Global Information Assurance Certification Paper

Using A Reverse Proxy To Filter HTTP and HTTPS

GIAC Security Essentials Certification (GSEC)
Practical Assignment
Version 1.4b
Option 2 - Case Study in Information Security
Mattison Ward
August 31, 2002

Abstract

To achieve defense in depth, HTTP and/or HTTPS traffic to a vulnerable web server can be filtered with a reverse proxy gateway. This paper discusses implementing a reverse proxy with Apache on Linux. The execution of the reverse proxy uses novel method to filter URLs for strings associated with attacks and only allows certain specified characters such as letters and numbers to pass through. Vulnerability scans are run against an unpatched IIS server running Outlook Web Access before and after filtering by the reverse proxy. Various uses of a reverse proxy, such as proxying two web servers, are discussed as well as the merits of restricting access to the reverse proxy with RSA's SecurID. The reverse proxy, as implemented in this paper, presents a lost-cost method of increasing the level of security for one or more vulnerable web servers.

Introduction

Based on server-side scripting requirements, many web sites and/or web applications are deployed on web server software that is less than ideal from a security standpoint. For example, many small to medium sized businesses deploy Outlook Web Access (OWA) using IIS on their Microsoft (MS) Exchange server in order to allow remote access to email without the complication and administrative hassles of a Virtual Private Network (VPN) client. Due to the large number of ports that must be open between Outlook clients and their Exchange server, many small to medium sized companies position their Exchange server on the internal network. Using security best practices, TCP port 25 traffic to the Exchange server should pass through an email gateway on a DMZ interface of the external firewall. Even companies that will take this step, however, sometimes open up TCP port 80 and 443 access to the Exchange server on the internal interface of the firewall. This action introduces a large threat since new vulnerabilities are discovered for IIS on a weekly or monthly basis. A search of the Common Vulnerability and Exposures (CVE) database at icat.nist.gov shows 54 vulnerabilities for IIS 5, 69 vulnerabilities for IIS 4, and 5 vulnerabilities for OWA 5/2000. Since a company's Exchange server often contains potentially sensitive data in the email database and will often be a domain controller containing domain account information allowing direct access to it from the Internet presents an unnecessary risk. To guard against this specific, prevalent threat, as well as to protect other vulnerable, sensitive web sites and/or web applications, a reverse proxy can be installed between the OWA/Exchange server and the Internet. A reverse proxy acts as an application layer firewall for the HTTP and HTTPS protocols. It is a gateway to a web server(s) through which all requests from the outside must pass. For a general overview of the concept of the reverse proxy see <http://rr.sans.org/web/reverse_proxy.php>.

A reverse proxy can be deployed in a DMZ and the protected web server moved to the internal interface of the firewall such that all external HTTP and HTTPS traffic to the protected web server must first pass through the reverse proxy. Additionally, the firewall rule set can be configured so that only TCP port 443 or 443 and 80 on the reverse proxy are accessible from the Internet. Additionally, only TCP ports 80 and/or 443 need to be open between the reverse proxy and the protected web server.

In order to minimize software cost and maximize flexibility, a reverse proxy can be implemented with relatively inexpensive hardware, the Linux OS, and the Apache web server. The filtering done by a reverse proxy improves security on a number of levels. Anyone trying to exploit the protected web server has to either compromise the reverse proxy or pass exploit traffic through it. Since the reverse proxy, unlike a 'front-end' Exchange Enterprise 2000 IIS server, needs only to send HTTP or HTTPS traffic to the protected web server, a breach will not necessarily allow exploitation of the protected web server. For example, after breaking into a 'front-end' IIS server a hacker could use Netbios vulnerabilities to exploit the 'back-end' Exchange server.

