Penetration Testing Professional

Information Gathering

Section 5: Web App Security – Module 2

2. Information Gathering

- 2.1 Gathering Information on Your Targets
- 2.2 Infrastructure
- 2.3 Fingerprinting Frameworks and Applications
- 2.4 Fingerprinting Custom Applications
- 2.5 Enumerating Resources
- 2.6 Information Disclosure Through Misconfiguration
- 2.7 Google Hacking
- 2.8 Shodan HQ

Gathering Information On Target

Information gathering is the very first and most critical step of every penetration test.

**It does not matter if you have to assess the security of an entire network or a single web application, you need to know as much detail as possible of your target or targets.

Most pentesting jobs are black-box tests.

During a *black-box* test, penetration testers simulate an external hacker's attack. By design, they do not know the inner processes, technology or any other internal information. Therefore, it makes the information they discover crucial.

Gathering information about the target is the initial phase of any penetration test. You will quickly find that in general, this is the most important part of the entire engagement.

At this stage, there is no unnecessary information; everything you collect should be noted for future use. The wealth of information you collect will become useful in both understanding application logic and during the attack phase.

What sorts of information are we going after?

- Infrastructure (Web server, CMS, Database…)
- Application Logic
- IPs, Domains and Subdomains
- Virtual hosts

During this chapter on information gathering techniques, you will be given valuable tips on how to store this information efficiently.

We highly recommend that you follow our approach. The better organized the information you collect, the easier it will be to find and exploit vulnerabilities. Please refer to our methodology documents for more advice.

For simplicity, we will pretend to be unaware of what systems and individuals are behind a given website.

The first step of information gathering usually starts away from the organizations network. It begins with their electronic footprint, not just of their employees, but also of their network and websites.

Here are some tools to help you do that.

WHOIS lookups are used to look up domain ownership details from different databases. They were traditionally done using a command line interface, but a number of simplified web-based tools now exist.

Web-based WHOIS clients still rely on the WHOIS protocol to connect to a WHOIS server to perform lookups and command-line execution. WHOIS clients are still widely used by system administrators.

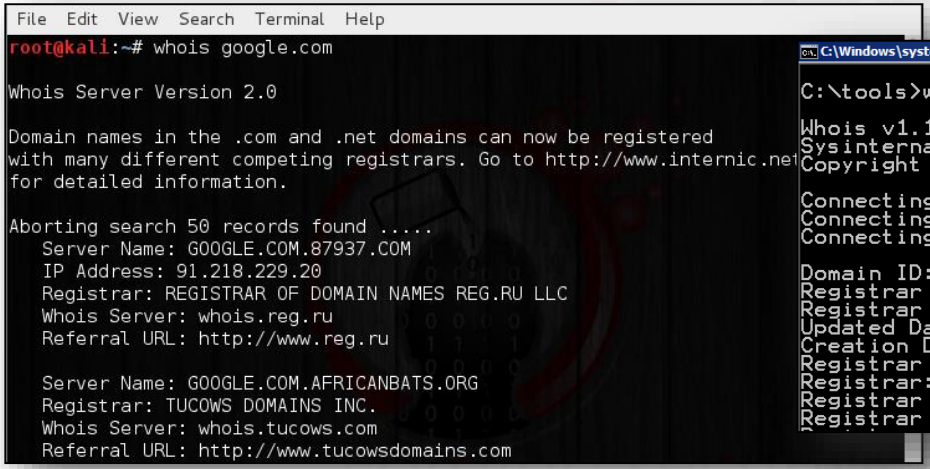
WHOIS normally runs on TCP port 43.

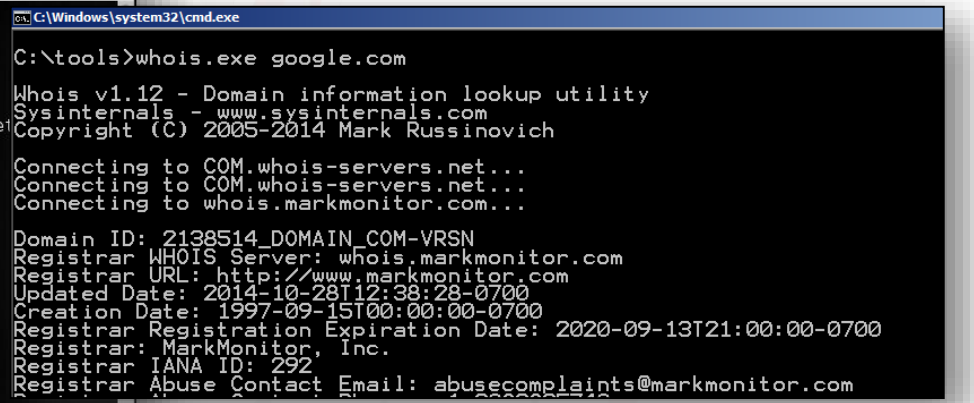
The following slides will guide you through the steps needed to use the Whois service to get information about a website.

We will first use *Whois* with a command line utility and then we will take a look at an online, web-based tool.

Whois Example - Command Line:

You can perform *Whois* queries by using the `whois` *nix command or by installing the Sysinternal Whois utility.





https://docs.microsoft.com/en-us/sysinternals/downloads/whois

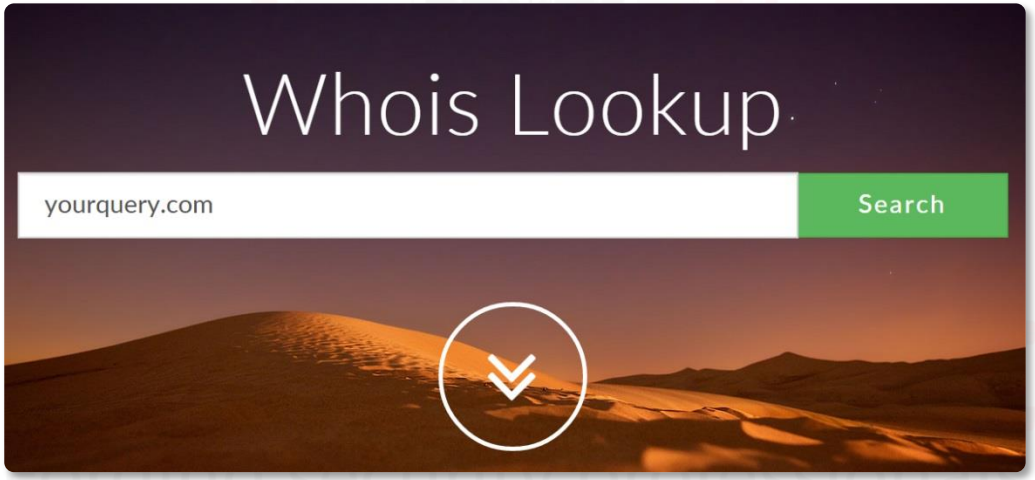
Whois Example - Command Line:

Note: on *nix systems you can set or specify many options. To list them, just run `whois -h`.

```
root@kali:~# whois -h
whois: option requires an argument -- 'h'
Usage: whois [OPTION]... OBJECT...

-h HOST, --host HOST   connect to server HOST
-p PORT, --port PORT   connect to PORT
-H                     hide legal disclaimers
      --verbose        explain what is being done
      --help           display this help and exit
      --version        output version information and exit

These flags are supported by whois.ripe.net and some RIPE-like servers:
-l                     find the one level less specific match
-L                     find all levels less specific matches
-m                     find all one level more specific matches
-M                     find all levels of more specific matches
-c                     find the smallest match containing a mnt-irt attribute
-x                     exact match
-b                     return brief IP address ranges with abuse contact
-B                     turn off object filtering (show email addresses)
-G                     turn off grouping of associated objects
-d                     return DNS reverse delegation objects too
-i ATTR[,ATTR]...      do an inverse look-up for specified ATTRibutes
-T TYPE[,TYPE]...      only look for objects of TYPE
```

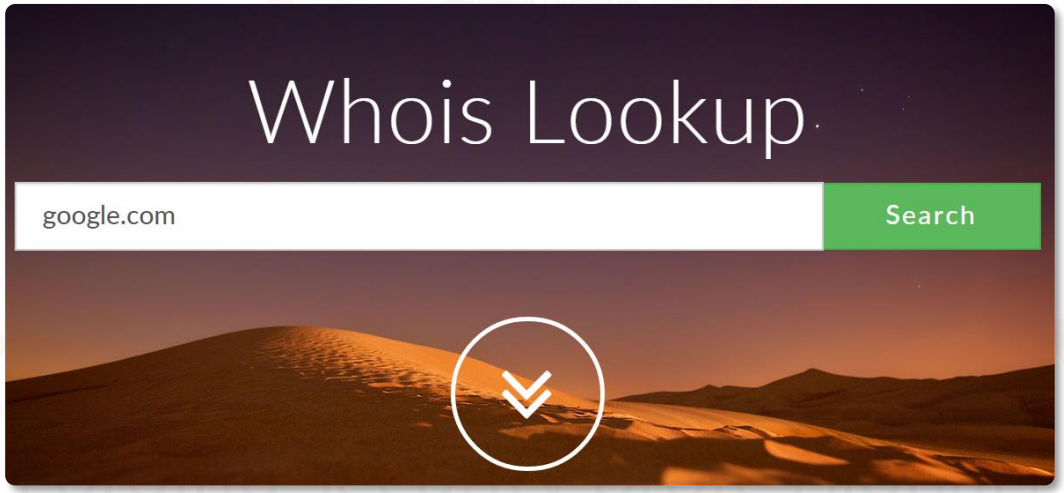
Whois Example - Web Based tools

Instead of using the command line tools, you can also use web-based tools such as: whois.domaintools.com



http://whois.domaintools.com/

The *Whois* database contains public information, so you can freely check it. In this example we will perform a lookup of `google.com`. This is what we get:



We get administrative contact information...

```
Registry Admin ID:
Admin Name: DNS Admin
Admin Organization: Google Inc.
Admin Street: 1600 Amphitheatre Parkway
Admin City: Mountain View
Admin State/Province: CA
Admin Postal Code: 94043
Admin Country: US
Admin Phone: +1.6506234000
Admin Phone Ext:
Admin Fax: +1.6506188571
Admin Fax Ext:
Admin Email: dns-admin@google.com
```

... Technical contact information...

```
Registry Tech ID:
Tech Name: DNS Admin
Tech Organization: Google Inc.
Tech Street: 2400 E. Bayshore Pkwy
Tech City: Mountain View
Tech State/Province: CA
Tech Postal Code: 94043
Tech Country: US
Tech Phone: +1.6503300100
Tech Phone Ext:
Tech Fax: +1.6506181499
Tech Fax Ext:
Tech Email: dns-admin@google.com
```

More importantly, we get the IP address of one of the machines of the organization we are studying

Registrar Status	clientDeleteProhibited, clientTransferProhibited, clientUpdateProhibited, serverDeleteProhibited, serverTransferProhibited, serverUpdateProhibited
Dates	Created on 1997-09-15 - Expires on 2020-09-14 - Updated on 2011-07-20
Name Server(s)	NS1.GOOGLE.COM (has 11,892 domains) NS2.GOOGLE.COM (has 11,892 domains) NS3.GOOGLE.COM (has 11,892 domains) NS4.GOOGLE.COM (has 11,892 domains)
IP Address	74.125.129.99 - 78 other sites hosted on this server
IP Location	- California - Mountain View - Google Inc.
ASN	AS15169 GOOGLE - Google Inc. (registered Mar 30, 2000)
Whois History	4,487 records have been archived since 2001-05-03

Now that we have some valuable information about our target, we can start digging further into the data to identify individual targets.

A valuable source for such information is the Domain Name System (DNS). We can query it for some of the IP addresses that we received from the WHOIS database.

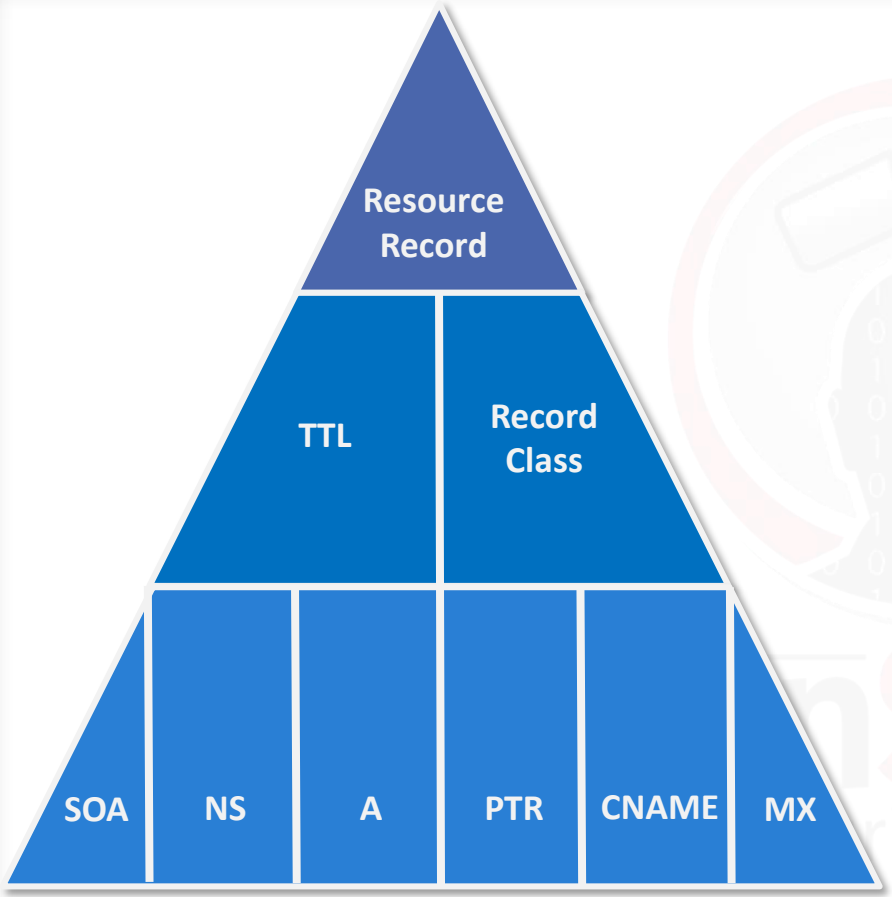
The DNS structure contains a hierarchy of names. The root, or highest level of the system is unnamed.

Top Level Domains (TLDs) are divided into classes based on rules that have evolved over time. Most TLDs have been delegated to individual country managers, whose codes are assigned from a table known as ISO-3166-1. These are maintained by an agency of the United Nations and are called country-code Top Level Domains, or ccTLDs. (Ex: .us, .uk, .il, .de, .fi, .fr)

In addition, there are a limited number of "generic" Top Level Domains (gTLDs) which do not have a geographic or country designation. (Ex. .com, .org, .net, .gov, .edu)

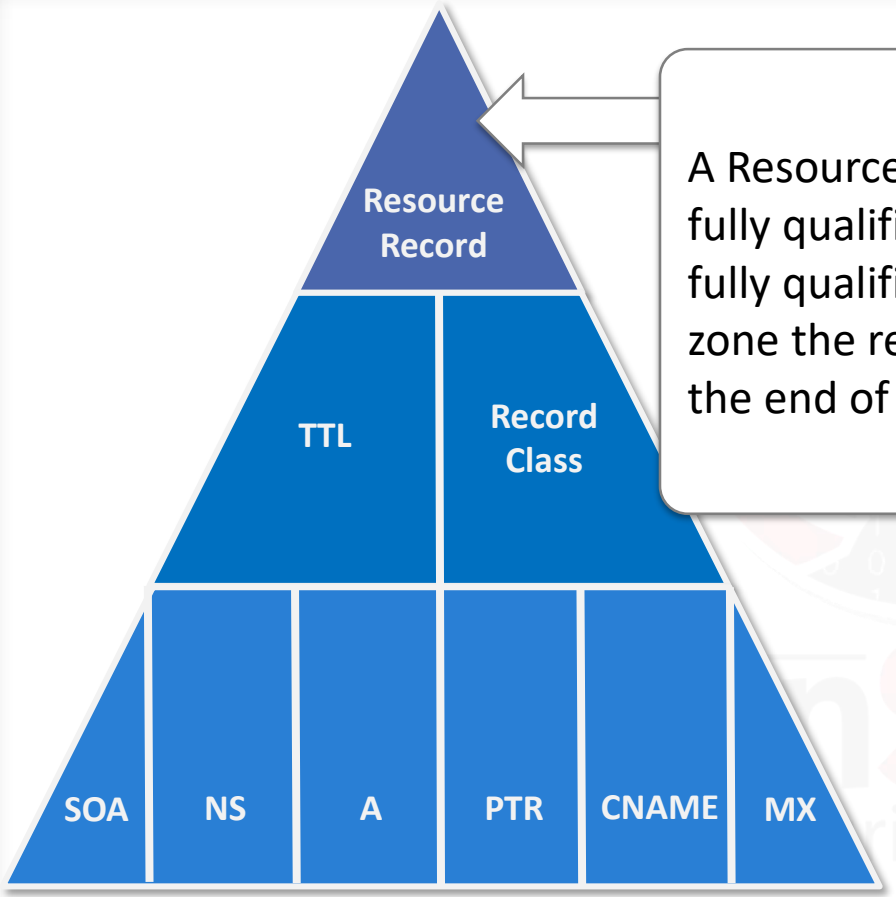
Responsibility for procedures and policies for the assignment of Second Level Domain Names (SLDs) and lower level hierarchies of names has been delegated to TLD managers - subject to the policy guidance contained in ISO-3166-1.

Country code domains are organized by a manager for that country; these managers perform a public service on behalf of the Internet community.



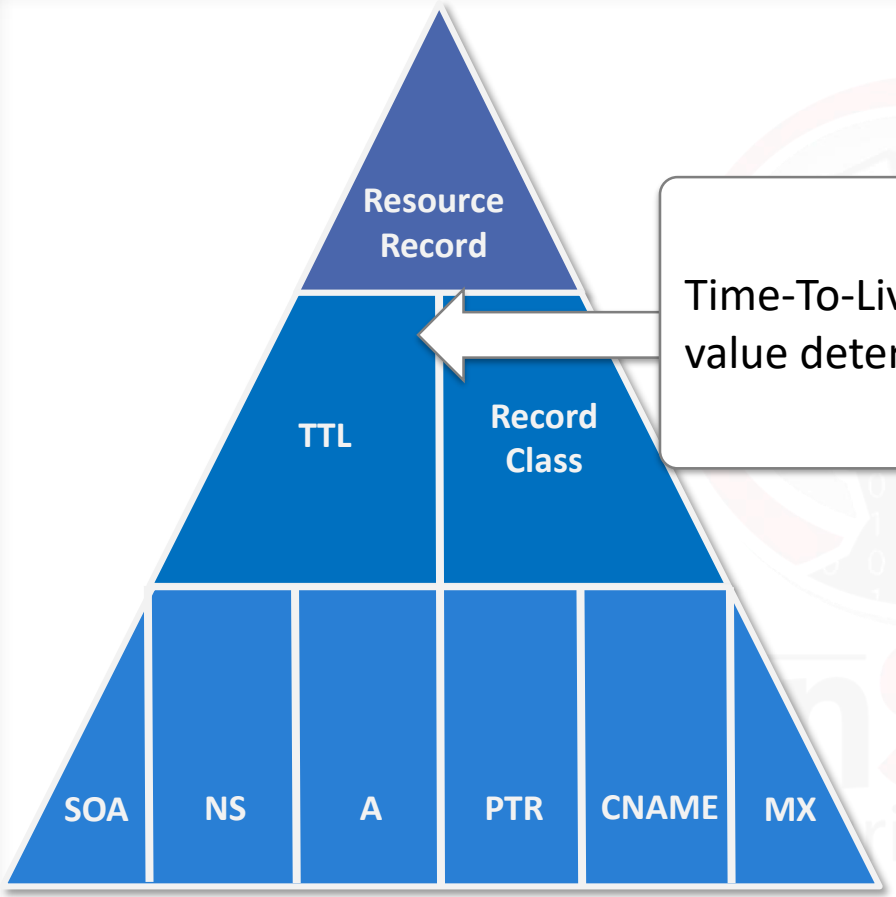
DNS queries produce listing called Resource Records.

This is a representation of a Resource Record.



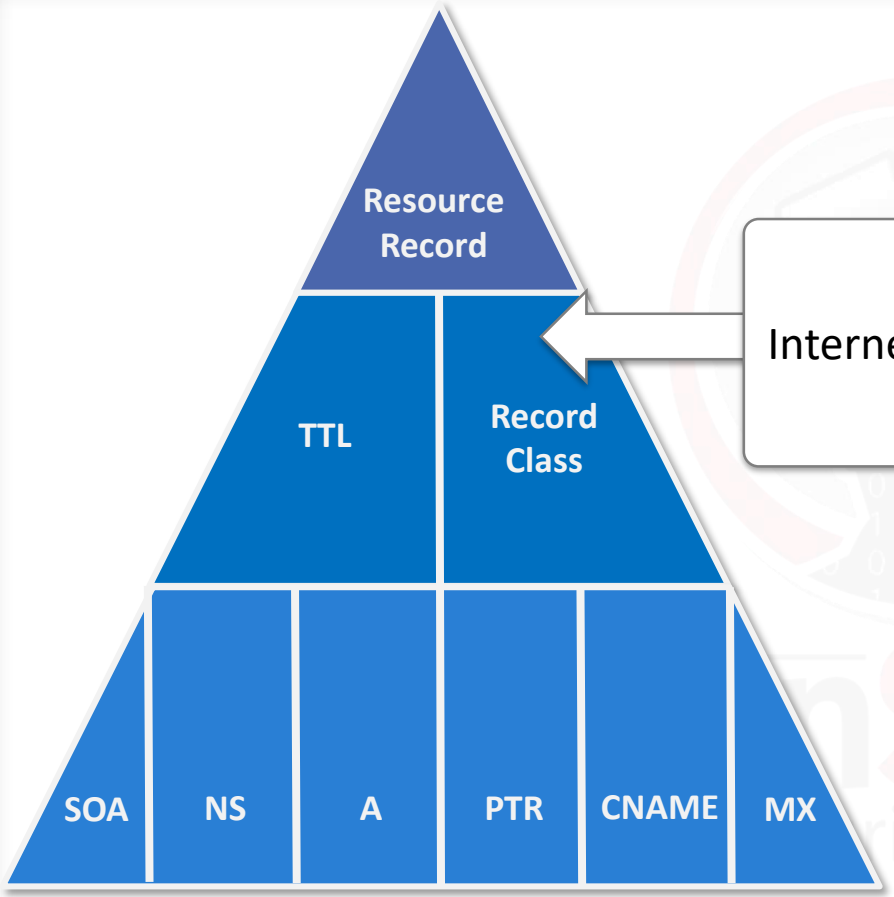
A Resource record starts with a domain name, usually a fully qualified domain name. If anything other than a fully qualified domain name is used, the name of the zone the record is in, will automatically be appended to the end of the name.

