

Exchange 2000 in the Enterprise: Tips and Tricks Part Three

Timothy M. Mullen 2003-02-20

Welcome to Part Three of our two-part series on Exchange2000 in the Enterprise! It seems that we had a bit too much content to cover in just two articles, so we bolted on a third.

When we left off in the [last article](#), we had finished talking about Exchange and OWA, and some of the security ramifications of direct server access and front-end server models. After a simple recommendation to use IPSec between front-end and back-end servers to ensure the encryption of credentials passed by the required Basic Authentication model, I realized how often that recommendation is made without providing step-by-step instructions on how to do so. Since we have a little more room to talk in this segment, let's go over just how to do that.

Configuring IPSec between Front-end and Back-end OWA Servers

The Win2K implementation of IPSec has a relatively simple hierarchical architecture: a system has IPSec policies containing rules. The rules contain filter lists, and the individual filter lists contain specific filters. The authentication method, tunnel setting, connection type, and filter action are set at the rule level. Any given policy can contain many different rules, each with their own authentication method or filter action imposed upon the various filter lists.

Though multiple policies can share individual Filter Lists, only one policy can be "assigned" at a time.

By default, your system will contain three policies: Client (Respond Only), Secure Server (Require Security), and Server (Request Security). Be careful when you play around with these; for instance, if you assign "Secure Server (Require Security)", all IP traffic to and from that box will immediately be required to be secure, meaning that any box trying to reach the server with IP will have to have a matching policy to do so.

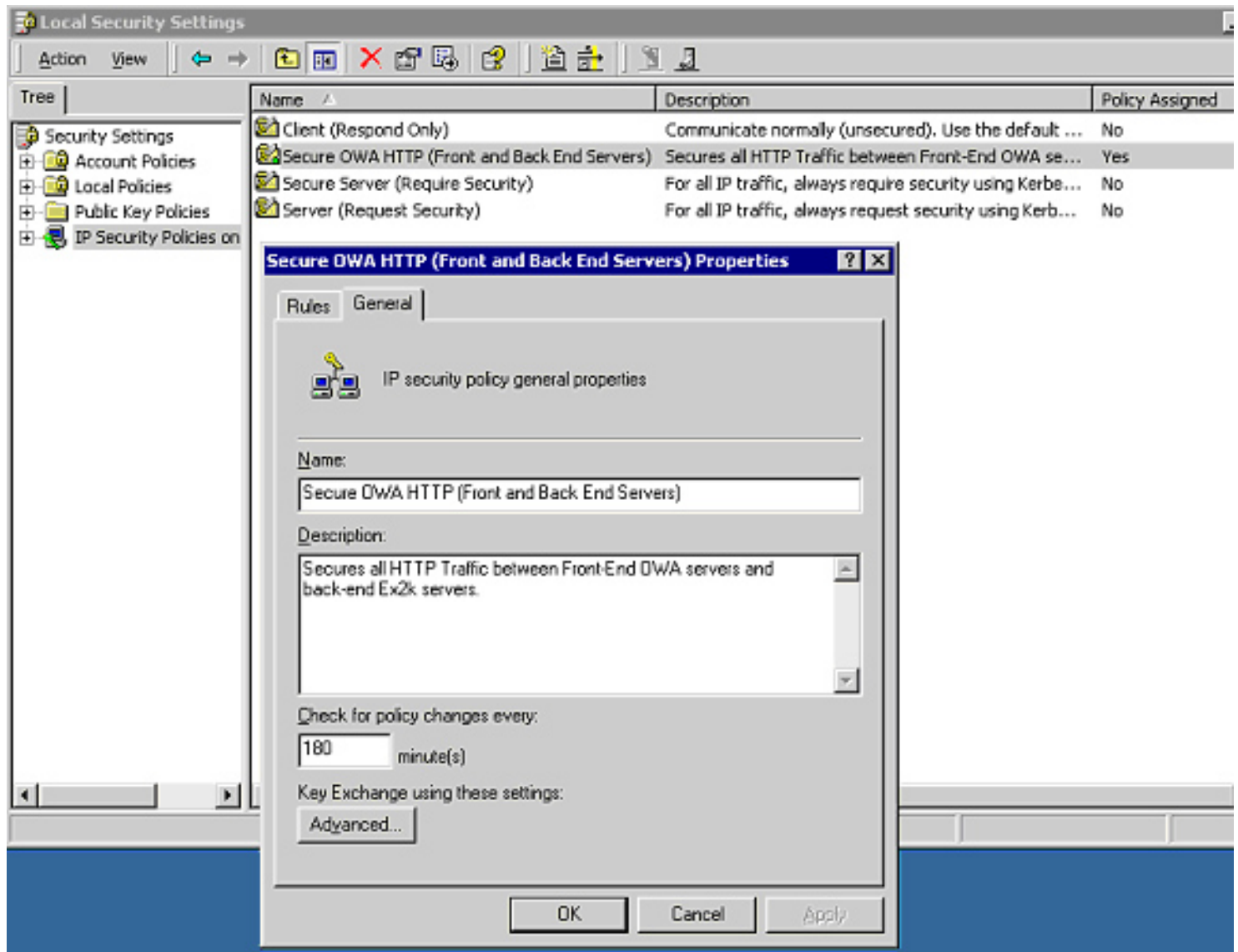
(As an aside, this can be a great practical joke to play on your admins. Just assign "Secure Server", and see how long it takes them to figure out what the hell is going on. It could very well get you fired, but it is great for laughs!)