Furthermore, far fewer exploits have been found for the 1.3.x series of Apache web server than for IIS. A search of the CVE database at icat.nist.gov shows 8 vulnerabilities for Apache on Red Hat Linux. When exploits for Apache do surface, patches can be applied with minimal or no service interruption as opposed to a reboot on the protected web server. It is difficult to pass exploit traffic through the reverse proxy for the following reasons:

- Many web exploits involve passing invalid or malformed HTTP requests to a server and Apache will only pass RFC compliant HTTP or HTTPS traffic
- Traffic to the protected web server can be restricted to select portions of the protected web server with mod_rewrite
- Traffic to the protected web server can be filtered based on URL content
- Authentication other than the Windows domain credentials can be required to access the reverse proxy

In conjunction with thorough host security on the OWA server, I will show that a reverse proxy is a cost-effective way to greatly increase the difficulty of breaking into OWA or other protected web servers.

Before Implementation

My initial experimentation with a reverse proxy was conducted in a controlled lab environment. This allowed me to try various methods of reverse proxying with the Apache configuration as well as to conduct vulnerability tests on an OWA server before and after filtering traffic with the reverse proxy. I have subsequently deployed this solution for about a dozen organizations so the implementation section of my paper will represent the sum of those experiences as well as the test setup in a lab environment. I performed vulnerability tests on an unpatched IIS/Exchange 2000/OWA server using the Nessus, twwwscan, whisker, and N-Stealth vulnerability scanners. The most serious vulnerabilities included: the PROPFIND vulnerability which allows a denial of service attack (<http://www.guninski.com/iispropover.html>); the index server (.ida) vulnerability which gives remote SYSTEM level access to the web server (http://www.eeye.com/html/Research/Advisories/AD20010618.html); the cross site-scripting vulnerability which could result in various buffer overflows; FrontPage vulnerabilities that reveal server configuration information; and example files such as the scripts samples. The index server vulnerability leaves this server susceptible to the Code Red virus (http://www.eeye.com/html/Research/Advisories/AL20010717.html) while the other security flaws indicate that it could be manually hacked with a minimum of effort by anyone with an Internet connection.

The majority of companies who have requested a reverse proxy were looking for a way to expose OWA or Novell Groupwise web access securely allowing their employees to access email remotely instead of deploying and supporting VPN to their laptops and home computers. About half of these companies had already exposed OWA and the other half would not consider exposing it until some secure method, such as the reverse proxy, had been put in place. For other businesses, a reverse proxy was needed to provide secure,

restricted access through a requirement of credentials different from the MS domain account to an intranet site on IIS. One client hosts their main web site on IIS in their DMZ so the reverse proxy needed to filter all of the traffic for this site as well as OWA on the same IIS server. Finally, another client needed to use one proxy to filter two different IIS servers: one running OWA and one running Citrix Nfuse.

Since most employees interact with email at some level, I have found it easy to justify the value of the reverse proxy to management staff with a simple risk assessment of OWA on Exchange server. A compromise of OWA on the Exchange server could at the very least cause a service interruption while the compromised server is rebuilt. This is assuming an attack was caught before any further damage was done. The cost of not being able to send or receive email could be large if the interruption lasts for an extended period of time. At worst, confidential internal emails and attachments could be accessed or destroyed and domain accounts could be harvested.

Implementing The Solution

The hardware minimums I have tried to use for reverse proxies are a Pentium 2 (or the equivalent) or above and at least 128 MB RAM. The busiest site I have protected, however, employs a dual Pentium 200 processor system with 128 MB RAM. That system is able to filter 6,155 hits per day and 24 megabytes per day with no discernible difference between access speed before and after filtering by a reverse proxy.