2.1.1.2. DNS

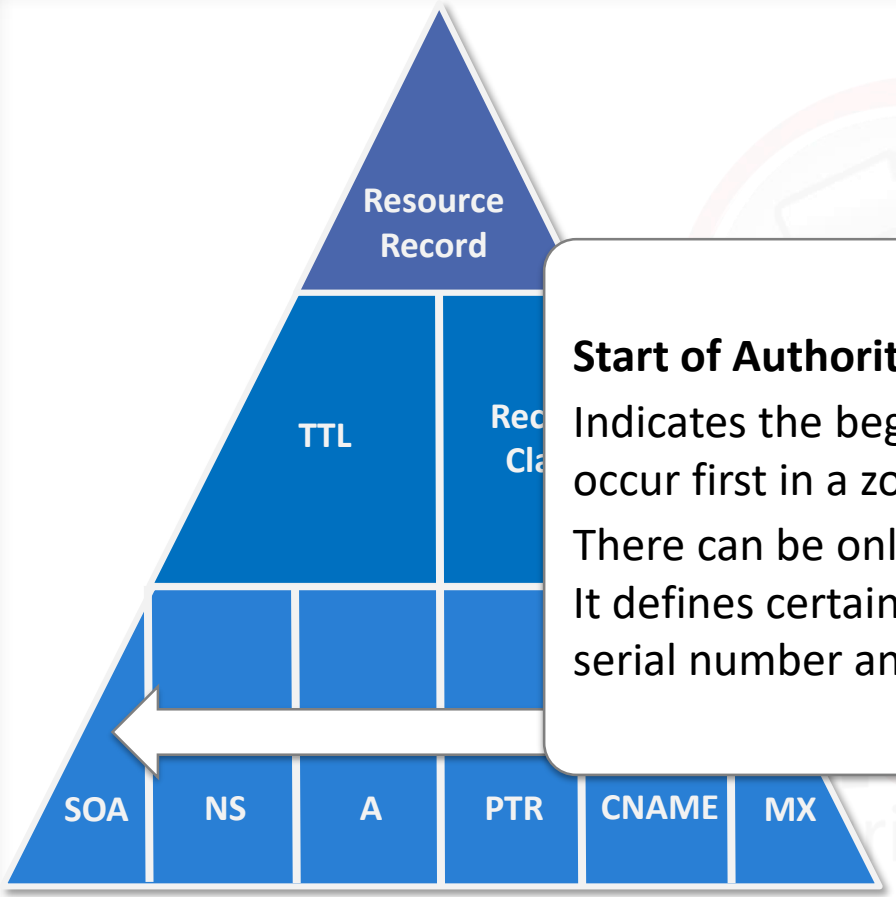


Time-To-Live (TTL), in seconds, defaults to the minimum value determined in the SOA record.

2.1.1.2. DNS



Internet, Hesiod, or Chaos

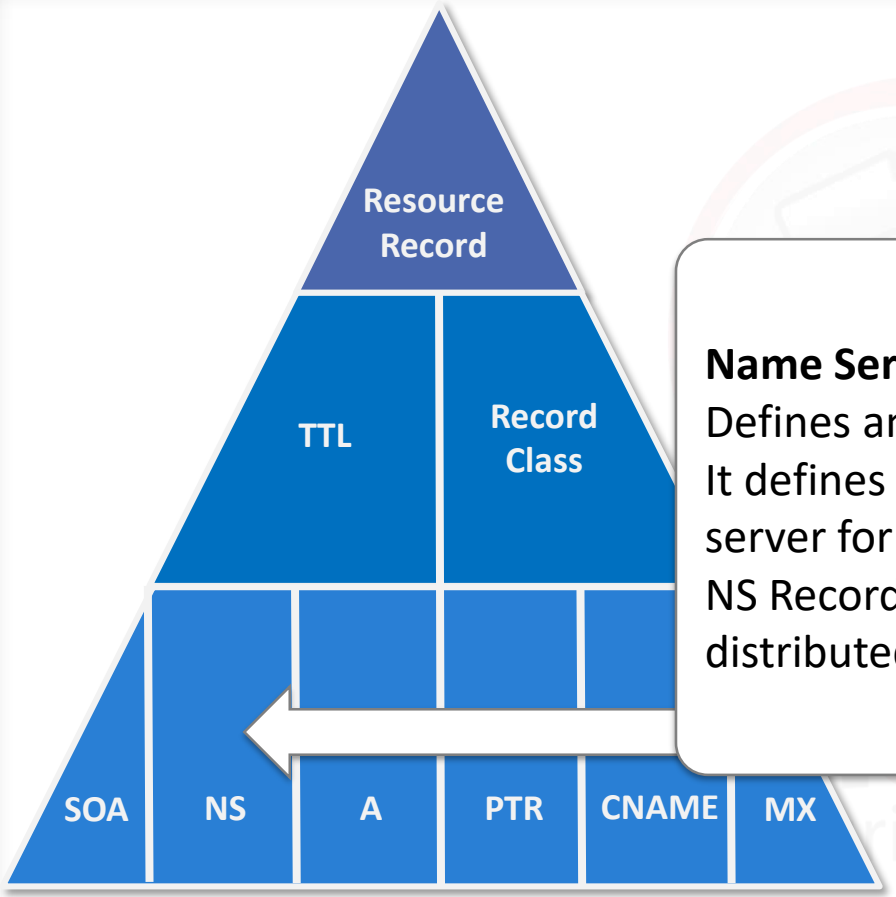


Start of Authority

Indicates the beginning of a zone and it should occur first in a zone file.

There can be only one SOA record per zone.

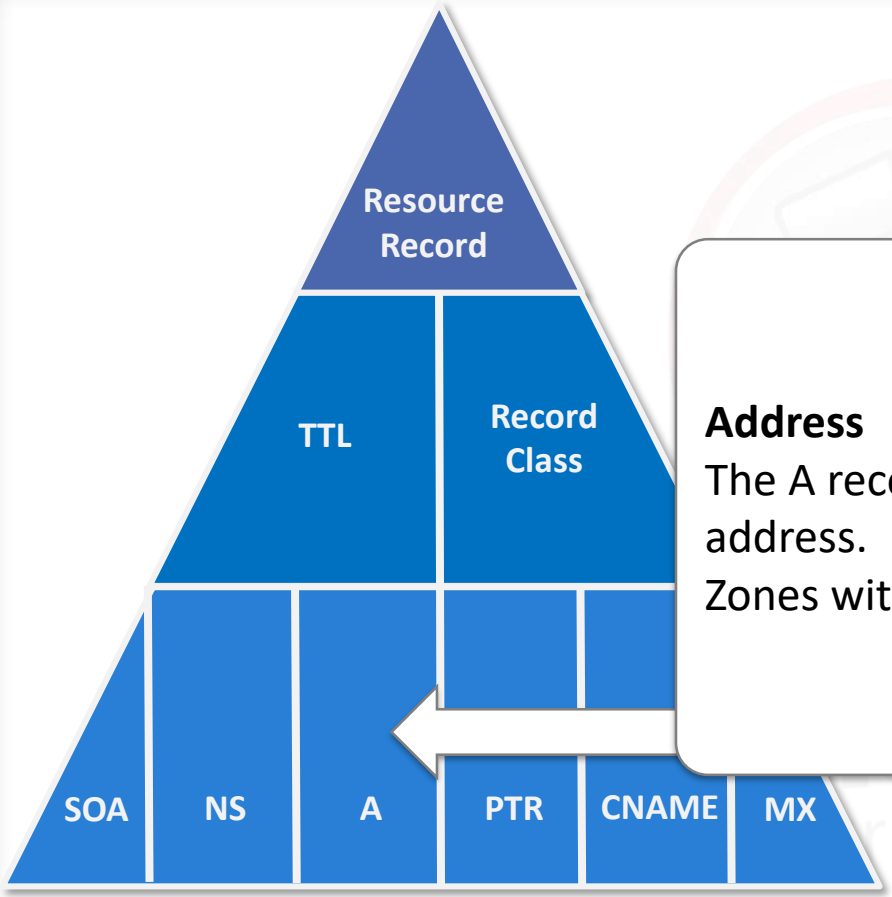
It defines certain values for the zone such as a serial number and various expiration timeouts



Name Server

Defines an authoritative name server for a zone. It defines and delegates authority to a name server for a child zone.

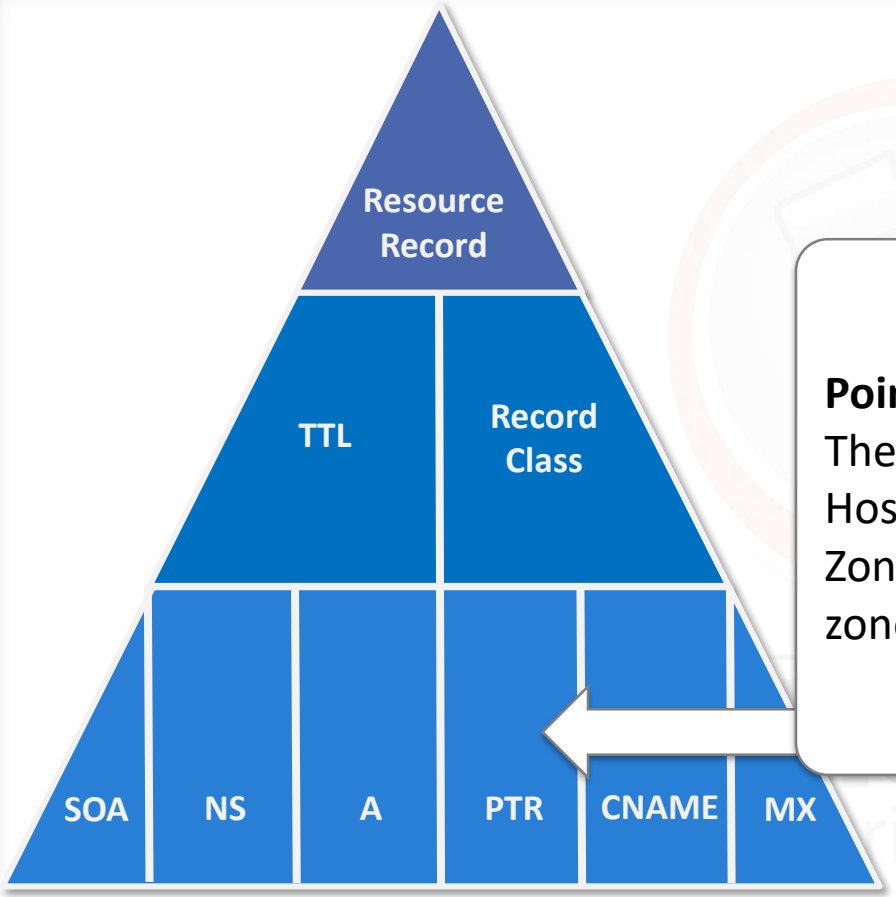
NS Records are the GLUE that binds the distributed database together.



Address

The A record simply maps a hostname to an IP address.

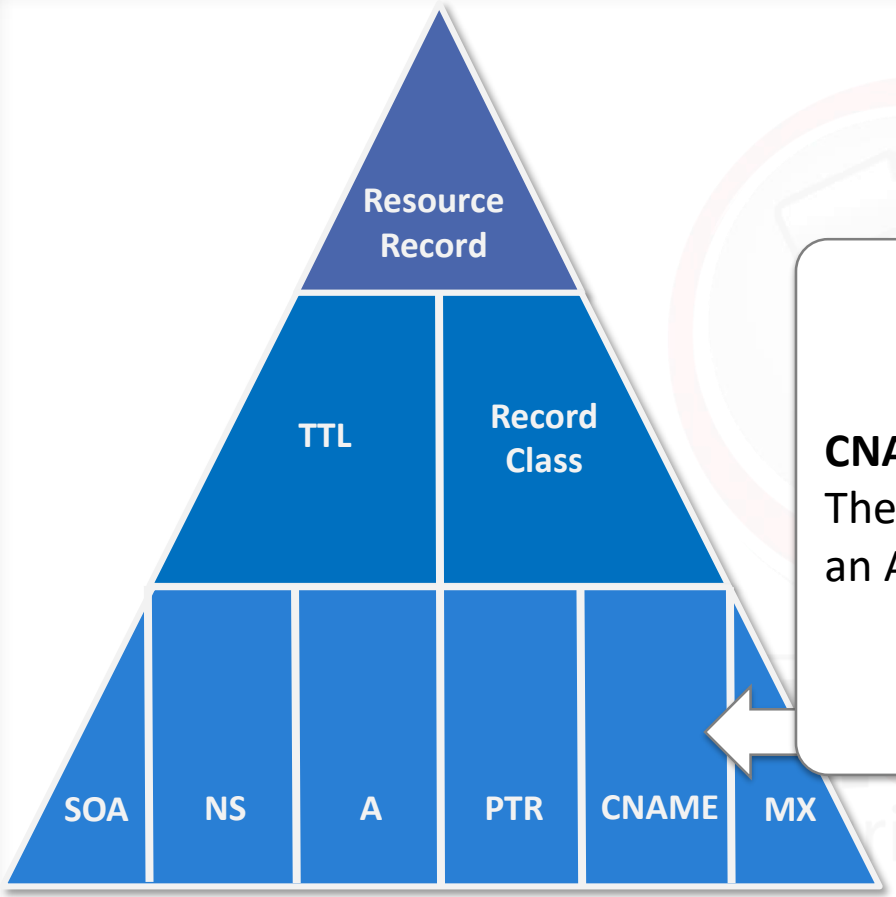
Zones with A records are called 'forward' zones.



Pointer

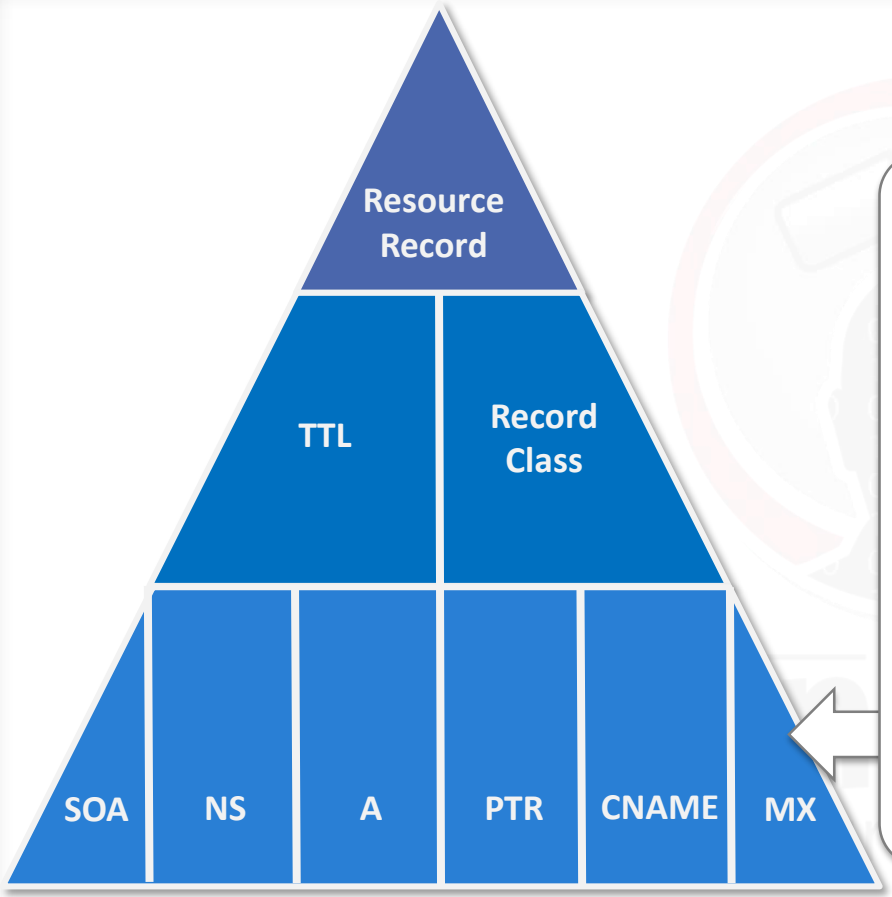
The PTR record maps an IP address to a Hostname.

Zones with PTR records are called 'reverse' zones.



CNAME

The CNAME record maps an alias hostname to an A record hostname.



Mail Exchange

The MX record specifies a host that will accept email on behalf of a given host.

The specified host has an associated priority value.

A single host may have multiple MX records.

The records for a specific host make up a prioritized list.

The domain name system (DNS) is a distributed database arranged hierarchically. Its purpose is to provide a layer of abstraction between Internet services (web, email, etc.) and the numeric addresses (IP addresses) used to uniquely identify any given machine on the Internet.

https://icannwiki.com/Domain_Name_System

This has several advantages:

- It permits the use of names instead of numbers to identify hosts (usually servers).
- Names are much easier to remember.
- It permits a server to change numeric addresses without requiring notification of everyone on the Internet, by simply pointing the name to the new numeric address.
- One name can refer to multiple hosts, to share the load

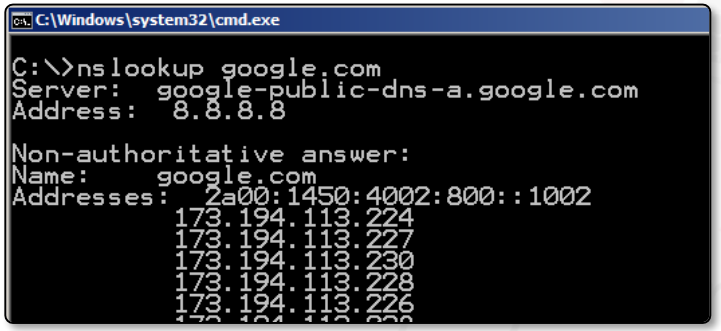
Nslookup is another very handy tool that lets you translate hostnames to IP addresses and vice versa.

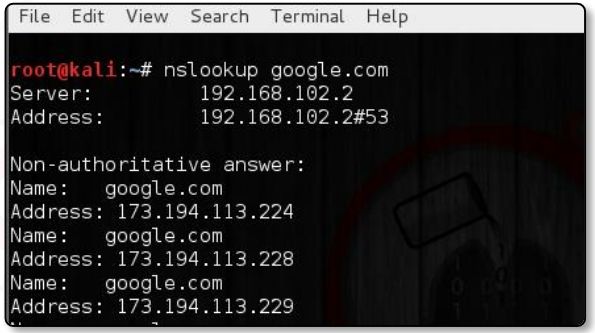
Let us quickly review its main features in the next interactive slide.

Nslookup

- Under *Windows*, click `Start>Run>cmd`.
- Under **nix* systems, open a console and type:

```
nslookup google.com
```





Reverse Lookup

The previous query is referred to as a lookup or as referenced in the previous section, it is an "A".

If you provide a domain name, DNS returns the IP addresses for the matching hosts.

Reverse Lookup

Let us try with a **reverse lookup**.
If you provide an IP address, DNS returns the domain name associated with that IP.

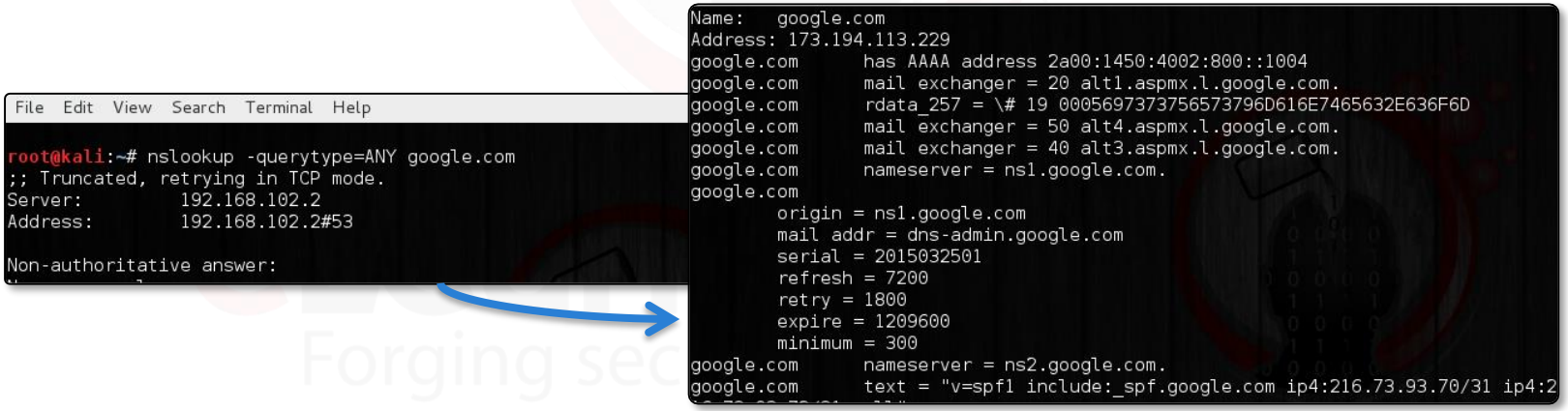
```
File  Edit  View  Search  Terminal  Help
root@kali:~# nslookup -type=PTR 173.194.113.224
Server:         192.168.102.2
Address:        192.168.102.2#53

Non-authoritative answer:
224.113.194.173.in-addr.arpa    name = mil01s18-in-f0.1e100.net.

Authoritative answers can be found from:

root@kali:~#
```

172.194.113.224 is the IP address we have found in the previous step

Records

In this step we will query the DNS server for the whole record associated with *google.com*:

```
nslookup -querytype=ANY google.com
```



Review results

From the previous command we are given the following important information:

- Name Servers
- A records
- CNAME's
- MX

You will have to save this information for the subsequent steps of your penetration testing engagement.

2.1.1.3. Nslookup

Every IP address on the Internet is assigned to an organization.

An organization can purchase a block of IP addresses according to their needs and it will "own" that entire block.

The `whois` database tracks the owners of public IP addresses as well as domain names.

