Web Application Footprints and Discovery

Methodology for Web servers hosting Multiple Web applications

Abstract

Web application assessment begins with IP address and ports (80/443) – this is very common practice. But there is flaw in this method. What if a web server is running with multiple virtual hosts? In other words, one server is running more than one web application.

In such a scenario, a web application assessment done on such IP/Port combinations may fail and produces partial results. Doing a reverse DNS on the IP and using it as HOST field in HTTP is an option, but may also fail most of the time.

So, where does the solution to this problem lie? The solution lies in the *WHOIS* information database and DNS server.

This paper describes how to fetch this information and follow up with the discovery process for web applications.

Shreeraj Shah

Co-Author: "Web Hacking: Attacks and Defense" (Addison Wesley, 2002) and published several advisories on security flaws.

n e t - s q u a r e
http://www.net-square.com
shreeraj@net-square.com

Problem Domain

Background

Let us consider an example where a web server is running on IP address 203.88.128.10 – a hypothetical one – on the Internet. Port 80 is open and HTTP traffic moves back and forth on this server. This web server may be hosting more than one web application.

A characteristic of the HTTP protocol is that HOST information is read in with every request made to the server and response obtained from the server, depending on the HOST tag supplied with the HTTP request.

For our example, let us assume that Apache web server is running on 203.88.128.10. Its httpd.conf resembles like the screenshot below:

httpd.conf on 203.88.128.10

```
<VirtualHost *:80>
    #   ServerAdmin webmaster@dummy-host.example.com
    DocumentRoot /usr/local/apache2/htdocs
    #   ErrorLog logs/dummy-host.example.com-error_log
    #   CustomLog logs/dummy-host.example.com-access_log common
</VirtualHost>

<VirtualHost *:80>
    #   ServerAdmin webmaster@dummy-host.example.com
    DocumentRoot /usr/local/apache2/htdocs/blue
    ServerName www.blue.com
    #   ErrorLog logs/dummy-host.example.com-error_log
    #   CustomLog logs/dummy-host.example.com-access_log common
</VirtualHost>

<VirtualHost *:80>
    #   ServerAdmin webmaster@dummy-host.example.com
    DocumentRoot /usr/local/apache2/htdocs/red
    ServerName www.red.com
    #   ErrorLog logs/dummy-host.example.com-error_log
    #   CustomLog logs/dummy-host.example.com-access_log common
</VirtualHost>
```

From the above blocks we deduce that the web server is running in the virtual host model. There are two hosts defined and each has an application running on them:
www.red.com
www.blue.com

The first block of the virtual host section had the *DocumentRoot* as */usr/local/apache2/htdocs*, the default root for web server – a client sending an *http* request without HOST information will be served by this root.

Now let's evaluate web server responses to three different *http* requests.

Request

```
C:\Documents and Settings\Administrator> nc 203.88.128.10 80
HEAD / HTTP/1.0
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 11 Jan 2005 20:17:40 GMT
Server: Apache/2.0.50 (Unix) mod_ssl/2.0.50 OpenSSL/0.9.7d mod_jk2/2.0.4
Content-Location: index.html.en
Vary: negotiate,accept-language,accept-charset
TCN: choice
Last-Modified: Fri, 04 May 2001 00:01:18 GMT
ETag: "1c4d0-5b0-40446f80;1c4e6-961-8562af00"
Accept-Ranges: bytes
Content-Length: 1456
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en
Expires: Tue, 11 Jan 2005 20:17:40 GMT
```

The above request is served by the *default root* and we get a default page of size 1456.

Request

```
C:\Documents and Settings\Administrator> nc 203.88.128.10 80
HEAD / HTTP/1.0
Host: www.blue.com
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 11 Jan 2005 20:17:45 GMT
Server: Apache/2.0.50 (Unix) mod_ssl/2.0.50 OpenSSL/0.9.7d mod_jk2/2.0.4
Last-Modified: Tue, 04 Jan 2005 23:10:29 GMT
ETag: "1865-b-f991a340"
Accept-Ranges: bytes
Content-Length: 11
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

The above request is served by *DocumentRoot /usr/local/apache2/htdocs/blue directive*. A page size of 11 was sent for the web application associated with Host *www.blue.com*. This indicates that an entirely new application was served by web server depending on Host tag of the GET/HEAD/POST request. In our case, the request was HEAD.

Request

