

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin

At this point in the series, I am sure that some of you are saying, "This is a lot of work to implement a secure wireless environment." And to those individuals, I would have to reply, "Yes, it is. But you went through the same thing when you set up your LAN, didn't you?" Up to this point in this series, we have implemented no "service" you should not already have in place on your LAN. Hopefully, you learned something new about how to set up those services, and maybe even made them more secure.

There is no point in implementing a proxy server if no one is going to use it. And, because the environment we are setting up is for "guest" users, we ideally want it to work without any (or, at least as little) support required. So, this month we will look at implementing Web Proxy Auto Discovery Protocol (WPAD).

When firewalls were first introduced, they were all proxy based. So, in a firewalled environment, users needed to manually configure proxies for services such as http, ftp, and gopher. As you can imagine, working in a "proxy" environment does have some drawbacks: First, you need a client that can support proxies. Second, you need to get the proxy information. The second problem is particularly troublesome for guest users because they have to get the information in order to connect to the network. This often resulted in calls to the support desk for internal users (who have forgotten what the proxies are) and the guests who need access.

With the advent of stateless firewalls, the requirements for proxies faded away, except in the case of Web caching. In an environment in which a large group of users retrieve very similar content, Web caching can improve performance and save Internet bandwidth (between 10% and 30%), especially if you have low-speed connections. There are also security considerations for using Web caching, which we have covered in past articles. However, to get the security and the savings, users need to use a proxy.

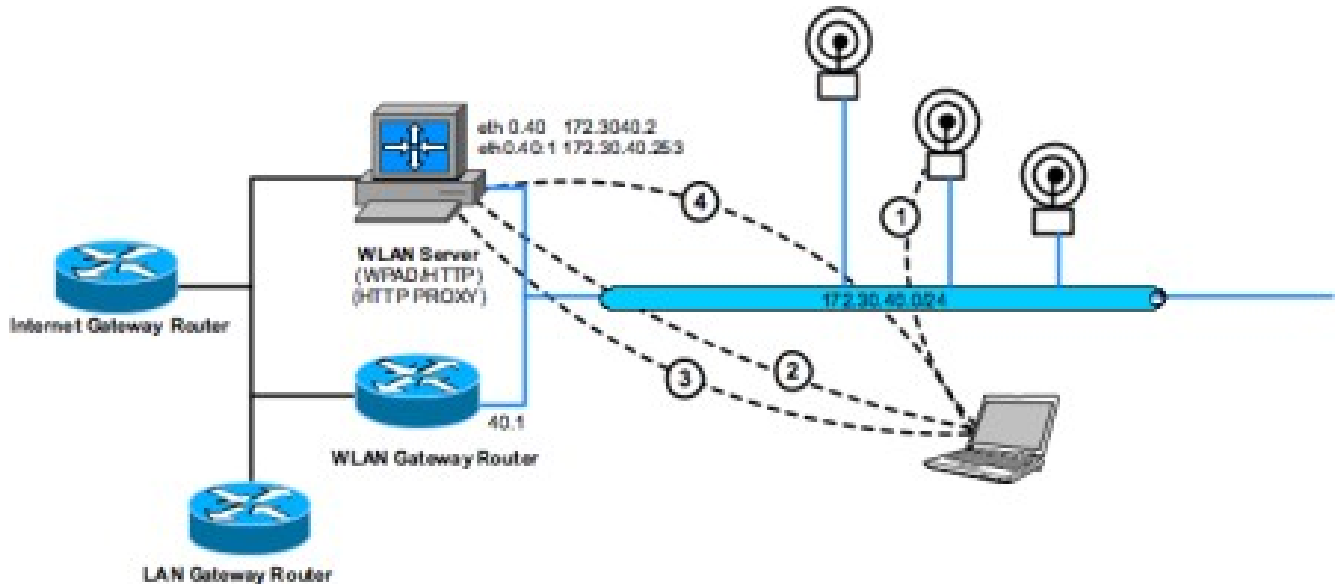
Enter WPAD. Invented by Microsoft, it was developed into an IETF draft (now expired). WPAD is more a process than a protocol. Web browsers can employ the process to automatically discover Web proxy server information and configure the Web browser to utilize the server for the duration the application operates. Although WPAD has expired as a standard, Microsoft's Internet Explorer (IE5 and higher) and the Mozilla Web browsers support WPAD. In the case of IE, it is enabled by default. To make sure it's enabled (some corporate Active Directory environments disable it as a security measure), select the Tools>Internet Options>Connections>LAN Settings> menu, then check the "Automatically detect settings" checkbox. In Mozilla or Firefox, you can check by selecting the Tools>Options>Connections Settings> menu. Make sure "Auto-detect proxy settings for this network" is selected.

