

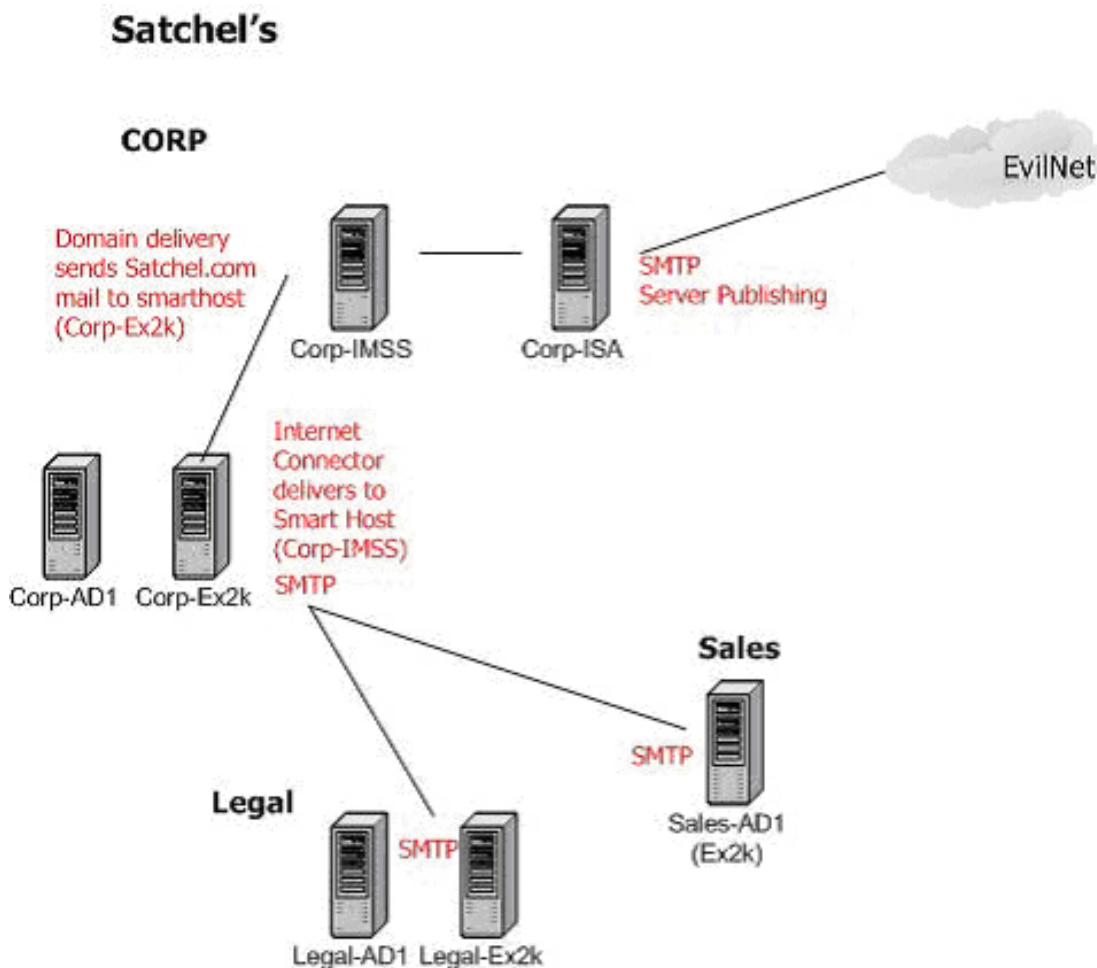
## Exchange 2000 in the Enterprise: Tips and Tricks Part Two

Tim Mullen 2003-01-15

This is the second installment in a two-part series on securing Exchange 2000 in the enterprise. When we [left off](#), we had finished up building a messaging infrastructure that handled many of the issues mail administrators must contend with. Since Part One was published, Microsoft has released a new [feature pack](#) for ISA Server, which includes many new features including an Enhanced SMTP Application Filter (allowing you to filter sender names and domains), and encrypted RPC between an Outlook client and an Exchange Server if you wanted VPN-less encryption for client-to-Exchange connections over the Internet. One quick note- the SMTP filter does not support SMTPS; well, it is supposed to- it just doesn't work. It breaks after STARTTLS- you should know that if you try to use the filter in conjunction with SMTPS.

Now it's time to address the security ramifications of publishing mail content to the Internet via Outlook Web Access.

Here is where we left the set-up:



There are basically two ways to publish mail content via OWA in Exchange 2000: direct server access, and "front-end/back-end" server configurations. Each has its associated pros and cons, so it is important to understand the security ramifications of both. In either case, this functionality is provided at the server level

by combining Exchange's support of the HTTP protocol with IIS's Web publishing. Any Exchange 2000 server housing a mailbox that you wish to access through OWA must have IIS installed on it.