To keep things simple, we will create a new policy and a new rule specifying the filters we will use to secure the HTTP traffic between our front- and back-end servers. First, we'll create the rules for our front-end server.

Go to the front-end server's local security policy (or use the MMC Snap-in to remotely manage it). Right click on "IP Security Policies", and select "Create IP Security Policy." In the wizard:

- Name your policy: I'm using "Secure OWA HTTP (Front and Back End Servers)"
- Leave "Activate the default response rule" checked. This is required to respond to requests for secure communication.
- Leave the initial authentication method at "Kerberos." You can change this to a certificate or pre-shared key if you like to suit your particular configuration.
- Leave "Edit properties" properties selected, hit "next", and then select the "General" tab.

You should end up with something like this:



Hit "OK" to get back to "IPSec Security Policies".

We'll need a filter list to select when we use the Rules Wizard, so let's create that now. This will be the filter that will enforce the encryption on a host-by-host basis. You can also create the Filter List within the wizard, but I want to break it all up manually. Right click again on "IP Security Policies" (or in the whitespace on the right pane) and select "Manage IP filter lists and filter actions".

- Name the Filter List - I'm calling mine "OWA HTTP." Click "ADD" to add a filter to the filter list. This invokes the IP Filter Wizard. Hit "next" at the info screen.
- Here, you will select the source address. You may want to change the way you enter this information, as you will use group policy or a policy export to replicate policies to other systems. But for now, I'll just select "My IP Address".
- Next, select the destination address - this will be the first back-end Ex2K server we want to reference. You can enter a DNS name or the IP address.
- Select the protocol - in this case, it is TCP.
- Since this is the front end server, it will be making calls TO port 80 on the back-end server FROM a high local port. So here, we'll select "From any port," "To this port" and enter "80" for the port number.
- You're done. No need to edit properties, so just hit "Finish".

We have created a single filter that specifies a source and destination computer, and a source and destination port. When we create the rule that will contain the filter list that contains this single filter, we will specify an action to take - in this case, to "Require Security".

