and select the "Assign" menu item. A green plus (+) sign should appear on the folder icon next to the IP Security policy, indicating that this policy is active.

Testing the Policy

To ensure that the Terminal Service connection from your client to your host is actually encrypted using this policy, Microsoft has provided an IPSEC monitoring tool with Windows 2000. Simply run IPSECMON.EXE to launch the IPSEC monitoring application. The application's monitoring window is automatically refreshed every 15 seconds, and can be changed through the "Options" button. A status in the lower-right corner should indicate that IP Security is enabled on this computer. If it does not, make sure the policy is assigned in the Group Policy Editor. Launch a Terminal Services connection from the client to your host. The IPSEC

Monitoring utility should now indicate a connection in its monitoring window. Figure 11 indicates a successful IPSEC connection.

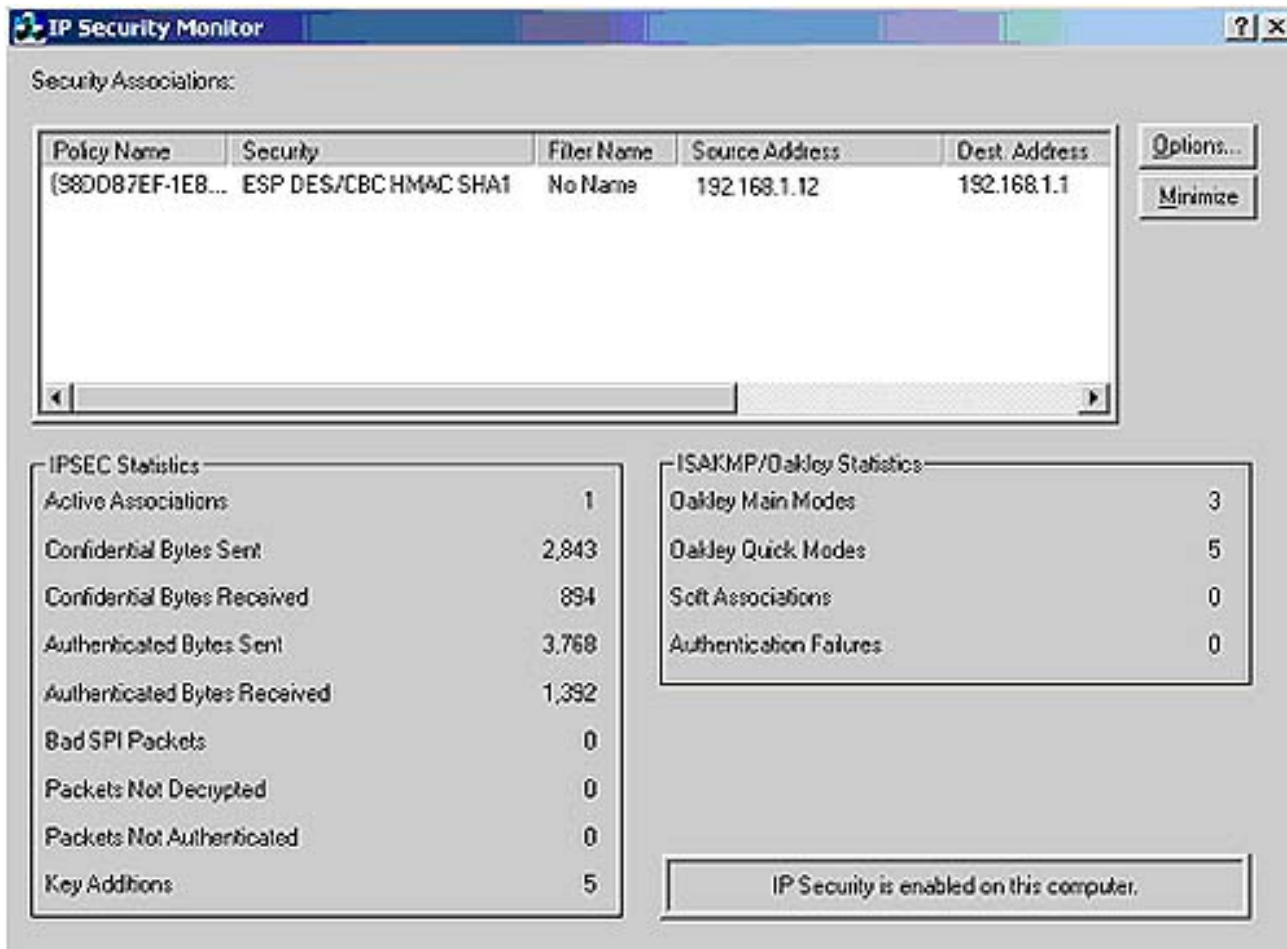


Fig 11: Successful IPSEC Negotiation

Domain Controller to Domain Controller Secure Communication

Since each filter you set an encryption filter action on will create a new encryption session, a host may be bogged down with encryption duties and degrade performance. Replacing the network card with a card that will perform IPSEC operations in hardware will reduce the CPU overhead on the server, limiting encryption to only certain critical services will also reduce CPU overhead. But what if you want to secure the communications between domain controllers to prevent replication from unauthorized hosts? What if you need to replicate to another domain controller that is behind a firewall? In the case of the firewall, one would need to open up numerous ports in the firewall(s) to support the RPC requests for replication, not to mention the NetBIOS ports and any other ports Windows utilizes for domain controller communications. However, a more firewall-friendly solution is available, one that will thwart connectivity from

