Global Information Assurance Certification Paper

Securing Microsoft Outlook Web Access
(with help from Apache)

Case Study in Information Security
GSEC v. 1.4b

Cory Steers
April 4, 2003

Securing Microsoft Outlook Web Access (with help from Apache)

Cory Steers

April 4, 2003

Abstract

The focus of this paper is on how to extend an Exchange 2000 email environment to allow users to read their email over the Internet with a web browser. This paper will discuss how to enable and secure Microsoft's OWA (Outlook Web Access), and use Apache as a web accelerator or caching proxy server to help control and protect what content is served up.

The Problem

As time moves on, it seems that business needs require only additional functionality from their IT departments. Of course, this new functionality seems to increase the network exposure to attack, making it increasingly difficult to mitigate risk. In the not so distant past a simple web server, serving static content to your customers and allowing your associates to send and receive Internet email was enough. Now, if you're not providing your Internet customers with a dynamically generated web portal, or letting your associates utilize instant messaging, you're sure to be put out of business by your competitors ... or so the business areas think.

My situation was similar, but not nearly as complicated. When I took over as the IT administrator for my church, one of the first issues my predecessor left was a buggy email solution. He was running Exchange/Outlook for office email, and the office personnel with a need for Internet email capabilities had a second POP3 email account built into their Outlook client. The POP3 account was the default email account for them, so Internet email left directly from the client machine. An off site business hosted the domain MX record and incoming email was being delivered there. The outlook clients would periodically pop the email down to the local workstation.

At first this didn't seem like a bad solution, but it did have its problems. Outlook would still use Exchange as its mail queue, so if outgoing mail could not be immediately delivered for some reason, it would queue it up on the exchange server to be sent later. Again, not a problem by itself, but all of the internal workstations and servers were named "<hostname>.<domain name>.local", which did not resolve on the Internet. In a world where spam is a problem for everyone, an ever increasing number of ISPs would block email from servers that didn't have an Internet resolvable IP address. Anything that ended in ".local" was never going to resolve to an Internet address (at least we hope), so mail queued up on the Exchange server destined for these ISPs email accounts would always be rejected.

Another quirk that I could not figure out was that sometimes, and I stress "sometimes", exchange would send out its own copy of the email. This was in addition to the outlook client successfully sending the original. So, either the recipient would get

two copies of the email, or if the recipient's ISP was checking for valid Internet resolvable hostnames, the recipient would get one copy and the sender would get an error message.

Tracking these errors down was filling up much of my day; therefore it was time for a change. I got a DynDNS account and configured it to point A records and MX records for our domain to the church's firewall address. The firewall was configured to be an SMTP relay, and I changed the outlook clients to have a single Exchange email account. Magically, all the weird email problems went away.

One feature of the previous email setup was the ability of the office personnel to check email from the Internet with a web browser. It wasn't fully functional. The outlook clients were configured so that when they popped the email down, they'd leave a copy of the email on the server for 10 days. The offsite ISP would then allow web access to those accounts. When I implemented my solution that fixed all of the other problems, this feature went away. I knew that they were going to lose this capability, but I underestimated how "needed" it was. I thought it was just a nice option to have and was rarely even used; however, the reality was that it was used by people when they were sick or out of the office for other reasons. For example, one user was a volunteer and only used the web feature. When she was no longer able to check mail from home, her church email account became useless to her. I needed a solution to replace the web mail solution I took away.

I was hesitant to implement a solution because the churches web site was hosted somewhere else and currently, the only service the church's Internet connection had running was SMTP for email. Securing an email relay was something I had experience in, so I wasn't worried about running that service. No matter what solution I came up with, I was going to have to allow additional traffic in. It was going to be a challenge to implement a secure solution, and stick to my budget.

The Solution

My two main concerns were security and money. I wasn't comfortable with the fact that I needed to allow additional access into the church's network. Since I'm a member of the church, I know exactly where the money for expenses comes from ... people like me, so I wanted to spend the money wisely. The church takes pledges every fall and builds a budget from those figures. There was money budgeted for computer expenses, but that budget didn't anticipate this need.