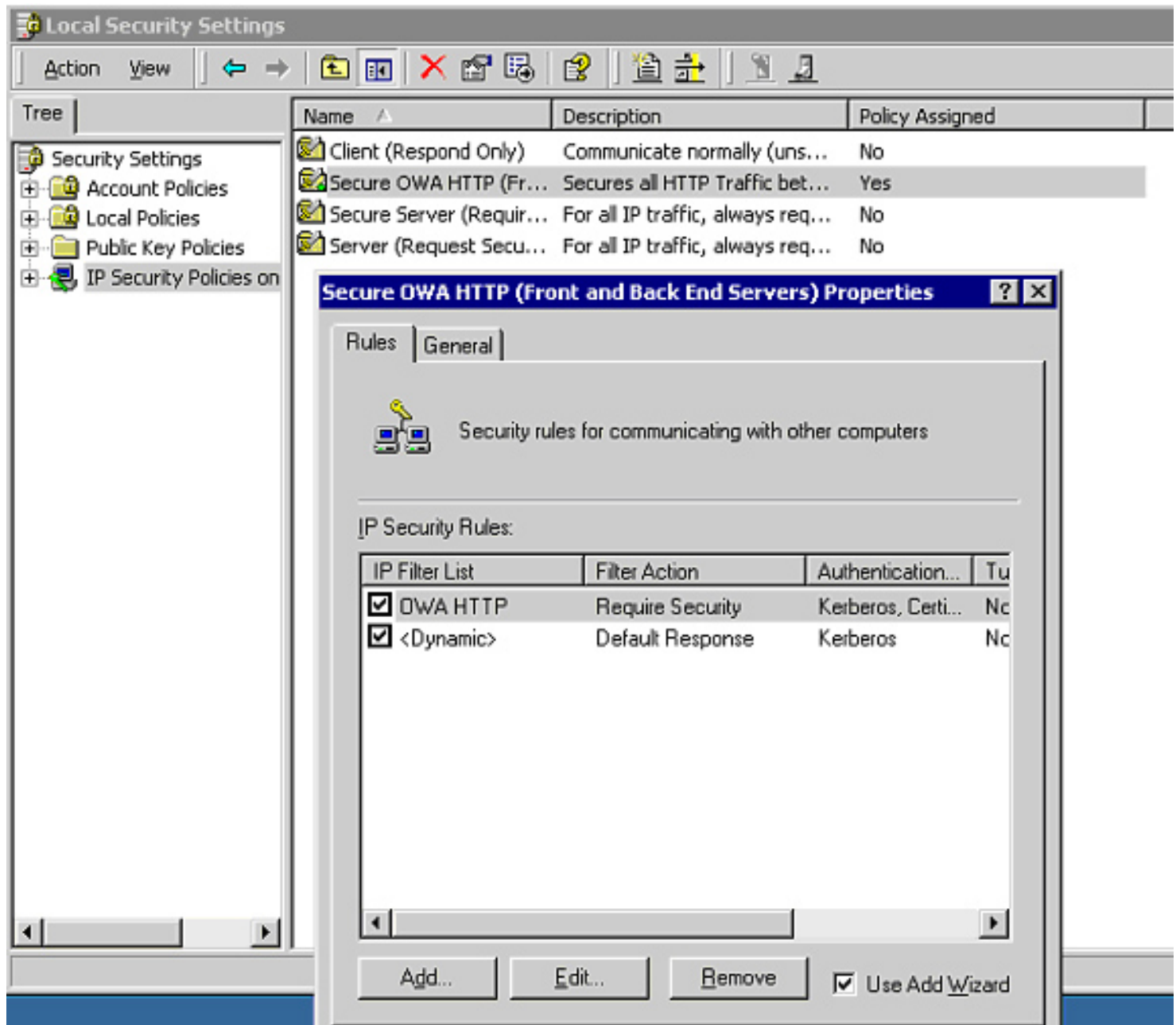
Note that we did not have to specify the individual host systems. We could have created a generic rule that would map from any source to a particular subnet, or whatever would work best in your case; but for these examples I'm doing host-to-host filters.

Repeat the process for each back-end server unless you have created subnet rules that do the job. Note that the Win2k IPSec GUI won't let you build a custom subnet(although the XP/2003 GUI and command line will). For instance, if you want to cover all subnets in a 192.168.0.0 range, you can't just build a 192.168.0.0 255.255.0.0 set: the GUI thinks the 192.168.0.0 address is strictly a class C, and won't let you cut it up into a class B.

