

Information Gathering



Instructor: C. Pu (Ph.D., Assistant Professor)

Lecture 03

puc@marshall.edu



Introduction to Intelligence Gathering

- Information gathered about a target can help refine the steps that will come later.
- During this process, you should seek to use as many methods as reasonable to observe and collect information about your target.
- You should be paying special attention to anything that may have the potential to be exploited later
 - Though it will take some experience to develop an eye for what is useful and what is not.
- Eventually you should be able to pick out items that have the potential to be helpful later in the pen testing process.
- Until you develop your “eye” and are able to detect the useful information carefully, examine what information you are uncovering and the details that are included.



Losing Control of Information

- From the client's standpoint, there can be several negative results from the gathering of intelligence in regard to their infrastructure and business operations:
- **Business Loss**
 - If customers or vendors discover that their information or other data is not properly secured, it could easily erode their confidence and cause them to go elsewhere.
- **Information Leakage**
 - This includes information that either deliberately or accidentally is made public, such as project information, employee data, personal details, financial information, or any of a number of possibilities.



Losing Control of Information

- Privacy Loss
 - This is a particularly bad situation where information that is supposed to be kept confidential is disclosed.
 - The biggest threat with this is not just a loss of confidence, but the legal repercussions that can result.
- Corporate Espionage
 - Information that is uncovered through the footprinting process can also be uncovered by well-financed and curious competitors looking for details about what a company is doing.



Categorizing the Types of Information

- Generally, when investigating a client, you are seeking to collect as much information as possible from a multitude of different sources.
- You can expect to find a lot of information about a target, including
 - Technical information
 - Operating system information, network information, applications present, IP address ranges, and even device information
 - Additionally, you can expect to be able to locate webcams, alarm systems, mobile devices, and much more



Categorizing the Types of Information

- Generally, when investigating a client, you are seeking to collect as much information as possible from a multitude of different sources.
- You can expect to find a lot of information about a target, including
 - Administrative information
 - Organizational structure, corporate policies, hiring procedures, employee details, phone directories, vendor information, and much more.



Categorizing the Types of Information

- Generally, when investigating a client you are seeking to collect as much information as possible from a multitude of different sources.
- You can expect to find a lot of information about a target, including
 - Physical details
 - Location data, facility data, people details, and social interactions with individuals, to name a few.
 - Expect to be able to view location details of a facility through simple surveillance or by using resources such as Google Street View to gain an understanding of the layout of an area.



Categorizing the Types of Information

- Within these categories, there exists a tremendous amount of information to be unearthed.
- The question is how much of it is useful and how much could you be overlooking.
- In fact, be prepared to experience something known as “information overload,” which is where you become overwhelmed by the amount of data being collected to the point where it cannot be processed effectively (if at all).



Categorizing the Types of Information

- Remember that too much information can be a dangerous thing.
- It is easy to become so enamored by what is being revealed that you end up gathering information that may not even be useful.
- Learn from your intelligence gathering and from the experience you gain from later stages which information is most useful and which may be less so.



Categorizing the Gathering Methods

- During the information-gathering phase, you should be able to formulate an attack strategy as well as gain an understanding of what information an organization releases.
- Information gathering typically falls into three categories.
 - Passive
 - Passive methods are those that do not interact with or engage the target.
 - By not engaging the target, the hope is that they are given little or no indication of your impending attack.



Categorizing the Gathering Methods

- During the information-gathering phase, you should be able to formulate an attack strategy as well as gain an understanding of what information an organization releases.
- Information gathering typically falls into three categories.
 - Active
 - Methods that fall into this category are making phone calls to the company, help desk, employees, or other personnel.
 - Anything that requires you to actively engage the target would fit into this category.



Categorizing the Gathering Methods

- During the information-gathering phase, you should be able to formulate an attack strategy as well as gain an understanding of what information an organization releases.
- Information gathering typically falls into three categories.
 - Open Source Intelligence Gathering
 - As far as intelligence gathering goes, open source or passive information gathering is the least aggressive.
 - Basically, the process relies on obtaining information from those sources that are typically publicly available and out in the open.
 - Potential sources include newspapers, websites, discussion groups, press releases, television, social networking, blogs, and innumerable other sources.



Testing for Information Gathering

- Understanding the deployed configuration of the server hosting the web application is almost as important as the application security testing itself.
- After all, an application chain is only as strong as its weakest link.
- Application platforms are wide and varied, but some key platform configuration errors can compromise the application in the same way an unsecured application can compromise the server.