Since the church's office mail solution is Exchange 2000 for the server, and Outlook 2000 and Outlook XP for clients, I decided to utilize Microsoft's OWA (Outlook Web Access) as the solution. OWA is a free extension of Exchange, so it wouldn't cost the church any extra money to turn it on. The church office consists of 10 to 12 employees, so I wasn't very concerned with server capacity.

Of course, now I had a whole new set of issues. By using OWA I would have to

open up access, either directly or indirectly, to the churches main server. Microsoft generally recommends putting different services on different servers. I agree with them, but it's much cheaper to buy one server and have one copy of Windows 2000 server, Exchange 2000, and whatever other services needed running on a single server. As a result, the church has a single server acting as the domain controller, exchange server, print server, and file server. Any access being allowed would be allowed to the heart of the church's computer systems. This scenario saves an attacker a lot of work. If they break into one server, they automatically gain access to everything.

I know from my SANS training that I want a layered approach. Changing iptables rules on the Linux firewall to redirect port(s) 80 and/or 443 straight to the server didn't sound like much of an additional layer. I decided that I should run some type of server on the firewall to proxy traffic to the exchange server. Ultimately, I chose Apache. It has both proxy and reverse proxy capabilities. Other products such as Squid do too, but I thought Apache would give me more flexibility in the future, as you never know what the users are going to want next. Once I have this working, our domain will be responding to web requests as well as SMTP requests; therefore I may have to figure out how to redirect www requests to the current website offsite while still allowing web mail to come in.

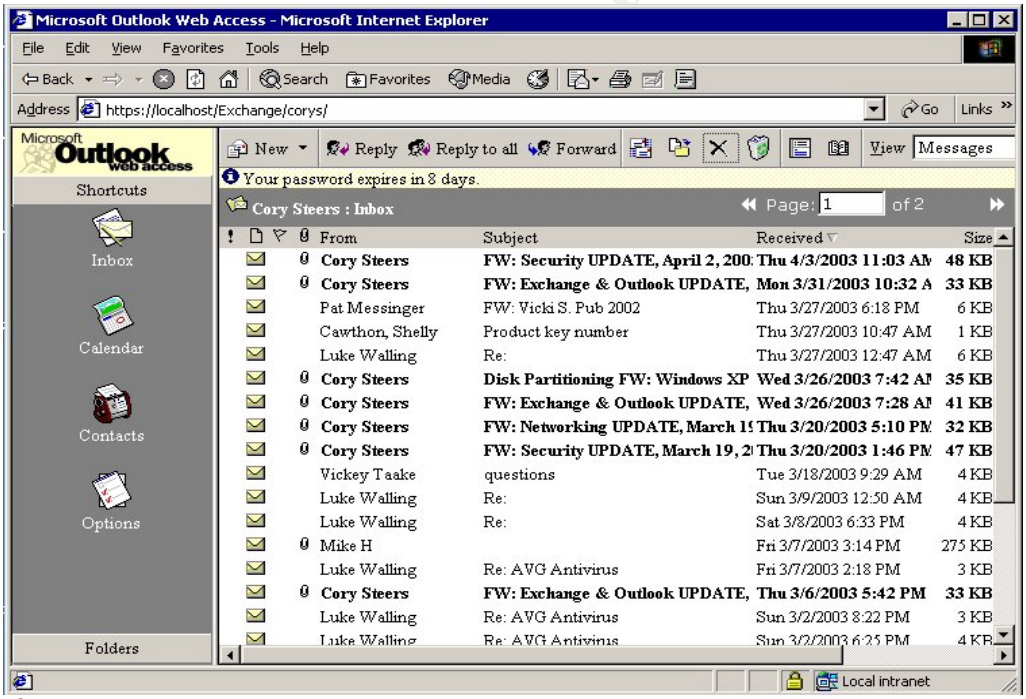

Figure 1.