We're ready to build a rule now. Double-click your "Secure OWA HTTP" policy and make sure you are on the "Rules" tab. You'll see the Default Response Filter List item you activated in the wizard. Now we will add an IP Security Rule by clicking "Add" and walking through the wizard:

- Intro screen - hit "next".
- Since this rule will secure communications between two computers, it will not specify a tunnel, so leave the default "This rule does not specify a tunnel" option selected.
- You can leave the "Network Type" as "All network connections" unless you have a need to limit it to LAN or Remote Access connections.
- Again, we'll leave the default authentication to Kerberos.
- Under "IP Filter Lists" - select the OWA HTTP Filter List we just created.
- This is where we specify the action to take when a match is found for the filter. In this case, we will select "Require Security", that way, if the traffic can't be secured, no connection will be made.

You should have something like this:





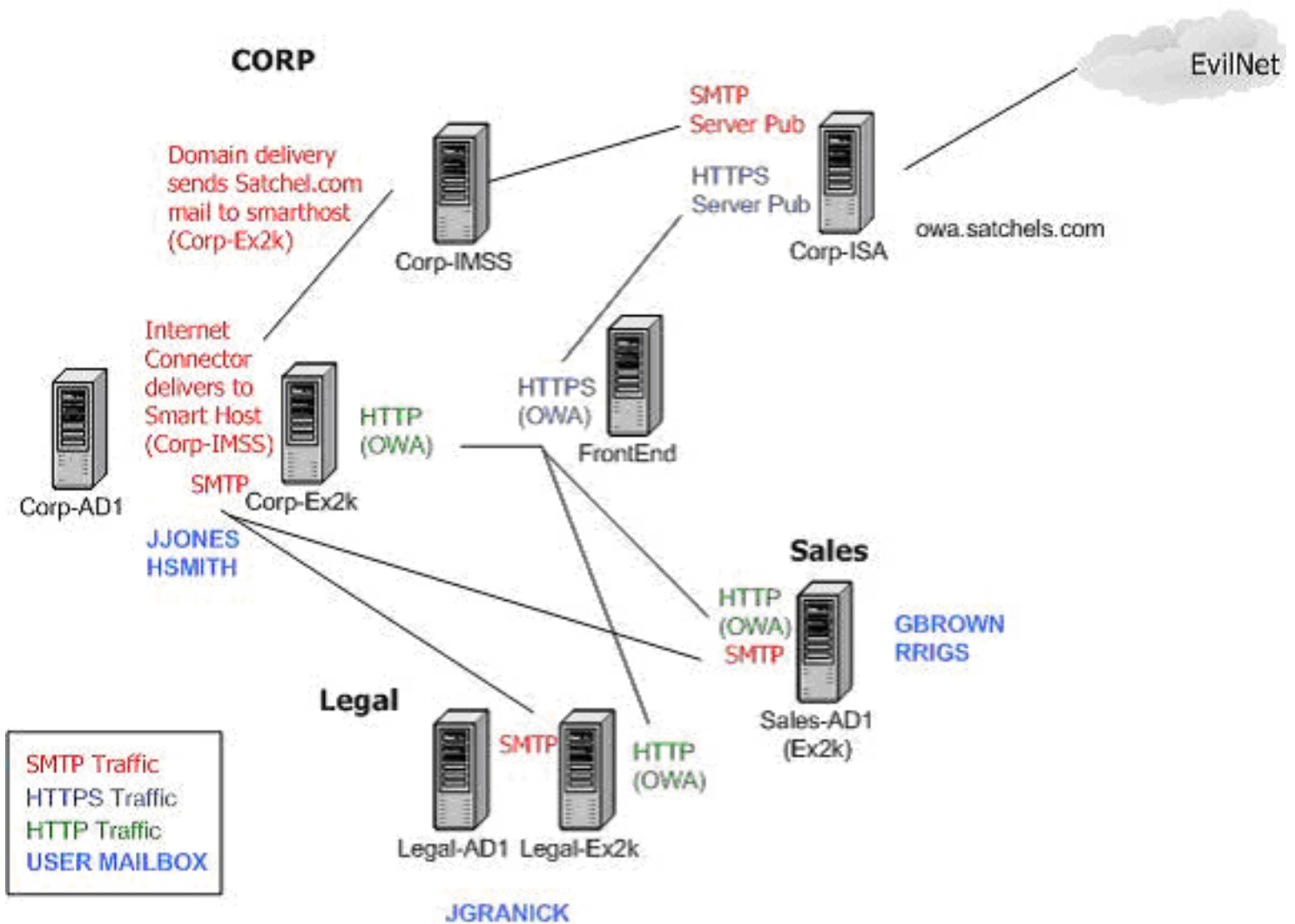
And we're done with the front-end server, simply assigning the policy by right clicking on "Assign" will now require secured traffic. Before this will work though, we have to create policies on our back-end servers. To do this, you will follow the exact same procedure, except that when you go to make your host-to-host filter, you will select "My IP Address" for the source address, and the front-end server IP as the destination address. Since the back-end server is receiving the HTTP traffic, you will filter traffic *from* port 80 *to* any port (the reverse of the front-end server filter.)