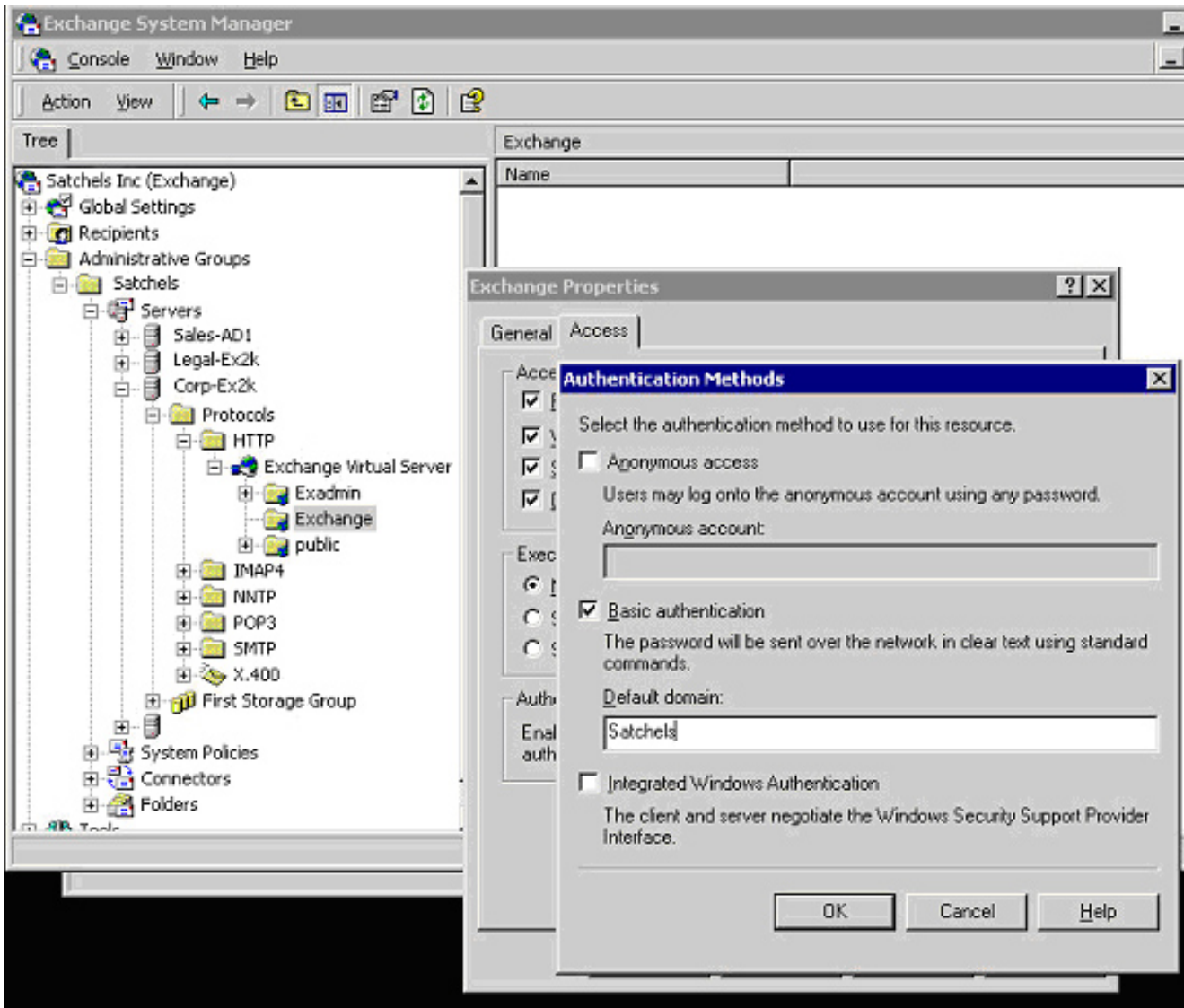
We'll start with the "direct access" method of publishing Exchange content. But first, I want to point out Exchange Server's dependence on and extensive use of actual system host names in its operation. You will see how this could potentially leak out more information than you may be comfortable with disclosing? More on that later.

## **Direct Publishing**

First, let's get the HTTP/IIS configuration completed on Corp-Ex2k; then we'll use ISA server to publish the Exchange OWA site to the outside world.