Sometimes, organizations are not actually the owners of the IP addresses they use for their presence on the internet.

They may rely on ISPs and hosting companies that lease one or more smaller netblocks (among those owned) to them.

Finding the netblock owner and the ISPs that our target organization relies on, is an important step that we will study in the next slide.

This time we want to know which ISP's hosting and IP addresses our target organization uses.

Using Nslookup we get the IP addresses associated to each subdomain. We will perform a whois request for each of these IP addresses to uncover the ISP's that these IP addresses belong to.

Note: When the organization is big, net-blocks may be assigned directly to it, so no Hosting services are involved.

Note: A corporation is not limited to having only one hosting company.

Requirements:

- nslookup (Windows/Linux)
- web browser

2.1.1.3.1. Finding Target ISP's

The first step is to gather all the IP addresses related to a domain or subdomain. For this example we will research the *statcounter.com* website.

Now is your chance to use FreeMind to map your research!

Let's start from the domain "*statcounter.com*":

```
nslookup statcounter.com
```

2.1.1.3.1. Finding Target ISP's


```
root@kali:~# nslookup statcounter.com
Server:         192.168.102.2
Address:        192.168.102.2#53

Non-authoritative answer:
Name:   statcounter.com
Address: 104.20.2.47
Name:   statcounter.com
Address: 104.20.3.47
```

As you can see there are two ip addresses ("A" records in the DNS) registered for that domain. We will store these two IP addresses for steps 3 and 4 of this process.

2.1.1.3.1. Finding Target ISP's

Continuing to perform a per-subdomain ip survey, we move on to "*www.statcounter.com*" and find out that it has different IP addresses associated with it.

```
nslookup www.statcounter.com
```

This command returns other two IP addresses:

- 93.188.134.172
- 93.188.134.237

Please note that these addresses may change in your tests

2.1.1.3.1. Finding Target ISP's

The returned IP addresses must be saved for further checks against *whois* in step 5.

We can continue this survey against all of the organization's domains and subdomains, but we will stop here and start our ISP recognition phase. We have to check:

1. 104.20.2.47
2. 104.20.3.47
3. 93.188.134.172
4. 93.188.134.237

2.1.1.3.1. Finding Target ISP's

Using online tools such as arin.net, whois.domaintools.com or ripe.net we will uncover the ISP's that our organization relies upon.

Let us start querying **104.20.2.47**.

http://whois.arin.net/rest/net/NET-108-162-192-0-1/pft
http://whois.domaintools.com/
https://apps.db.ripe.net/db-web-ui/#/query

2.1.1.3.1. Finding Target ISP's

This IP address belongs to "CloudFlare".

A netblock for this ISP is 104.16.0.0/12.

IP Location	Singapore Singapore Cloudflare Inc.
ASN	AS13335 CLOUDFLARENET - CloudFlare, Inc. (registered J
Whois Server	whois.arin.net
IP Address	104.20.2.47

```
NetRange:       104.16.0.0 - 104.31.255.255
CIDR:           104.16.0.0/12
NetName:        CLOUDFLARENET
NetHandle:      NET-104-16-0-0-1
Parent:         NET104 (NET-104-0-0-0-0)
NetType:        Direct Assignment
OriginAS:       AS13335
Organization:   CloudFlare, Inc. (CLOUD14)
RegDate:        2014-03-28
Updated:        2014-03-28
Comment:        https://www.cloudflare.com
Ref:            http://whois.arin.net/rest/net/NET-104-16-0-0-1

OrgName:        CloudFlare, Inc.
OrgId:          CLOUD14
Address:        665 Third Street #207
City:           San Francisco
```

We can move on with the other two IP address: **93.188.134.172** and **93.188.134.23**. We have now uncovered that the `www.statcounter.com` subdomain is handled by another organization: *CDNetworks*.

IP Address 93.188.134.149

```
% Abuse contact for '93.188.134.0 - 93.188.134.255'

inetnum:        93.188.134.0 - 93.188.134.255
netname:        CDNETEU1
descr:          CDNetworks Inc.
country:        IT
admin-c:        CDN57-RIPE
tech-c:         CDN57-RIPE
```

IP Address 93.188.134.23

```
% Abuse contact for '93.188.134.0 - 93.188.134.255' is

inetnum:        93.188.134.0 - 93.188.134.255
netname:        CDNETEU1
descr:          CDNetworks Inc.
country:        IT
admin-c:        CDN57-RIPE
tech-c:         CDN57-RIPE
status:         ASSIGNED PA
```

2.1.1.3.1. Finding Target ISP's

At the end of this process, we can build a table with all the IP addresses used by the organization and the ISP/Hosting services that these IP addresses belong to.

To perform a thorough pen test, this information must be included in your penetration testing documentation.

This information will become useful when mapping the attack surface.

2.1.1.3.2. Finding Target ISP's with Netcraft

A faster way to uncover the organization's hosting scheme and ownership is by using **Netcraft**.

Netcraft has a wealth of information for us and we will use it often in this module.

Visiting www.netcraft.com and doing a search for *statcounter.com* will reveal the Hosting provider for *statcounter.com* as well as its IP netblock.

2.1.1.3.2. Finding Target ISP's with Netcraft

The *statcounter.com* example is a good example for demonstrating how an organization may rely upon different netblocks (and hosting) for different servers.

By just querying a domain we get all the information in 1 page.

Let us try it by visiting [netcraft.com](https://www.netcraft.com/)

https://www.netcraft.com/

2.1.1.3.2. Finding Target ISP's with Netcraft

Netcraft

Network

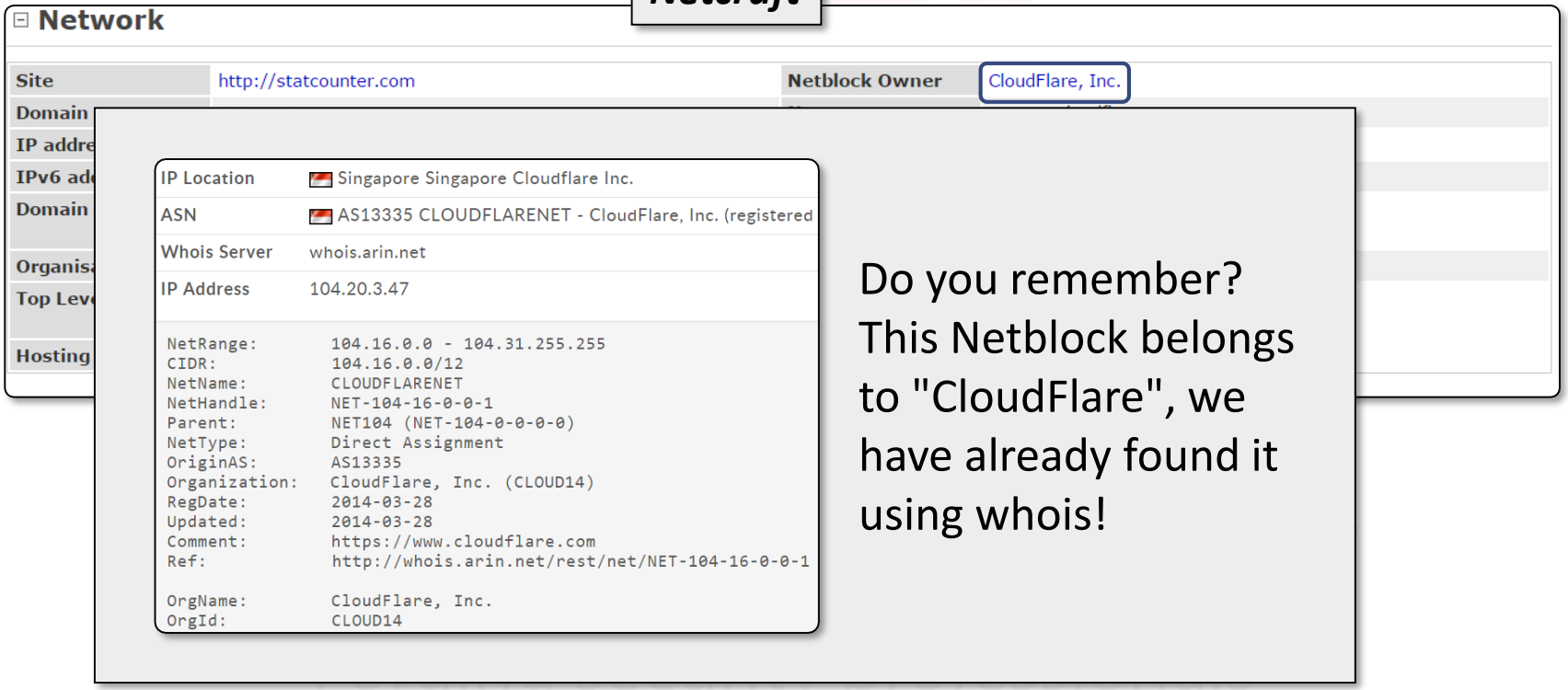
Site	http://statcounter.com	Netblock Owner	CloudFlare, Inc.
Domain	statcounter.com	Nameserver	may.ns.cloudflare.com
IP address	104.20.3.47	DNS admin	dns@cloudflare.com
IPv6 address	*Not Present*	Reverse DNS	*unknown*
Domain registrar	*unknown*		
Organisation	*unknown*		
Top Level Domain	Commerc		
Hosting country	US		

```
root@kali:~# nslookup statcounter.com
Server:         192.168.102.2
Address:        192.168.102.2#53

Non-authoritative answer:
Name:   statcounter.com
Address: 104.20.2.47
Name:   statcounter.com
Address: 104.20.3.47
```

This is the IP address that we found using nslookup.

2.1.1.3.2. Finding Target ISP's with Netcraft

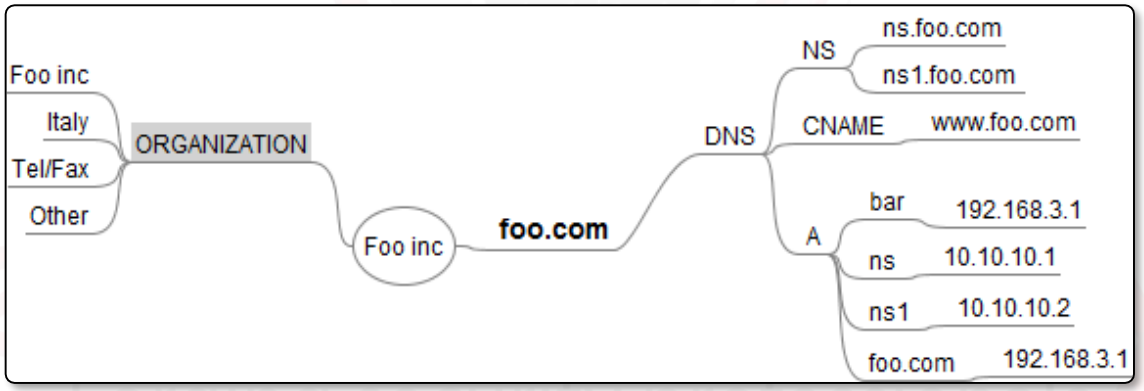
Netcraft



Do you remember? This Netblock belongs to "CloudFlare", we have already found it using whois!

2.1.1.3.2. Finding Target ISP's with Netcraft

If you have read and applied our methodological approach, this map should be similar to the one you created up to this point:



If you have a FULL or ELITE plan you can click on the image on the left to start the video

INFRASTRUCTURE

The infrastructure behind a web application is what supports it and allows it to function.

This includes the web server that is directly involved in the execution of any web application.

The two most common web servers used on the internet today are Apache and Microsoft IIS.

2.2. Infrastructure

Discovering what kind of web server is behind your application will give you a hint about what OS the server is running. This helps you to research what known vulnerabilities may exist.

For example, discovering an IIS (Internet Information Service) web server will tip us off that the server is running an OS in the Windows Server OS family.

IIS version 6.0 is installed by default on all Windows Server 2003 boxes, Windows Server 2008 supports IIS 7 and Windows server 2012 is the only one to support IIS 8.0.

https://www.iis.net/

These guesses are correct in 90% of the cases when dealing with IIS and Windows Server however, the same cannot be said for the many different Linux and BSD distributions. These may run different versions of the Apache web server.

Although hacking into the server operating system is beyond the scope of our web application test engagement, having a clear understanding of the infrastructure will be useful in the next testing steps.

2.2.1. Fingerprinting The Webserver

Uncovering both the web server type and version will give us enough information to mount many different attacks against its components (during later stages of the test).

IIS components, usually called ISAPI extensions, work as dynamic libraries, extending the functionalities of the web server and performing different tasks for the web server.

These include: URL rewriting, load balancing, script engines (like PHP, Python or Perl) and many others.

A rewriter changes "ugly" web application URLs such as news.php?id=12 to a more search-engine-friendly URL like news/12.html or a route like news/12.

Also an IDS is a web application firewall that detects and prevents intrusions coming from the HTTP/S protocol.

2.2.1. Fingerprinting The Webserver

The presence of either of these two modules can alter the results of our tests significantly, so performing a careful fingerprint of the web server and its components is obviously a great help.

Let's have a look at the first and simplest way to retrieve the web server version along with other useful information. Sometimes, this information is leaked through the HTTP headers in response to a trivial HTTP request to the web server.

2.2.1. Fingerprinting The Webserver

Request

```
GET / HTTP/1.1
Host: test.xxx
```

Response

```
HTTP/1.x 200 OK
Date: Sat, 18 Apr 2009
13:08:40 GMT
Server: Apache
Content-Length: 88750
```

As we can see from this response, the web server has quietly provided its name to us!

In this case, unfortunately, we have no versioning information. That info is extremely important to us in order to understand the level of exposure to known vulnerabilities.

2.2.1. Fingerprinting The Webserver

For the version info, we can use Netcraft; it provides web server analysis via its enormous information database.

It can be reached at: www.netcraft.com.

2.2.1. Fingerprinting The Webserver

You will find Netcraft to be very useful not only for this web server fingerprinting step but also, for subsequent steps like collecting all available subdomains for a domain.

Let's try our web server identification using Netcraft.

2.2.1. Fingerprinting The Webserver

By searching for the domain name from the Netcraft home page, we are presented with a great deal of information regarding our target.

This includes the web server version, name server and IP addresses of the different web servers in use.

2.2.1. Fingerprinting The Webserver

The Netcraft site report also shows the webserver and historical OS information about the domain microsoft.com.

Background

Site title	Microsoft – Official Home Page	**Date first seen**	August 1995
Site rank	1082	**Primary language**	English
Description	At Microsoft our mission and values are to help people and businesses throughout the world realise their full potential.		
Keywords	*Not Present*		

Network

Hosting History

Netblock owner	IP address	OS	Web server	Last seen Refresh
Microsoft Corp One Microsoft Way Redmond WA US 98052	134.170.188.221	unknown	Microsoft-IIS/8.5	29-Mar-2015
Microsoft Corp One Microsoft Way Redmond WA US 98052	134.170.185.46	unknown	Microsoft-IIS/8.5	25-Mar-2015
Microsoft Corp One Microsoft Way Redmond WA US 98052	134.170.188.221	unknown	Microsoft-IIS/8.5	24-Mar-2015
Microsoft Corp One Microsoft Way Redmond WA US 98052	134.170.185.46	unknown	Microsoft-IIS/8.5	22-Mar-2015
Microsoft Corp One Microsoft Way Redmond WA US 98052	134.170.188.221	unknown	Microsoft-IIS/8.5	21-Mar-2015
Microsoft Corp One Microsoft Way Redmond WA US 98052	134.170.185.46	unknown	Microsoft-IIS/8.5	15-Mar-2015
Microsoft Corp One Microsoft Way Redmond WA US 98052	134.170.188.221	unknown	Microsoft-IIS/8.5	14-Mar-2015

We searched for domain info on microsoft.com and Netcraft responded with a Web Server version Microsoft-IIS/8.5.

We are also given a list of the web servers and IP addresses. Microsoft uses a server farm and the HTTP request may be routed to different web servers based on load and availability at the moment we visit.

This is not to confuse us, but it must be taken into account when we perform our web application tests.

It is not uncommon to find corporations or even small businesses using load balancers that route HTTP request to different servers that may even run different web servers versions.

The advice here is to take note of all web server version-to-IP couplets for further use.

2.2.1. Fingerprinting The Webserver

The Nameserver is the DNS server that replies to all lookup queries regarding the namespace of a domain. An nslookup query for microsoft.com involves a request to ns1.msft.net, for example.

```
C:\>nslookup -type=NS microsoft.com
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
microsoft.com   nameserver = ns3.msft.net
microsoft.com   nameserver = ns4.msft.net
microsoft.com   nameserver = ns1.msft.net
microsoft.com   nameserver = ns2.msft.net
```

2.2.1. Fingerprinting The Webserver

In addition to web server version, IP addresses and Nameservers, Netcraft provides the following information we can capture:

- Server version
- Uptime stats
- IP address owner
- Host provider

2.2.1. Fingerprinting The Webserver

Sometimes Netcraft does not provide us with enough information regarding our target web server version.

In addition, there are cases where Netcraft cannot be used, such as with Internal Web Servers that are not attached to Internet.

When this is the case we can use manual testing techniques and tools to identify a server such as: `netcat`, `httprint`, `whatweb`, `wappalyzer`.

https://github.com/urbanadventurer/WhatWeb
https://wappalyzer.com/

2.2.1. Fingerprinting The Webserver

Some of these freely available tools rely on common web server characteristics in order to accurately identify them.

They probe the web server with a series of requests and compare the responses to their database of signatures in order to both find a match, and accurately guess the following information:

- Web server version
- Installed modules
- Web enabled devices (routers, cable modems, etc.)

2.2.1. Fingerprinting The Webserver

The most important feature of these tools is that they do not solely rely on the service banner.

They are capable of fingerprinting the web server version even when the banner or the HTTP response header has been manually obfuscated / altered using security modules (mod_security…).

https://www.modsecurity.org/

Let's start inspecting these tools.

The first one we want to use for manual fingerprinting is **Netcat**. This is a simple utility that reads and writes data across network connections.

By using *Netcat* we can establish a connection to the Web Server and look at the Server field in the HTTP response header.

The following is an example of what we can get by using nc (Netcat) against a Web Server that resides in our network:

```
root@kali:~# nc 192.168.102.136 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 30 Mar 2015 14:40:06 GMT
Server: Apache/2.2.22 (Debian)
Last-Modified: Thu, 05 Feb 2015 21:12:05 GMT
ETag: "1847cb-b1-50e5dc184b340"
Accept-Ranges: bytes
Content-Length: 177
Vary: Accept-Encoding
Connection: close
Content-Type: text/html
```

Remember that once we establish the connection with netcat, we have to send `HEAD / HTTP/1.0` and hit enter two times

From the server field we can see that we are running Apache version 2.2.22 on Linux OS

Here is another Web Server address. As you can see, the field order changes as well as their values:

```
root@kali:~# nc 134.170.185.46 80
HEAD / HTTP/1.0

HTTP/1.1 301 Moved Permanently
Cache-Control: private
Content-Length: 23
Content-Type: text/html
Location: http://www.microsoft.com
Server: Microsoft-IIS/8.5
Set-Cookie:
ASPSESSIONIDACRQQCDQ=LKKMCDHAFINIAMHBICPIMLJF; path=/
…
```

The following output shows us that the remote Web Server is using IIS version 8.5

Beyond the Server header we should also look at the X-Powered-By header, which may reveal the technology behind the Web Application.

```
root@kali:~# nc 134.170.188.221 80
HEAD / HTTP/1.0

HTTP/1.1 301 Moved Permanently
..stripped output…
Server: Microsoft-IIS/8.5
X-Powered-By: ASP.NET
X-UA-Compatible: IE=EmulateIE7
Date: Tue, 31 Mar 2015 07:48:01 GMT
Connection: close
```

In this case, the header tells us the Web App is using ASP.NET. Other possible values are PHP, JSP, JBoss and so on.

Cookies are also an interesting resource that may reveal useful information in this phase. Each technology has its default cookies names. Therefore, we can potentially guess the web server by inspecting the cookie header. Here is a short list of what you may encounter:

Server	Cookie
PHP	PHPSESSID=XXXXX
.NET	ASPSSESSIONIDYYYY=XXXXX
JAVA	JSESSION=XXXXX

As you can imagine, there may be many different result outputs depending on the service running on the machine, the version, Operating System and so on.

Here and here you can find a few more examples and information about different Web server outputs.

https://www.owasp.org/index.php/Testing_for_Web_Application_Fingerprint_(OWASP-IG-004)
https://www.owasp.org/index.php/Fingerprint_Web_Application_Framework_(OTG-INFO-008)

Another very useful tool is WhatWeb. It is a command line tool that can be used to recognize website technologies, Web server versions, blogging platforms, JavaScript libraries and much more.

Pentesting distributions such as Kali Linux have it installed by default, so you can start using it by running the following command:

```
root@kali:~# whatweb -h
```

https://github.com/urbanadventurer/WhatWeb

If you are using a clean environment, you can use git clone to download it on your machine. Here is how to clone the git repository to the tools folder:

```
root@kali:~/tools# git clone https://github.com/urbanadventurer/WhatWeb.git
Cloning into 'WhatWeb'...
remote: Counting objects: 17838, done.
remote: Total 17838 (delta 0), reused 0 (delta 0), pack-reused 17838
Receiving objects: 100% (17838/17838), 6.65 MiB | 224 KiB/s, done.
Resolving deltas: 100% (9360/9360), done.
root@kali:~/tools# cd WhatWeb/
root@kali:~/tools/WhatWeb# ./whatweb -h
```

This tool is very easy to use. You only need to type the name of the tool followed by the address (IP or URL) of the target and hit enter. Note that you can specify multiple targets in the command or even IP ranges.

Moreover, it offers options that allow us to specify different user agents, HTTP basic authentication credentials, cookies, proxy and much more.

Let us try to run the tool against www.elearnsecurity.com.