Some of my colleagues had set up reverse proxies for OWA on Exchange 5.5, but were unable to get it to work correctly with OWA on Exchange 2000 because of differences in HTML references to absolute versus relative paths. After many unsuccessful tries, we had given up on reverse proxying OWA on Exchange 2000 until I found an article in Google groups that pushed me in the right direction (http://groups.google.com/groups?q=mod_proxy_add_forward.c&hl=en&lr=&ie=UTF-8&selm=2prG6.1195%24O7.30461%40news1.mts.net&rnum=4). I learned from this posting that one must add a line to the /etc/hosts file with the private IP address of the OWA backend and the FQDN that points to the public IP address of the reverse proxy. Unlike OWA on Exchange 5.5, all references in the Apache configuration file must be to the FQDN used to access the reverse proxy, not the internal IP address of the Exchange server. Additionally, the posting specified a module that had been created for Apache that does not come with the Red Hat distribution called mod_proxy_add_forward. The source code is available at <http://freshmeat.net/projects/mod_proxy_add_forward.c>. This module adds a 'X-Forwarded-For' header to proxy requests. Together, these three items allowed for the reverse proxying of OWA on Exchange 2000. In contrast with the implementation of the reverse proxy with ProxyPass and ProxyPassReverse as is suggested in the Google posting or a RewriteRule that passes requests regardless of URL content as is suggested in <http://rr.sans.org/web/reverse_proxy.php>, I discovered that I could improve filtering of the reverse proxy dramatically by employing RewriteRules that would only pass characters that I specified.

To test this new method of implementing a reverse proxy with Apache, I set up a test box for vulnerability scanning with Red Hat Linux. I employed the latest version of Apache 1.3 and Apache modules. Routine security measures were employed on the server: Services were disabled such that the server only listened on TCP ports 22 (SSH), 80 (HTTP), and 443 (HTTPS), /tmp was mounted nosuid, and SSH was disabled for root. TCP wrappers and firewall rules were used to restrict SSH access to a small number of management workstations. My implementation of reverse proxy has been tested with Red Hat 7.2 and 7.3 and all of the accompanying revisions of Apache. I would expect the reverse proxy to work on any Apache server with the appropriate modules, though the vulnerabilities for Apache on Win32 exclude Windows as a platform.

In the /etc/hosts file of the reverse proxy there must be an entry that reads:

```
10.121.124.2   webmail.foo.com
```

where 10.121.124.2 is the IP address of the OWA server and webmail.foo.com is the FQDN for the IP of the reverse proxy. Though this entry is not strictly necessary for reverse proxying OWA on Exchange 5.5, it is mandatory for Exchange 2000. Unnecessary modules should be commented out of httpd.conf. The modules that I kept are:

```
LoadModule vhost_alias_module modules/mod_vhost_alias.so
LoadModule env_module         modules/mod_env.so
LoadModule config_log_module  modules/mod_log_config.so
LoadModule agent_log_module   modules/mod_log_agent.so
LoadModule referer_log_module modules/mod_log_referer.so
LoadModule mime_module        modules/mod_mime.so
LoadModule negotiation_module modules/mod_negotiation.so
LoadModule autoindex_module   modules/mod_autoindex.so
LoadModule dir_module         modules/mod_dir.so
LoadModule cgi_module         modules/mod_cgi.so
LoadModule action_module      modules/mod_actions.so
LoadModule userdir_module     modules/mod_userdir.so
LoadModule alias_module       modules/mod_alias.so
LoadModule rewrite_module     modules/mod_rewrite.so
LoadModule access_module      modules/mod_access.so
LoadModule auth_module        modules/mod_auth.so
LoadModule anon_auth_module   modules/mod_auth_anon.so
LoadModule db_auth_module     modules/mod_auth_db.so
LoadModule proxy_module       modules/libproxy.so
LoadModule expires_module     modules/mod_expires.so
LoadModule headers_module     modules/mod_headers.so
LoadModule setenvif_module    modules/mod_setenvif.so
LoadModule perl_module        modules/libperl.so
LoadModule ssl_module         modules/libssl.so
LoadModule proxy_add_forward_module /usr/lib/apache/mod_proxy_add_forward.so
```

Since this server will have few users, to conserve memory I minimized the number of processes Apache would use in httpd.conf:

```
MinSpareServers 1
MaxSpareServers 10
StartServers 1
```