Now, as mentioned above, WPAD is more of a process than a protocol. The process workings are illustrated here:

- Host joins the wireless segment
- Client launches Web browser with WPAD enabled. Browser performs WPAD server lookup.
- Web browser successfully resolves wpad.x, downloads wpad.dat.
- Web browser establishes connection with web proxy server and starts to send URL requests to proxy server.

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin



In order to get this process to work, an environment needs to have a few different elements in place. Two very important elements that regular readers should be very familiar with are DHCP and DNS. DHCP provides client IP addressing and the host's fully qualified domain name (FQDN) information. The DNS server provides the name "resolving" service that enables the host to figure out what it's FQDN is and resolves the wpad.x, a record query that tells the Web client the IP address of the server that is hosting the wpad.dat file. This is the point where we need to "expand" our infrastructure, so to speak. We have the above DHCP and DNS resources in place (theoretically, at least). In order to get our WPAD service up and running, we need to complete two tasks: build a Web server and create our wpad.dat file. Then we'll need to make the required DNS/DHCP changes.

Building a Web Server

In the Unix/Linux world, Apache is the "standard" for Web servers. However, Apache is a little bulky when it comes to configuration and is far more than we need to simply host a WPAD server. As an alternative, we will implement a lightweight but very effective http service called `< href=http://www.acme.com/software/mini_httpd/mini_httpd-1.19.tar.gz>mini_httpd`. Of course, all build efforts start with getting the source code:

```
[root@localhost src]# /usr/local/bin/wget
http://www.acme.com/software/mini_httpd/mini_httpd-1.19.tar.gz
--07:58:42--          http://www.acme.com/software/mini_httpd/mini_httpd-
1.19.tar.gz
=> `mini_httpd-1.19.tar.gz'
Resolving www.acme.com... 216.27.178.28
Connecting to www.acme.com|216.27.178.28|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 42,063 (41K) [application/x-tar]

100%[=====>] 42,063          39.92K/s

07:58:44 (39.82 KB/s) - `mini_httpd-1.19.tar.gz' saved [42063/42063]

[root@localhost src]#
Once the code is on our server, we need to expand the tar ball:
```

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin

```
[root@localhost src]# rm -rf mini_httpd-1.19
[root@localhost src]# tar xvfz mini_httpd-1.19.tar.gz
mini_httpd-1.19/
mini_httpd-1.19/README
mini_httpd-1.19/Makefile
...
mini_httpd-1.19/version.h
mini_httpd-1.19/FILES
[root@localhost src]#
```