Forcing Replication to a Single Port

By default, Windows will replicate Active Directory changes to other domain controllers over RPC. The RPC service runs the associated services on random ports above 1024. This makes it very difficult to restrict someone from promoting a Windows 2000 server to Domain Controller status and performing unauthorized replication with the Windows 2000 Active Directory domain. However, it is possible to restrict the port used for replication and create IP Security filters that will permit only defined domain controllers to replicate with each other. To restrict the port that Active Directory replication uses, the following key must be created on every domain controller using REGEDIT:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\NTDW\Parameters  
Registry Value: TCP/IP Port  
Value Type: REG_DWORD  
Value Data: *
```

*The IANA (Internet Assigned Numbers Authority) specified range for private ports is 49152-65535

After setting this key, the domain controllers must be rebooted in order for it to take affect. Once this key is set and activated, the Windows 2000 domain controllers will utilize the defined method (RPC, IP, or SMTP) to contact its peer domain controller and will establish a connection on the defined port for replication. This, in itself, minimizes the ports required to be open in a firewall and allows for the creation of IP Security filters to prohibit unwanted replication attempts. Later, we will see how we can further reduce the ports that are open through a firewall.

Restricting Replication with Filters

Now that the domain controllers are replicating over a defined port, you can create the IP Security filters to allow the communication between authorized domain controllers while denying replication from unauthorized sources. These filters will need to be installed on all domains controllers. Since these are explicit filters, each domain controller will need the appropriate filters to be defined for each domain controller to which it replicates. This makes the initial set-up lengthy, but provides for the most secure configuration. Please also note that if you attempt to bring up a new authorized

domain controller into a domain with these filters, it will not be able to replicate unless specific filters permitting its communication are defined on all the domains controllers.

In this example, we have two domains controllers and have set the registry key on each to force replication on port 50000:

- DC1 has an IP address of 10.1.1.1
- DC2 has an IP address of 10.2.2.2

We will create the following IP Security filters on DC1 (by now, you should be familiar with creating IP Security Filters). In both of these examples, we will use the Mirror option (which is enabled by default):

Filter Name: Replication with DC2

- Source: My IP Address
- Destination: 10.2.2.2
- Protocol: TCP
- Destination Port: 50000
- Filter Action: Require Security
- Source: 10.2.2.2
- Destination: My IP Address
- Protocol: TCP
- Destination Port: 50000
- Filter Action: Require Security

Filter Name: Deny Unauthorized Replication

- Source: ANY
- Destination: My IP Address
- Protocol: TCP
- Destination Port: 50000
- Filter Action: Block

Consequently, we will also create the following IP Security filters on DC2:

Filter Name: Replication with DC1

- Source: My IP Address
- Destination: 10.1.1.1
- Protocol: TCP
- Destination Port: 40000
- Filter Action: Require Security
- Source: 10.1.1.1
- Destination: My IP Address
- Protocol: TCP
- Destination Port: 50000
- Filter Action: Require Security

Filter Name: Deny Unauthorized Replication

- Source: ANY
- Destination: My IP Address
- Protocol: TCP
- Destination Port: 50000
- Filter Action: Block

Once these IP Security filters are defined and assigned to a policy, simply assign the policy through the Group Policy Editor and launch the IPSEC Monitor application (IPSECMON.EXE). During the next replication, you should notice a connection established in the IPSEC monitor window.

Encrypting All Domain Controller Communications

The previous example still requires a handful of ports to be open in a firewall and is not entirely secured against sniffing if replication takes place over a public network such as the Internet. To go one step further, we will be encrypting *all* communication between domain controllers with IP Security policies, thus requiring even less ports to be open through a firewall, further ensuring that it will be difficult for unauthorized clients to decode the communications.