Conduct Search Engine Discovery for Information Leakage: Summary

- There are direct and indirect elements to search engine discovery.
 - Direct methods relate to searching the indexes and the associated content from caches.
 - Indirect methods relate to gleaning sensitive design and configuration information by searching forums, newsgroups, and tendering websites.



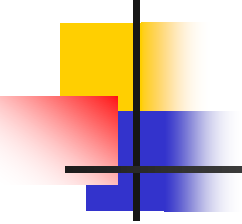
Conduct Search Engine Discovery for Information Leakage: Test Objectives

- To understand what sensitive design and configuration information of the application/system/organization is exposed both directly (on the organization's website) or indirectly (on a third party website).



Conduct Search Engine Discovery for Information Leakage: How to Test

- Use a search engine to search for:
 - Network diagrams and configurations
 - Archived posts and emails by administrators and other key staff
 - Log on procedures and username formats
 - Usernames and passwords
 - Error message content
 - Development, test, UAT and staging versions of the website

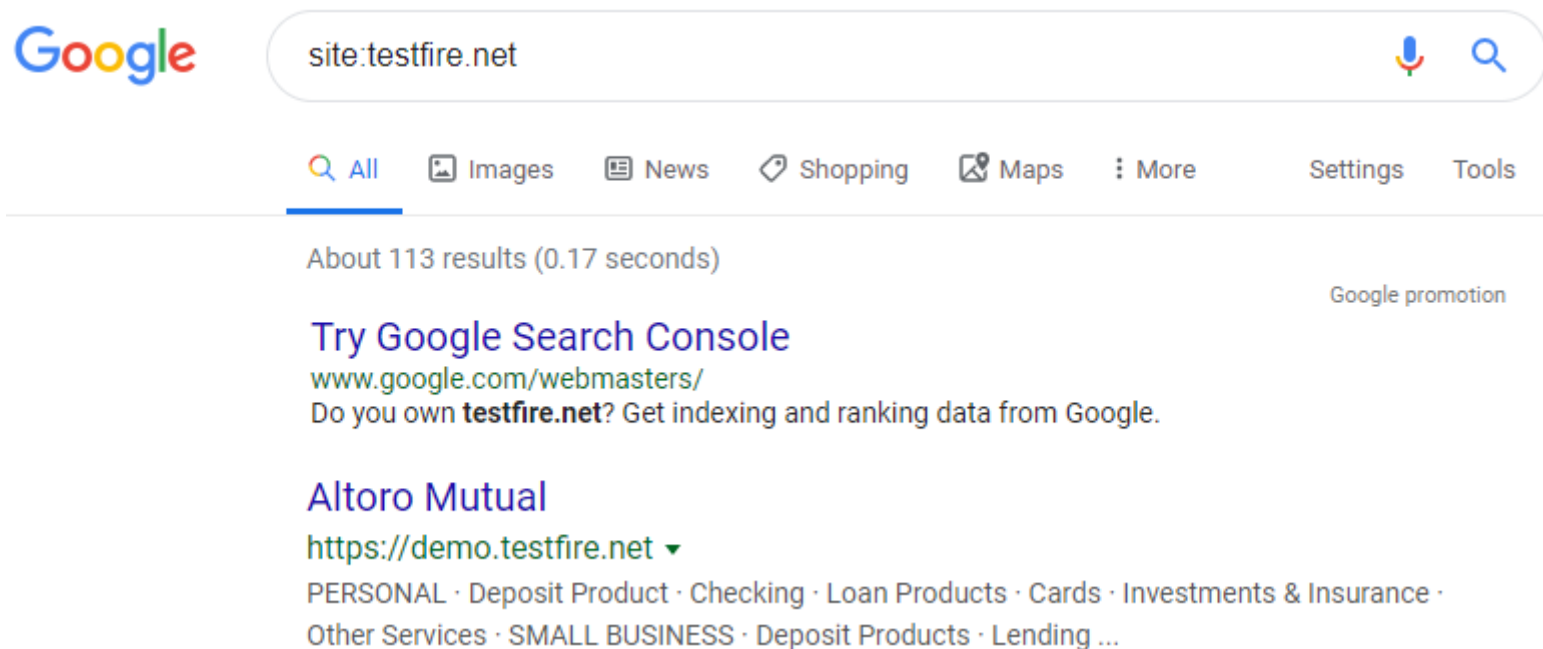


Conduct Search Engine Discovery for Information Leakage: Search Operators

- Using the advanced “site:” search operator, it is possible to restrict search results to a specific domain
- Do not limit testing to just one search engine provider as they may generate different results depending on when they crawled content and their own algorithms.
- Consider using the following search engines:
 - Baidu
 - binsearch.info
 - Bing
 - Duck Duck Go
 - Startpage
 - Google
 - Shodan
 - PunkSpider