Now, we move to the source directory, type "make," and we have our applications.

```
[root@localhost mini_httpd-1.19]# make
rm -f mime_encodings.h
sed < mime_encodings.txt > mime_encodings.h \
  -e 's/#.*//' -e 's/[ ]*$//' -e '/^$/d' \
  -e 's/[ ]*[ ]*/", 0, "/" -e 's/^{ "/" -e 's/$/', 0 },/'
rm -f mime_types.h
sed < mime_types.txt > mime_types.h \
  -e 's/#.*//' -e 's/[ ]*$//' -e '/^$/d' \
  -e 's/[ ]*[ ]*/", 0, "/" -e 's/^{ "/" -e 's/$/', 0 },/'
gcc -O -c mini_httpd.c
gcc -O -c match.c
gcc -O -c tdate_parse.c
gcc -O -s mini_httpd.o match.o tdate_parse.o -lcrypt -o mini_httpd
gcc -O -c htpasswd.c
gcc -O -s htpasswd.o -lcrypt -o htpasswd
[root@localhost mini_httpd-1.19]#
```

There are two applications: htpasswd and mini_httpd. These applications and the configuration files on which they rely now all must be installed in a chroot jail. This will make the Web server as secure as the other applications running in our proxy environment.

```
[root@localhost mini_httpd-1.19]# adduser httpd -s /sbin/nologin
[root@localhost mini_httpd-1.19]# mkdir -m 0700 /usr/local/httpd
[root@localhost mini_httpd-1.19]# mkdir -p /usr/local/httpd/htdocs
[root@localhost mini_httpd-1.19]# mkdir -p /usr/local/httpd/bin
[root@localhost mini_httpd-1.19]# mkdir -p /usr/local/httpd/etc
[root@localhost mini_httpd-1.19]# mkdir -p /usr/local/httpd/cgi-bin
[root@localhost mini_httpd-1.19]# mkdir -p /usr/local/httpd/lib
[root@localhost mini_httpd-1.19]# mkdir /usr/local/man/man8
[root@localhost mini_httpd-1.19]# mkdir /usr/local/man/man1
```

Now we install the binaries and man pages:

```
[root@localhost mini_httpd-1.19]# cp ./mini_httpd /usr/local/httpd/bin
[root@localhost mini_httpd-1.19]# cp ./htpasswd /usr/local/httpd/bin
[root@localhost mini_httpd-1.19]# cp mini_httpd.8 /usr/local/man/man8
[root@localhost mini_httpd-1.19]# cp htpasswd.1 /usr/local/man/man1
```

The last step is to copy the libraries on which htpasswd and mini_httpd depend:

```
[root@localhost mini_httpd-1.19]# ldd mini_httpd
```

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin

```
libcrypt.so.1 => /lib/libcrypt.so.1 (0x00b18000)
libc.so.6 => /lib/tls/libc.so.6 (0x00659000)
/lib/ld-linux.so.2 (0x00640000)
[root@localhost mini_httpd-1.19]# ldd httpasswd
libcrypt.so.1 => /lib/libcrypt.so.1 (0x00b18000)
libc.so.6 => /lib/tls/libc.so.6 (0x00659000)
/lib/ld-linux.so.2 (0x00640000)
[root@localhost mini_httpd-1.19]#
```

We need to make one more directory:

```
mkdir -p /usr/local/httpd/lib/tls
```

```
[root@localhost mini_httpd-1.19]# cp /lib/libcrypt.so.1 usr/local/httpd/lib
[root@localhost mini_httpd-1.19]# cp /lib/ld-linux.so.2 /usr/local/httpd/lib
[root@localhost mini_httpd-1.19]# cp /lib/tls/libc.so.6 /usr/local/httpd/lib/tls
```

With the application installed, we must install the support files for mini_httpd. There are three:

- mime_encodings.txt
- index.html,
- mini_httpd.cnf.