2.2.1.2. WhatWeb


```
root@kali:~/tools/WhatWeb# ./whatweb www.elearnsecurity.com
http://www.elearnsecurity.com [302] HTTPServer[Microsoft-IIS/7.5], IP[199.193.116.231],
 Microsoft-IIS[7.5], RedirectLocation[https://www.elearnsecurity.com/], Title[Document
Moved], UncommonHeaders[x-xss-protection,x-frame-options,strict-transport-security], X-
Frame-Options[sameorigin], X-Powered-By[MOS 6502], X-XSS-Protection[1; mode=block]
https://www.elearnsecurity.com/ [200] HTML5, HTTPServer[Microsoft-IIS/7.5], IP[199.193.
116.231], JQuery, Microsoft-IIS[7.5], PoweredBy[eLearnSecurity,], Script[text/javascrip
t], Title[eLearnSecurity - IT Security training courses for individuals and corporation
s], UncommonHeaders[x-xss-protection,x-frame-options,strict-transport-security], X-Fram
e-Options[sameorigin], X-Powered-By[MOS 6502], X-UA-Compatible[IE=edge], X-XSS-Protecti
on[1; mode=block]
root@kali:~/tools/WhatWeb#
```

As you can see we have a great deal of information in the output. Moreover, you should notice that the tool automatically follows redirections (302): in the output we have the results for both HTTP and HTTPS websites.

2.2.1.2. WhatWeb

If you are using a newer version of Kali you may be able to run whatweb directly from the terminal window by typing: whatweb <website>

```
root@kali:~# whatweb www.elearnsecurity.com
/usr/share/whatweb/lib/tld.rb:85: warning: key "2nd_level_registration" is duplicated and overwritten on line 85
/usr/share/whatweb/lib/tld.rb:93: warning: key "2nd_level_registration" is duplicated and overwritten on line 93
/usr/share/whatweb/lib/tld.rb:95: warning: key "2nd_level_registration" is duplicated and overwritten on line 95
/usr/share/whatweb/plugins/wordpress.rb:436: warning: key "2.7-beta1" is duplicated and overwritten on line 453
/usr/share/whatweb/lib/extend-http.rb:102:in `connect': Object#timeout is deprecated, use Timeout.timeout instead.
http://www.elearnsecurity.com [302] Country[UNITED STATES][US], HTTPServer[Microsoft-IIS/7.5], IP[199.193.116.231]
, Microsoft-IIS[7.5], RedirectLocation[https://www.elearnsecurity.com/], Title[Document Moved], UncommonHeaders[st
rict-transport-security], X-Frame-Options[sameorigin], X-Powered-By[MOS 6502], X-XSS-Protection[1; mode=block]
/usr/share/whatweb/lib/extend-http.rb:102:in `connect': Object#timeout is deprecated, use Timeout.timeout instead.
/usr/share/whatweb/lib/extend-http.rb:140:in `connect': Object#timeout is deprecated, use Timeout.timeout instead.
https://www.elearnsecurity.com/ [200] Country[UNITED STATES][US], HTML5, HTTPServer[Microsoft-IIS/7.5], IP[199.193
.116.231], JQuery[1.11.0], Microsoft-IIS[7.5], PoweredBy[eLearnSecurity,], Script[text/javascript], Title[eLearnSe
curity - IT Security training courses for individuals and corporations], UncommonHeaders[strict-transport-security
], X-Frame-Options[sameorigin], X-Powered-By[MOS 6502], X-UA-Compatible[IE=edge], X-XSS-Protection[1; mode=block]
```

2.2.1.2. WhatWeb

The previous output may seem a bit messy. If you want a more readable output, use the -v option.

As you can see, we now have all the information well organized.

```
root@kali:~/tools/WhatWeb# ./whatweb -v www.elearnsecurity.com
http://www.elearnsecurity.com/ [302]
http://www.elearnsecurity.com [302] HTTPServer[Microsoft-IIS/7.5], IP[199.193.116.231], Mi
ectLocation[https://www.elearnsecurity.com/], Title[Document Moved], UncommonHeaders[x-xss
ions,strict-transport-security], X-Frame-Options[sameorigin], X-Powered-By[MOS 6502], X-XS
ock]
URL     : http://www.elearnsecurity.com
Status  : 302
   HTTPServer ----------------------------------------------------------------
        Description: HTTP server header string. This plugin also attempts to
                     identify the operating system from the server header.
        String     : Microsoft-IIS/7.5 (from server string)

   IP ------------------------------------------------------------------------
        Description: IP address of the target, if available.
        String     : 199.193.116.231

   Microsoft-IIS -------------------------------------------------------------
        Description: Microsoft Internet Information Services (IIS) for Windows
                     Server is a flexible, secure and easy-to-manage Web server
                     for hosting anything on the Web. From media streaming to
                     web application hosting, IIS's scalable and open
                     architecture is ready to handle the most demanding tasks. -
                     homepage: http://www.iis.net/
        Version    : 7.5

   RedirectLocation ----------------------------------------------------------
        Description: HTTP Server string location. used with http-status 301 and
                     302
        String     : https://www.elearnsecurity.com/ (from location)

   Title ---------------------------------------------------------------------
        Description: The HTML page title
```

2.2.1.3. Wappalyzer

As we have seen, WhatWeb successfully identified the target Web Server.

Although we are not going to inspect all the tool options, feel free to explore.

Instead we are going to see another very useful tool that can be used directly from our web browser. It is called wappalyzer and it is a Web Browser plugin based tool that works both on *Firefox* and *Chrome*.

https://wappalyzer.com/download

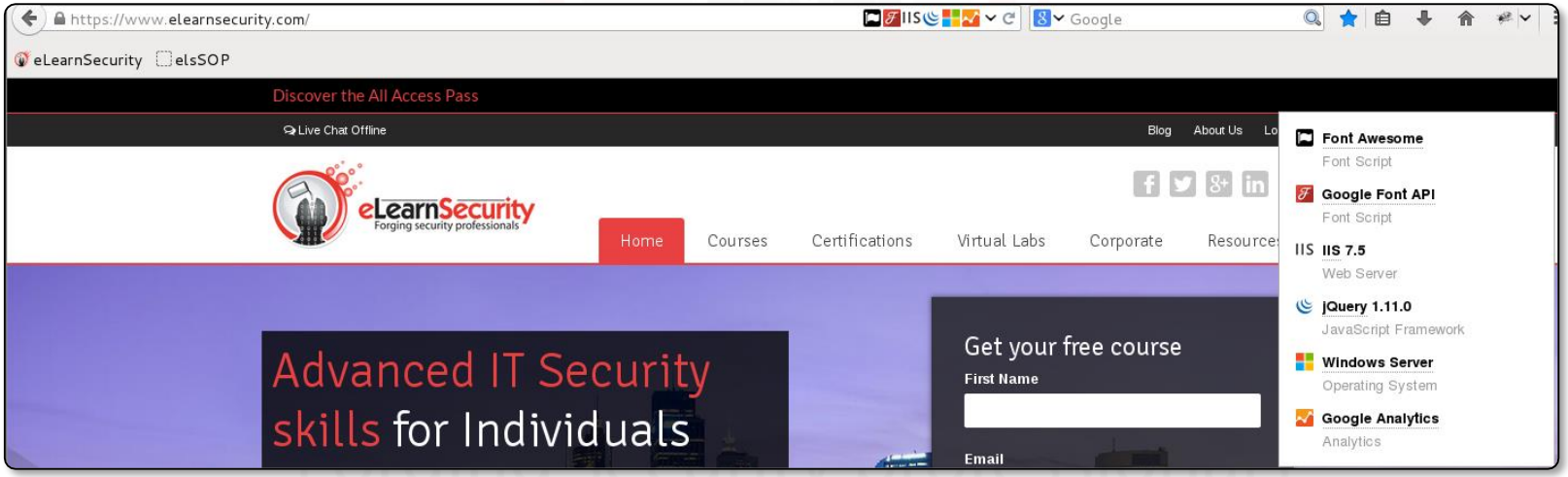
2.2.1.3. Wappalyzer

Once you install the plugin from the previous link, you have to navigate to your target website: you will see some icons in your address bar.

Each icon gives you information about the Web Server, such as the Operating System, The Web Server, JavaScript frameworks and much more.

2.2.1.3. Wappalyzer

In order to inspect the information found, just click on an icon and a pop up will appear on the right, listing all the information gathered.



3..2.1. Video - Web App Fingerprinting

If you have a FULL or ELITE plan you can click on the image on the left to start the video

2.2.1.4. Fingerprinting Webserver Modules

Along these same lines of how we fingerprinted the web server version, we can fingerprint what modules are installed and in use on the server.

Modules we are looking for are ISAPI modules (for IIS) or Apache modules that may interfere with or alter our test results.

2.2.1.4. Fingerprinting Webserver Modules

Nowadays, more and more websites use search engine and human-friendly URLs (SEF URLs).

So-called "ugly" URLs are the ones that carry query string parameters and values that are meaningful to the web server but not representative of the content on the page.

For example, www.example.com/read_doc.php?id=100 tells the server to query the database to fetch the document with id=100. This is not helpful to search engines looking for the document's contents.

A search engine-friendly version would be www.example.com/read/Buffer_Overflow.html.

So, how are the two translated?

When a user requests read_doc.php?id=100 the server side module in charge of translating the URL will use regular expressions to match a **Rewrite rule** and will translate the URL according to the rules specified by the administrator.

In the case above, *Buffer overflow* is the title field in the database at id=100.

URL rewriting is done on Apache with the **mod_rewrite** module or .**htaccess**.

On IIS it is handled by *Ionic Isapi Rewrite* or *Helicon Isapi Rewrite*.

The presence of URL-rewriting is easy to recognize as in the above example and should be kept in mind during the testing phase when attempting input validation attacks.

This type of attacks involves the use of malformed input (among the other data input) using the URL parameter.

Not having the real URL, and only the rewritten URL, will make it much more difficult for a penetration tester to try these attacks on URLs. However, still be possible to carry malformed payload using other input "*channels*" such as: forms, cookies, and headers.

Search engine friendly URLs are **not** a security feature at all; input validation attacks are still possible if you can reverse-engineer the translation rules.

However, there will be only rare cases in which the rewritten URL is easy to reverse engineer to its original form.

Also note that input from forms are still intact for us to tamper with.

```
Example: www.example.com/news_read/112
```

We can make a guess by requesting www.example.com/news_read.php?id=112.

If the two pages match and no 404 error is returned, we have found the URL rewriting rule.

As you can see, we are guessing on the parameter name- (id), which usually does not appear in the rewritten URL.

2.2.2. Enumerating Subdomains

Since we have already discussed DNS, this is the right place to mention **subdomain enumeration**.

The enumeration exercise starts by mapping all available subdomains within a domain name.

This will widen our attack surface and sometimes reveal hidden management backend panels or intranet web applications that the network administrators intended to protect through the old disgraced method of **security through obscurity**.

2.2.2. Enumerating Subdomains

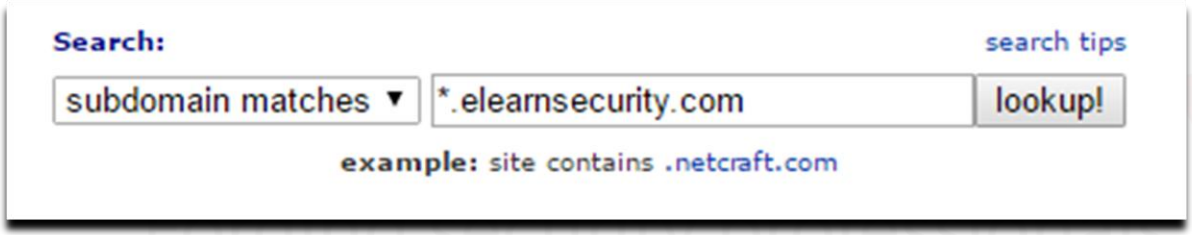
There are lots of ways to enumerate subdomains:

- Netcraft
- Google
- Crawling / Brute force
- Tools
- Zone transfers

2.2.2.1. Enumerating subdomains with Netcraft

We have already used Netcraft to gather information from a specific domain, but Netcraft can also be used to enumerate subdomains.

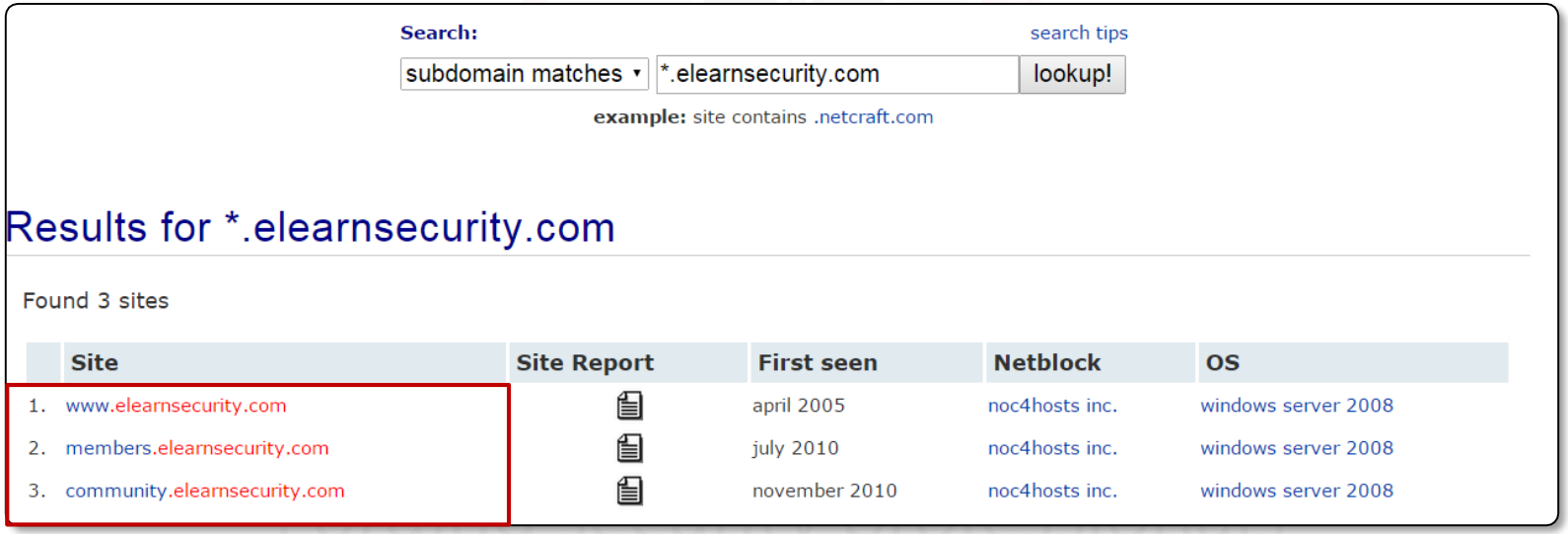
In order to list all the subdomains of a specific target we need to open the Netcraft search page, select "*subdomain matches*" from the dropdown menu and type in our string:



http://searchdns.netcraft.com/

2.2.2.1. Enumerating subdomains with Netcraft

If the target has any subdomain, we will see them listed on the results page. If you want to get more information about a specific subdomain, just click on the "*Site Report*" icon.



Search: search tips

subdomain matches | *.elearnsecurity.com | lookup!

example: site contains .netcraft.com

Results for *.elearnsecurity.com

Found 3 sites

	Site	Site Report	First seen	Netblock	OS
1.	www.elearnsecurity.com		april 2005	noc4hosts inc.	windows server 2008
2.	members.elearnsecurity.com		july 2010	noc4hosts inc.	windows server 2008
3.	community.elearnsecurity.com		november 2010	noc4hosts inc.	windows server 2008

2.2.2.2. Enumerating subdomains with Google

Although tools such as Netcraft are very useful in finding subdomains, search engines are sometimes an even better option. We will exploit the power of Google search operators to tweak the results and enumerate a list of subdomains.

Let us see some example that can be run with Google.

http://www.googleguide.com/advanced_operators_reference.html

2.2.2.2. Enumerating subdomains with Google

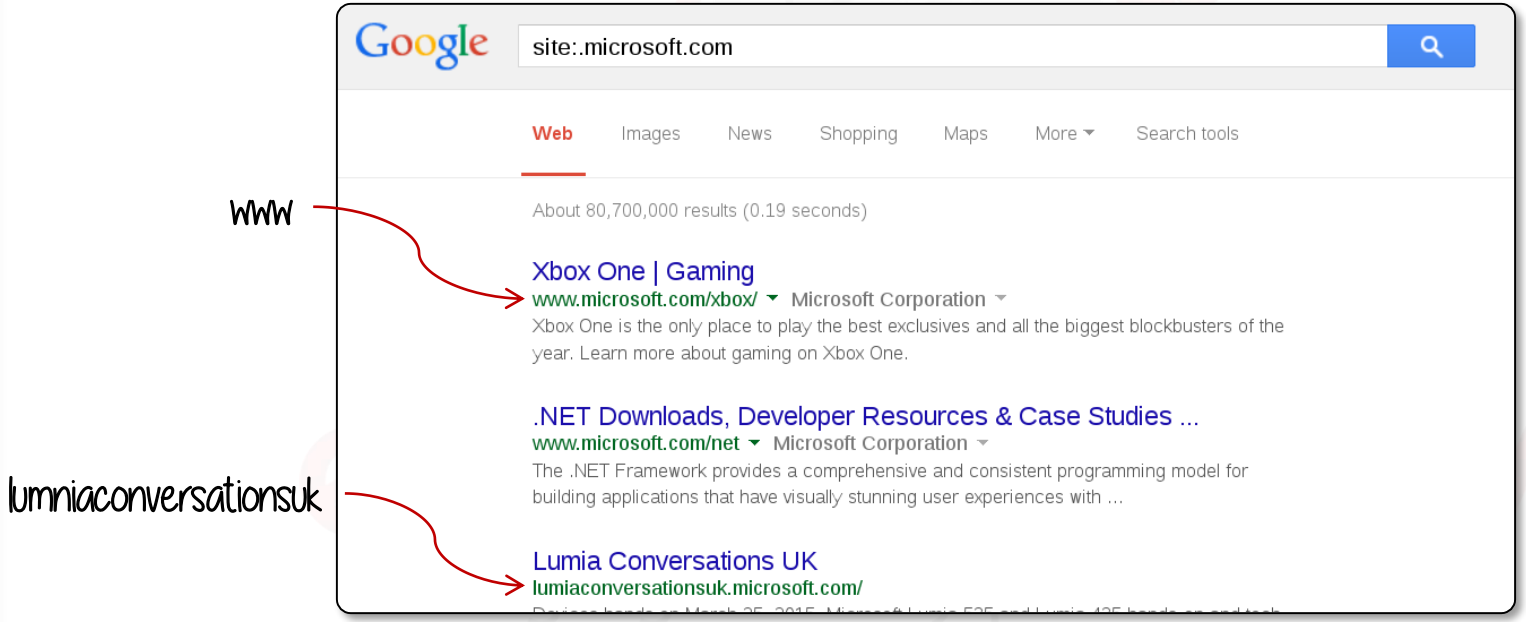
Let us suppose our target is microsoft.com, and our goal is to obtain a list of all its subdomains.

Our first search query string will be something like this:

```
site:.microsoft.com
```

The site: operator restricts the search results to the domain specified. In our case means that we will get results that have the domain .microsoft.com. Let's see what we get.

As we can see, each result displayed is part of the domain microsoft.com.



2.2.2.2. Enumerating subdomains with Google

As you have probably noticed there are some subdomains that appear more often than others (such as www). Moreover, once you discover some subdomains, we have to further tweak our search query in order to delete them from the results. To do this we can use the minus operator (-) in conjunction with site or inurl:

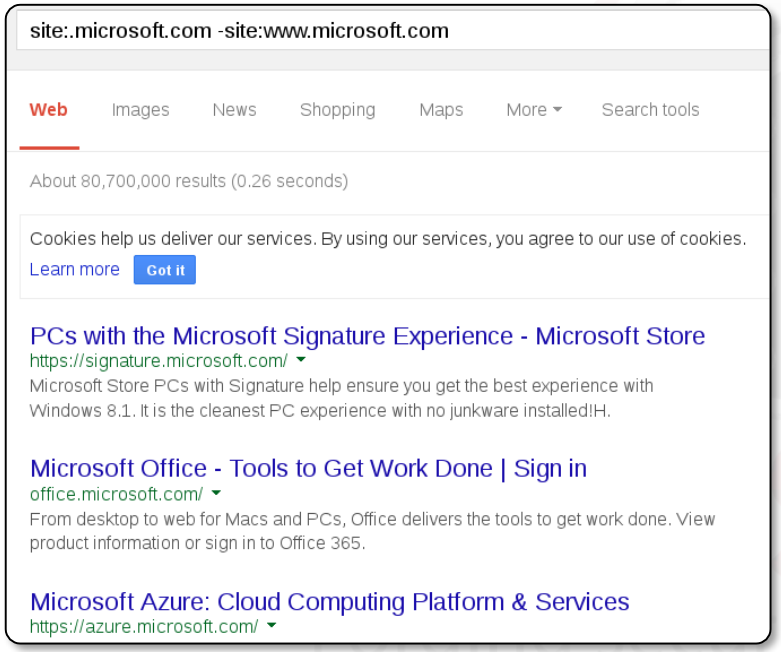
```
site:.microsoft.com -inurl:www.
```

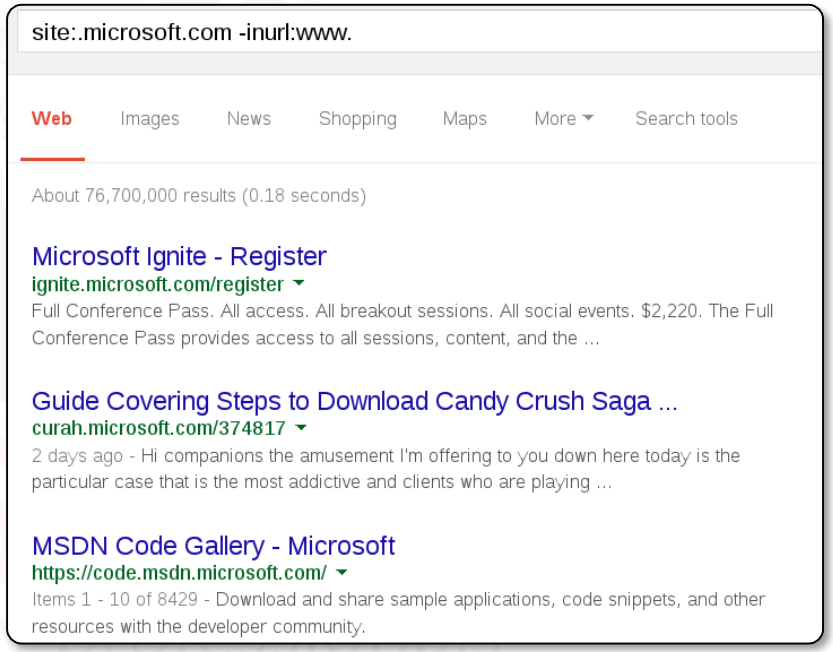
or

```
site:.microsoft.com -site:www.microsoft.com
```

2.2.2.2. Enumerating subdomains with Google

As we can see in the results, the subdomain www no longer appears in the output:





2.2.2.2. Enumerating subdomains with Google

Now we can continue tweaking our search query by removing the new subdomains found. So we will keep adding "-site" or "-inurl" until we find all the subdomains:

```
site:microsoft.com -site:subdomain1.microsoft.com
site:subdomain2.microsoft.com -inurl:subdomain3.microsoft.com
```

As you can imagine the process can be continually exhaustive, and why it is important to take good notes and use your mindmapping software.