Given the filters that we previously created on servers DC1 and DC2, we will modify the IP Security filters on both DC1 and DC2:

DC1 Filter Name: Replication with DC2

- Source: My IP Address
- Destination: 10.2.2.2
- Protocol: TCP
- Destination Port: ANY
- Filter Action: Require Security
- Source: 10.2.2.2
- Destination: My IP Address
- Protocol: TCP
- Destination Port: ANY
- Filter Action: Require Security

DC2 Filter Name: Replication with DC1

- Source: My IP Address
- Destination: 10.1.1.1
- Protocol: TCP
- Destination Port: ANY
- Filter Action: Require Security
- Source: 10.1.1.1
- Destination: My IP Address
- Protocol: TCP
- Destination Port: ANY
- Filter Action: Require Security

On both DC1 and DC2, we will retain the "Deny Unauthorized Replication" filters to prevent unauthorized replication from other domain controllers. The filters we just created will encrypt *all* communications between DC1 and DC2, while blocking connections to port 50000 from any other hosts. However, some services, such as Kerberos authentication between the domain controllers, which is considered a secure authentication mechanism, and Multicast, are not encrypted by these filters.

If we were to secure a third domain controller's communications, we would need to build the same IP Security filters with the IP address of the third domain controller on each of the existing domain controllers. Please note that there may be difficulties with promoting

a new domain controller into this configuration without first creating the appropriate IP Security filters on each existing domain controller. To briefly describe the filters to be created on each existing domain controller to allow replication with the new Windows 2000 server being promoted:

DC1 Filter Name: Replication with DC3

- Source: My IP Address
 - Destination: 10.3.3.3
 - Protocol: TCP
 - Destination Port: ANY
 - Filter Action: Permit
-
- Source: 10.3.3.3
 - Destination: My IP Address
 - Protocol: TCP
 - Destination Port: ANY
 - Filter Action: Permit

As you can see, we simply created a new IP Security Filter and changed the Filter Action from "Require Security" to "Permit". Once the new domain controller is promoted and has replicated, you will need to create the appropriate IP Security filters on it to communicate with the other domain controllers, then go back to each existing domain controller and change the Filter Action from "Permit" to "Require Security".

Implementation Suggestions

By now, you should be familiar with creating and implementing IP Security filters and associated encryption methods. But what should you implement in your environment? Should you encrypt all the communications, but increase CPU usage and restrict which clients can connect, or just encrypt some of the important services? Or do you want to simply restrict certain services to select clients, while maintaining open connectivity to the other services? Below, we will look at a few of the common scenarios and suggest a filter scheme that will best fit the task.

General Usage Server (Web Server)

For a general use server, such as a Web server, you would not want to utilize IPSEC encryption; rather, you would want to reduce the scope of clients that can use the Web server versus the other services on the server (such as drive shares). For example, if you had a Web server that was to serve Web pages to the general Internet community, but also had the administrative drive shares enabled for administration, you could create a default "Deny" ruleset allowing only HTTP traffic in from the Internet while allowing anything else from the local subnet:

Policy Name: My Web Server

Filter Actions Required: Permit, Block

Filter 1 Name: Allow HTTP from Internet

- Source: Any
 - Destination: My IP Address
 - Protocol: TCP
 - Destination Port: 80
 - Filter Action: Permit
-
- Source: My IP Address
 - Destination: Any
 - Protocol: TCP
 - Source Port: 80
 - Filter Action: Permit

Filter 2 Name: Allow Everything From Local Subnet

- Source: IP: 10.0.0.0 Subnet Mask: 255.0.0.0
 - Destination: My IP Address
 - Protocol: Any
 - Filter Action: Permit
-
- Source: My IP Address
 - Destination: IP: 10.0.0.0 Subnet Mask: 255.0.0.0
 - Protocol: Any
 - Filter Action: Permit

Filter 3 Name: Deny Anything Not Defined

- Source: Any
- Destination: My IP Address
- Protocol: Any
- Filter Action: Deny

Windows Terminal Server