To obscure the version of Apache I changed the following lines in httpd.conf:

```
ServerSignature Off
ServerTokens Prod
ProxyVia Off
```

In my implementation of a reverse proxy, incoming URL requests are filtered with RewriteRules. RewriteRules are made up of regular expressions and are processed in order. The main regular expressions used in the rewrite rules are:

^ = beginning of a line
() = anything in the parentheses will be passed to the variable '1'
$ = end of line
. = any one character
[a-zA-Z0-9] = any character between a and Z or 0 and 9
* = matches the preceding element zero or more times
.* = any string
\ = 'escape' character - treat the following symbol as character rather than a regular expression metacharacter - always put it before symbols (e.g. $, ?, etc.) in rewrite rules if you are not sure if a symbol is a metacharacter.

See http://etext.lib.virginia.edu/helpsheets/regex.html for further discussion of regular expressions. See http://httpd.apache.org/docs/misc/rewriteguide.html for further discussion of rewrite rules. The following is the relevant section of httpd.conf for the Rewrite rules that specify what traffic will be passed to the OWA server:

```
<VirtualHost _default_:443> # https (tcp port 443) section title
RewriteEngine  On
# items here are almost never legitimate traffic so they are sent to forbidden [F]
RewriteRule     ^(http|ftp)://.*    -           [F]
RewriteRule     ^(.*)?/iisadmin/?-          [F]
RewriteRule     ^(.*)?/samples/?            -           [F]
RewriteRule     ^(.*)?/scripts/?   -          [F]
RewriteRule     ^(.*).ida$          -          [F]
RewriteRule     ^(.*).htw$          -          [F]
RewriteRule     ^(.*)./_vti/_.      -          [F]
RewriteRule     ^(.*).idq$          -          [F]
RewriteRule     ^(.*).exe$      -       [F]
RewriteRule     ^(.*)?/winnt/?    -       [F]
# only reverse proxy [P] exchange directories public, exchweb, exchange
# only proxy letters, numbers, forward slash, dot, underscore, hyphen, space
# other characters can be added as needed (e.g. \=\@\#\$\"\&\%)
RewriteRule ^/public([a-zA-Z0-9/\.\_-\ ]*)$          https://webmail.foo.com/public/$1        [P]
RewriteRule ^/exchweb([a-zA-Z0-9/\.\_-\ ]*)$     https://webmail.foo.com/exchweb/$1    [P]
RewriteRule ^/exchange([a-zA-Z0-9/\.\_-\ ]*)$   https://webmail.foo.com/exchange/$1   [P]
# send everything else to forbidden
RewriteRule .* - [F]
RewriteLog      /var/log/httpd/rewrite_log
RewriteLogLevel          1
```

When reverse proxying OWA on an Exchange 2000 server, authentication must be changed from NTLM to basic authentication in Exchange Manager and IIS manager and the 'allow scripts' permission must be set for both. NTLM or 'Challenge-Response' authentication cannot be proxied. OWA in Exchange 2000 does not follow the HTTP standards and is thus is difficult to proxy. To allow even more control over HTTP traffic and to simplify the proxying, enable the downlevel client option (Microsoft Knowledge Base article q296232). The downlevel client option forces all browsers, not just non-MS and IE before 5, to view a simplified version of OWA. If the downlevel client option is used, HTTP methods can be limited in httpd.conf:

```
<Limit POST GET>
</Limit>
```