Conduct Search Engine Discovery for Information Leakage: Example

- To find the web content of testfire.net indexed by a typical search engine, the syntax required is:
site:testfire.net



Conduct Search Engine Discovery for Information Leakage: Example

- To display the index.html of testfire.net as cached, the syntax is:
cache:testfire.net

A search bar with the text "cache:testfire.net" entered. To the right of the text is a small "X" icon and a microphone icon.

Google Search

I'm Feeling Lucky

This is Google's cache of <https://demo.testfire.net/>. It is a snapshot of the page as it appeared on Dec 20, 2019 11:08:47 GMT. The [current page](#) could have changed in the meantime. [Learn more.](#)

[Full version](#) [Text-only version](#) [View source](#)

Tip: To quickly find your search term on this page, press Ctrl+F or ⌘-F (Mac) and use the find bar.



Fingerprint Web Server: Summary

- Web server fingerprinting is a critical task for the penetration tester.
- Knowing the version and type of a running web server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing.
- There are several different vendors and versions of web servers on the market today.
- Knowing the type of web server that is being tested significantly helps in the testing process and can also change the course of the test.



Fingerprint Web Server: Summary

- This information can be derived by sending the web server specific commands and analyzing the output, as each version of web server software may respond differently to these commands.
- By knowing how each type of web server responds to specific commands and keeping this information in a web server fingerprint database, a penetration tester can send these commands to the web server, analyze the response, and compare it to the database of known signatures.



Fingerprint Web Server: Test Objectives

- Find the version and type of a running web server to determine known vulnerabilities and the appropriate exploits to use during testing.



Fingerprint Web Server: How to Test

- The simplest and most basic form of identifying a web server is to look at the Server field in the HTTP response header.
- Netcat
 - Netcat is a featured networking utility which reads and writes data across network connections, using the TCP/IP protocol.
 - It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs and scripts.
 - At the same time, it is a feature-rich network debugging and exploration tool, since it can create almost any kind of connection you would need and has several interesting built-in capabilities.



Fingerprint Web Server: Netcat

- We will use the following syntax for nc command.

netcat options destination port

- options used to set some special behavior like timeout, help, etc.
- destination is used to specify remote system IP or Hostname
- port is the remote system port number



Fingerprint Web Server: Netcat

- Help

`netcat -h`

- netcat command provides a lot of different options



Fingerprint Web Server: Netcat

- Port Scan

```
netcat -z -v destination port#_range
```

- -z option: zero-I/O mode
- -v option: detailed information



Fingerprint Web Server: Netcat Tutorial

- If you are new to netcat, please find some netcat tutorials to study.
- For your reference, I upload one netcat tutorial on Blackboard.
- You also can find some other netcat tutorials online for study.
 - All netcat tutorials are very similar.



Fingerprint Web Server: Example

- Look at the Server field in the HTTP response header to identify a web server

```
root@kali:~# nc testfire.net 80
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=9D783EDD4942C2463C18CAF1092DABE8; Path=/; HttpOnly
Content-Type: text/html; charset=ISO-8859-1
Date: Sat, 21 Dec 2019 21:00:52 GMT
Connection: close

root@kali:~#
```



Fingerprint Web Server: Example

- Look at the Server field in the HTTP response header to identify a web server

```
root@kali:~# nc testfire.net 80
GET / HTTP/3.0

HTTP/1.1 505 HTTP Version Not Supported
Server: Apache-Coyote/1.1
Date: Sat, 21 Dec 2019 21:03:07 GMT
Connection: close

root@kali:~#
```



Fingerprint Web Server: Example

- Look at the Server field in the HTTP response header to identify a web server

```
root@kali:~# nc testfire.net 80
GET / HTTP/1.0

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=8C66E7204E09212B5BF9D6A5CDA4C476; Path=/; HttpOnly
Content-Type: text/html; charset=ISO-8859-1
Date: Sat, 21 Dec 2019 21:04:03 GMT
Connection: close

<!-- BEGIN HEADER -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/
DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" >
```