```
C:\Documents and Settings\Administrator> nc 203.88.128.10 80
HEAD / HTTP/1.0
Host: www.red.com
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 11 Jan 2005 20:17:57 GMT
Server: Apache/2.0.50 (Unix) mod_ssl/2.0.50 OpenSSL/0.9.7d mod_jk2/2.0.4
Last-Modified: Tue, 04 Jan 2005 23:16:57 GMT
ETag: "1cc0b-9-10b20c40"
Accept-Ranges: bytes
Content-Length: 9
Connection: close
Content-Type: text/html; charset=ISO-8859-1
```

The above request is served by the *DocumentRoot /usr/local/apache2/htdocs/red directive*. A page size of 9 was sent for the web application associated with Host www.red.com. This means that a completely new application was served by the web server depending on the HOST tag of a GET/HEAD/POST request. In our case, the request was HEAD.

Problem

In the example illustrated above, we assume that we have access to the Apache configuration file *httpd.conf*. Based on this assumption, we can send a "HOST" tag with the correct value. This is not the case in real life. All the information that is at your disposal is just one IP address (e.g. 203.88.128.11) or one host name (e.g. www.yahoo.com)

What we should be able to do is to make a list of possible hosts pointing to same IP address using this inadequate bit of information. How do we go about gathering more information? This issue is addressed in this paper.

Solution

Step 1: *Find Nameservers* for a particular IP Address

Our first objective is to find a nameserver for a particular IP address. For example let's take IP 203.88.128.10. We need to find the block in which this IP address is assigned and the nameserver for this particular IP address, so that we can run some queries if needed.

Running the following (hypothetical) *whois* query on an ARIN database, produces this result:

```
C:\Program Files\GnuWin32\bin>jwhois -h whois.arin.net 203.88.128.10
[Querying whois.arin.net]
[whois.arin.net]

OrgName:    XYZ corp
OrgID:      XYZC
Address:    101 First Avenue
City:       NYC
StateProv:  NY
PostalCode: 94089
Country:    US

NetRange:   203.88.128.0 – 203.88.128.255
CIDR:       203.88.128.0/20
NetName:    XYZC-4
NetHandle:  NET-203-88-128-0-1
Parent:     NET-203-0-0-0-0
NetType:    Direct Allocation
NameServer: ns1.xyz.com
NameServer: ns2.xyz.com
Comment:
RegDate:    2003-07-17
Updated:    2003-07-17

OrgTechHandle: NA098-ARIN


OrgTechName:   Netblock Admin
OrgTechPhone:  +1-212-999-9999
OrgTechEmail:  netblockadmin@xyz.com

# ARIN WHOIS database, last updated 2005-01-10 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.

C:\Program Files\GnuWin32\bin>
```

We now have nameserver information for this IP address.

We can perform a port scan on an entire range and look for UDP port 53 as well. This would give us any alternative nameservers running on this range.

We now have one or more IP addresses, running as nameservers for our target IP address. These nameservers can be used as our enumeration points.

Step 2: *Look for PTR records* on the nameserver

Next, we use *nslookup* and try to query the PTR record for the target IP address as shown in the screenshot below:

```
C:\Documents and Settings\Administrator>nslookup
Default Server:  ns1.icenet.net
Address:  203.88.128.7

> server ns1.xyz.com
Default Server:  [203.88.128.250]
Address:  203.88.128.250

> 203.88.128.10
Server:  [203.88.128.250]
Address:  203.88.128.250

Name:    www.blue.com
Address:  192.168.7.50

> set type=PTR
> 203.88.128.10
Server:  [203.88.128.250]
Address:  203.88.128.250