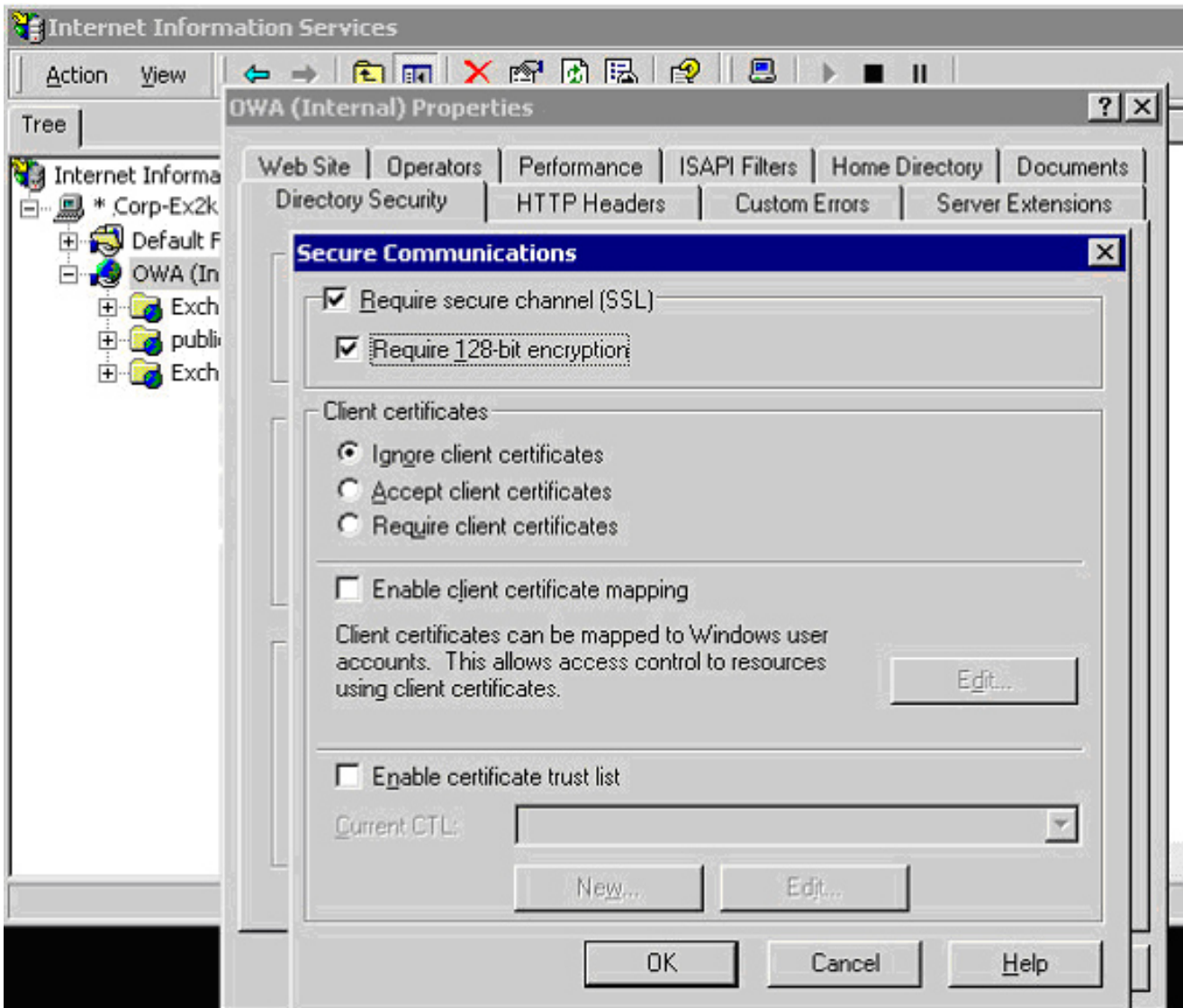
To fully configure the server, we have to (unfortunately) do so in two different places, the Exchange System Manager and the IIS MMC. Some settings, such as authentication mechanisms, are duplicated in both MMCs - it is important to remember that Exchange configuration information is stored in Active Directory, and the IIS configuration is in the local metabase. Shared information you change in System Manager will overwrite the same settings in the IIS metabase when the AD changes propagate. For instance, if you set the authentication method in the Exchange System Manager for the Exchange virtual directory in the HTTP Exchange Virtual Server to "Basic Authentication," and then later go into the IIS MMC and change the virtual directory's access to both "Basic Authentication" and "Integrated Windows Authentication," the new changes made in the IIS console will apply for a short period of time, but will then be overwritten (back to "Basic" only) when AD updates the metabase. Don't let this bite you.

An idiosyncrasy that compounds this issue is a misleading message one gets when the properties of the Exchange Virtual Server (under the HTTP protocol settings in Exchange System Manager) are requested, saying "You must use the IIS Admin to manage this Virtual Server's settings." This is not exactly correct, as all authentication settings must be configured/changed in System Manger by expanding out the Exchange Virtual Server and applying them at the virtual directory level as follows:



Knowing we will be using SSL to access OWA, we will set the authentication method to "Basic Authentication". As we will soon see in our "front-end/back-end" configuration, Basic Authentication will be required when we build that model. If we will have internal users that we wish to log on with their current credential when accessing the site, we can select "Integrated Windows Authentication" as well.

Now it's a quick trip over to the IIS MMC to set other security options like IP restrictions and settings for secure communications and SSL. Any time an external user logs on to OWA, it should be over SSL- validating Web users with basic authentication over HTTP is simply foolish.



You'll see that we have required SSL (after acquiring the appropriate certificate) and have required 128-bit encryption as well. Browsers that do not support high encryption will not be able to log in with this configuration, and users won't be able to mistakenly access the site over standard HTTP. Of course, you could set any client certificate or mapping settings as well.

OWA is now set up and listening on Corp-Ex2k. Internal users can already access the site by visiting <https://corp-ex2k.satchels.com/exchange> (resolved by internal DNS), but we are not finished yet. We have already brought the system up to the current patch level but we need to add a level of protection- and can do so with the free IISLockdown tool from Microsoft, which includes URLScan. The IISLockdown portion of the tool will remove unneeded ISAPI extensions and re-map them explicitly to the included 404.dll, set permissions appropriately, and remove unneeded services. URLScan will add a level of filtering that allows us to set what URL elements we will allow to make it to the server before they are parsed by the IIS engine. When we install IISLockdown, we need to be sure to select the Exchange 2000 template so that the URLScan.ini file will be properly built to allow commands like "SUBSCRIBE" and "SEARCH".

But be aware that there is a caveat here. By default, URLScan will filter out certain character sequences like double dot, percent, dot slash and the ampersand (even with the Exchange template selected). When

messages are stored in the virtual M: drive store, the subject line is used as part of the filename, terminated with an ".eml" extension. So a message subject like "This is important." becomes the filename "This is important..eml"

The link to the message from the OWA InBox simply calls the message store filename - in this case it contains ".." which URL will filter out - thus, you would not be able to read, delete, forward, or reply to that particular email in OWA (it results in an HTTP 404 - File not found error). The same with a subject line of "We are up 5%" or "I need this & that."

Personally, I would prefer to not read the mail message via OWA at all than to allow things like ".." just in case someone discovers another weird directory traversal bug, but I doubt your field crew will go for that. In these cases, you will just have to change the URLScan.ini file so that it allows these character sequences. Note that if you have other sites on that server, the URLScan options are server-wide, not site specific; all sites would allow the URL sequence. [SecureIIS](#), a product by eEye, has the ability to configure filter settings not only at the site level but at the folder level as well. In fact, there is a specific configuration of SecureIIS specifically for OWA. Those of you who would like more granular control over your filtering application should check out eEye's site for more information.