Assign the IPSec policies on both systems, test, and you are good to go!

What's In Your Headers?

When we left off in the [last article](#), our topology looked like this:

Satchel's



Part of our concern in this series was that someone could immediately ID our mail server type and name with a simple Telnet to port 25. We solved this by the placement of our Corp-IMSS SMTP Gateway box with a custom banner. However, this information is still being dispensed in our email headers when we send out Internet mail. Ex2k automatically appends the host information to email headers when it receives an SMTP message from a host - this is by RFC. Based on the configuration we have in place now, here is what a standard email header from R Riggs in Sales would look like to the recipient:

```
Received: from mail.satchels.com ([144.51.5.2]) by mail.RecipientDomain.com
with Microsoft SMTPSVC(5.0.2195.2966);
    Mon, 23 Dec 2002 14:00:00 -0500
```

```
Received: from sales-ad1.satchels.com ([192.168.2.100]) by mail.satchels.com
with Microsoft SMTPSVC(5.0.2195.2966);
    Mon, 23 Dec 2002 14:00:00 -0500
```

Subject: Berman Sold You, Now Buy Berman!

Date: Mon, 23 Dec 2002 14:00:00 -0500

Message-ID: <4729F3EF39A58F4FB6612F16DF36BB1C01FF8A@sales-ad1.satchels.com >

MIME-Version: 1.0

Content-Type: multipart/alternative;

boundary="----_=_NextPart_001_01C1FEAD.44389CF2"

X-MS-Has-Attach:

X-MS-TNEF-Correlator:

content-class: urn:content-classes:message

Thread-Topic: Berman Sold You, Now Buy Berman!

X-MimeOLE: Produced By Microsoft Exchange V6.0.4712.0

Thread-Index: AcJ+rEQwB2FlomAnrr7RtafFFekZooRopA==

From: "Riggs"

To: "Joe Mamma"

Return-Path: rriggs@satchels.com

X-OriginalArrivalTime: 23 Dec 2002 14:00:49.0613 (UTC) FILETIME=[45C091D0:01C1FEAD]

As you can see, the name of our Sales-AD1 box is displayed directly in the headers, along with its *internal* IP address. This is actually by design. If you are wondering why the name of Corp-Ex2k does not show up in the headers, it is because the Trend IMSS box does *not* append the server information into the header when it receives a SMTP object from Corp-Ex2k. We'll use this "feature" in a little bit. If it were not for the IMSS box, though, by default, the name Corp-Ex2k.satchels.com would indeed be displayed in the header.