My first step was to get OWA enabled and test that out on the internal network. In true Microsoft fashion, turning OWA on was very easy. Exchange 2000 server enables OWA for each user by default, so if your server isn't already hosting some type of web content that would require a separate virtual server, you should be able to just point a browser on the internal network to http://servername/Exchange/. If you find that it's not

enabled, you can use this guide from Microsoft. At that point, you'll be provided with user credentials (or not if you're logged in to a domain computer with your domain account), and then you'll get a web page that looks much like a default outlook window does. If that doesn't work, try http://servername/Exchange/<username>/ where <username> is your account user id. If it still isn't working, you can verify that your user account has OWA enabled by:

Start Active Directory Users and Computers
From the View menu, select Advanced Features.
Double-click the user's name to open the Properties dialog
Click on the Exchange Advanced tab, click Protocol Settings
In the Protocols dialog box, double-click HHTP
Select Enable For Mailbox
Click OK 3 times

Figure 1 shows what your browser window will look like by accessing OWA from an IE browser. As you can see, it's easy to mistake it for a true outlook client window.

The next step is to secure the server as much as possible. First, I ran Windows Update to make sure that I had all the latest security patches. I learned in the SANS GSEC training that it's a good idea to sanity check Windows Update with the Network Security Hotfix Checker hfnetchk. Occasionally, Windows Update is missing necessary security patches that hfnetchk will find.

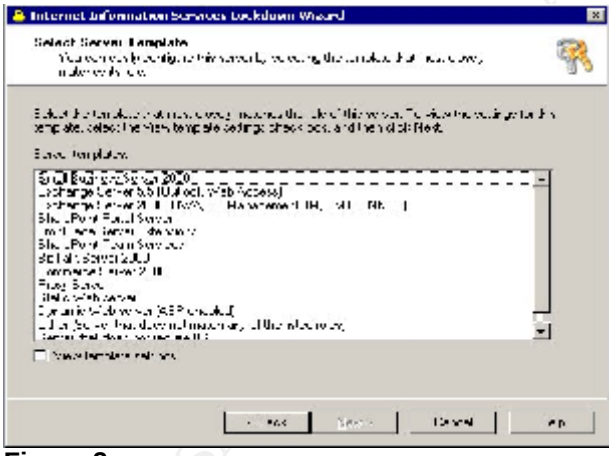

Figure 2.

With the server and its applications up to their current patch level, the next step is to download and install the IIS Lockdown Tool and URLscan. The IIS Lockdown tool will turn off unnecessary features in IIS, thereby reducing risk of Attack. When you run this tool, it has a typical windows initial wizard screen and you click “next” to proceed.

If you take a look at figure 2, you'll see a screen capture from the IIS Lockdown tool. This screen is where you can inform the IIS Lockdown tool what type of server you

have. It ranges from not running IIS at all to needing most, if not all, of the IIS functionality. Fortunately, there is a selection for Exchange servers. This is the ideal setting for a server that's only running IIS for the OWA features of Exchange. This turns off most of the pieces that have all the exploits except WebDav, which is a key component needed for OWA. Interestingly enough, as I was working on this, a severe exploit requiring WebDav was released. Microsoft had a patch out in short order. For more detail you can check the “view template settings” box to see what it's going to do.

The IIS Lockdown Tool installs URLscan 2.0, but there's a 2.5 upgrade, and you should download and install it too. With version 2.5, you now have two choices; baseline URLscan and URLscan-srp. They both include the urlscan.dll and urlscan.ini, but URLscan-srp builds a more restrictive default configuration. According to Microsoft at http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/tools/tools/urlscan.asp the main difference between the URLscan-SRP configuration and that of the Baseline URLscan, is how it's configured to handle chunked encoding data transfers. These transfers are blocked by default in SRP, and not in the baseline version. The SRP version also restricts data uploads to 30MB by default (is 30MB supposed to be restrictive?).

URLscan-SRP installed without introducing any problems to OWA. As long as this is an Exchange server only, and not doing any other web server functionality, you should be safe with the SRP version. If your results differ, just edit the urlscan.ini file.

At this point, I feel I’ve done all I can to secure the exchange server, and OWA is working great on the internal network via unencrypted http traffic. I wasn’t concerned about the traffic being encrypted over the internal network, so it was my intention to deal with ssl on the firewall running Apache.

Now I needed to make it work from the Internet. The key steps needed to make it work from the Internet are:

- Make sure Internet DNS is configured so that OWA is reachable via URL name.
- Change the firewall rules to allow https traffic to the web server on the firewall
- Install and Configure Apache to allow access to the internal OWA Exchange server
- Build SSL key and get OWA working via SSL
- Patch and Secure Apache to prevent exploit of Apache itself

Since I'm using DynDNS to host my DNS records for me, I need to log onto my DynDNS account and add a CNAME record for the single A record <domainname>.org that resolves to my firewall's IP address. I've chosen the name webmail.<domainname>.org, instead of www. This provides a little obfuscation to help hide my OWA server, and gives me the flexibility to use www later for an actual web site.