2.2.2.3. Enumerating subdomains with Tools

In addition to search engines, there are a plenty of tools that can be used to enumerate subdomains. Some of them parse search engine results, while others use wordlists to verify if a specific set of domains exist. The following is a small list of these tools:

- dnsrecon: https://github.com/darkoperator/dnsrecon
- subbrute: https://github.com/TheRook/subbrute
- fierce: https://github.com/davidpepper/fierce-domain-scanner
- Nmap: https://nmap.org/book/man-host-discovery.html
- dnsenum: https://code.google.com/archive/p/dnsenum/downloads
- knock: https://github.com/guelfoweb/knock
- theHarvester: https://github.com/laramies/theHarvester
- recon-ng: https://bitbucket.org/LaNMaSteR53/recon-ng/wiki/Usage%20Guide

Since they are all very similar, we are going to inspect only a few (subbrute, dnsrecon and theHarvester). We encourage you try them by yourself to verify how they work and what features they offer.

The first tool, as the name suggests, uses a default wordlist to find the subdomains of a specific target. These types of tools are very useful if we cannot rely on search engines (IE: performing an internal pentest).

subbrute

If you do not have the tool already installed on your machine, you can simply run the following command:

```
git clone https://github.com/TheRook/subbrute.git
```

Once you have cloned the repository on your machine, you can launch the tool and display its options with the following command:

```
python subbrute.py -h
```

subbrute

By running the previous command, you should receive an output similar to the following screenshot. As you can see, the tool uses, by default, a wordlist named names.txt.

```
root@kali:~/tools/subbrute# python subbrute.py -h
Usage: subbrute.py [options] target

Options:
  -h, --help            show this help message and exit
  -s SUBS, --subs=SUBS  (optional) list of subdomains,  default = 'names.txt'
  -r RESOLVERS, --resolvers=RESOLVERS
                        (optional) A list of DNS resolvers, if this list is
                        empty it will OS's internal resolver default =
                        'resolvers.txt'
  -t TARGETS, --targets_file=TARGETS
                        (optional) A file containing a newline delimited list
                        of domains to brute force.
  -o OUTPUT, --output=OUTPUT
                        (optional) Output to file (Greppable Format)
  -j JSON, --json=JSON  (optional) Output to file (JSON Format)
  -a, -A                (optional) Print all IPv4 addresses for sub domains
                        (default = off).
  --type=TYPE           (optional) Print all reponses for an arbitrary DNS
                        record type (CNAME, AAAA, TXT, SOA, MX...)
  -c PROCESS_COUNT, --process_count=PROCESS_COUNT
                        (optional) Number of lookup theads to run. default =
                        16
  -f FILTER, --filter_subs=FILTER
                        (optional) A file containing unorganized domain names
                        which will be filtered into a list of subdomains
                        sorted by frequency.  This was used to build
                        names.txt.
  -v, --verbose         (optional) Print debug information.
root@kali:~/tools/subbrute#
```

subbrute

Let's try to run the tool and see what subdomains it is able to enumerate. We are not going to use any particular option at this time.

Note: the whole task may take a while

```
root@kali:~/tools/subbrute# python subbrute.py microsoft.com
microsoft.com
www.microsoft.com
home.microsoft.com
cs.microsoft.com
my.microsoft.com
members.microsoft.com
blogs.microsoft.com
search.microsoft.com
i.microsoft.com
feeds.microsoft.com
forums.microsoft.com
math.microsoft.com
news.microsoft.com
games.microsoft.com
dev.microsoft.com
mail.microsoft.com
info.microsoft.com
music.microsoft.com
support.microsoft.com
help.microsoft.com
s.microsoft.com
e.microsoft.com
office.microsoft.com
profile.microsoft.com
member.microsoft.com
```

subbrute

As shown in the previous screenshot, by simply running the tools against the microsoft.com domain, it is able to enumerate an acceptable number of subdomains.

In case you want to use a custom wordlist you can simply run the following command:

```
python subbrute.py -h -s [path_to_file.txt]
```

dnsrecon

The second tool we want to use is dnsrecon. If you are using Kali Linux, it is already installed on your machine and you can simply run it with the following command:

```
dnsrecon -h
```

Similarly to subbrute, dnsrecon can leverage wordlists to enumerate subdomains. However, in addition to this, it also offers the possibility to use search engines like google.

2.2.2.3. Enumerating subdomains with Tools

dnsrecon

As we can see in the following screenshot, it offers many more options.

```
root@kali:~# dnsrecon -h
Version: 0.8.8
Usage: dnsrecon.py <options>

Options:
   -h, --help                 Show this help message and exit
   -d, --domain      <domain> Domain to Target for enumeration.
   -r, --range       <range>  IP Range for reverse look-up brute force in formats (first-last)
                              or in (range/bitmask).
   -n, --name_server <name>   Domain server to use, if none is given the SOA of the
                              target will be used
   -D, --dictionary  <file>   Dictionary file of sub-domain and hostnames to use for
                              brute force.
   -f                         Filter out of Brute Force Domain lookup records that resolve to
                              the wildcard defined IP Address when saving records.
   -t, --type        <types>  Specify the type of enumeration to perform:
                              std      To Enumerate general record types, enumerates.
                                       SOA, NS, A, AAAA, MX and SRV if AXRF on the
                                       NS Servers fail.

                              rvl      To Reverse Look Up a given CIDR IP range.

                              brt      To Brute force Domains and Hosts using a given
                                       dictionary.

                              srv      To Enumerate common SRV Records for a given

                                       domain.

                              axfr     Test all NS Servers in a domain for misconfigured
                                       zone transfers.

                              goo      Perform Google search for sub-domains and hosts.
```

dnsrecon

The option we are interested into is -g: *perform Google enumeration with standard enumeration*. Moreover, it is multi-threaded, so we can speed up the process by setting more threads (--threads). Let us use microsoft.com once again and run the following command:

```
dnsrecon -d microsoft.com -g
```

2.2.2.3. Enumerating subdomains with Tools

dnsrecon

As we can see in the following outputs, first, it execute some general enumeration by checking the DNS configuration and then begins enumerating the domains via Google.

```
root@kali:~# dnsrecon -d microsoft.com -g
[*] Performing General Enumeration of Domain: microsoft.com
[-] DNSSEC is not configured for microsoft.com
[*]      SOA ns1.msft.net 208.84.0.53
[*]      SOA ns1.msft.net 2620:0:30::53
[*]      NS ns3.msft.net 193.221.113.53
[*]      NS ns3.msft.net 2620:0:34::53
[*]      NS ns4.msft.net 208.76.45.53
[*]      NS ns4.msft.net 2620:0:37::53
[*]      NS ns1.msft.net 208.84.0.53
[*]      NS ns1.msft.net 2620:0:30::53
[*]      NS ns2.msft.net 208.84.2.53
[*]      NS ns2.msft.net 2620:0:32::53
[*]      MX microsoft-com.mail.protection.outlook.com
```

```
[*] Performing Google Search Enumeration
[*]      CNAME www.microsoft.com toggle.www.ms.akadns.net
[*]      CNAME toggle.www.ms.akadns.net www.microsoft.com-c.edgekey.net
[*]      CNAME www.microsoft.com-c.edgekey.net www.microsoft.com-c.edgekey.net.globalredi
[*]      CNAME www.microsoft.com-c.edgekey.net.globalredir.akadns.net e10088.dspb.akamaie
[*]      A e10088.dspb.akamaiedge.net 72.247.197.45
[*]      CNAME ieonline.microsoft.com any.edge.bing.com
[*]      A any.edge.bing.com 204.79.197.200
[*]      CNAME windows.microsoft.com origin.windows.microsoft.com.akadns.net
[*]      A origin.windows.microsoft.com.akadns.net 134.170.119.140
[*]      CNAME support.microsoft.com wildcard.support.microsoft.com.edgekey.net
[*]      CNAME wildcard.support.microsoft.com.edgekey.net e10315.g.akamaiedge.net
[*]      A e10315.g.akamaiedge.net 2.17.104.63
```

theHarvester

TheHarvester is a tool for gathering subdomain names from different public sources such as search engines or PGP key servers. Among its subdomain enumeration features, theHarvester is also able to retrieve data related to the target organization from many websites such as *LinkedIn*, *People123*, *Twitter*, *Google+* and few more.

theHarvester

Kali Linux already has theHarvester installed by default. We can simply run it as follows:

```
theharvester [options]
```

-d	Domain to search
-l	Limit the results to work with
-b	Data source (bing, google, linkedin, pgp, all,…)
-f	Output to HTML or XML file (optional - good for long lists)

theHarvester

Now that we know its basic options, let's try to run the tools against microsoft.com. We will use *google* as the search engine, we will limit the results to 200, and also store the results into a HTML file.

Our command will look like this:

```
theharvester -d microsoft.com -b google -l 200
-f /root/Desktop/msresults.html
```

2.2.2.3. Enumerating subdomains with Tools

theHarvester

Even if we set an output file, the results are also displayed in the console.

Console

```
[+] Emails found:
------------------
oss@microsoft.com
secure@microsoft.com
joakim.karlen@microsoft.com
Edvard.bergstrom@microsoft.com
dinei@microsoft.com
cormac@microsoft.com
@microsoft.com
hiballan@microsoft.com
thomkar@microsoft.com
antr@microsoft.com
simonpj@microsoft.com

[+] Hosts found in search engines:
------------------------------------
[-] Resolving hostnames IPs...
72.247.197.45:www.microsoft.com
134.170.119.140:windows.microsoft.com
168.62.198.20:commerce.microsoft.com
```

HTML

E-mails names found:

- oss@microsoft.com
- secure@microsoft.com
- joakim.karlen@microsoft.com
- Edvard.bergstrom@microsoft.com
- dinei@microsoft.com
- cormac@microsoft.com
- @microsoft.com
- hiballan@microsoft.com
- thomkar@microsoft.com
- antr@microsoft.com
- simonpj@microsoft.com

Hosts found:

- 72.247.197.45:www.microsoft.com
- 134.170.119.140:windows.microsoft.com
- 168.62.198.20:commerce.microsoft.com
- 2.17.104.63:support.microsoft.com
- 191.235.177.147:azure.microsoft.com

theHarvester

Another, very useful feature of *theHarvester*, is the ability to collect data from other sources, such as LinkedIn. Obviously, in order to list all the people that are somehow related to our target.

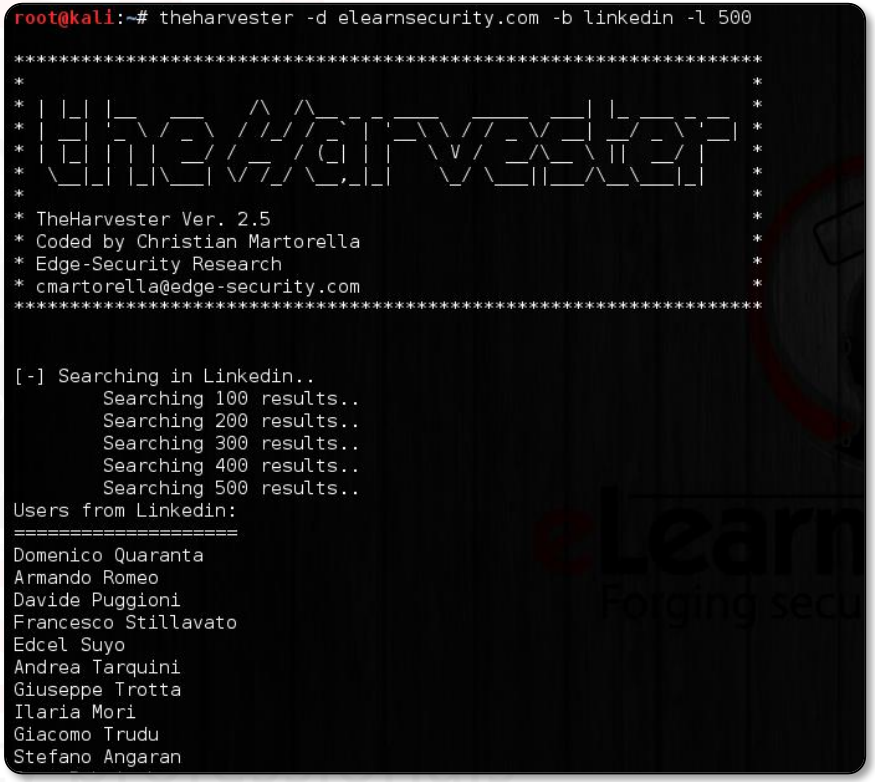
For example, let us see what happens if we use the target eLearnSecurity:

```
theharvester -d elearnsecurity.com -b linkedin -l 200
```

theHarvester

We will obtain something similar to the following screenshot.

All the people listed here are related to eLearnSecurity (employees, customers, partner and so on).



2.2.2.4. Enumerating subdomains via Zone Transfer

In addition to search engines and tools, there are other ways we can discover information about domains and subdomains. One of these is through a Zone Transfer.

"A Zone Transfer is the term used to refer to the process by which the contents of a DNS Zone file are copied from a primary DNS server to a secondary DNS server."

Explanation of a DNS Zone Transfer. Microsoft Support.

https://en.wikipedia.org/wiki/DNS_zone_transfer

2.2.2.4. Enumerating subdomains via Zone Transfer

Zone transfers are usually the result of a misconfiguration of the remote DNS server. They should be enabled (if required) only for trusted IP addresses.

When zone transfers are available, we can enumerate all of the DNS records for that zone. This includes all the subdomains of our domain (A records).

2.2.2.4. Enumerating subdomains via Zone Transfer

On Windows systems, we can gather information from Zone Transfer by running the following commands:

```
nslookup
server [NAMESERVER FOR mydomain.com]
ls -d mydomain.com
```

- [NAMESERVER] = you can find this by just doing:

```
nslookup -type=NS mydomain.com
```

2.2.2.4. Enumerating subdomains via Zone Transfer

This screenshot shows the output from the previous commands run against the target elsfoo.com.

```
C:\>nslookup -type=NS elsfoo.com
Server:  google-public-dns-a.google.com
Address:  8.8.8.8

Non-authoritative answer:
elsfoo.com      nameserver = ns.elsfoo.com
elsfoo.com      nameserver = ns6.dnsmadeeasy.com
elsfoo.com      nameserver = ns5.dnsmadeeasy.com
elsfoo.com      nameserver = ns7.dnsmadeeasy.com

C:\>nslookup
Default Server:  google-public-dns-a.google.com
Address:  8.8.8.8

> server ns.elsfoo.com
Default Server:  ns.elsfoo.com
Address:  74.50.103.103

> ls -d elsfoo.com
[ns.elsfoo.com]
 elsfoo.com.                    SOA     ns.elsfoo.com bernyreed.elsfoo.com. (41 900 600 86
 elsfoo.com.                    NS      ns6.dnsmadeeasy.com
 elsfoo.com.                    NS      ns7.dnsmadeeasy.com
 elsfoo.com.                    NS      ns.elsfoo.com
 elsfoo.com.                    NS      ns5.dnsmadeeasy.com
 elsfoo.com.                    MX      5       alt1.aspmx.l.google.com
 elsfoo.com.                    MX      5       alt2.aspmx.l.google.com
 elsfoo.com.                    MX      10      aspmx2.googlemail.com
 elsfoo.com.                    MX      10      aspmx3.googlemail.com
 elsfoo.com.                    MX      1       aspmx.l.google.com
 elsfoo.com.                    TXT             "google-site-verification=omyr9Nazb1WYCs_l
axvvtCjTI2EA4_S-Q"

 elsfoo.com.                    TXT             "v=spf1 include:_spf.google.com ~all"

 ns6.dnsmadeeasy.com.           A       208.80.124.13
 ns7.dnsmadeeasy.com.           A       208.80.126.13
 ns5.dnsmadeeasy.com.           A       208.94.148.13
 ns5.dnsmadeeasy.com.           AAAA    2600:1800:5::1
 admin                          A       74.50.103.103
 intranet                       A       74.50.103.103
 ns                             A       74.50.103.103
 private                        A       74.50.103.103
 www                            A       74.50.103.103
 elsfoo.com.                    SOA     ns.elsfoo.com bernyreed.elsfoo.com. (41 900 600 86
>
```

2.2.2.4. Enumerating subdomains via Zone Transfer

On Linux systems you can run the following command:

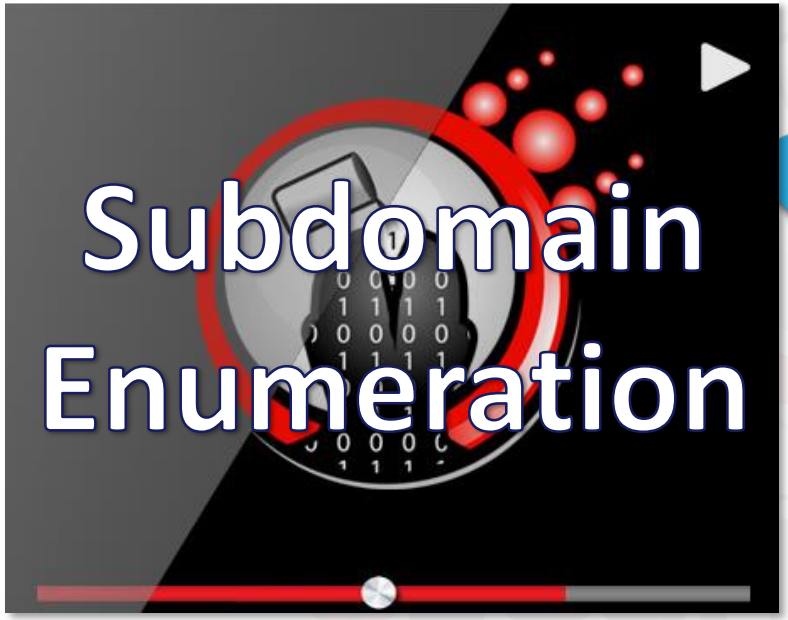
```
dig @nameserver axfr mydomain.com
```

- nameserver is a nameserver for mydomain.com
- axfr is the mnemonic opcode for the DNS zone transfer.

2.2.2.4. Enumerating subdomains via Zone Transfer

This screenshot shows the output from the previous command against the target elsfoo.com.

```
root@kali:~# dig @ns.elsfoo.com AXFR elsfoo.com

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> @ns.elsfoo.com AXFR elsfoo.com
; (1 server found)
;; global options: +cmd
elsfoo.com.             3600    IN      SOA     ns.elsfoo.com. bernyreed.elsfoo.com. 41 900 600 8640(
elsfoo.com.             3600    IN      NS      ns6.dnsmadeeasy.com.
elsfoo.com.             3600    IN      NS      ns7.dnsmadeeasy.com.
elsfoo.com.             3600    IN      NS      ns.elsfoo.com.
elsfoo.com.             3600    IN      NS      ns5.dnsmadeeasy.com.
elsfoo.com.             3600    IN      MX      5 alt1.aspmx.l.google.com.
elsfoo.com.             3600    IN      MX      5 alt2.aspmx.l.google.com.
elsfoo.com.             3600    IN      MX      10 aspmx2.googlemail.com.
elsfoo.com.             3600    IN      MX      10 aspmx3.googlemail.com.
elsfoo.com.             3600    IN      MX      1 aspmx.l.google.com.
elsfoo.com.             3600    IN      TXT     "google-site-verification=omyr9Nazb1WYCs_DW29VGj_zMJ
elsfoo.com.             3600    IN      TXT     "v=spf1 include:_spf.google.com ~all"
ns6.dnsmadeeasy.com.    3600    IN      A       208.80.124.13
ns7.dnsmadeeasy.com.    3600    IN      A       208.80.126.13
ns5.dnsmadeeasy.com.    3600    IN      A       208.94.148.13
ns5.dnsmadeeasy.com.    3600    IN      AAAA    2600:1800:5::1
admin.elsfoo.com.       3600    IN      A       74.50.103.103
intranet.elsfoo.com.    3600    IN      A       74.50.103.103
ns.elsfoo.com.          3600    IN      A       74.50.103.103
private.elsfoo.com.     3600    IN      A       74.50.103.103
www.elsfoo.com.         3600    IN      A       74.50.103.103
elsfoo.com.             3600    IN      SOA     ns.elsfoo.com. bernyreed.elsfoo.com. 41 900 600 8640(
;; Query time: 144 msec
;; SERVER: 74.50.103.103#53(74.50.103.103)
;; WHEN: Tue Mar 31 12:01:01 2015
;; XFR size: 22 records (messages 1, bytes 800)
```



If you have a FULL or ELITE plan you can click on the image on the left to start the video

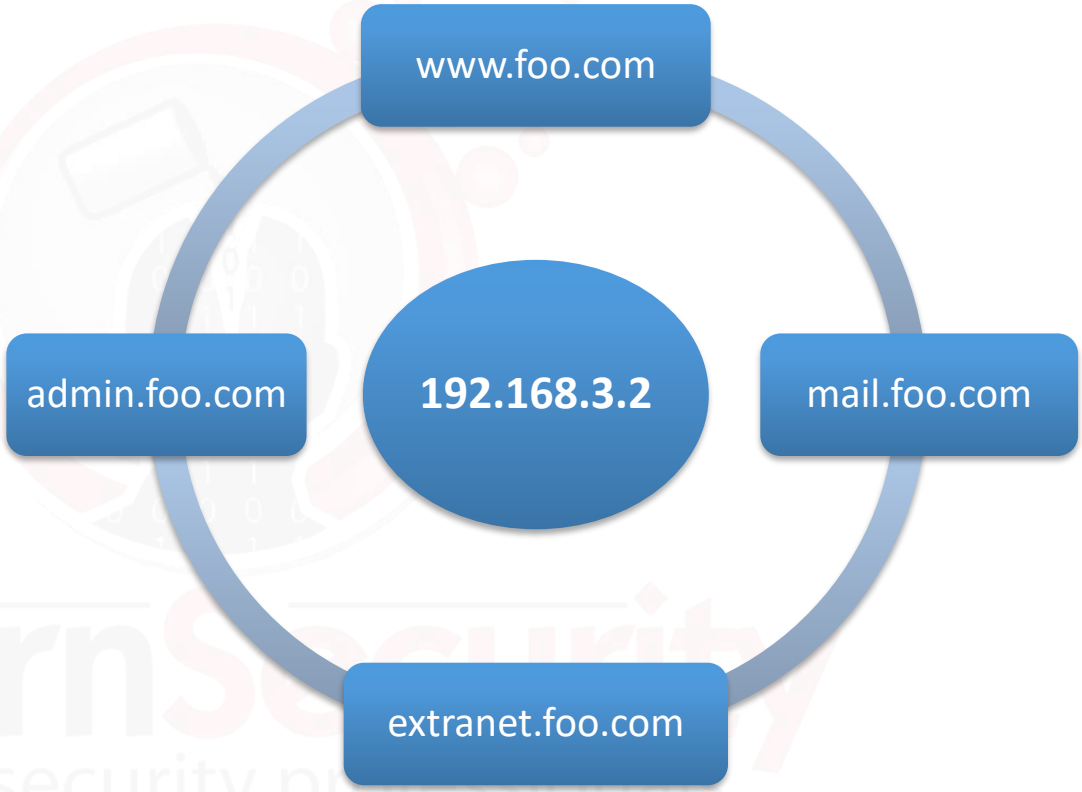
A virtual host is simply a website that shares an IP address with one or more other virtual hosts.