10.128.88.203.in-addr.arpa     name = www.blue.com
10.128.88.203.in-addr.arpa     name = www.red.com
>
```

We are able to get hold of their PTR record. We can see there are two domains pointing to same IP address. The same procedure can be repeated using the *dig* command as well.

Step 3: *PTR record fails*

Many times we don't get hold of PTR records on the nameserver or entries are simply not created on the nameserver. In this case we cannot locate the right information about the target IP address. One of the ways to solve this complicated problem is by querying a *WHOIS* server.

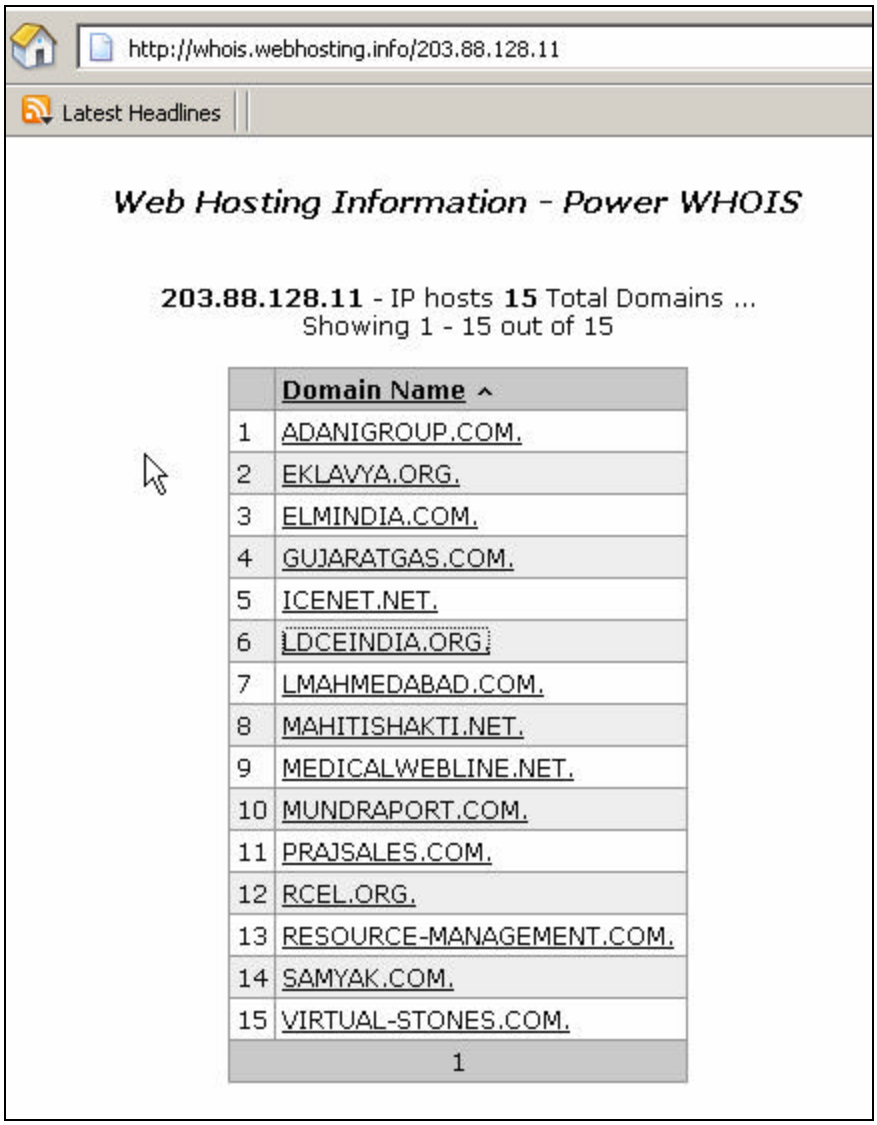
A normal *whois* query, supported by the whois protocol and server is limited in its capacity to retrieve the type of information sought, since such a query is not supported. But there are a few whois server databases that have this database created. Tapping these databases is essential to

web application footprining. This query can be called *reverse IP query*. One of the databases we can query is webhosting.info.

To illustrate how this is done, let us use the IP address 203.88.128.11 [Real IP on the Net], one which doesn't have a PTR record on the nameserver.