In this next scenario, we have a Web server with Microsoft Terminal Services installed for easy management from the local subnet. Since it is not a good idea to expose Terminal Services to the Internet, and since this server has had all the listening services disabled with the exception of the Web service and the Terminal Services, we will utilize the default "Allow" ruleset:

Policy Name: My Web Server

Filter Actions Required: Permit, Block

Filter 1 Name: Allow Terminal Services from Local Network

- Source: IP: 10.0.0.0 Subnet Mask: 255.0.0.0
 - Destination: My IP Address
 - Protocol: TCP
 - Destination Port: 3389
 - Filter Action: Permit
-
- Source: My IP Address
 - Destination: IP: 10.0.0.0 Subnet Mask: 255.0.0.0
 - Protocol: TCP
 - Source Port: 3389
 - Filter Action: Permit

Filter 2 Name: Deny Terminal Services from Everywhere Else

- Source: Any
- Destination: My IP Address

- Protocol: TCP
- Destination Port: 3389
- Filter Action: Block

- Source: My IP Address
- Destination:
- Protocol: TCP
- Source Port: 3389
- Filter Action: Block

Intranet Domain Controller Communications

The following scenario is an example of an IP Security policy that may be used between domain controllers located within your organization and that do not replicate over a public network. In this scenario, we simply want to limit the replication to the known defined domain controllers:

Policy Name: My Domain Controller

Filter Actions Required: Permit, Block

Other Requirements: Set the registry key

(HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\NTDW\Parameters to a specific port for replication.

Filter Name: Replication with DC2

- Source: My IP Address
- Destination: 10.2.2.2
- Protocol: TCP
- Destination Port: 50000
- Filter Action: Require Security

- Source: 10.2.2.2
- Destination: My IP Address
- Protocol: TCP
- Destination Port: 50000
- Filter Action: Require Security

Filter Name: Deny Unauthorized Replication

- Source: ANY
- Destination: My IP Address
- Protocol: TCP
- Destination Port: 50000
- Filter Action: Block

Communication Through a Public Network

In this final scenario, we will be attempting to send all communications through a firewall, destined for a host across a public network. We want to encapsulate all the communications into a single "service", so numerous ports do not need to be defined in the firewall. Also, we want to encrypt the data to ensure that no one can easily intercept and read the communications. We would also like local communications to continue without hindrance or encryption.

Policy Name: My Web Server

Filter Actions Required: Permit, Block

DC1 Filter Name: Replication with DC2

- Source: My IP Address
 - Destination: 10.10.10.1
 - Protocol: TCP
 - Destination Port: ANY
 - Filter Action: Require Security
-
- Source: 10.10.10.1
 - Destination: My IP Address
 - Protocol: TCP
 - Destination Port: ANY
 - Filter Action: Require Security

Conclusion

The use of IP Security filters is just a part of the total defense-in-depth security model.

Your organization will still want to implement firewalls and keep up-to-date on system patches and service packs. However, IP Security policies will allow you to define, with more granularity, who can utilize the services a Windows host offers, if any, and how the communication is encrypted, if at all. IP Security policy filters may even be implemented in the reverse, allowing only certain services to originate from the local machine (in the case of a public use kiosk). But remember, IP Security policy filters are simply packet filters: if adequate controls are not in place at a network level, an attacker may still perform attacks such as denial of service attacks and spoofing techniques. Lastly, IP Security filters will not protect against an application exploit to a service that is allowed by a filter.

Joe Klemencic is currently performing Data Security responsibilities at Fermi National Accelerator Laboratory in Batavia, Illinois. He has spent the past 11 years in network architecture support, design and security.

Relevant Links

[Securing Windows 2000 Communications with IP Security Filters, Part One](#)

Joe Klemencic, SecurityFocus

[Using IPSEC to Lock Down a Server](#)

Steve Riley, Microsoft Service Providers

[Active Directory Replication Over Firewalls](#)

Steve Riley, Microsoft Service Providers

[Microsoft Q254949 Article: Domain Controller IPSEC Support](#)

Microsoft Product Support Services

[How to Enable IPSEC Through a Firewall](#)

Microsoft Product Support Services

[Privacy Statement](#)

Copyright 2006, SecurityFocus