Review Webserver Metafiles for Information Leakage: Summary

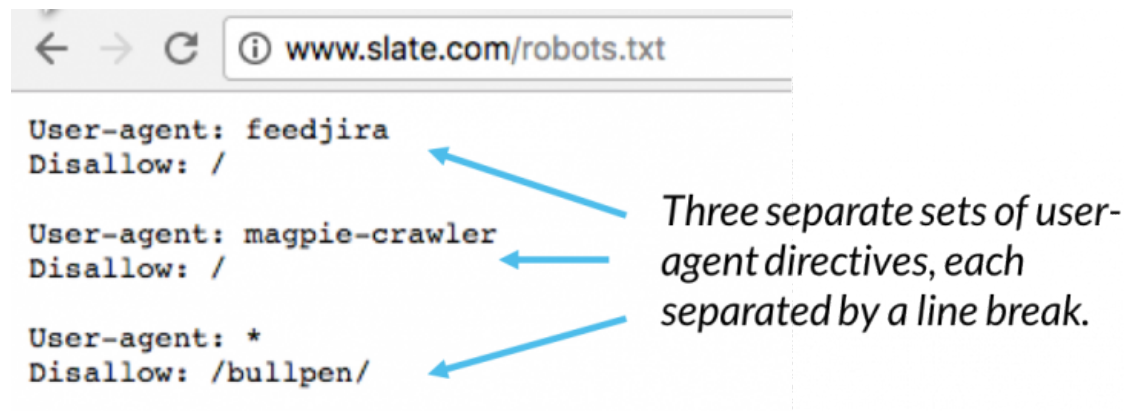
- Test the robots.txt file for information leakage of the web application's directory or folder path(s).

Basic format:

```
User-agent: [user-agent name]Disallow: [URL string not to be crawled]
```

Review Webserver Metafiles for Information Leakage: Summary

- Test the robots.txt file for information leakage of the web application's directory or folder path(s).



```
← → ↻ ⓘ www.slate.com/robots.txt

User-agent: feedjira
Disallow: /


User-agent: magpie-crawler
Disallow: /

User-agent: *
Disallow: /bullpen/
```

Three separate sets of user-agent directives, each separated by a line break.

Review Webserver Metafiles for Information Leakage: Summary

- Test the robots.txt file for information leakage of the web application's directory or folder path(s).

← → ↻  Secure <https://www.buzzfeed.com/robots.txt>

```
User-agent: msnbot
Crawl-delay: 120
Disallow: /*.xml$
Disallow: /buzz/*.xml$
Disallow: /category/*.xml$
Disallow: /mobile/
Disallow: *?s=mobile
Disallow: *?s=lightbox
Disallow: /bfmp/
Disallow: /buzzfeed/
Disallow: /contest
Disallow: /contests
Disallow: /plugin/
Disallow: /embed/
Disallow: /_comments/
```

Buzzfeed.com wants msnbot to wait 120 msc before crawling each page and NOT crawl any of these URL strings.

AND

```
User-agent: *
Disallow: /buzz/*.xml$
Disallow: /category/*.xml$
Disallow: /mobile/
Disallow: *?s=lightbox
Disallow: /bfmp/
Disallow: /buzzfeed/
Disallow: /contest
Disallow: /contests
Disallow: /_ga/
Disallow: /static/
Disallow: /dashboard/
Disallow: /plugin/
Disallow: /api/
Disallow: /buzzfeed/api/
Disallow: /embed/
Disallow: /_comments/
```

Buzzfeed.com wants all other user-agents (except for msnbot, discobot, and Slurp) to NOT crawl any of these URL strings

AND

```
User-agent: discobot
Disallow: /
```

Discobot should not crawl ANY URLs on buzzfeed.com.

AND

```
User-agent: Slurp
Crawl-delay: 4
```

Slurp (Yahoo's user-agent) should wait 4 msc before crawling each page, but crawl all URLs on buzzfeed.com.



Review Webserver Metafiles for Information Leakage: Test Objectives

- Information leakage of the web application's directory or folder path(s).