ISA Server Feature Pack 1 also has a new "URLScan for ISA" that will allow you to perform the URLScan functions at the ISA Server in addition to the URLScan installation on the individual servers. The ISA version does not have the template options that the normal URLScan does, but it does include what they call an "OWA" option that basically sets up a separate "urlscan\_owa.ini" file for use with OWA sites. You could actually use it for all the sites published by the ISA server but it is designed for use with OWA. Though the feature pack "read me" file insinuates that one could protect the internal IIS installation with a single installation of URLScan on ISA, I would not recommend doing so. Practice security in depth and use it in conjunction with server installations of URLScan.

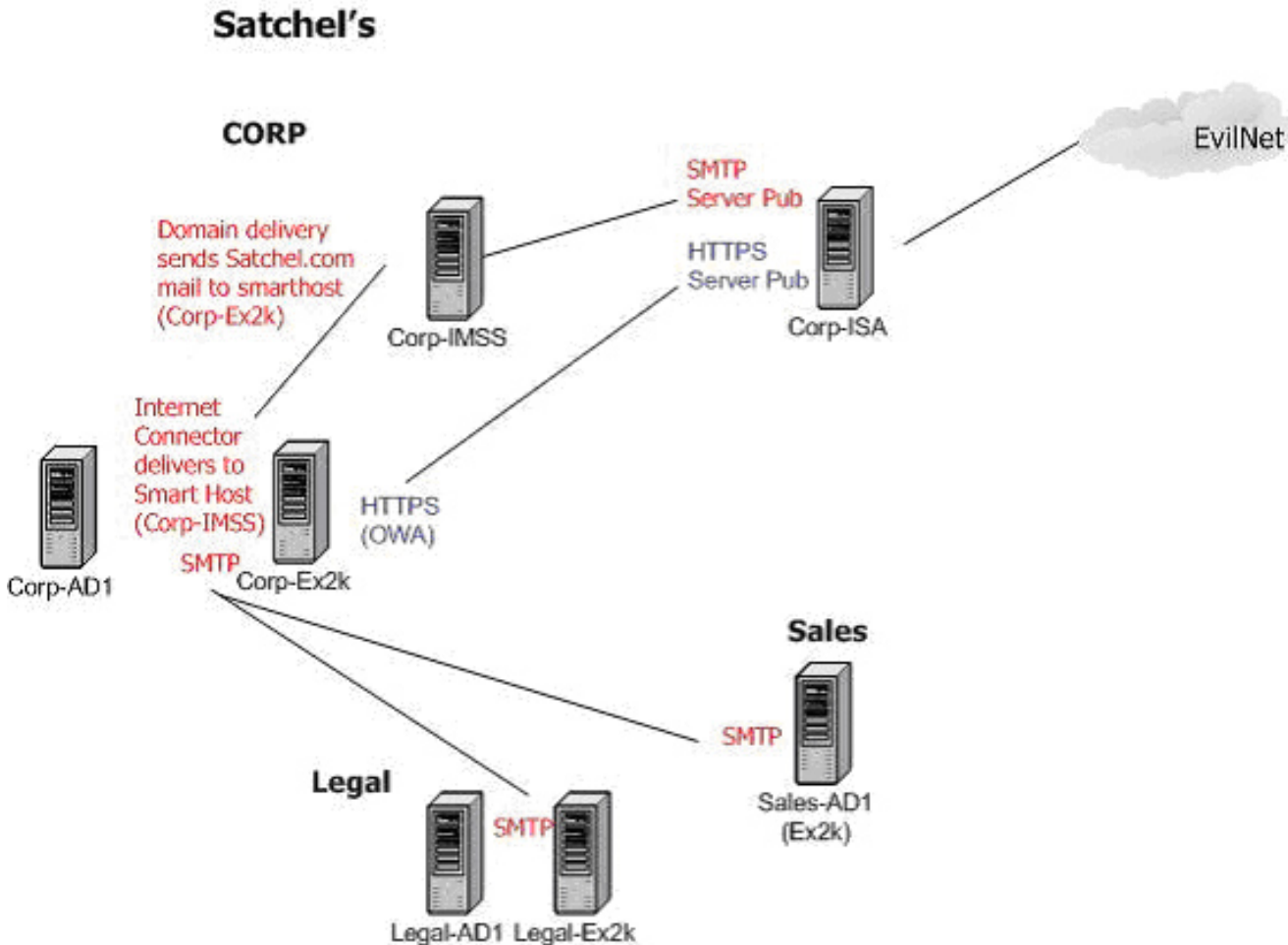
Now it is time to configure ISA server to publish external requests for our OWA site to the internal site.

As far as ISA is concerned, we can Web publish the site, where the ISA server is configured with Web listeners to act as the external site (a proxy) and forward requests to the internal network, or we can Server publish, where an address/protocol set is redirected internally. For purposes of this article, it does not really matter how you choose to do it - but I'll use Server publishing so that the actual IP address of the connecting client is referred to in the OWA logs and not the IP of the ISA server, as would be the case in a Web publishing environment. Note that ISA Feature Pack 1 includes a new OWA Web Publishing Wizard, which goes through and configures up your destination sets for you. Each method has its benefits, for more information, check out [www.isaserver.org](http://www.isaserver.org) for tutorials and technical articles on the subject.

We'll just run the Server publishing wizard to redirect HTTPS traffic from the external interface to our Corp-Ex2k machine. (Note that we have changed the IP address from the original example in Part One of this series, since we moved the Corp-IMSS machine into that slot.)