```
C:\Documents and Settings\Administrator>nslookup
Default Server:  ns1.icenet.net
Address:  203.88.128.7

> server 203.88.128.250
Default Server:  icedns1.icenet.net
Address:  203.88.128.250

> 203.88.128.11
Server:  icedns1.icenet.net
Address:  203.88.128.250

Name:    ice.128.client11.icenet.net
Address:  203.88.128.11

> set type=PTR
> 203.88.128.11
Server:  icedns1.icenet.net
Address:  203.88.128.250

Non-authoritative answer:
11.128.88.203.in-addr.arpa      name = ice.128.client11.icenet.net
> 203.88.128.11
Server:  icedns1.icenet.net
Address:  203.88.128.250

Non-authoritative answer:
11.128.88.203.in-addr.arpa      name = ice.128.client11.icenet.net
>
```

Not quite there yet. We still do not get the number of hosts residing on the server, but we can run a *whois query on webhosting.info.* Take a look at the results below. [This is a live IP for an ISP]



Bingo! 15 hostnames pointing to the same IP address. We now know that 15 applications are running on this server. This database is evolved by the service provider over a period of time and may not be accurate but is a good starting point for the discovery phase.

Step 4: *Discovering* each of these hosts

We can send a HEAD request to an IP address using each of the above hosts and see what kind of response we get from the server.

Request 1 [Default]

```
C:\Documents and Settings\Administrator>nc 203.88.128.11 80
HEAD / HTTP/1.0
```

Response

```
HTTP/1.1 404 Object Not Found
Server: Microsoft-IIS/4.0
Date: Thu, 27 Jan 2005 10:12:16 GMT
Content-Type: text/html
Content-Length: 102

<html><head><title>Error</title></head><body>The system cannot find the file spe
cified. </body></html>
```

We sent a request with no HOST specified and got back a **404** response.

Request 2 [*junk* as host]

```
C:\Documents and Settings\Administrator>nc 203.88.128.11 80
HEAD / HTTP/1.0
Host: junk
```

Response

```
HTTP/1.1 404 Object Not Found
Server: Microsoft-IIS/4.0
Date: Thu, 27 Jan 2005 10:14:37 GMT
Content-Type: text/html
Content-Length: 102

<html><head><title>Error</title></head><body>The system cannot find the file spe
cified. </body></html>
```

In this request, we sent **"junk"** as a value for the Host tag and we got a **404** once again.

Request 3

```
C:\Documents and Settings\Administrator>nc 203.88.128.11 80
HEAD / HTTP/1.0
Host: icenet.net
```

Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Content-Location: http://icenet.net/index.htm
Date: Tue, 11 Jan 2005 10:07:12 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Wed, 05 Jan 2005 06:52:02 GMT
ETag: "0553fff3f2c41:b3ae6"
Content-Length: 33442
```

In this request we sent “icenet.net” as host value which we derived from step 3 and got 200 back as a response, with 33442 as content length and some ETag value. This is sure Host residing on this server.

Request 4

```
C:\Documents and Settings\Administrator>nc 203.88.128.11 80
HEAD / HTTP/1.0
Host: adanigroup.com
```

Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Content-Location: http://adanigroup.com/index.htm
Date: Tue, 11 Jan 2005 10:07:24 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Wed, 28 Apr 2004 14:51:55 GMT
ETag: "80771d59302dc41:b3ae6"
Content-Length: 806
```

Similarly, we can get access to “adanigroup.com”.

Request 5

```
C:\Documents and Settings\Administrator>nc 203.88.128.11 80
HEAD / HTTP/1.0
Host: www.mundraport.com
```

Response

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Content-Location: http://www.mundraport.com/index.htm
Date: Tue, 11 Jan 2005 10:09:56 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Thu, 01 Jul 2004 05:59:09 GMT
ETag: "80f45486305fc41:b3ae6"
Content-Length: 607
```

We have also obtained access to www.mundraport.com.

Continuing in this manner, we can have a list of all live hosts mapped to one particular IP address on the server. Each host can be treated as a separate web application falling within the scope of assessment.

Conclusion

Web applications are an integral part of real-time business automation today. This has resulted in a proportionate increase in security incidents making web application assessments – defined, but still evolving – the norm rather than the exception.

Serious challenges arise when doing web application assessments on web servers that host multiple virtual hosts and all this with zero knowledge about the number of web applications mapped to a single IP address. Using the manual techniques outlined in the paper, the methodology pinpoints specific ways to discover applications and enhance web application assessment with tangible results.

Acknowledgement

Lyra Fernandes for her help on documentation.