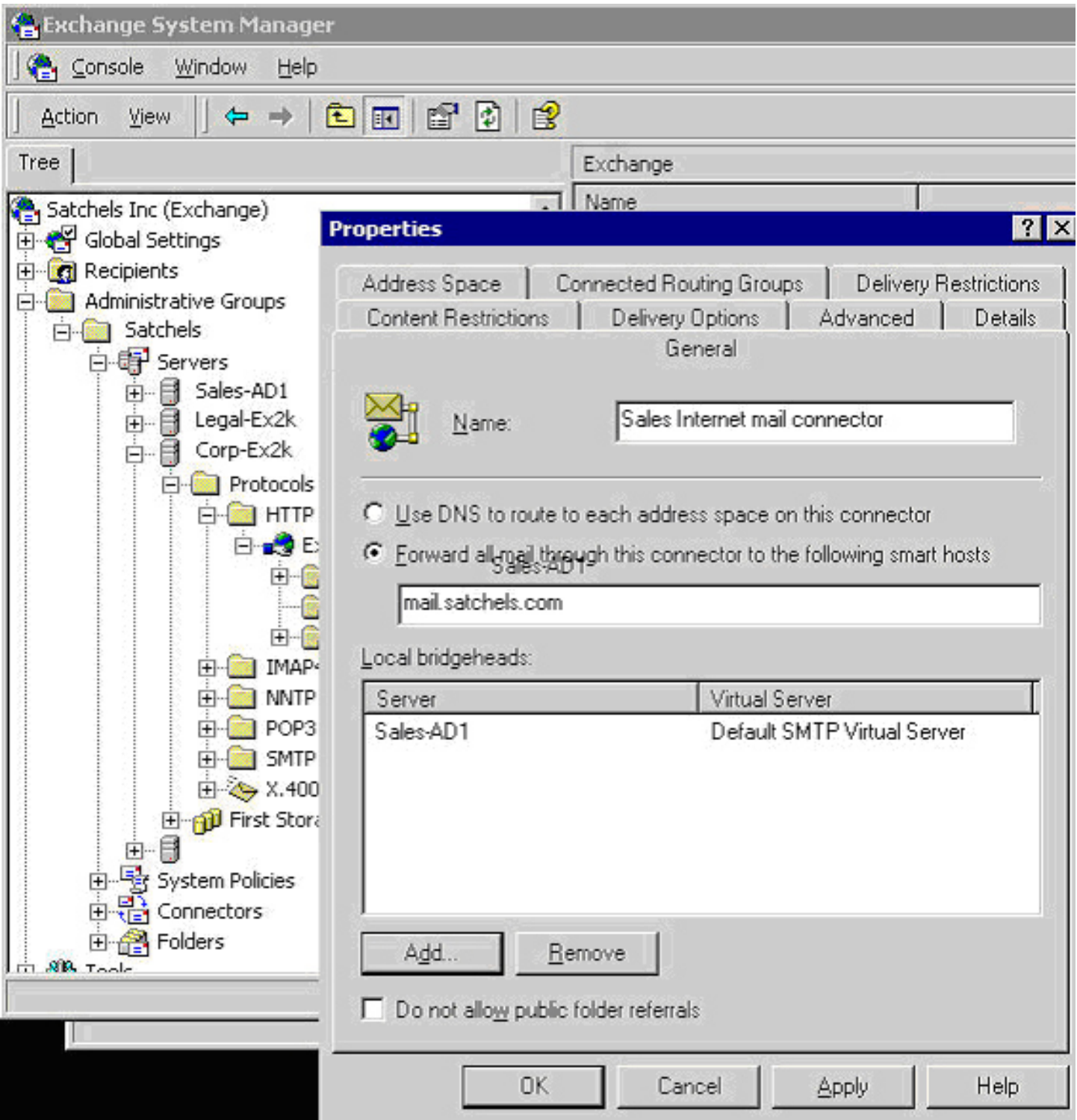
So, not only could someone easily find out the actual host name of our internal servers, they could also get the internal IP scheme. Within deeper infrastructures, you can actually get a decent topology map to the infrastructure from a single email header.