My firewall is running Mandrake Linux 8.2. Mandrake uses Red Hat Linux as a base for making its distribution, so most of the “behind the scenes” configuration is identical

to Red Hat. From this point forward, I'll reference how Mandrake does things and not differentiate whether Mandrake inherited it from Red Hat or not. Mandrake uses a standard iptables binary, but executing "/etc/init.d/iptables save" executes the binary iptables-save putting the information in /etc/sysconfig/iptables, and "/etc/init.d/iptables start" executes the binary iptables-restore pointing to /etc/sysconfig/iptables. I mentioned this, because once I have a base firewall rule-set, I usually just edit /etc/sysconfig/iptables with my favorite editor rather than using the iptables binary directly. Since I already have an incoming rule to allow SMTP (TCP 25) traffic directly to the firewall, it's very easy to just copy that line and modify the ports for ssl (TCP 443). So, I could determine the rule number I want this to be, and execute "iptables -I INPUT <rule number> -i <interface> -p tcp –dport 443 -m state –state !INVALID -j ACCEPT" or I could just find the existing input rule for smtp in /etc/init.d/iptables, copy that line to the appropriate place within the file and modify it for port 443. That would look something like: "[0:0] -A INPUT -i <interface> -p tcp -m tcp –dport 443 -m state –state NEW,RELATED,ESTABLISHED -j ACCEPT".

Notice the line that I typed in had "--state !INVALID", but the line in the iptables file in /etc/sysconfig had "--state NEW,RELATED,ESTABLISHED". Iptables will accept either, but typically won't write to the /etc/sysconfig/iptables file with the negative. So, while I'm a lazy typist and would rather type "!INVALID", iptables loves to type and would rather write out "NEW,RELATED,ESTABLISHED". The other nice thing about editing the /etc/iptables/sysconfig file directly is that I can comment out this line (or un-comment it) directly and then run "/etc/init.d/iptables start" to add or remove the access. I did this a lot as I was working on the solution. I didn't want to leave the access open until everything was properly configured and hardened.

Now it was time to install and configure Apache. I decided to use unencrypted http to get everything working, and then add the ssl encryption later. Installing Apache on Mandrake is easy using the rpm files. Mandrake compiles Apache to use the /etc/httpd directory for its configuration files. "/etc/httpd/conf/httpd.conf" is the main configuration file for Apache. Httpd.conf includes the file commonhttpd.conf. I decided to put all my modifications in commonhttpd.conf. After doing some research on http://www.apache.org, I realized that the module I needed to use was the "mod_proxy" module, specifically the ProxyPass directive. You can read more about it at http://www.apache.org/docs/mod/mod_proxy.html.

The first entry I made in commonhttpd.conf is "ProxyPass / http://<servername>/Exchange/". When a user types in http://webmail.<domainname>.org/ I want them to go to the /Exchange directory on the exchange server. I don't want them to be able to get to the root directory of the webserver on the Exchange server at all. This line takes care of this. Attempting to connect to the root directory of the apache web server redirects you to the sub directory /Exchange on the Exchange server.

Now, it's time to try it out. I open my browser and type in http://webmail.<domainname>.org /. I quickly realize I have a problem. My browser shows

some errors and it's complaining about not being able to resolve <servername>, which is the name of the internal server. Of course, this name does not have a corresponding Internet address. The problem is that I did correctly retrieve the initial html/asp file from the exchange server; however, within that file are several requests to various other pieces of information such as pictures for the icons. Because the request came to the exchange server as <servername>/Exchange, it built all those additional requests with <servername> instead of webmail.<domainname>.org.

To fix this, I'll have to make webmail.<domainname>.org resolve to the internal IP address of the exchange server for the internal machines on the churches network. I added a zone for <domainname>.org to the churches internal DNS, and made an A record for webmail to point to the exchange server. I also change my Apache configuration to “ProxyPass / http://webmail.<domainname>.org/Exchange/”.