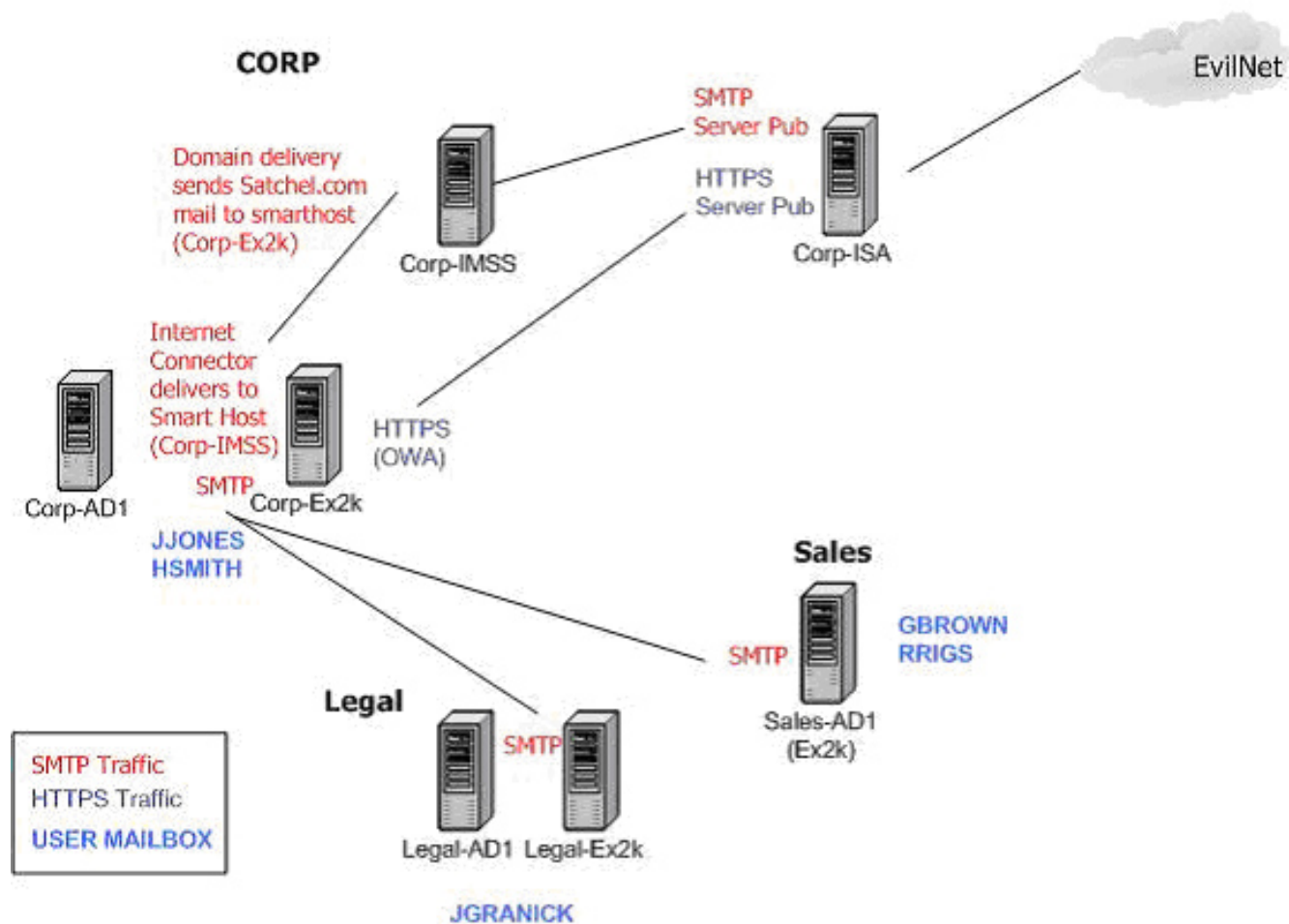
External requests will now be routed to the Corp-Ex2k machine, and we may now provide access to all the server's mailboxes from the Internet. This could be done directly against the IP address or via a DNS host name that you add to your external DNS servers (or that you have your ISP add). Note that differences in the hostname entry and the name on the certificate will make the user have to first validate the connection.

Here's what we have now:



For direct access to this server, it does not matter what the Internet hostname is. Although it ultimately connects up to the internal Corp-Ex2k host, the Internet hostname could be "owa.satchels.com" or anything else. But this will only work as long as users with mailboxes on this server log in: if they log in to this server but have a mailbox on another server, OWA will automatically try to redirect to the actual host-name of the other Exchange node. This is part of the "host-name" dependence we talked about earlier. Consider the following user distribution:

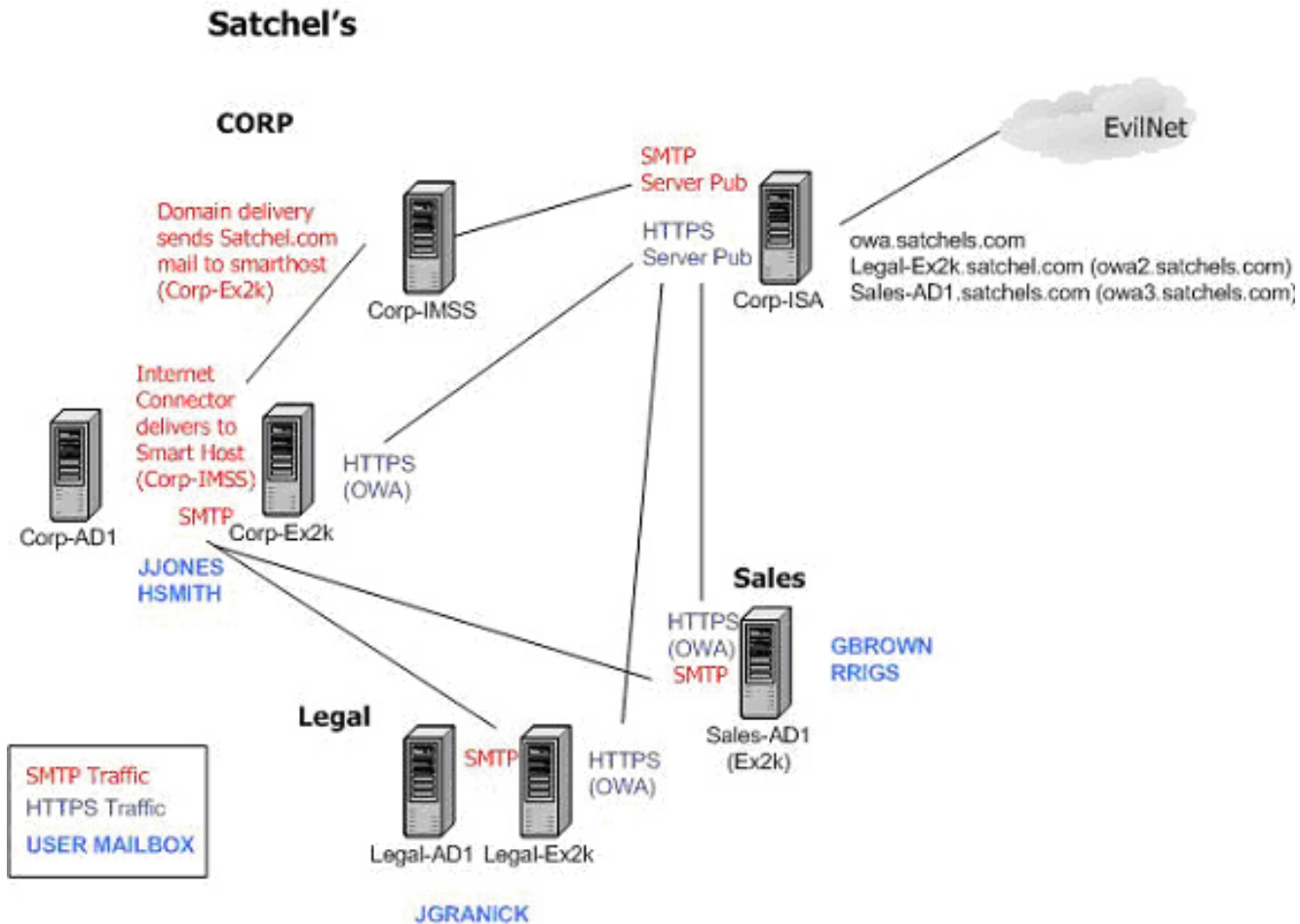
## Satchel's



Given that owa.satchels.com publishes to Corp-Ex2k, we will have no issues if JJones or HSmith log on to <https://owa.satchels.com/exchange>. However, if JGranick logs on to <https://owa.satchels.com/exchange>, Corp-Ex2k will know the mailbox is on Legal-Ex2k, and will automatically re-route the client browser to <https://Legal-Ex2k.satchels.com/exchange>. Likewise, if GBrown logs on to <https://owa.satchels.com/exchange>, the client browser will be redirected to <https://Sales-AD1.satchels.com/exchange>. Since none of these host addresses are valid to an external client, these users will not be able to log in. The only way around this- that is, the only way to have direct server publishing where the client only needs to know one host address (when you have multiple servers), is to add host names to the external DNS servers for the actual host names of the internal machines (with an external address of course)- something you may not want to do. The security ramifications of adding actual host names to your external DNS are somewhat dubious, but many consider any such information as too much information.

Alternately, you could Sever Publish each internal machine as its own publishing rule, each with its own external IP address (Web publishing would let you base this on the URL) and tell each user what host name to use for their own mail server. This may seem like a lot of work, but it does ensure that each OWA server is set to only use HTTPS (you'll need certificates for all servers), and you have direct control of how each server gets published. Whether you create DNS entries for the actual host names (so that the end user only knows one address) or make up separate host names for each individual OWA machine (as in owa2.satchels.com,

owa3.satchels.com, etc.) you know that every user that logs in does so directly to the server hosting their mailbox, and that SSL is used at all points in the log-on process. Since all access is ultimately a direct client-to-server connection, one can better control things like IP restrictions and authentication methods as one could with any individual server-based configuration. The layout looks like this now:



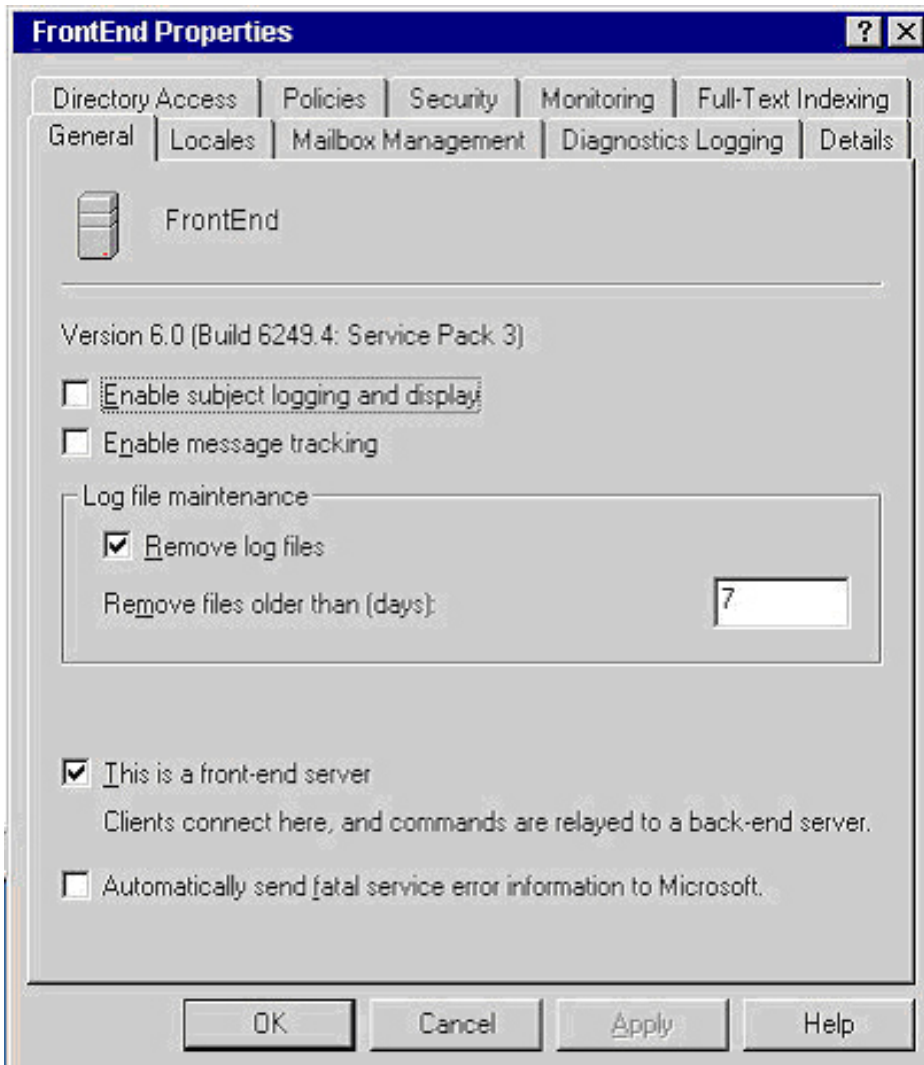
This would not be practical in huge environments, but it does offer tight control and security. In fact, even in smaller environments this could be an administrative burden (consider multiple domain structures), but there are certainly circumstances where the security value added justifies the configuration overhead.

## Front End Servers

The other, more common, solution is to deploy a Front End Server configuration where a single server is used for all log-ons with data requests being proxied to the appropriate back-end server. This is a much easier way to configure things, but it too has issues.

In our current topography, we are not able to use a Front End server. Why? Because all of our Exchange servers have mailboxes on them! Only a virgin Exchange node without user mailboxes can be designated as a front-end server. This is not exactly intuitive, as the "This is a front-end server" checkbox is not even visible in the server properties unless that condition is met. I would rather have the option grayed-out so you would know it is an option in the first place, but there's not much we can do about that. We'll set up a new

Exchange node called FrontEnd, at which point we can set it as a front-end server:

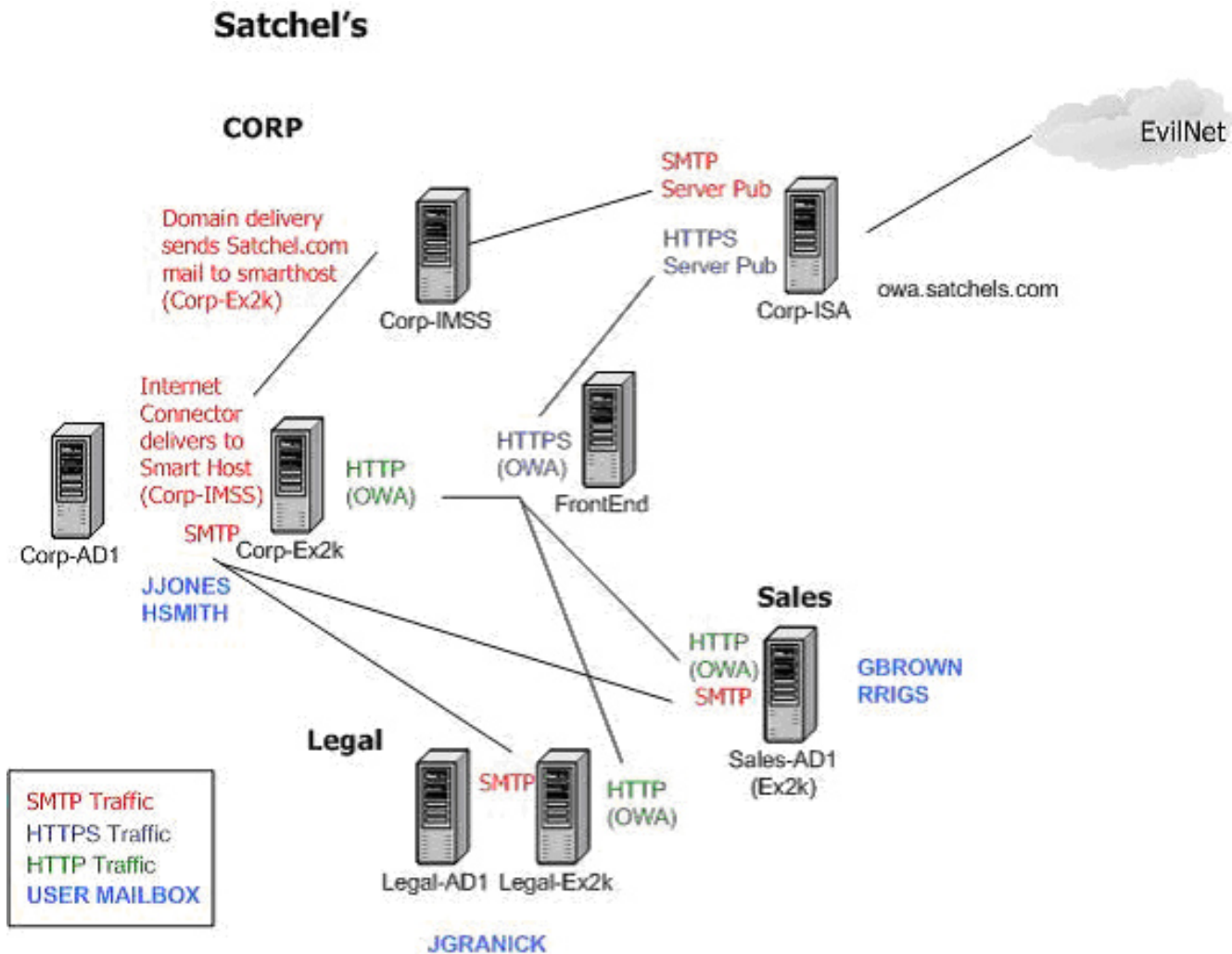


This will change our configuration substantially. For starters, we must live in a "basic authentication" mode only. Well, to be precise, a front-end server can be set for anonymous authentication and thus require the back-end server to first accept the anonymous connection and then validate the user itself, but that is a silly way to do it (from a security standpoint) so I won't even consider it.

If you expand out the authentication methods of FrontEnd now (in the Exchange System Manager), you will see that "Integrated Windows Authentication" is grayed-out and is no longer an option. Our SSL options have also changed: we will use Basic Authentication over HTTPS (and still require 128 bit) to the FrontEnd server, but our back-end servers can no longer be set to require SSL. This is because the Front End Server model requires basic authentication over HTTP for front-end-to-back-end communications. You can still mix up Basic and Windows authentication (if you want direct internal access) on the back-end servers from an authentication standpoint, but you cannot require SSL (Note that you can still \*use\* SSL for direct access to the back-end servers if you choose, but it can not be a requirement). If you leave a back-end server set to require SSL and try to access it through the front-end server, you will get an HTTP 403.4 - Forbidden: SSL required message, even though you are looking at <https://owa.satches.com/exchange> in the URL.

The new options available in ISA Feature Pack 1 do not affect this functionality.

Let's see what this looks like now:



We've now published a single host that will allow all users to log into a single front-end server, and access their mail regardless of where the server physically resides (and thus have flattened the namespace). Administratively, this is pretty easy. Being a front end server, the namespace will always stay the same, as the mailbox access is made from FrontEnd to whatever back-end server houses the mailbox; no need to worry about client redirection and multiple host publication or DNS records.

The security issue is that all back-end communications take place via basic authentication over regular old nasty HTTP. In our configuration, everything between the client and FrontEnd will be over 128-bit SSL, but all back end communication will be in the clear.

When RRigs logs on, anyone anywhere between FrontEnd and the back-end server with a sniffer could grab the following each time the client browser submitted a GET, SEARCH, SUBSCRIBE, etc to the Exchange server:

```
HTTP: GET Request for Client
```

```
HTTP: Request Method = GET
HTTP: Uniform Resource Identifier = /exchange/rriigs/Inbox/?Cmd=contents
HTTP: Protocol Version = HTTP/1.1
HTTP: Connection = Keep-Alive
HTTP: Accept = image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vnd.
HTTP: Accept-Language = en-us
HTTP: Connection = Keep-Alive
HTTP: Host = owa.satchels.com
HTTP: Referer = https://owa.satchels.com/exchange/
HTTP: User-Agent = Exchange-Server-Frontend-Proxy/6.0 Mozilla/4.0 (compatible; MSIE
HTTP: Cookie = sessionid=d12d9cc-4ffe-2234-ccea-d9ea3f9e8bb4,0x399
HTTP: Authorization = Basic V2VyZSB5b3UgcmlVhbGx5IGV4cGVjdGluZyBjcmVkdW50aWFscz8=
HTTP: Accept-Encoding =
HTTP: Front-End-HTTPS = on
```

(Yes, "Connection = Keep-Alive" repeats, and yes, the HTTP standards body spelled "referer" incorrectly.)

Basic authentication is not encrypted - it is just encoded with Base64, and it takes about two milliseconds to decode it. People like Kevin Poulsen can even decode Base64 in their head. Just the act of logging on pushes out the user's credentials about eight times- and that's before the user actually *does* anything. Given the fact that administrators and corporate executives alike will be using OWA for their email, this poses a substantial risk- even if the traffic is all internal. After all, if we could trust all of our users, we would not need different user names and passwords, would we?

In front-end server models, you should seriously consider some sort of point-to-point encryption such as IPSEC to protect the traffic between your front-end server and any back-end servers you have in place considering how much unencrypted credential traffic is created in this type of model.

The lesson here is that no simple, default of OWA is secure all by itself. Whether you have to do a bit of extra work and publish servers separately to secure them, or do a bit of extra work to secure the traffic between front- and back-end servers, you'll still have to do a bit more than set up IIS and point a browser at it.

Alas, we have run out of time, so it looks like we will have a Part Three in this two-part series. Stay tuned for an expanded review of what information you may be disclosing in your email headers, and what techniques you can leverage to mitigate exposure. Until then...

*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*

[Privacy Statement](#)

Copyright 2006, SecurityFocus