Review Webserver Metafiles for Information Leakage: How to Test

- Web Spiders, Robots, or Crawlers retrieve a web page and then recursively traverse hyperlinks to retrieve further web content.
- Their accepted behavior is specified by the Robots Exclusion Protocol of the robots.txt file in the web root directory.
- As an example, the beginning of the robots.txt file from <https://www.google.com/robots.txt> sampled on 21 Dec 2019 is quoted below:

```
User-agent: *  
Disallow: /search  
Allow: /search/about  
Allow: /search/static  
Allow: /search/howsearchworks
```



Review Webserver Metafiles for Information Leakage: How to Test

- The robots.txt file is retrieved from the web root directory of the web server.
- For example, to retrieve the robots.txt from www.google.com using wget:

```
wget https://www.google.com/robots.txt
```

- Open robots.txt

```
head -n5 robots.txt
```

Review Webserver Metafiles for Information Leakage: How to Test

- The robots.txt file is retrieved from the web root directory of the web server.
- For example, to retrieve the robots.txt from www.google.com using wget:

```
root@kali:~# wget https://www.google.com/robots.txt
--2019-12-21 16:08:43-- https://www.google.com/robots.txt
Resolving www.google.com (www.google.com)... 172.217.7.132, 2607:f8b0:4004:802::2004
Connecting to www.google.com (www.google.com)|172.217.7.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: 'robots.txt'

robots.txt           [  =>  ]  6.84K  --.-KB/s  in 0s

2019-12-21 16:08:43 (14.8 MB/s) - 'robots.txt' saved [7004]

root@kali:~# head -n5 robots.txt
User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
root@kali:~#
```



Enumerate Applications on Webserver: Summary

- A paramount step in testing for web application vulnerabilities is to find out which particular applications are hosted on a web server.
- Many applications have known vulnerabilities and known attack strategies that can be exploited in order to gain remote control or to exploit data.
- In addition, many applications are often misconfigured or not updated, due to the perception that they are only used “internally” and therefore no threat exists.



Enumerate Applications on Webserver: Test Objectives

- Enumerate the applications within scope that exist on a web server



Enumerate Applications on Webserver: How to Test

- While web applications usually live on port 80 (http) and 443 (https), there is nothing magic about these port numbers.
- In fact, web applications may be associated with arbitrary TCP ports, and can be referenced by specifying the port number as follows: `http[s]://www.example.com:port/`.
 - For example, `http://www.example.com:20000/`.



Enumerate Applications on Webserver: How to Test

- It is easy to check for the existence of web applications on non-standard ports.
- A port scanner such as nmap is capable of performing service recognition by means of the -sV option, and will identify http[s] services on arbitrary ports.

Enumerate Applications on Webserver: Example

- It is easy to check for the existence of web applications on non-standard ports.
- A port scanner such as nmap is capable of performing service recognition by means of the -sV option, and will identify http[s] services on arbitrary ports.

```
root@kali:~# nmap -sV testfire.net
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-21 16:20 EST
Nmap scan report for testfire.net (65.61.137.117)
Host is up (0.015s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Apache Tomcat/Coyote JSP engine 1.1
443/tcp   open  ssl/https?
8080/tcp   open  http         Apache Tomcat/Coyote JSP engine 1.1

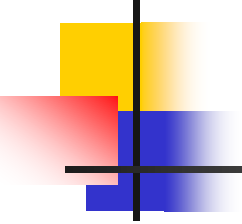
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 73.74 seconds
root@kali:~#
```

Review Webpage Comments and Metadata for Information Leakage: Summary



- It is very common, and even recommended, for programmers to include detailed comments and metadata on their source code.
- However, comments and metadata included into the HTML code might reveal internal information that should not be available to potential attackers.
- Comments and metadata review should be done in order to determine if any information is being leaked.

Review Webpage Comments and Metadata for Information Leakage: Test Objectives



- Review webpage comments and metadata to better understand the application and to find any information leakage.

Review Webpage Comments and Metadata for Information Leakage: How to Test



- HTML comments are often used by the developers to include debugging information about the application.
- Sometimes they forget about the comments and they leave them on in production.
- Testers should look for HTML comments which start with “”.

Review Webpage Comments and Metadata for Information Leakage: Example

- Check HTML source code for comments containing sensitive information that can help the attacker gain more insight about the application.
- It might be SQL code, usernames and passwords, internal IP addresses, or debugging information.

```
...  
<div class="table2">  
  <div class="col1">1</div><div class="col2">Mary</div>  
  <div class="col1">2</div><div class="col2">Peter</div>  
  <div class="col1">3</div><div class="col2">Joe</div>  
  
<!-- Query: SELECT id, name FROM app.users WHERE active='1'  
-->  
  
</div>  
...
```

```
<!-- Use the DB administrator password for testing: f@keP@a$$w0rD -->
```

```
<META name="Author" content="Andrew Muller">
```