These hosts are domains and subdomains.

This is very common in a shared hosting environment where a multitude of websites share the same server/IP address.

For example:

There are multiple virtual hosts associated with the IP address 192.168.3.2.



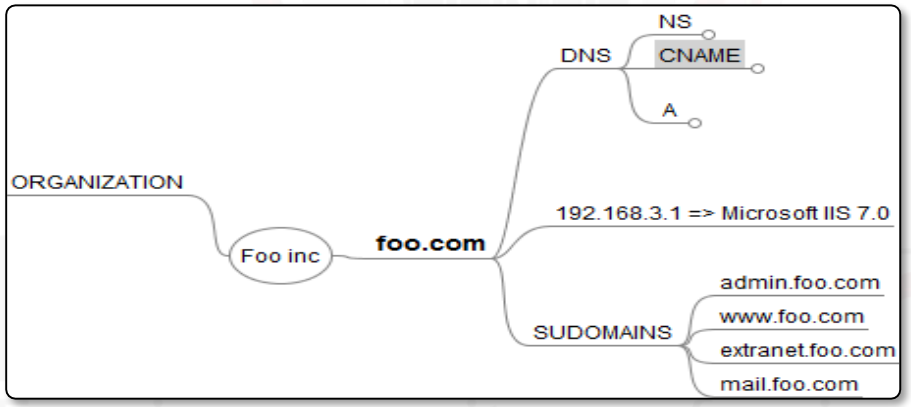
2.2.3. Finding Virtual Hosts

Many of the tools we have seen previously for enumerating subdomains can also be used in finding virtual hosts. The following is an example of what the tool **fierce** is able to discover starting from the domain elearnsecurity.com.

```
root@kali:~/tools/hostmap# fierce -dns elearnsecurity.com
DNS Servers for elearnsecurity.com:
        ns1.elearnsecurity.com
        ns.elearnsecurity.com
        ns5.dnsmadeeasy.com
        ns6.dnsmadeeasy.com
        ns7.dnsmadeeasy.com

Trying zone transfer first...
        Testing ns1.elearnsecurity.com
                Request timed out or transfer not allowed.
        Testing ns.elearnsecurity.com
                Request timed out or transfer not allowed.
        Testing ns5.dnsmadeeasy.com
                Request timed out or transfer not allowed.
        Testing ns6.dnsmadeeasy.com
                Request timed out or transfer not allowed.
        Testing ns7.dnsmadeeasy.com
                Request timed out or transfer not allowed.

Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...
Nope. Good.
Now performing 2280 test(s)...
162.220.56.82   blog.elearnsecurity.com
162.220.56.82   community.elearnsecurity.com
199.193.116.231 lib.elearnsecurity.com
199.193.116.231 members.elearnsecurity.com
199.193.116.231 mgmt.elearnsecurity.com
199.193.116.232 ns.elearnsecurity.com
199.193.116.233 ns1.elearnsecurity.com
199.193.116.231 webmail.elearnsecurity.com
199.193.116.231 www.elearnsecurity.com
```

Have you been updating your map with the latest information recovered in this chapter?

Remember that keeping track of your findings is important!



Fingerprinting Frameworks And Applications

Once we have a list of subdomains, we will apply the techniques that follow in this module to all of them.

We will basically start looking at the webpages running on each of the subdomains we have found.

Common applications are software that is available for anyone to use (aka COTS - Common off the shelf).

They can be either open source or commercial, but what makes them interesting for our analysis is the fact that we have access to their source code (and other security researchers may have looked at it before us).

We are able to read both the application logic and the security controls implemented (or not implemented) therefore, we gain a big advantage over applications built in-house. For a pentester, the logic of in-house applications is a "guesstimate" task to some degree.

Common applications may be:

- forums (like phpBB, vBulletin)
- CMS's (like Joomla or Drupal)
- CRM's, blogging platforms (like WordPress or Movable types)
- social networking scripts and a number of other applications.

For example, web scripts are available online at sites like: www.hotscripts.com.

Almost all of these freely (meaning open to anyone, regardless of their price) available applications suffered from some kind of vulnerability in their history.

Understanding what piece of commonly available software the web server is running will give us the possibility for an easy exploitation by just looking online for a publicly available exploit.

Even here, obtaining the name of the application will not be enough for us. We need the exact version in order to look for a working exploit online.

A basic step for fingerprinting the application involves browsing the website and looking at its URLs, appearance and logic.

Sometimes it is as easy as looking for the application's name in the web page content.

In other cases, we need to look at the web page source; the name and version is usually included in HTML comments or even in the HTTP headers.

Sending a GET request: This is how a Joomla powered website responds to a normal GET request:

Request

```
GET / HTTP/1.1
Host: www.joomla.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0)
Gecko/20100101 Firefox/31.0 Iceweasel/31.5.3
```

Response

```
HTTP/1.x 200 OK
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Server: LiteSpeed
Vary: Accept-Encoding
X-Content-Encoded-By: Joomla! 2.5
```

As you can see, by just using telnet, Burp suite, web browsers or any other way that will let us read the raw response, headers will reveal useful information about the website; this includes the CMS running on it: **Joomla 2.5**.

Moreover, remember that response headers may give other valuable information such as PHP version, exact web server version and modules installed.

Other applications may behave differently. The HTTP header exposing the CMS version can be suppressed so we would need to move on, examining the web page content for hints.

The open source community behind these projects (but many commercial applications act similarly), usually require the final user to keep a footer notification in place that gives credit to the project for more support and acknowledgment.

This also happens for commercial software like the famous forum application vBulletin; it reveals its presence both in the website title and footer:

Sometimes we have to look more in-depth to find what we are looking for.

We may need to read the web page source code and look for information in the META tags and in HTML comments. This is WordPress, the most popular blogging open source web application:

```
<link rel="wlwmanifest" type="application/wlwmanifest+xml" href
includes/wlwmanifest.xml" />
<meta name="generator" content="WordPress 4.2-beta3-31946" />
<script type="text/javascript" src="//s.w.org/wp-includes/js/jq
<script>document.cookie='devicePixelRatio='+((window.devicePixe
```

Many different kinds of CMSs are available online for free or licensed commercially. Very common examples of open source CMSs are *Joomla*, *Drupal* or *Mambo*.

These have a large customer base and an ever-growing support community providing free add-ons, components and extensions, which add more functionality to the core application.

These add-ons are usually poorly coded and contain vulnerabilities.

While the core parts of these projects are usually built following the best practices of secure coding, thousands of free-to-use extensions are coded by amateurs, and most of the time, these are affected by many different web application vulnerabilities.

In the case of Joomla, (this discussion applies to many other, similar projects) fingerprinting installed add-ons is as easy as looking at the website URLs.

Joomla URLS consist of 3 main parts:

```
index.php?option=%component_name%&task=%task_value%
```

Index.php is the only script you will ever see on Joomla. It is in charge of loading the specified component passed in, via the option parameter.

More tasks and arguments are passed to that component with subsequent parameters.

2.3.1. Fingerprinting Third-Party Add-Ons

The following is an example for the very popular Docman, document manager, component:

```
index.php?option=com_docman&task=doc_view&gid=100
```

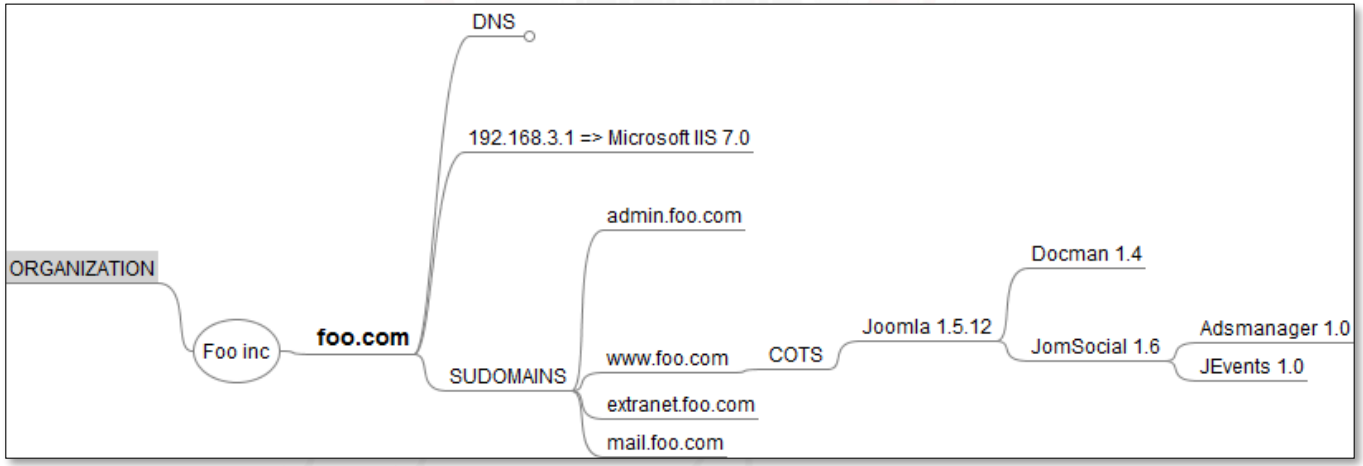
Here we are loading the Docman add-on, declaring that we want to view the document with id=100.

It should be clear that by looking at the option parameter in the URL, we can easily understand what potentially vulnerable third party add-ons are installed and in use on the website.

2.3.1. Fingerprinting Third-Party Add-Ons

In our Information gathering process, we will not only list all the common applications in use, but also, all the third party add-ons in use in that application. These will likely be useful for our penetration tasks later on.

Our map is starting to get filled up with the applications found on the subdomain (Joomla). We can (and should) also list 3rd party add-ons (such as *JomSocial*) and add-ons to 3rd party add-ons (such as *Jevents*, *Adsmanager*)



Fingerprinting Custom Applications

When you are not in front of a commonly available application, you have to go through an initial overview of the application logic.

In this case we have an in-house application, customized for the organization we are auditing. The inner logic is unknown to us but can be reverse engineered with a careful analysis of its behavior.

Our first step in this case will be to consider the overall scope of the application:

- What is it for?
- Does it allow user registration?
- Does it have an administration panel?
- Does it take input from the user?
- What kind of input?
- Does it accept file uploads?
- Does it use JavaScript or Ajax or Flash? And so on.

These questions can be answered by just visiting the website and taking notes of anything we come across in the process.

Spidering (or crawling) the application is addressed at a later stage, but valuable in this case.

First we want to understand what our application does and how it does it.

Another important aspect to take into consideration is the possibility that we may find common software intertwined with custom code.

Forums and blogs are great examples when we are dealing with a custom application. It is very likely that we find open source or commercial applications implementing blogs, forums, shopping carts and a number of other functions.

This applies to small company websites as well as corporate ones. Recognizing them among the custom code is important and should be noted on our functional graph.

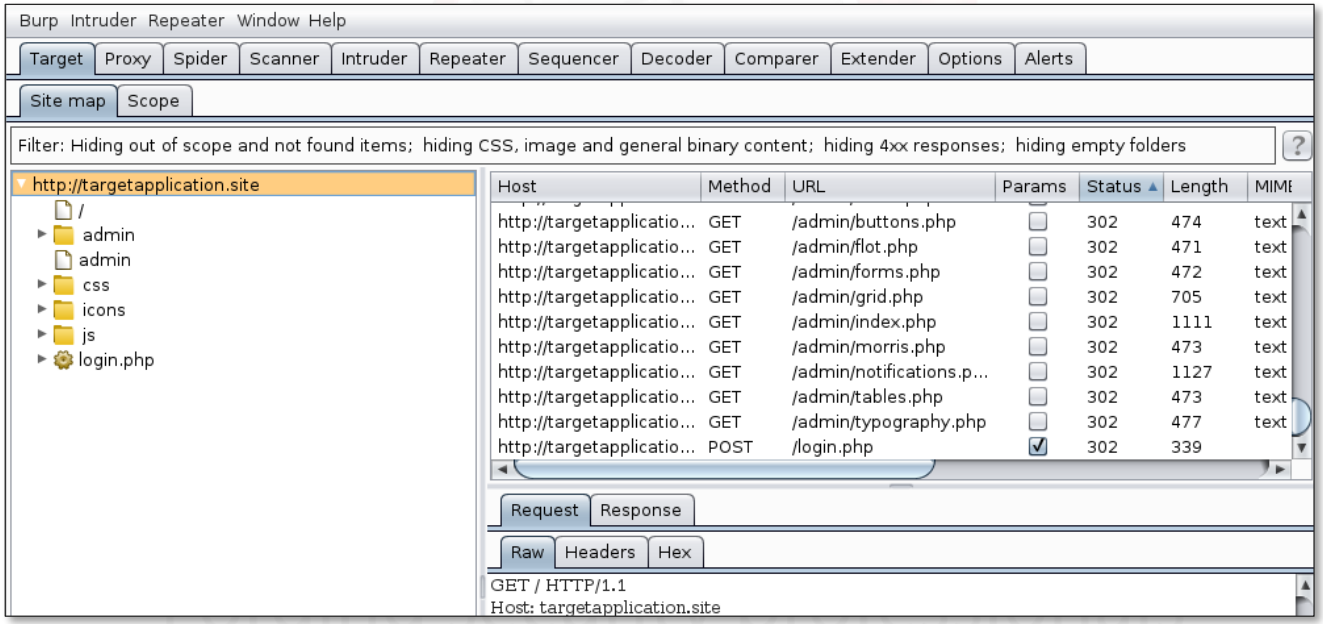
At this point, it would be helpful to browse the application with a proxy in the middle of our requests, collecting all the headers and responses. This will help us analyze the results later.

A good recommended proxy for this stage is Burp Proxy

https://portswigger.net/burp

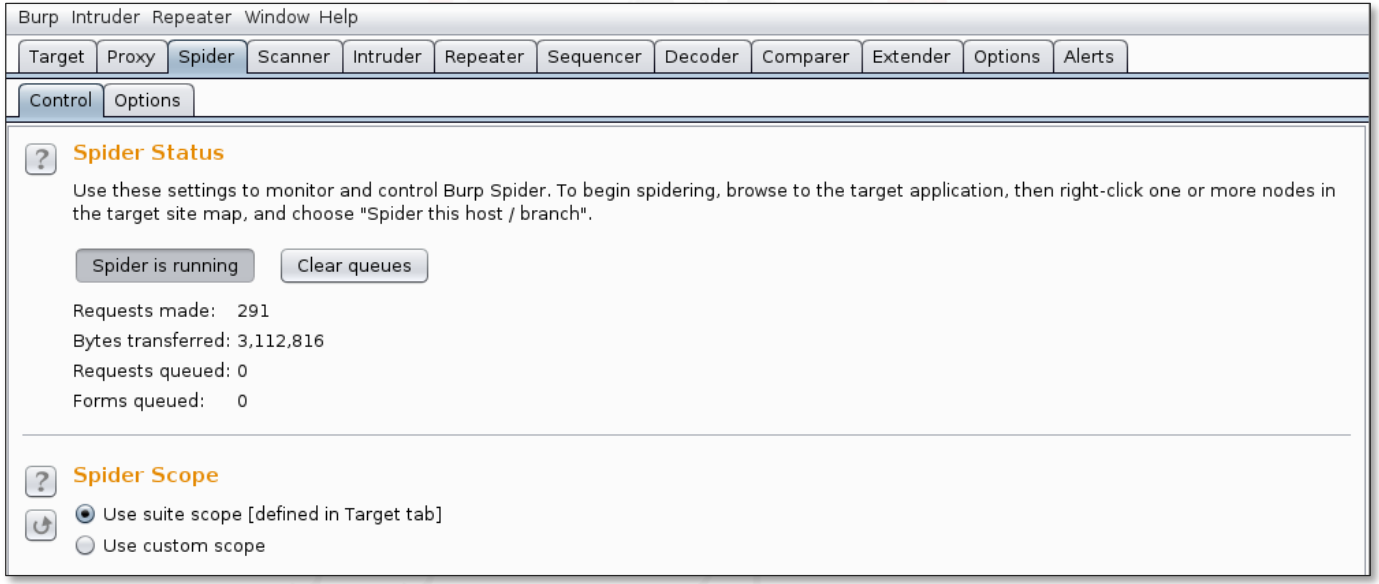
2.4.1. Burp Target Crawler

This is the Burp Target tool. Burp lets us configure our scope through simple regular expressions.

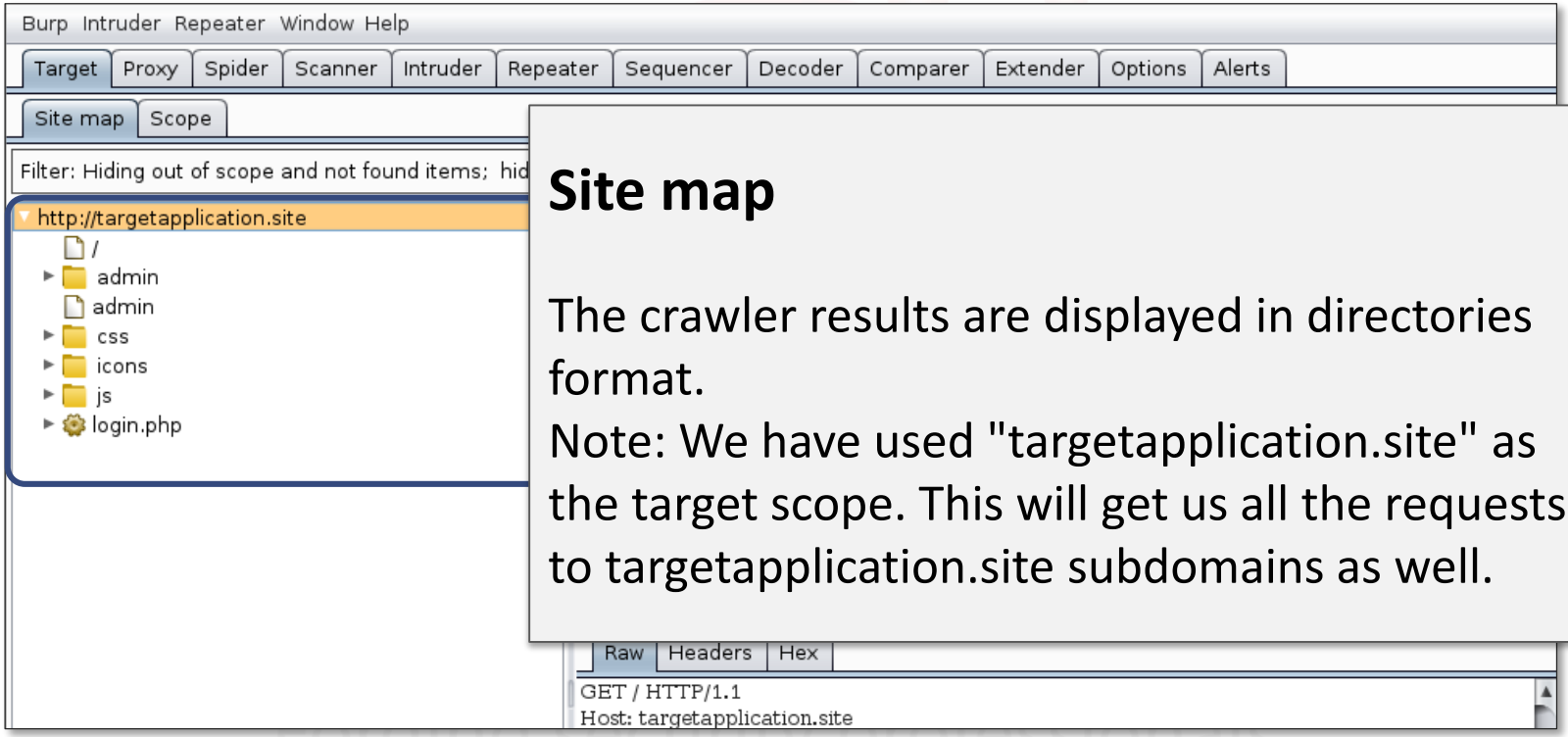


2.4.1. Burp Target Crawler

While browsing, and if enabled, the crawler (Spider tab) automatically generates and records requests and response headers for further inspection.