The mime_encodings.txt and mini_httpd.cnf files can be copied from the build directory to the /usr/local/httpd/etc directory. These files are part of the default configuration and do not require any editing.

```
[root@localhost mini_httpd-1.19]# cp ./mime_encodings.txt
/usr/local/httpd/etc
[root@localhost mini_httpd-1.19]# cp ./mini_httpd.cnf /usr/local/httpd/etc
```

The index.html file is not required for WPAD, but is required for normal Web server operation. You can use the one supplied with the distribution for testing.

```
[root@localhost mini_httpd-1.19]# cp ./index.html /usr/local/httpd/etc
```

The last file required is the server configuration file. This file is called from the command line and tells the daemon what IP address and service port to listen on, the user to run the server as and the directory structure. Here is an example:

```
dir=/usr/local/httpd
data_dir=/usr/local/httpd/htdocs
user=httpd
host=172.30.71.21
port=80
```

These items can all be defined using command line flags, but this option is far easier. To start the mini-http daemon, run the following:

```
/usr/local/httpd/bin/mini_httpd -C /usr/local/httpd/etc/mini_httpd.conf
```

```
[root@localhost ~]# /usr/local/httpd/bin/mini_httpd -C
/usr/local/httpd/etc/mini_httpd.conf
```

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin

```
/usr/local/httpd/bin/mini_httpd: started as root without requesting  
chroot(), warning only  
[root@localhost ~]#
```

This will start the service. To make sure the server is operating correctly, connect to the server address with your Web browser and you should see an index page.

Creating The Wpad.Dat File

Our second task is to create a wpad.dat file. The file is essentially a Netscape proxy.pac file with a new name. Prior to WPAD, Netscape came up with a feature that allowed network administrators to build a proxy configuration file that the Web browser could download and then use to configure proxy settings. This approach had the advantage of making it easier to manage a proxy environment, but it still had the downside of requiring the user (in most cases) to configure the browser to look for the proxy.pac file. Microsoft did develop a method to provide users the location of the proxy.pac file via DHCPINFORM message and/or an Active Directory policy (on certain Windows OS variants), but this approach is only supported by Windows systems. A very simple proxy configuration file is required to support WPAD. It has three rules:

Rule 1: If the Web site requested is local, do not proxy; send directly to the server.

Rule 2: If the Web site is not covered by rule #1, then send to the proxy.

Rule 3: If the proxy fails, then send the request directly to the server.

This basic rule set covers most proxy environments. You do not normally want local Web sites to be accessed thru a proxy server; performance is always better direct. You want external sites to go to the proxy. However, you can add rules to make exceptions for external sites if you need to. You also want users to be able to access Web sites if the proxy is down. Here is an example using the Web site domain name as the "local" qualifier:

```
function FindProxyForURL(url, host)
{
    if (isPlainHostName(host) ||
        dnsDomainIs(host, "any.net"))
        return "DIRECT";
    else
        return "PROXY proxy.open.outlan.net:8080; DIRECT";
}
```

Alternatively, you can use a host's IP address as the "local" qualifier:

```
function FindProxyForURL(url, host)
{ if(isPlainHostName(host) ||
    isInNet(host,"172.30.71.0","255.255.255.0")) return "DIRECT";
  else return "PROXY 172.30.80.101:8080; DIRECT";
}
```

In both of these examples, as I mentioned above, if the proxy is down, the HTTP client attempts to connect the site with a direct request. In our WLAN proxy environment, we do not want users directly connecting, so we need to remove rule #3. This is done easily by removing the "DIRECT" statement following the proxy server designation. Here is an example:

```
function FindProxyForURL(url, host)
{
```

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin

```
if (isPlainHostName(host) ||
    dnsDomainIs(host, "any.net"))
    return "DIRECT";
else
    return "PROXY proxy.open.outlan.net:8080;";
}
```

With the wpad.dat, if the proxy server is down, the user gets an error page from the client.

Changes to DNS/DHCP

The third and final step is to make the required DNS/DHCP changes. On the DNS side, you need to create A-records and CNAME (canonical name) pointers for wpad.x.x and the proxy server. On the DHCP side, you must make sure that you are propagating OPTION 15 – Domain Name to your DHCP clients, otherwise the hosts are unable to properly resolve the WPAD server. For example, let's say we have three IP subnets we want to configure WPAD for.