If you cannot enable the downlevel client option, you may have to use less restrictive rewrite rules such as:

```
RewriteCond %{REQUEST_URI} ^(.*)?Outbox?
RewriteRule ^/exchange(.*)$ https://webmail.foo.com/exchange$1 [P,L]
RewriteRule ^/public(.*)$ https://webmail.foo.com/public$1 [P,L]
RewriteRule ^/exchweb(.*)$ https://webmail.foo.com/exchweb$1 [P,L]

RewriteCond %{REQUEST_URI} ^(.*)?Inbox?
RewriteRule ^/exchange(.*)$ https://webmail.foo.com/exchange$1 [P,L]
RewriteRule ^/public(.*)$ https://webmail.foo/public$1 [P,L]
RewriteRule ^/exchweb(.*)$ https://webmail.foo/exchweb$1 [P,L]

RewriteCond %{REQUEST_URI} ^(.*)?Drafts?
RewriteRule ^/exchange(.*)$ https://webmail.foo.com/exchange$1 [P,L]
RewriteRule ^/public(.*)$ https://webmail.foo.com/public$1 [P,L]
RewriteRule ^/exchweb(.*)$ https://webmail.foo.com/exchweb$1 [P,L]

RewriteCond   %{REQUEST_URI}      ^(.*)
RewriteRule ^/exchange([a-zA-Z0-9/\.\_-\ ]*)$ https://webmail.foo.com/exchange$1    [P]
RewriteRule ^/public([a-zA-Z0-9/\.\_-\ ]*)$ https://webmail.foo.com/public$1     [P]
RewriteRule ^/exchweb([a-zA-Z0-9/\.\_-\ ]*)$ https://webmail.foo.com/exchweb$1     [P]
```

When reverse proxying OWA on Exchange 5.5 one can use rewrite rules that utilize TCP port 80 to pass traffic between the reverse proxy and the IIS server. On OWA on Exchange 2000, the rewrite rules must use HTTPS or the end user will see a warning in their browser that not all objects on the page are encrypted. Therefore, reverse proxying OWA on Exchange 2000 involves obtaining a SSL certificate for IIS 5. Since the end user browser only connects directly to the reverse proxy, a certificate for IIS can be signed by an untrusted certificate authority (CA) such as the CA that comes with Windows 2000 server. After installing the CA on a Windows 2000 server on your network, it is fairly simple to install the certificate on any IIS 5 server that can access the IIS service running on the CA server. To do this, open a browser on the OWA server and enter the IP address of the CA server and /CertSrv. After filling out the information for requesting an Advanced Certificate and approving the certificate request from the MMC on the CA, the certificate can be downloaded from the /CertSrv site and installed with a wizard from the IIS MMC on the OWA server (http://www.microsoft.com/windows2000/techinfo/planning/walkthroughs/default.asp#section7).

While some companies do not mind the browser warning that results from using the untrusted certificate that comes with Apache by default on a reverse proxy, others requested a certificate signed by a trusted CA. Certificate Signing Requests and private keys can be generated with OpenSSL. Certificates can be purchased online and installed with little trouble (See http://www.entrust.net/tech/apachemod_ssl/index.htm for a walk-through). I found a problem in that the private key generated by the Entrust documentation required a passphrase to be entered every time the httpd service was started. Luckily, I found a method to have the passphrase entered automatically. In the Apache configuration file change 'SSLPassPhraseDialog builtin' to 'SSLPassPhraseDialog exec:/etc/httpd/conf/ssl-passphrase.' The ssl-passphrase file should contain 'echo "thepassphrase"' and the permissions set so that it is only accessible by root.

Although I suggested that companies only open TCP port 443 through the firewall to the reverse proxy to improve confidentiality, many requested that TCP port 80 traffic also be allowed to the Apache server since some users are not familiar with typing https into the location bar of their browser. For those implementations, I added a rule in the virtual hosts section of httpd.conf that would redirect all requests to the reverse proxy on TCP port 80 to the Exchange directory on TCP port 443:

```
RewriteRule .*  https://webmail.foo.com/exchange/       [L]
```

Some of companies requested a means of viewing a summary of the traffic going through the reverse proxy. To accommodate this request, I modified the rewrite rules to allow access to the files created by webalizer with a rewrite rule that would not pass traffic on to the IIS server. I placed this rule before the one directing all traffic to https and restricting access to internal IP addresses:

```
RewriteRule     ^/usage([a-zA-Z0-9/\.\_-\ ]*)    /usage$1       [L]
```