I tried it again, but it still didn't work. I was getting the same, “not found” errors, but the URL now contained webmail.<domainname>.org not <servername>. After a lot of testing and traffic sniffing, I saw what was happening. The requests not being found contained webmail.<domainname>.org/Exchange/Exchange/, with /Exchange being in there twice. With the ProxyPass command that I have in the Apache configuration, what I’m really saying is /<anything> is getting passed on as /Exchange/<anything>. So, I’m attempting to “GET” from a non-existent directory and the Exchange server.

I need to precede this line with another directive: “ProxyPass /Exchange http://<servername>/Exchange/”. It's important that this new line precedes the first line. The first line that matches is the one that's used, and remember that “/” matches everything. This new preceding entry keeps from making requests like /Exchange/Exchange.

At this point, OWA is functional from the Internet, however, the page doesn't quite look right. There are several icons that aren't being retrieved. As it turns out, there's a subdirectory called exchweb that holds these icons. Again, we have a similar problem where /exchweb is getting modified to /Exchange/exchweb. One more entry of “ProxyPass /exchweb http://<servername>/exchweb/” fixes this. Now, we have full access to OWA from the Internet via http. Here are my configuration changes again:

```
ProxyPass /exchweb http://webmail.<domainname>.org/exchweb/
ProxyPass /Exchange http://webmail.<domainname>.org/Exchange/
ProxyPass / http://webmail.<domainname>.org/Exchange/
```

Now I’m ready to get my webmail setup working over SSL/https instead of http. Since I’m more concerned with costs, rather than having an SSL key that’s signed by a recognized authority, I’m not going to bother paying someone like VeriSign to sign the key that I generate. The upside is that I saved some money. The downside is that every time you check your email via the Internet, you’ll get a warning message about not being able to verify the server SSL key. This, in turn, introduces another security issue. Because the users will be accustomed to just clicking past the warning message,

the risk of “man in the middle” attacks is greater. I decided I could accept this risk. The church's email isn't really sensitive, so it doesn't have direct value to anyone else.

The Apache website has a nice document on how to generate a certificate. That sight also has a link for creating your own CA (certificate authority). This is needed if you're going to skip paying a company to certify your key. You need to edit the ssl.default-vhost.conf file in the ssl directory and add/change the SSLCertificateFile entry to point to the name of your CRT file.

Now I’m ready to try OWA via ssl. First, I want to make some other minor Apache configuration changes. In the main httpd.conf file there are directives for Listening on various ports for proxy mode, http, and ssl. I commented out the lines “Listen 8080” under the “if defined APACHEPROXIED” section, and “Listen 80” under the “if defined !SSL” section. Now, when I restart Apache, it’s only listening on port 443. I also need to change my iptables rules to allow TCP 443 inbound, not TCP 80. Right away I notice that things aren’t working. Eventually I have to sniff the traffic again to determine the root cause. Because all my ProxyPass statements in the Apache Configuration are http, and not https, the traffic appears to switch to http between my browser and Apache as well. Apache is no longer listening on port 80, and even if it were my firewall is no longer allowing port 80. In any case this is not what I want.

I wasn’t able to find any documentation on the Internet to help me out with this. There’s probably another way to make it work, but ultimately I decided to add SSL to IIS on the Exchange server, and make it run over SSL end to end. I found another document on the Internet on how to add an SSL key to IIS. Interestingly enough it leverages the CA I built on my Linux box. I’ll have the same issues with this key not being signed by a recognized CA like VeriSign … been there done that. Again, I have to make one last change to the ProxyPass commands in the Apache configuration. They now look like:

```
ProxyPass /exchweb https://webmail.<domainname>.org/exchweb/
ProxyPass /Exchange https://webmail.<domainname>.org/Exchange/
ProxyPass / https://webmail.<domainname>.org/Exchange/
```

It is basically the same, except http has been changed to https.