IP Subnet	Domain Name
172.30.71.0	usr.outlan.net
172.30.80.0	open.outlan.net
192.168.20.0	cell.outlan.net

There are few approaches one can take. The first (and simplest) is to use a common domain name for each network: outlan.net. The second, (which can be more complex) is for each IP subnet to have its own sub-domain.

The browser will not derive its domain-name information by performing a reverse lookup of the host's assigned IP address. In order for WPAD to work, the domain name needs to be defined using OPTION 15. So, unless you plan to implement forward and reverse lookup entries and require that hosts on each subnet can be identified by sub-domain for each DHCP assigned client address, the "single domain" is fine for most implementations. That said, host identification aside, if you wish to have subnet-specific proxy handling rules, implementing the "sub domain" option with the subnet-specific wpad.dat files is the only way to go. It is also possible to strike a happy medium by having each DNS resolve to a common WPAD server using a CNAME record. Let's take a look at some configuration examples.

On the DHCP side, the configuration to support the "single domain" approach would look like this:

```
subnet 172.30.71.0 netmask 255.255.255.0 {
range 172.30.71.10 172.30.71.100
option routers 172.30.71.1;
option domain-name "outlan.net";
option domain-name-servers 172.30.80.101;
option subnet-mask 255.255.255.0;
}
```

```
subnet 172.30.80.0 netmask 255.255.255.0 {
range 172.30.80.10 172.30.80.100
option routers 172.30.80.1;
option domain-name "outlan.net";
option domain-name-servers 172.30.80.10;
```

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin

```
option subnet-mask 255.255.255.0;
}

subnet 192.168.20.0 netmask 255.255.255.0 {
range 192.168.20.10 192.168.20.100
option routers 192.168.20.1;
option domain-name "outlan.net";
option domain-name-servers 172.30.80.101;
option subnet-mask 255.255.255.0;
}
```

The DHCP configuration for the sub-domains would look like this:

```
subnet 172.30.71.0 netmask 255.255.255.0 {
range 172.30.71.10 172.30.71.100
option routers 172.30.71.1;
option domain-name "usr.outlan.net";
option domain-name-servers 172.30.80.101;
option subnet-mask 255.255.255.0;
}
```

```
subnet 172.30.80.0 netmask 255.255.255.0 {
range 172.30.80.10 172.30.80.100
option routers 172.30.80.1;
option domain-name "lab.outlan.net";
option domain-name-servers 172.30.80.101;
option subnet-mask 255.255.255.0;
}
```

```
subnet 192.168.20.0 netmask 255.255.255.0 {
range 192.168.20.10 192.168.20.100
option routers 192.168.20.1;
option domain-name "wlan.outlan.net";
option domain-name-servers 172.30.80.101;
option subnet-mask 255.255.255.0;
}
```

The DNS configuration for the "single domain" implementation requires that only a forward and in-addr zone file be configured for the outlan.net domain. In our WLAN environment, we are running the proxy, DHCP and DNS on the same server, so only one A-Record is required. The wpad and proxy server entries are set with CNAME records. Here is the forward zone file:

```
@                IN          SOA      outlan.net. hostmaster.outlan.net. (
                    5              ; Serial
                    8H            ; Refresh
                    2H            ; Retry
                    4W            ; Expire
                    1D)           ; Minimum TTL

outlan.net.      IN          NS       tisiphone.outlan.net.
                IN          MX 10    outlan.net
tridant.outlan.net. IN A        172.30.80.101
wpad.outlan.net.  IN CNAME    tridant.outlan.net.
proxy.outlan.net. IN CNAME    tridant.outlan.net.
```

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin

And here is what the in-addr zone file looks like:

```
@                IN          SOA      outlan.net. hostmaster.outlan.net. (  
                6          ; Serial  
                8H        ; Refresh  
                2H        ; Retry  
                4W        ; Expire  
                1D)       ; Minimum TTL  
                IN          NS      outlan.net.  
  
101.80.30.172.in-addr.arpa.      IN PTR    tridant.outlan.net.
```