Using this technique, HTTP requests for files on the reverse proxy can be filtered and served at the same time that requests to other directories are filtered and passed to the protected server. Finally, to reverse proxy IIS on two different servers multiple rewrite rules need to be used. For example, the following rewrite rules would filter requests to the root directory of the reverse proxy to an IIS server running Nfuse, while requests to the Exchange directories would go to the OWA server.

```
RewriteRule ^/([a-zA-Z0-9/\.\_-\ ]*)$              https://citrix.foo.com/nfuse/$1             [P,L]
RewriteRule ^/public([a-zA-Z0-9/\.\_-\ ]*)$        https://owa.foo.com/public/$1               [P,L]
RewriteRule ^/exchweb([a-zA-Z0-9/\.\_-\ ]*)$     https://owa.foo.com/exchweb/$1
[P,L]
RewriteRule ^/exchange([a-zA-Z0-9/\.\_-\ ]*)$    https://owa.foo.com/exchange/$1        [P,L]
```

After Implementation

After setting up the test reverse proxy, I first verified that from outside the firewall I could log in to OWA at https://webmail.foo.com/exchange/. I reran the vulnerability tests against the reverse proxy. Nessus discovered the exchange directory, but noted only that it required authentication. Neither N-Stealth nor twwwscan found any vulnerabilities. Since the IIS server was not patched and had not been 'secured' with iislockdown or urlscan before the reverse proxy implementation it may seem obvious that vulnerabilities were found on it while they were not on the patched and secured Apache server. The point of the vulnerability tests is to show that these sample vulnerabilities on the IIS server could be filtered through the reverse proxy without any changes to the Exchange server. Furthermore, implementing the reverse proxy in the DMZ only protects the OWA server to one vector of attack: outside the network. Before connecting the OWA server to the network, one should, at the very least, install iislockdown and urlscan and apply service packs and hotfixes to the server.

The various businesses for which I implemented a reverse proxy all had fairly trouble-free installations. The ones for which I reverse proxied OWA on Exchange 2000 introduced the most difficulty. Until I made the RewriteRules more permissive as outlined at the end of the implementation section, I received complaints from users who were unable to open certain emails. By searching the httpd access log for 403 Forbidden errors I was able to track these false positives down. Since OWA puts the subject lines of messages into the URL when viewing a message, subjects containing symbols such as '!' or '...' or '&' were blocked if those characters were not allowed by the rules. The following shows an example of a message that could not be viewed because of an ellipsis in the subject line:

```
10.152.182.145 - - [28/Jul/2002:23:10:11 -0400] "GET /exchange/mward/Inbox/
Re:%20An%20idea....EML?Cmd=open HTTP/1.1" 403 254
```

Additionally, some users complained that some attachments could not be downloaded. OWA on Exchange 2000 does unique encoding of attachments names that required passing any characters through the RewriteRules. Other problems arose when the logrotate script that ran nightly would HUP the httpd process and it would not restart correctly. I modified the script to kill the httpd process and restart it to correct that issue. The logrotate configuration also had to be modified to run monthly rather than weekly for webalizer to work correctly. Finally, on versions of OWA on Exchange 5.5 before service pack 4, I found that I had to allow all characters through the rewriterules on URLs containing the forms directory in order for users to reply to messages:

```
RewriteRule ^/exchange/forms(.*)$   http://webmail/exchange/forms/$1    [P,L]
```

In contrast to the aforementioned problem with downloading attachments, one client was concerned that deployment of a reverse proxy and OWA would allow confidential documents transmitted via email within the company to be downloaded to the insecure home PCs of end users. On OWA for Exchange 5.5, it was trivial to add a rewrite rule that would prevent access to attachments. The almost limitless flexibility to restrict access to any web application by crafting rewrite rules with regular expressions shows the real advantage of using an Apache-based reverse proxy to filter web traffic over a 'canned' commercial application.