2.4.1. Burp Target Crawler

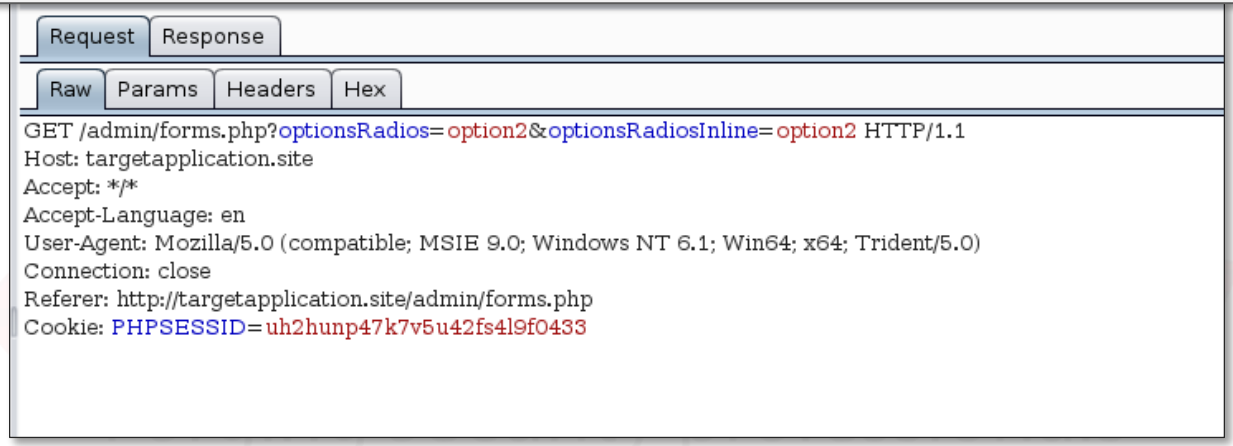


Site map

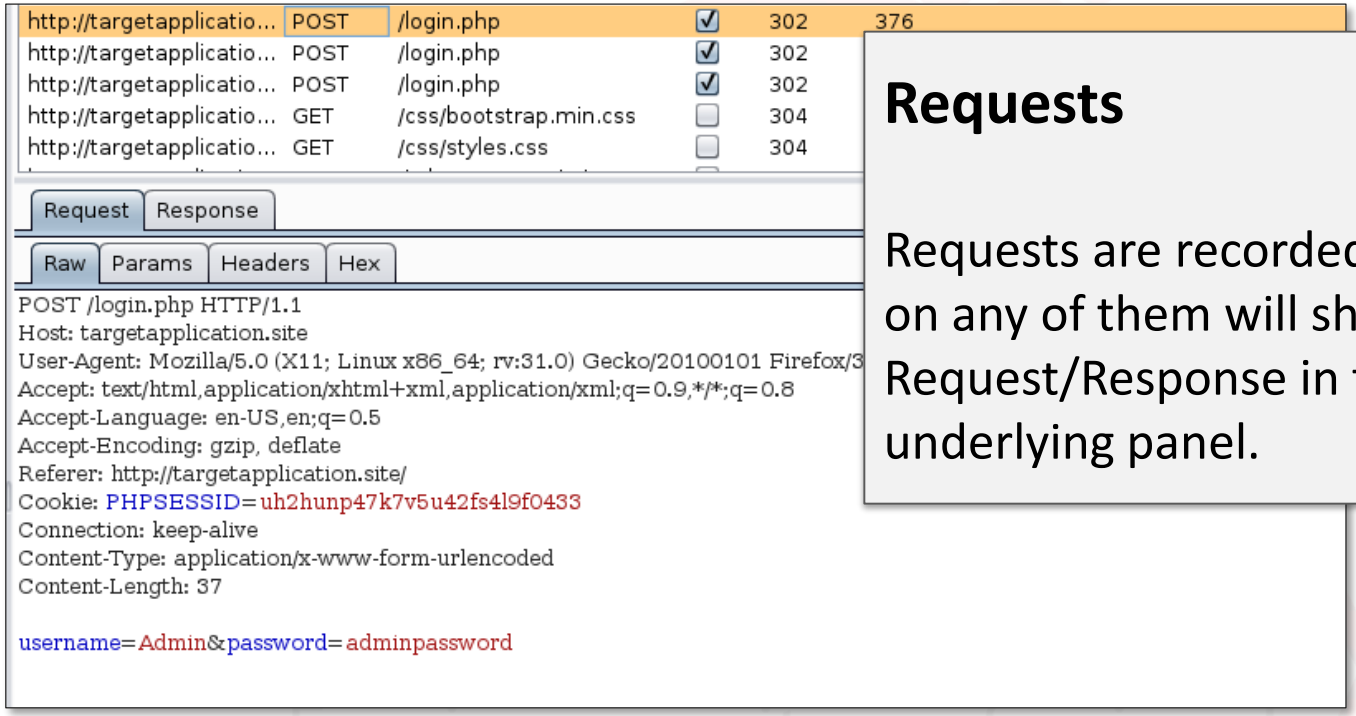
The crawler results are displayed in directories format.
Note: We have used "targetapplication.site" as the target scope. This will get us all the requests to targetapplication.site subdomains as well.

Headers

For each request made by the web browser (or the crawler), Burp stores the HTTP Request and Response headers.
This will let us inspect the web application behavior carefully.



2.4.1. Burp Target Crawler



Requests

Requests are recorded. Clicking on any of them will show the Request/Response in the underlying panel.

2.4.1. Burp Target Crawler

Burp is a very powerful tool.

It lets us store all the data traveling to/from our browser, while:

- Using the crawler
- Manually navigating the website from the web browser

We will then be able to carefully inspect the web application

2.4.1. Burp Target Crawler

Your browsing of the most important parts of the target website will allow Burp to collect enough information for us to analyze.

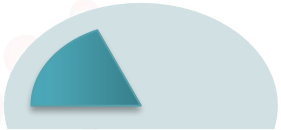
In the analysis phase, may be necessary to further test the application to draw a more clear, functional graph.

We would draw the most important parts of the entire web application.

Pentesting a complex web application is challenging as you have to keep close attention to small details while not forgetting the big picture.

The advice is to study the whole target behavior, then split the tests in smaller tasks and take care of each one.

Study The Target



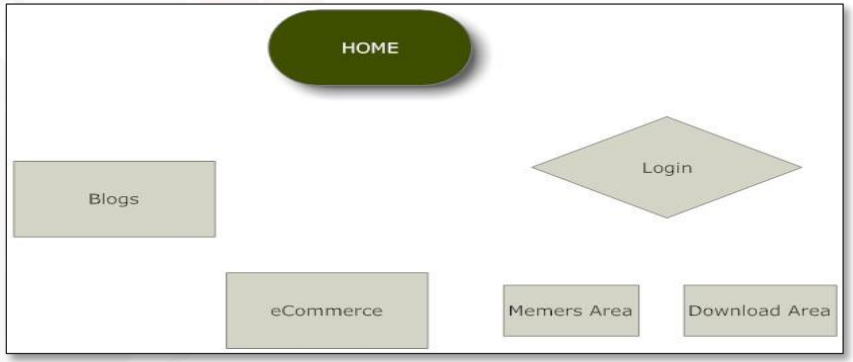
In this phase you would use the web browser to study the target under the behavioral point of view. No technical analysis is involved.

This purpose of this phase is to recognize the blocks by which the target website is made of.

The following questions should help guide you:

- What is the purpose of the website/web application?
  - Sell online?
  - Corporate online presence?
  - Blogging?
- What seems to be the core of the website?
  - Selling products?
  - Do they sell memberships? digital contents?
- Does it require a login to perform certain actions?
- What are the main areas of the website?
  - Blogs?
  - eCommerce area?

The answers to the above questions will help you illustrate the website blocks on paper.



Make sure to use different colors for the main area of the website (*eCommerce*). If you find a login protected area use a rhombus.

We have used a dark green from the HOME.

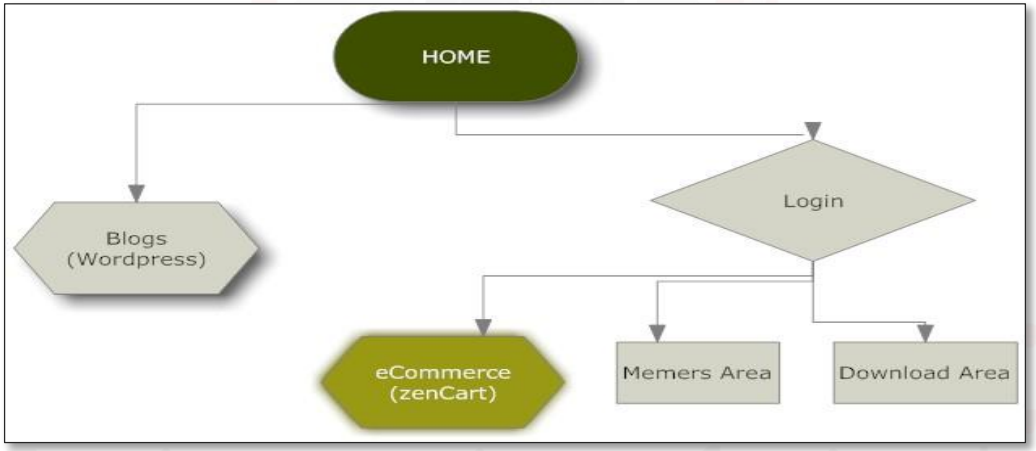
Study The Blocks

Now we will repeat our process for each block more closely.

We want to know if:

- Any block uses a third-party application (we will change the shape of the block if so and note the kind of application)
- Any block can only be accessed through the login (we will create a first path using arrows)

eCommerce is a green Hexagon because it is a third-party application and at the same time is the Core of the whole website.



Functional Graph

The goal of the functional graph is to visually spot important information at a glance.

We will use this functional graph as a basis for further charting of our information and prepare it for the testing part.

This step is vital and we recommend taking it seriously as it will allow you to concentrate your testing efforts on smaller parts instead of the entire application.

The ancient motto *divide et impera* (divide and conquer) is the most efficient solution when dealing with complex applications.

For each smaller part of the application, we will add notes and comments including (but not limited to):

- Client side logic (usually JavaScript code in use)
- flash applications
- cookies (a new cookie may be set when browsing this area of the website)
- authorization required
- forms and so on

It should be clear that our attack methodology will vary depending on the data collected in this phase and, as such, the more information we retrieve from direct inspection, the greater chance we will have in identifying exploitable vulnerabilities.

2.4.3. Mapping The Attack Surface

The attack surface is the area of the application in which we will focus all of our security testing efforts.

The more we know about our target web application, the wider the attack surface will be.

Use the suggested graphing techniques or create one you are comfortable with and stick with it.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

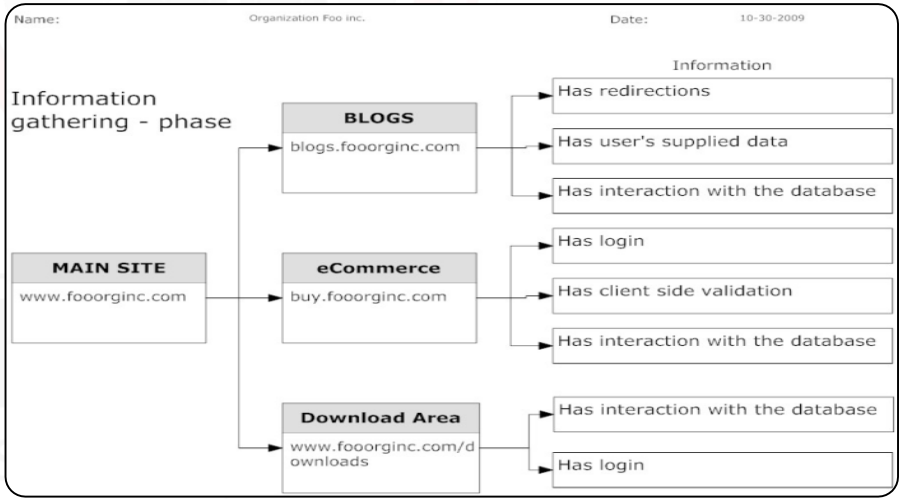
In our previous step, we dissected the application into smaller blocks, noting its most important functionalities and features.

Going deeper into this process, we will have to add more detailed information that will serve as a checklist in our testing phase.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

We will collect this information on a per block basis. A suggested graph is given in the image below. An alternative is given in the Charting tab. We will inspect it later on.



2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

Let's begin mapping the attack surface.

User submitted data through web forms can be validated on the client side, server side or both.

Recognizing where the validation occurs will allow us to manipulate the input in order to mount our input validation attacks like SQL injection, Cross site scripting or general logical flaws.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

We can reveal the presence of client side validation by inspecting the web page source code and looking at the JavaScript functions triggered upon form submittal.

We can do this different ways. Our favorite is Firebug, the very popular Firefox Add-on that lets you to inspect the web page source code easily without having to read hundred lines of code before finding the interesting information.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

Detecting database interaction will allow us to look for SQL injection vulnerabilities in the testing phase.

By database interaction, we mean that the user input changes the appearance of the page because either different data is fetched from the database or, new data is added.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

This hints that the SQL queries are generated from our input and if the input is not properly sanitized, may result in SQL injections..

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

It is important to note the pages that use the database in an active way.

While you may want to skip all the pages that do not directly connect to our input, they are in fact retrieving data from the database and therefore it make them important.

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

It is not uncommon to encounter web pages providing dynamic downloads according to a parameter provided by the user.

For example:

```
www.example.com/download.php?file=document.pdf
```

2.4.3. Mapping The Attack Surface

This kind of behavior, if not handled correctly, can lead to a number of nasty attacks including Remote and Local File Inclusion vulnerabilities. They will be explained in the next few modules.

Note: In this phase we are not interested in direct downloads like

```
www.example.com/document.pdf
```

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

File upload forms are very common in forums, social networks and CMSs. Desired files types can be anything from images, documents and even executables.

Handling these uploads is a critical task for the web developer.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

Mistakes in the way these documents are validated upon upload can lead to critical vulnerabilities, so we will make sure to note any page that offers this feature.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

This is one of the most common features in a dynamic website.

Finding displayed user supplied data will bring us to the top web application vulnerability: Cross site scripting.

We will analyze in depth on next module.

2.4.3. Mapping The Attack Surface

Redirections are server side or client side directives to automatically forward the visitor of a web page to another web page.

From the server side perspective the server can issue two different HTTP Response codes to make a redirect : 301 or 302.

The difference between the two is not important for our analysis. We will just have to remember that 3xx code stands for redirect.

From the client perspective, the redirection is handled by the web browser. It recognizes the 3xx HTTP Response code and makes a request to the page contained in the Location header.

This is a sample of a server side redirect:

Request:

```
GET /index.php?id=100
HTTP/1.1
Host: www.example.com
```

Response:

```
HTTP/1.x 301 Moved Permanently
Content-Length: 0
Content-Type: text/html
Location:
http://www.example.com/index.php?=500
```

The client browser will make a request to http://www.example.com/index.php?=500.

Another kind of refresh is the so-called meta refresh. The meta HTML tags are used to add metadata information to a web page. This data is usually read by search engines to better index the web page. Meta redirect, instead, is a way to generate a redirect either after x seconds or immediately if x=0.

Example:

```
<meta http-equiv="Refresh" content="0;
url=http://www.example.com/">
```

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

Finding redirects is an important part of our attack surface as HTTP response splitting and other Header manipulation attacks can be performed when redirects are not handled properly.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

Login pages will reveal the presence of restricted access areas of the site.

We will employ authentication bypass techniques as well as password brute forcing to test the security of the authentication routines in place.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

While at this stage, we will not intentionally cause the application to generate errors (we will see later how it can be a great source of information), we will collect all the errors we may encounter while browsing it.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

Since we are professionals, and not random amateurs firing up tools against a target, we want to keep all our information well organized.

This will let us spot the attack surface and perform our tests in a much easier and scientific manner.

2.4.3. Mapping The Attack Surface

- Client Side Validation
- Database Interaction
- File Uploading And Downloading
- Display Of User Supplied Data
- Redirections
- Access Controls And Login Protected Pages
- Error Messages
- Charting

We advise to add the information found for each block visually.

Blocks	Information					
	Client side validation	Redirections	Database Interaction	Errors	Displays users data	Login
Blogs	X	✓	✓	X	✓	X
eCommerce	✓	X	✓	X	X	✓
Downloads	X	X	✓	X	X	✓
…						
…						
…						
…						
…						

2.4.3. Mapping The Attack Surface

If you do not like the graph given at the beginning of this section, you can use the above table (or any other graph you are comfortable with).

You can also add more on the information retrieved, e.g. the URL inside the block where you encountered it.

During the process of mapping the attack surface, we have introduced two alternative charting techniques:

The tree-based chart is especially good if there are just a few blocks. Its value is in visually spotting information.

The table-based chart is what we can actually use in our testing phase, where a test for a given vulnerability will be triggered by a V in the table.

Example

A detected *interaction with a database* should be tied to a test for a SQL Injection.

An *access restricted page* may be checked against authentication bypass techniques.

This process will guarantee the best results while making sure that you do not miss any testable areas.

Enumerating Resources

2.5. Enumerating Resources

The resources we are going to enumerate in this step of our information gathering process are: subdomains, website structure, hidden files, configuration files and any additional information leaked because of misconfiguration.

This information will have to be saved, as usual, for later use (if you have not read the Methodology document already, this is the right time do so).

https://members.elearnsecurity.com/course/resources/name/ptp_v5_section_2_module_1_attachment_eLearnSecurity_Handling_Information

2.5.1. Crawling The Website

Crawling a website is the process of browsing a website in a methodical or automated manner to enumerate all the resources encountered along the way.

It gives us the structure of the website we are auditing and an overview of the potential attack surface for our future tests.

A crawler finds files and folders on a website because these appear in web page links, comments or forms.

In the section 2.4, we saw a manual, direct-browsing, crawling mechanism using Burp Proxy.

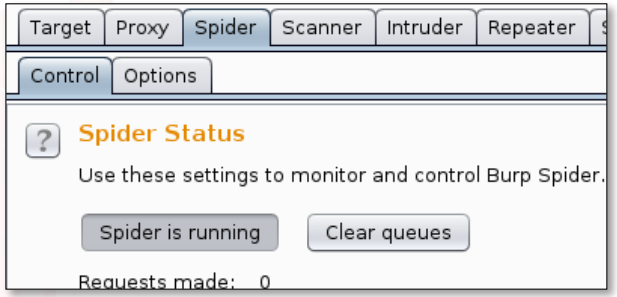
We can have Burp Proxy perform an automatic and exhaustive crawling of a website. This gives us the hierarchical website structure in the form of folders and files.

To do that, the first step is to jump to the Target tab of **Burp** and then to the Scope subtab to set up our scope.

This is an example of what should be inserted in the host/IP range field to retrieve data only from a given domain name: ^www\.domain\.com$ where domain is in our test scope.

Once the scope has been set, we will have to make sure that the proxy is activated on port 8080.
To do this, we have to open the Proxy tab, select the Options subtab and make sure that the running checkbox is activated.

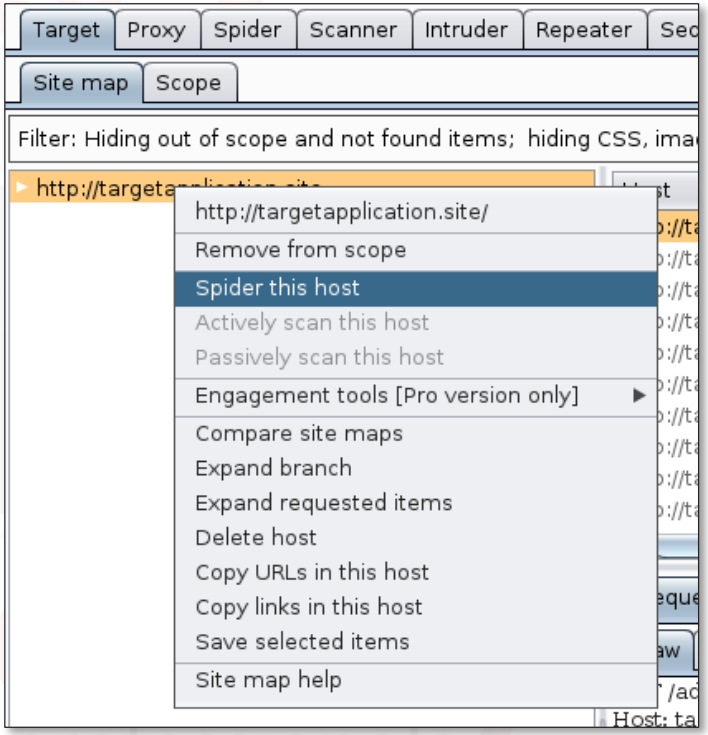
At this point, we activate the crawler by going to Spider tab and activating the spider is running checkbox.



We will set up our browser to use the proxy 127.0.0.1:8080 and navigate to the home page of the website we want to crawl. This should appear in the Target tab within Burp as seen in our previous tutorial on Burp Suite.

We have two ways to start the actual crawling process:

- right-click on the host and click spider this host
- enable the spider and start browsing the web app from our browser



2.5.1. Crawling The Website

We will be able to perform automatic form submittal, crawling pages that are reachable only through a POST request, as well as provide login data to crawl access-restricted areas of the site.

The ability to filter the data we receive makes our analysis phase easier. We will be able to see only the pages with redirects, or only pages with forms, etc.

One convenience of Burp is the built-in fuzzer and HTTP request editors in the same program. By right-clicking on any crawled web page, we will be able to send it to the Intruder - to fuzz it, or to Repeater - to manually alter the request in our tests.

Although Burp may not seem so intuitive at first, it is recommended that you get familiar with this tool. You will benefit from it in the long run.

If you have a FULL or ELITE plan you can click on the image on the left to start the video

2.5.2. Finding Hidden Files

While a crawler will only enumerate publicly available resources found through links and forms; *hidden files crawlers* and *fuzzers* like DirBuster will be able to find files that the web developer does not want us to see.

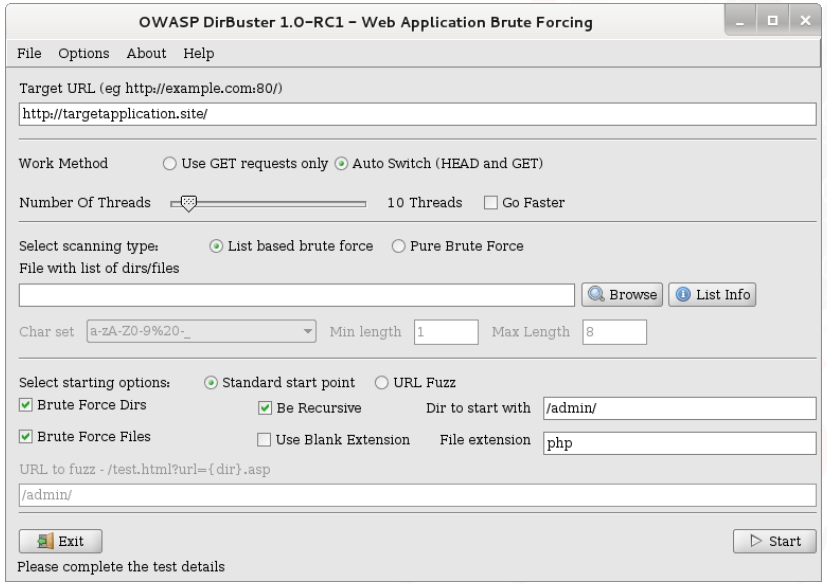
For this reason, these files are the most important source of information we can find and include: backup files, configuration files and a number of other resources that are very useful for our audit.