Let's address the machine's host name first. While some don't consider this a big deal, others don't like to give out any more information than they have to - I'm one of those. This can be handled by entering a fully-qualified domain name in the "Advanced Delivery" dialog of the SMTP Virtual Server's properties. When you enter or change this field, ensure that you have already created a DNS host name to match, this is the name other internal servers will use to connect to SMTP server. We'll enter mail1.satchels.com for Corp, mail2.satchels.com for the Sales unit, and mail3.satchels.com for the Legal unit (remember that the IMSS box is mail.satchels.com). So, the same email from R.Riggs would now refer to the following servers:

```
Received: from mail.satchels.com ([144.51.5.2]) by mail.RecipientDomain.com
with Microsoft SMTPSVC(5.0.2195.2966);
    Mon, 23 Dec 2002 14:00:00 -0500
Received: from mail2.satchels.com ([192.168.2.100]) by mail.satchels.com
with Microsoft SMTPSVC(5.0.2195.2966);
    Mon, 23 Dec 2002 14:00:00 -0500
```

You'll see that it still contains the internal IP address of the server, which is something I like even less than my host name being out there. A way to get around this (without purchasing third party software that strips your headers) is to build separate Internet Connectors for your SMTP servers so that they don't take the default route through your infrastructure. It also relieves other systems of the burden of routing the object. You may be logically tempted to try this by just adding a smart host value in the SMTP virtual server properties, but what we need to do is create new Internet mail connectors.

In system manager, navigate to "Connectors", right click on "Connectors", and create a new SMTP connector:



We've named this one "Sales Internet mail connector," told the connector to forward all mail to the mail.satchels.com smart host (Corp-IMSS) and set the local bridgehead to the Sales-AD1 Exchange Server.

Now, all SMTP mail will go directly to Corp-IMSS, where DNS will deliver it. The same mail item sent by RRiggs will now look like this:

```
Received: from mail.satchels.com ([144.51.5.2]) by mail.RecipientDomain.com
```

```
with Microsoft SMTPSVC(5.0.2195.2966);
    Mon, 23 Dec 2002 14:00:00 -0500
Subject: Berman Sold You, Now Buy Berman!
Date: Mon, 23 Dec 2002 14:00:00 -0500
Message-ID: <4729F3EF39A58F4FB6612F16DF36BB1C01FF8A@sales-ad1.satchels.com >
MIME-Version: 1.0
Content-Type: multipart/alternative;
    boundary="----_=_NextPart_001_01C1FEAD.44389CF2"
X-MS-Has-Attach:
X-MS-TNEF-Correlator:
content-class: urn:content-classes:message
Thread-Topic: Berman Sold You, Now Buy Berman!
X-MimeOLE: Produced By Microsoft Exchange V6.0.4712.0
Thread-Index: AcJ+rEQwB2FlomAnrr7RtafFFekZooRopA==
From: "Riggs"
To: "Joe Mamma"
Return-Path: rriggs@satchels.com
X-OriginalArrivalTime: 23 Dec 2002 14:00:49.0613 (UTC) FILETIME=[45C091D0:01C1FEAD]
```

No topology leaks, and for the most part, no internal server names. The only exception to this is the message-ID element. Unfortunately, the message-ID is created by the message store, and the actual server name will be appended to the message-ID as in the above:

```
Message-ID: <4729F3EF39A58F4FB6612F16DF36BB1C01FF8A@sales-ad1.satchels.com >
```

That line still gives up the true host name, but at least we have a way of restricting the message-routing and all the host names in between.

In a wish list to Microsoft, I'm going to suggest that an SMTP virtual server have an option to suppress appending SMTP host information on received objects, and that the message-ID use the fully qualified domain name of the SMTP virtual server that served the message.

Well, that's about it for this series. I appreciate your time, and hope it was valuable to you.

'Till next time...

Timothy M. Mullen is CIO and Chief Software Architect for AnchorIS.Com, a developer of secure, enterprise-

based accounting software. This article is an exclusive excerpt from Tim Mullen's upcoming Blackhat Win2K training, Microsoft Ninjitsu: Securely Deploying Microsoft Technologies.

Relevant Links

Exchange 2000 in the Enterprise: Tips and Tricks Part One

Tim Mullen

Exchange 2000 in the Enterprise: Tips and Tricks Part Two

Tim Mullen

[Privacy Statement](#)

Copyright 2006, SecurityFocus