After finalizing the configuration of the reverse proxy, I implemented host-based intrusion detection with Tripwire. One of the strengths of the reverse proxy is that even if it were to be compromised, it does not contain any sensitive data other than the internal IP address of the OWA server. Nevertheless, it is important to implement host-based and network-based intrusion detection, such as Snort, so that a compromise can be detected and handled before it can lead to any possible breach of other hosts on the network.

Conclusions

Vulnerability scanners and hot fixes can only detect and correct known vulnerabilities. The value of the reverse proxy is protection not only against known vulnerabilities, but also against future vulnerabilities that have not yet been disclosed. Since the reverse proxy will only pass traffic that is RFC compliant, certain attacks will never work through it. For example, if the rewrite rules are all replaced with the following line:

```
RewriteRule ^/(.*)$        http://webmail.foo.com/$1        [P]
```

then web traffic will be allowed through the proxy to any part of the IIS server regardless of the content of the URL. If a vulnerability scan is run against this configuration, then only the printer vulnerability is found. The more serious .ida, PROPFIND, and cross scripting vulnerabilities are blocked merely by passing through the Apache server.

The Microsoft solution to shielding an Exchange server from the Internet is to purchase Exchange 2000 Enterprise edition and deploy a 'front-end' Exchange server to handle OWA. This demands a premium in price to buy the Enterprise edition and is less flexible with respect to securing OWA. In addition, the following ports must be open between the front-end and back-end Exchange servers as specified in MSKB q280132:

```
53 (Transmission Control Protocol [TCP], User Datagram Protocol [UDP]) - Domain Name
System (DNS). 88 (Transmission Control Protocol [TCP], UDP) - Kerberos authentication. 123
(TCP) - Windows Time Synchronization Protocol (NTP). Note that this is not necessary for
Windows 2000 logon capability, but may be configured or required by the network
administrator. 135 (TCP) - EndPointMapper. 389 (TCP, UDP) - Lightweight Directory Access
Protocol (LDAP). 445 (TCP) - Server message block (SMB) for Netlogon, LDAP conversion
and distributed file system (Dfs) discovery. 3268 (TCP) - LDAP to global catalog servers. One
port for the Active Directory logon and directory replication interface. Exchange 2000 front-
end and back-end connectivity only requires that additional ports be open as needed for
whatever communication is desired (for example, Web client front-end and back-end
connectivity requires port 80 [TCP] open, IMAP 143 [TCP], and so on). Additionally, any
connectivity by secure protocols such as IPSec or Secure Sockets Layer (SSL)-secured
Hypertext Transfer Protocol (HTTP), Internet Message Access Protocol (IMAP), or Post Office
Protocol version 3 (POP3)
```

The requirement to have ports 135 and 445 open, not to mention the other listed ports, means that moving the IIS service from the 'main' Exchange server to a front-end server would only slightly delay a compromise of the IIS server in spreading to the other Exchange server(s).

To add non-repudiation to the solution, a reverse proxy server is deployed in an environment with a RSA ACE Server. A SecurID module can easily be added to Apache such that any access to the server can require authentication from webpages built into the module. The Apache module accesses a Linux ACE Agent that passes the credentials on to the ACE Server for verification.

The SecurID system utilizes a token that produces a tokencode that changes every minute. A user authenticates either with the tokencode and a memorized pin number or a hash of the two. The tokencode is synchronized with the ACE Server. The ACE Server will disable the account if a brute-force attack occurs and will ask for the next tokencode if the user enters the tokencode before or after the valid one (Harper, 2001).