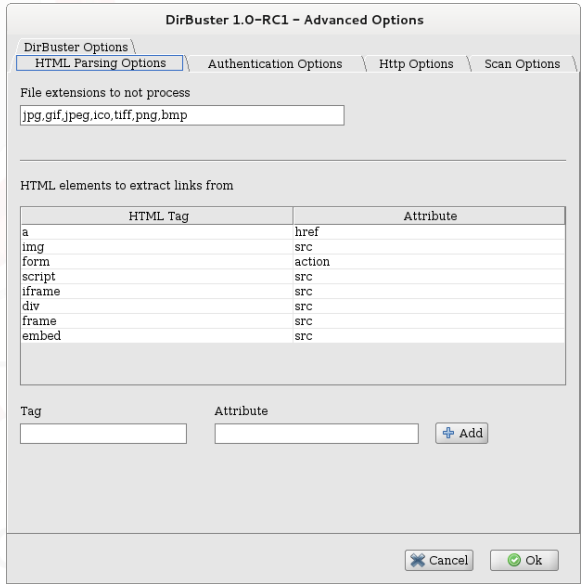
DirBuster is a mature OWASP project that allows us to crawl a website and also mount a dictionary or brute force discovery attack on the remote file system by probing each call and identifying a correct guess through the HTTP response code or web page content (in case the website uses a custom 404 page.)

The tool ships with few differently-sized dictionary lists that cover the most common folder and file names. We can choose to append an arbitrary extension to the items in the dictionary (matching the one used by our website).

For example, if the website is using PHP files, we will use .php as our extension.

You can use different settings for crawling: custom user-agent, authentication, html elements to extract links from, number of requests per second, etc.





2.5.2. Finding Hidden Files

DirBuster will present the results as a tree of folders and files.

We need to pay particular attention to files and folders that were not retrieved by Burp Proxy spider.

2.5.2.1. Back Up And Source Code File

Web developers are known to be lazy. They have to do their job well, but quickly. Their project time constraints often bring errors that can be fatal to the overall website security.

Looking for back up files and source code files left on a server is an often-overlooked step in the information gathering process that we will take seriously as it can give us not only useful information but also, sometimes, complete access to the organization's assets.

A web server is instructed to treat special file extensions as cgi files, where the source code is interpreted and not relayed to the client.
When the extension is altered for back up or maintenance purposes, we are often given the application logic and if we are lucky enough, credentials to access the databases connected to that application.

Such common, disgraced practices, involve renaming the extension in php.bak or asp.bak for example.

In Burp, we will try to probe the web server, for every file found by our crawlers in the previous steps, for the presence of back up files by appending common backup extensions like .bak or _bak or 01 and so on.

2.5.2.1. Back Up And Source Code File

A good list of back up files extension follows:

- bak
- bac
- old
- 000
- ~
- 01
- _bak
- 001
- inc
- Xxx

2.5.2.1. Back Up And Source Code File

The extension .inc stands for *include* and it has been an abused extension for a long time; in ASP 3.0 these files were used to contain source code to be included as part of the main asp page execution.

We recommend checking for their presence with DirBuster, especially when the site uses ASP as the server-side scripting engine.

If you have a FULL or ELITE plan you can click on the image on the left to start the video

2.5.3. Enumerating Users Accounts

Among the resources we can enumerate in a website, *usernames* are another important bit of information that may turn up useful information when we have to audit an authentication mechanism.

2.5.3. Enumerating Users Accounts

A badly designed system can reveal sensitive information even if wrong credentials have been inserted.

For example, a web application could reveal information about the existence of a user.

It is important to note, that while determining valid usernames is something not considered as a serious threat, usernames are half of what is needed to login.

2.5.3. Enumerating Users Accounts

While a user, at the login stage, wants to know whether the typed username or password is wrong, applications may reveal too much information. They can make an intruder's life easier by making it possible to enumerate the users.

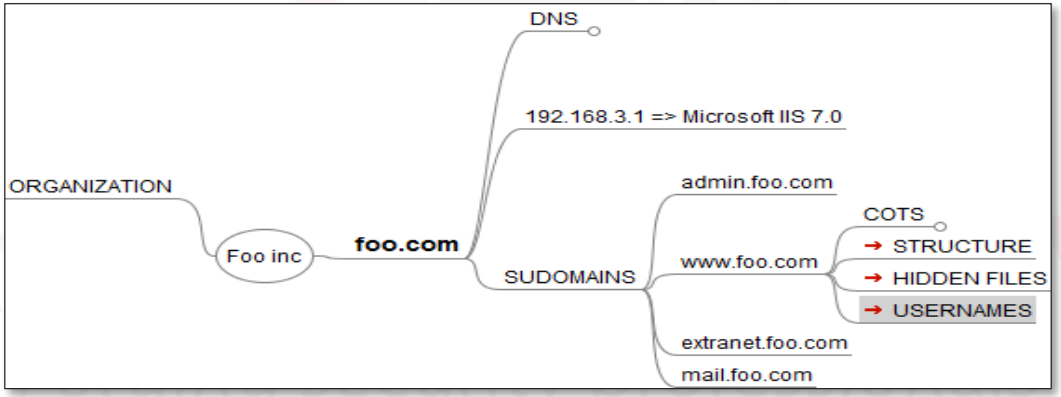
How many times have you seen the incorrect login messages like "*Login incorrect*" or "*Username blah does not exist*"?

What is different in these messages?

Depending on the application behavior we may be able to discover valid usernames.

We will see later how to use tools such as Burp Suite and Patator, to enumerate valid usernames on the target application.

Your information map nodes can be linked to the output of the tools we have just surveyed. Site directory tree, hidden files and usernames can be put in different TXT files and linked to tree nodes as the following image demonstrates:



Relevant Information Through Misconfigurations

2.6. Relevant Information Through Misconfigurations

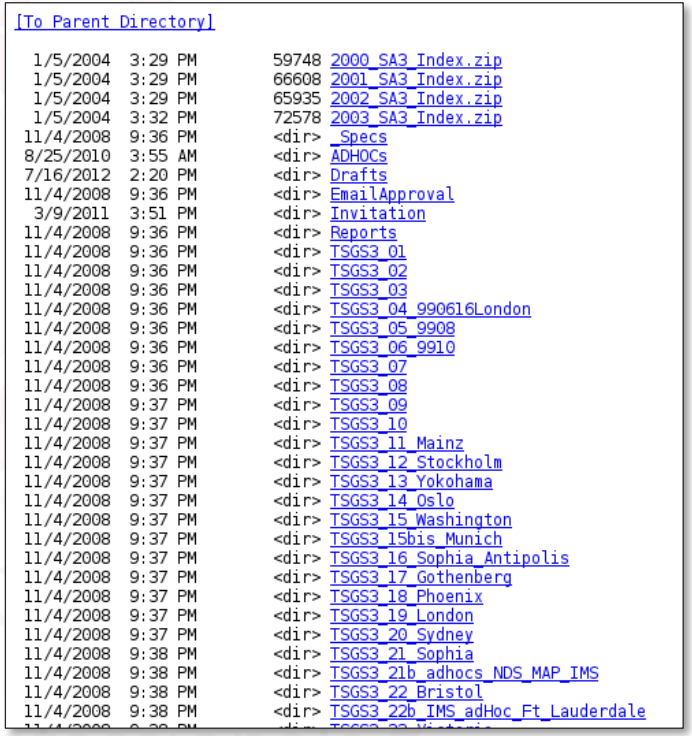
Sometimes we find that the best way to retrieve relevant information about our web applications is to look for potential mistakes in web server configuration.

A quick and very common way to gather information, files, source code and misconfigurations is by looking for open directory listings.

2.6.1. Directory Listing

These directories have been configured to show a list of the files and subdirectories in paths that we can access directly.

Note: 99% of the time, these directories have not been deliberately configured to show their content. They are just the result of a misconfiguration.



[To Parent Directory]

1/5/2004	3:29 PM	59748	2000_SA3_Index.zip
1/5/2004	3:29 PM	66608	2001_SA3_Index.zip
1/5/2004	3:29 PM	65935	2002_SA3_Index.zip
1/5/2004	3:32 PM	72578	2003_SA3_Index.zip
11/4/2008	9:36 PM	<dir>	_Specs
8/25/2010	3:55 AM	<dir>	ADHOCs
7/16/2012	2:20 PM	<dir>	Drafts
11/4/2008	9:36 PM	<dir>	EmailApproval
3/9/2011	3:51 PM	<dir>	Invitation
11/4/2008	9:36 PM	<dir>	Reports
11/4/2008	9:36 PM	<dir>	TSGS3_01
11/4/2008	9:36 PM	<dir>	TSGS3_02
11/4/2008	9:36 PM	<dir>	TSGS3_03
11/4/2008	9:36 PM	<dir>	TSGS3_04_990616London
11/4/2008	9:36 PM	<dir>	TSGS3_05_9908
11/4/2008	9:36 PM	<dir>	TSGS3_06_9910
11/4/2008	9:36 PM	<dir>	TSGS3_07
11/4/2008	9:36 PM	<dir>	TSGS3_08
11/4/2008	9:37 PM	<dir>	TSGS3_09
11/4/2008	9:37 PM	<dir>	TSGS3_10
11/4/2008	9:37 PM	<dir>	TSGS3_11_Mainz
11/4/2008	9:37 PM	<dir>	TSGS3_12_Stockholm
11/4/2008	9:37 PM	<dir>	TSGS3_13_Yokohama
11/4/2008	9:37 PM	<dir>	TSGS3_14_Oslo
11/4/2008	9:37 PM	<dir>	TSGS3_15_Washington
11/4/2008	9:37 PM	<dir>	TSGS3_15bis_Munich
11/4/2008	9:37 PM	<dir>	TSGS3_16_Sophia_Antipolis
11/4/2008	9:37 PM	<dir>	TSGS3_17_Gothenberg
11/4/2008	9:37 PM	<dir>	TSGS3_18_Phoenix
11/4/2008	9:37 PM	<dir>	TSGS3_19_London
11/4/2008	9:37 PM	<dir>	TSGS3_20_Sydney
11/4/2008	9:38 PM	<dir>	TSGS3_21_Sophia
11/4/2008	9:38 PM	<dir>	TSGS3_21b_adhocs_NDS_MAP_IMS
11/4/2008	9:38 PM	<dir>	TSGS3_22_Bristol
11/4/2008	9:38 PM	<dir>	TSGS3_22b_IMS_adHoc_Ft_Lauderdale

The previous figure shows a sample of a webserver-generated webpage showing the directory listing for the main folder.

As you can imagine, you may have access to interesting information and files that potentially contain database information, login credentials, absolute server path and so on.

Looking for directory listings is an easy task that anyone with a basic experience in scripting languages like Perl, Ruby, Python and so on, and can be automated in few minutes.

Starting with the DirBuster output, (from the previous step) which has uncovered a number of public and hidden directories on the server, let us do a GET request for each directory found. We will look at the web page content to search for patterns like *To parent directory* , *Directory Listing For*, *Index of*…

If the pattern is matched, we should be in front of a directory listing that we can navigate to using our web browser.

2.6.2. Log And Configuration Files

Logs are text files left on the web server by applications to keep track of different activities: errors, logins, informative messages and so on.

They usually contain valuable information that we will want to uncover.

Every web application has a configuration file placed somewhere in the application's folder structure.

For example, *Joomla* stores the configuration file in the application root folder with the name configuration.php.

2.6.2. Log And Configuration Files

Configuration files, as we saw in the previous paragraph, contain settings and preferences regarding the web applications installed.

They can contain the username and password that the application uses to connect to the database, or other administrative area.

This file in itself is not viewable because it has the .php extension, but we should look for backup alternatives (configuration.php.bak, configuration.php.old,…).

In the case of Joomla and other similar CMSs, the configuration file contains the username and password of the database user used to connect to the database.

Among the different mistakes an administrator can make in the configuration of the web server, leaving a directory writable by anyone is the biggest.

Writable directories are all those directories that allow us to upload our own files on the server through the PUT HTTP verb.

It is worth noting that the availability of the PUT verb among the allowed verbs does not imply that we can upload files on the server. We will be able to use the PUT verb only on writable directories.

The first step in this process is to check for the availability of the PUT verb among the allowed verbs.

To do this we will use Putty or Netcat to query the webserver directly issuing the following request:

```
OPTIONS / HTTP/1.1
Host: targetsite.com
```

The server will respond with the list of the available verbs.

```
OPTIONS / HTTP/1.1
Host: test.com

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Sat, 09 May 2009 17:01:57 GMT
X-Powered-By: ASP.NET
MS-Author-Via: DAV
Content-Length: 0
Accept-Ranges: none
DASL: <DAV:sql>
DAV: 1, 2
Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE, MKCOL, PROPFIN
D, PROPPATCH, LOCK, UNLOCK, SEARCH
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, COPY, MOVE, PROPFIND, PROPPATCH, SEARC
H, MKCOL, LOCK, UNLOCK
Cache-Control: private
```

We are lucky! The server allows a number of verbs.

Note: this was an example that we made up for educational purposes. Usually you will find out that only GET, POST, HEAD and TRACE verbs are enabled.

Another thing to note is that if the OPTIONS verb is not allowed, we will not be able to enumerate all the available verbs. This does not mean that PUT is not available. We will just have to test it directly.

In case OPTIONS verb is not allowed we will receive either a 4xx or 5xx error message according to the webserver.

```
OPTIONS / HTTP/1.1
Host: www.google.com

HTTP/1.1 400 Bad Request
Content-Type: text/html; charset=UTF-8
Content-Length: 1350
Date: Sat, 09 May 2009 17:11:54 GMT
Server: GFE/2.0
```

This is just the beginning. We need now to know what directory, if any, we can upload to.

It is important to understand the correlation between the directory privileges and the possibility of uploading files: if the server's local user with which the website is executed also has the write attribute enabled for a given folder, then we will be able to write to that folder.

In IIS, every configured website can be run by different local users. These users are assigned to the website visitor in order for him to browse the website.

If the user IUSR_test is set as the anonymous account for test.com then all the directories writable by IUSR_test will be our target (because we are indeed using IUSR_test's privileges to browse the web site).

2.6.3. HTTP Verbs and File Upload

Looking for writable directories is guess work. There is not a straight forward method how to verify directory privileges remotely.

2.6.3. HTTP Verbs and File Upload

Sometimes though, we can study the application and understand what directories are used to store user submitted avatars, files, attachments…

```
PUT /writable_folder/test.html HTTP/1.1
Host: test.com
Content-length: 184

HTTP/1.1 100 Continue
Server: Microsoft-IIS/5.1
Date: Sun, 10 May 2009 08:36:40 GMT
X-Powered-By: ASP.NET

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE> Uploaded file </TITLE>
 </HEAD>

 <BODY>
  <h1>Test uploaded file</h1>
 </BODY>
</HTML>
HTTP/1.1 201 Created
Server: Microsoft-IIS/5.1
Date: Sun, 10 May 2009 08:36:43 GMT
X-Powered-By: ASP.NET
Location: http://test.com/writable_folder/test.html
Content-Length: 0
Allow: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, COPY, MOVE, PROPFIND, PROPPATCH,
SEARCH, LOCK, UNLOCK
```

To do this, we will look for the path that brings us to these files.
For example, in a forum, we can find the path to a user's submitted avatar. This is useful because it is as folder configured to store these files.

Once we have a pool of candidate folders we will try our PUT command on them:

```
PUT /writable_dir/test.html HTTP/1.1
Content-length: 184
<RETURN>

[CONTENT OF TEST.HTML]
<RETURN>
<RETURN>
```

If the upload is successful, the server will respond with a 201 Created.

It is important to provide the Content-length of the payload that we want to upload in the file specified as an argument of the PUT. The actual file content is in the request payload.

To make sure that our file has been successfully uploaded to the server, we will look for it with our favorite browser.

Google Hacking

When we talk about Google hacking, we mean using Google's sophisticated search operators for our information gathering purposes.

Johnny Long has been one of the first to uncover using Google to find misconfigured web servers, sensitive information left on a server (that was crawled by Google bots), password files, log files, directory listings and many others.

His Google Hacking Database contains a list of Google searches for every purpose.

Here is an example:

Fingerprinting web servers is possible through Google by querying for Apache online documentation or special folder added by IIS to the web root.

https://www.exploit-db.com/google-hacking-database/

Search term:

```
intitle:"Apache HTTP Server" intitle:"documentation"
```

The operator intitle will search only the title tag of all the pages crawled by Google. We can be more specific and restrict our search to include only our scope of audit:

Search term:

```
intitle:"Apache HTTP Server" intitle:"documentation" site:target.com
```

Looking for open directory listings containing .bak files is another easy task with Google:

Search term:

```
"Index of" bak
```

Search term:

```
"Directory listing for" bak
```

Looking for files with a specific extension is as easy as:

```
filetype:"bak"
```

or

```
filetype:"inc"
```

To restrict the search only to the website in our scope add site:target.com.

```
filetype:"inc" site:target.com
```

Through Google Hacking, we may able to detect:

- Error messages that may contain valuable information
- Sensitive files and directories
  - Passwords, usernames, configurations, etc.
- Server or application vulnerabilities
- Pages that contain login portals
- and much more

For a full list of available Google Search Operators please refer to:

http://www.googleguide.com/advanced_operators.html

Google Hacking Database is available here:

https://www.exploit-db.com/google-hacking-database/

Shodan HQ

Similarly to Google Hacking, there is another great search engine that will be very useful for our information gathering process.

It is Shodan HQ. Shodan is a computer search engine that uses a different approach from other search engines like Google, Yahoo, Bing, etc. Indeed, instead of crawling data in web pages, Shodan scans the entire Internet and interrogates ports in order to gather information from the banners.

https://www.shodan.io/

Shodan searches includes the following protocols

- HTTP(S)
- SSH
- SNMP
- MySQL / MondoDB
- RDP
- FTP
- Telnet
- and few more

We can use it to search for:

- devices with default username and password,
- viewing the configuration of a device,
- detect server versions, and much more.

Like other search engines, it has Boolean operators and filters that can be used to narrow down the results:

- **[before/after] day/month/year**: search only for data that was collected before or after the given date
- **hostname**: filters results for hosts that contain the value in their hostname
- **port**: narrow the search for specific services
- **OS**: search for specific operating system

These are just a few of the available filters.

https://www.shodan.io/

It is important to know that in order to use most of its features (such as exporting results) and filters, you need to create an account.

Let’s look at some search examples. Suppose we want to find all the devices running apache and that are in Italy (IT).

Our search query will be something like this:

```
apache country:IT
```

Services

HTTP	5,211
HTTPS	2,252
HTTP Alternate	2,017
HTTPS Alternate	83
HTTP	65

Top Cities

Rome	500
Milan	374
Florence	127
Torino	119
Genova	92

151.32.117.212
Linux 3.x
WIND Telecomunicazioni S.p.A
Added on 06.11.2013
Torino
Details

ppp-212-117.32-151.iol.it

```
HTTP/1.0 200 OK
Date: Wed, 06 Nov 2013 02:12:30 GMT
Server: Apache/2.2.22 (Ubuntu)
Last-Modified: Mon, 16 Sep 2013 15:41:49 GMT
ETag: "361a26-b1-4e6820ae2ed8f"
Accept-Ranges: bytes
Content-Length: 177
Vary: Accept-Encoding
Content-Type: text/html
X-Pad: avoid browser bug
```

Elastix - Pagina di Connessione
93.63.209.104
Fastweb
Added on 06.11.2013
Rome
Details

```
HTTP/1.0 200 OK
Date: Wed, 06 Nov 2013 02:07:16 GMT
Server: Apache/2.2.3 (CentOS)
X-Powered-By: PHP/5.1.6
```

As we can see, we have a good list of matching devices:

If you need more detailed information about the host, you can click on Details and then the following page will appear:

Of course you can refine your searches by filtering results for specific hostname, ports and so on.

We suggest you try it yourself in order to understand the true power of this tool.

References

References

Sysinternal Whois

https://docs.microsoft.com/en-us/sysinternals/downloads/whois

whois.domaintools.com

http://whois.domaintools.com/

Arin.net

http://whois.arin.net/rest/net/NET-108-162-192-0-1/pft

ripe.net

https://apps.db.ripe.net/db-web-ui/#/query

Netcraft

https://www.netcraft.com/

IIS

https://www.iis.net/

WhatWeb

https://github.com/urbanadventurer/WhatWeb

Wappalyzer

https://wappalyzer.com/

References

ModSecurity

https://www.modsecurity.org/

OWASP WebApp Fingerprinting

https://www.owasp.org/index.php/Testing_for_Web_Application_Fingerprint_(OWASP-IG-004)

OWASP Framework Fingerprinting

https://www.owasp.org/index.php/Fingerprint_Web_Application_Framework_(OTG-INFO-008)

Google Search Operators

http://www.googleguide.com/advanced_operators_reference.html

dnsrecon

https://github.com/darkoperator/dnsrecon

subbrute

https://github.com/TheRook/subbrute

fierce

https://github.com/davidpepper/fierce-domain-scanner

Nmap

https://nmap.org/book/man-host-discovery.html

References

dnsenum

https://code.google.com/archive/p/dnsenum/downloads

knock

https://github.com/guelfoweb/knock

theHarvester

https://github.com/laramies/theHarvester

recon-ng

https://bitbucket.org/LaNMaSteR53/recon-ng/wiki/Usage%20Guide

Zone Transfer

https://en.wikipedia.org/wiki/DNS_zone_transfer

HotScripts

http://www.webhostingtalk.com/

Burp Suite

https://portswigger.net/burp

eLearnSecurity Methodology Document

https://members.elearnsecurity.com/course/resources/name/ptp_v5_section_2_module_1_attachment_eLearnSecurity_Handling_Information

References

Google Hacking Database

https://www.exploit-db.com/google-hacking-database/

Google Advanced Operators

http://www.googleguide.com/advanced_operators_reference.html

ShodanHQ

https://www.shodan.io/

Shodan Filters

https://www.shodan.io/

Web App Information Gathering

Subdomain Enumeration

Web app Fingerprinting

Crawling and Spidering

Dirbuster

Information Gathering

Fingerprint the web application, find hidden files, directories and much more