In a sub-domain environment, forward and in-addr zone files would be needed for each IP subnet, but the required entries would be quite similar. The exception would be if a single WPAD server were used to support all of the subnets. In this case, only CNAME entries would be needed for the two sub-domains that did not have their own WPAD server. For example, let's say that the WPAD server is physically located in the usr.outlan.net subnet. The forward zone file would look like this:

```
@                IN          SOA      usr.outlan.net. hostmaster.outlan.net. (  
                5          ; Serial  
                8H        ; Refresh  
                2H        ; Retry  
                4W        ; Expire  
                1D)       ; Minimum TTL  
usr.outlan.net.      IN          NS      tisiphone.usr.outlan.net.  
                IN          MX 10    outlan.net  
tridant.usr.outlan.net.      IN A        172.30.80.101  
wpad.usr.outlan.net.        IN CNAME    tridant.usr.outlan.net.  
proxy.usr.outlan.net.       IN CNAME    tridant.usr.outlan.net.
```

The other sub-domain forward DNS records would have only CNAME records:

```
@                IN          SOA      open.outlan.net. hostmaster.outlan.net. (  
                5          ; Serial  
                8H        ; Refresh  
                2H        ; Retry  
                4W        ; Expire  
                1D)       ; Minimum TTL  
open.outlan.net.      IN          NS      tisiphone.usr.outlan.net.  
  
wpad.open.outlan.net.      IN CNAME    tridant.usr.outlan.net.  
proxy.open.outlan.net.     IN CNAME    tridant.usr.outlan.net.
```

Most of the problems with implementing WPAD end up being issues with DHCP and DNS. As you can see, there are a number of ways that you can implement WPAD. You just need to figure out what works best for your environment. One of the easiest ways to troubleshoot problems with WPAD is to use TCPdump to monitor your web browser's behavior. If your WPAD implementation is working correctly when the Web browser loads up, you should see a DNS request for the WPAD server, followed by a DNS request for proxy server:

```
root# tcpdump -i en1 -A -n -s 96 host 192.168.20.254 and dst port 53  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
```

Building A WLAN Proxy Server - Implementing WPAD

Michael J. Martin

```
listening on en1, link-type EN10MB (Ethernet), capture size 1500 bytes
23:23:07.483746 IP 192.168.20.30.49193 > 192.168.20.254.53: 28215+
A? wpad.cell.outlan.net. (38)
23:23:23.558607 IP 192.168.20.30.49198 > 192.168.20.254.53: 49655+
A? proxy.outlan.net.
(39)
```

If you don't see these DNS requests, there is something wrong. First, make sure that the host knows what its domain name is. In the Mac OS X, this can be done using the <ipconfig> command, with the {getpacket <int>} option:

```
dhcp29:~ user$ ipconfig getpacket en1
op = BOOTREPLY
xid = 1099324492
secs = 3995
ciaddr = 172.30.80.29
yiaddr = 172.30.80.29
siaddr = 172.30.71.101
options:
Options count is 11
dhcp_message_type (uint8): ACK 0x5
server_identifier (ip): 172.30.71.101
lease_time (uint32): 0x2904
subnet_mask (ip): 255.255.255.0
router (ip_mult): {172.30.80.1}
domain_name_server (ip_mult): {172.30.80.101}
domain_name (string): outlan.net
ldap_url (string): ldap://ds.outlan.net/dc=outlan,dc=net
end (none):
dhcp29:~ user$
```

In the Windows environment, this information can be viewed using the <ipconfig> command.

If the host has the proper domain information (and is performing the DNS resolutions when the Web client starts), the next step is to check the wpad.dat file syntax. Use the examples above for comparison.

Well, that's all you should need to know about WAPD. Next time we will jump the last user access hurdle for WLAN proxy: how to passively redirect user Web traffic to allow for network access.