The SecurID system improves security in several ways. If the protected web server is password protected, as is the case with OWA, one must obtain a user's domain credentials, PIN number and SecurID token to access the server or somehow circumvent both authentication mechanisms. If HTTPS is not employed or one of the keys is compromised, sniffing the IP traffic to obtain the

passwords will reveal the domain credentials and RSA passcode, but since the passcode can only be used once and changes every minute the protected server cannot be accessed. Since the system only requires the user to remember a short PIN number, password management problems of resetting passwords or changing them periodically mostly disappear. Finally, since the SecurID system requires two-factor authentication a hacker must obtain a token that has been activated for the corresponding ACE Server as well as a PIN number that matches a valid username.

Unfortunately, the cost of an ACE server is often outside the reach of many small and medium sized businesses. If the business has Checkpoint's Firewall-1, access to the reverse proxy can be restricted by using a 'client-auth' rule such that users must authenticate to a web page generated by the firewall with usernames and passwords stored on the firewall and the requirement that passwords must be different from the user's MS domain password. If the business has a Cisco PIX firewall, access to the server can be restricted using the 'aaa authentication' command. Radius can be installed on the reverse proxy and the PIX set to use the Linux usernames and passwords, which can be easily managed by command-line-averse system administrators with Webmin, for authentication. Regrettably, both Firewall-1 and the PIX will transmit usernames and passwords in clear text by default.

In conclusion, the security of confidential or vulnerable web servers can be vastly improved at a low Total Cost of Ownership by filtering traffic to the servers with rewrite rules on an Apache-based reverse proxy. This filtering supplements host security on the web server(s) and is enhanced by requiring strong authentication with non-Microsoft credentials to access the reverse proxy. Since OWA is deployed insecurely on many Exchange servers, a reverse proxy is perfectly suited to shield OWA without having to purchase additional Microsoft software. Furthermore, the flexibility of rewrite rules allows the reverse proxy to host websites while proxying certain URLs, proxy for multiple websites, block the downloading of attachments and otherwise restrict any access to vulnerable servers that can be crafted with regular expressions.

References

Apache HTTP Server Project. "Apache 1.3 URL Rewriting Guide". December 1997. URL:http://httpd.apache.org/docs/misc/rewriteguide.html

eEye Digital Security. ".ida 'Code Red' Worm". July 17, 2001. URL:http://www.eeye.com/html/Research/Advisories/AL20010717.html

eEye Digital Security. "All versions of Microsoft Internet Information Services Remote buffer overflow (SYSTEM Level Access)". June 18, 2001. URL: <http://www.eeye.com/html/Research/Advisories/AD20010618.html>

Entrust. "Certificate Services: Apache Web Server with OpenSSL, mod_SSL". 2002. URL: http://www.entrust.net/tech/apachemod_ssl/index.htm

Guninski, Georgi. "IIS 5.0 PROPFIND DOS". 8 March 2001. URL: http://www.guninski.com/iispropover.html

Harper, Chris. "Securing Critical Network Resources with Two-Factor Authentication". June 11, 2001. <URL:http://rr.sans.org/authentic/twofactor.php>

Microsoft Knowledge Base. "XCCC: Empty Inbox When Using Internet Explorer 5 and Later to Gain Access to OWA". Apr 12, 2001. <URL:http://support.microsoft.com/default.aspx?scid=kb;[LN];Q296232>

Microsoft Knowledge Base. "XCCC: Exchange 2000 Windows 2000 Connectivity Through Firewalls". Nov 20 2000. URL:http://support.microsoft.com/default.aspx?scid=KB;EN-US;q280132&

Microsoft. "Windows 2000 Step-by-Step Guides". February 16, 2001. URL:http://www.microsoft.com/windows2000/techinfo/planning/walkthroughs/default.asp#section7

National Instititute of Science and Technology. "ICAT Metabase: A CVE Based Vulnerability Database". September 5, 2002. URL:http://icat.nist.gov/

Ramsay, Stephen. "Using Regular Expressions". October 5, 1997. URL:http://etext.lib.virginia.edu/helpsheets/regex.html

Stricek, Art. "A Reverse Proxy Is A Proxy By Any Other Name". January 10, 2002. URL: <http://rr.sans.org/web/reverse_proxy.php>