I try OWA over the Internet again, and this time it’s successful. At this point I have a working configuration. There are a couple of things to note. OWA is a pretty heavy web application, especially if you’re using a recent version of IE (Internet Explorer). OWA will present a rich client look to browsers that can support things like dynamic HTML (which IE 5.0 and later do). It also supports what it calls a reach client which works with any browser that is html 3.2 compliant. The reach client interface is more restrictive but also lighter than the rich client. Also by default, OWA tells the browser not to cache the icons, so every time you login to OWA it’s like the first time. This document (http://www.exchangeadmin.com/Articles/Index.cfm?ArticleID=38466) talks about how to turn off the content expiration headers and also how to force the reach client. I only

recommend the latter in a low bandwidth situation.

The second thing to take notice of is the lack of functionality in the OWA. Don't get me wrong, for a web email application the rich client interface is pretty nice; however, there are still some features saved for the actual outlook fat client. The spell checker is one of them. I needed to make sure my users understood that due to bandwidth and functionality, the OWA should be used as little as possible and shouldn't set expectations too high.

The last thing that I need to do at this point is harden Apache. I have the latest RPM for Apache 1.3, and there are no known vulnerabilities for a patched version of 1.3. To better protect myself for when a new bug is found, I want to turn off all unnecessary features. Mandrake's RPM of Apache installs with functionality in mind. There are several modules that are loaded by default that I'm not using. This wastes system resources and increases my exposure to any future exploits.

The only modules that I know I'm using are mod_ssl for SSL traffic and mod_proxy, for the ProxyPass directive; however, I can't be sure that other modules aren't also being used, and I'm just not aware of it. First, I head back to the Apache website where they have a listing of the modules. I read up on all the modules to help determine whether I need them or not. Next, I decide to disable modules in groups of five and retest OWA to make sure everything is still working. If I find that one or more of the last five modules disabled broke something, I can begin turning them back on, one by one, to find the needed module or modules.

I was surprised at how many modules were needed. After all, IIS was serving up all the content. Some of the modules may have only been needed because of the way the default configuration was set up. I found a few modules that could only be turned off after I changed the default configuration. It's possible that I could have turned off more modules, but couldn't determine what to change in the configuration to leave them off. I also left a couple on that I knew I'd possibly need for some later configurations like the rewrite module. In any case, I went from approximately 37 modules to twelve. Here's a list:

mod_log_config
mod_mime
libproxy
mod_alias
mod_rewrite
mod_access
mod_expires
mod_headers
mod_usertrack
mod_setenvif
libssl
mod_vhost_alias

Now I have a functional and secure solution for providing Internet mail access to the

Church's office mail. This new solution is even more functional than the web mail system that they had with the original email solution. With the exception of the Outlook features missing from OWA, the users can now do most of their email and calendar functions via the Internet as well.

To recap, here are the steps that I took:

- Enable OWA and get it working on the internal network
- Patch the Exchange Server
- Install the IIS Lockdown Tool and URLscan
- Open up http and ssl access on the firewall
- Install Apache on the firewall
- Configure Apache to proxy the traffic
- Build and install the SSL certificates
- Retest the solution over SSL and close the http access
- Harden Apache

Two final things I want to do is develop a system to apply patches to Windows, Exchange, Mandrake Linux, and Apache, as well as draft an acceptable use policy for the churches OWA. Now more than ever, it's important that I keep up with patches for the various systems. There's more access allowed in, and more processes installed and running on the systems, so I have to be ever diligent with applying patches. OWA is a heavy application, and the churches DSL connection has just enough bandwidth to be considered broadband. I need to make sure that the employees understand that this will affect their user experience and may affect other church business requiring Internet access. My hope is that this won't be used much and won't become the preferred method of reading email.

References

1) The Apache Software Foundation: http://www.apache.org

2) Exchange and Outlook Administrator: http://www.exchangeadmin.com

3) Using an OpenSSL demo CA to Sign/Create Certificates for IIS 5.0: http://wrightnet.dhs.org/Public/docs/OpenSSL%20demo%20CA%20Signing%20IIS%205.0%20Certs.htm

4) Microsoft: http://www.microsoft.com

5) Mandrake Linux http://www.mandrakelinux.com

6) W. Richard Stevens. The Protocols (TCP/IP Illustrated, Volume I) 2nd ed. Addison-Wesley Pub Co, 1994.

7) William R. Stanek. Microsoft Exchange 2000 Server Administrator's Pocket Consultant. Microsoft Press, 2000.