B | EHC

Ethical Hacking Class

3: Footprinting or Information Gathering(Part-2)

i

by Lokesh Singh

www.hackingloops.com

CONTENTS

OVERVIEW OF LAST BEH ISSUE: FOOTPRINTING OR INFORMATION GATHERING TECHNIQUES

In our last issue we have started learning about preparatory phase of any hacking attempt i.e. Information gathering or Footprinting. Let’s have a brief overview what we have covered in our last BEHC issue. We have started with the introduction part of Footprinting or information gathering and then we have covered several Footprinting and information gathering techniques namely how to get an IP address of victim, different techniques to steal IP address, Ping sweep, Flood Ping DDOS attack, Trace route, WHOIS information gathering, extracting history details of any domain, owner contact information extraction, DNS queries and DNS health check to discover domain level bugs. This was all what we covered in our last issue. In this issue we will continue learning about other information gathering techniques. The techniques that we learn in this issue are mentioned below:

1. Search Engine discovery or Reconnaissance
2. Spiders, Crawlers or Robots discovery
3. Web data Extraction
4. Reviewing Metadata and JavaScript’s
5. Web application fingerprint
6. Web server fingerprint
7. People Search

So without wasting much time let’s continue our learning.

SEARCH ENGINE DISCOVERY

As we all know search engine is the best friend of Hackers. How so? Either its personal information search or searching for details about organization or simply searching for vulnerable websites or more technically extracting our victims, search engine is the best friend. Search engine discovery consists of some technical terms like dorks, Google operators, Google indexing dumps and much more. Let’s learn all these things one by one.

Google Operators: Most of us might have tried these but for sure they are unaware of its real strength i.e. up to which extent these operators can be used. For newbie’s, Google has provided a list of Google operators i.e. nothing just filters, so that we can minimize or maximize or better call manipulate our searching scope on

Google. For Example: If I want to search something specific on Hackingloops like Hacking Email account. Then what we write in Google search box:

Site:www.hackingloops.com Hacking Email account

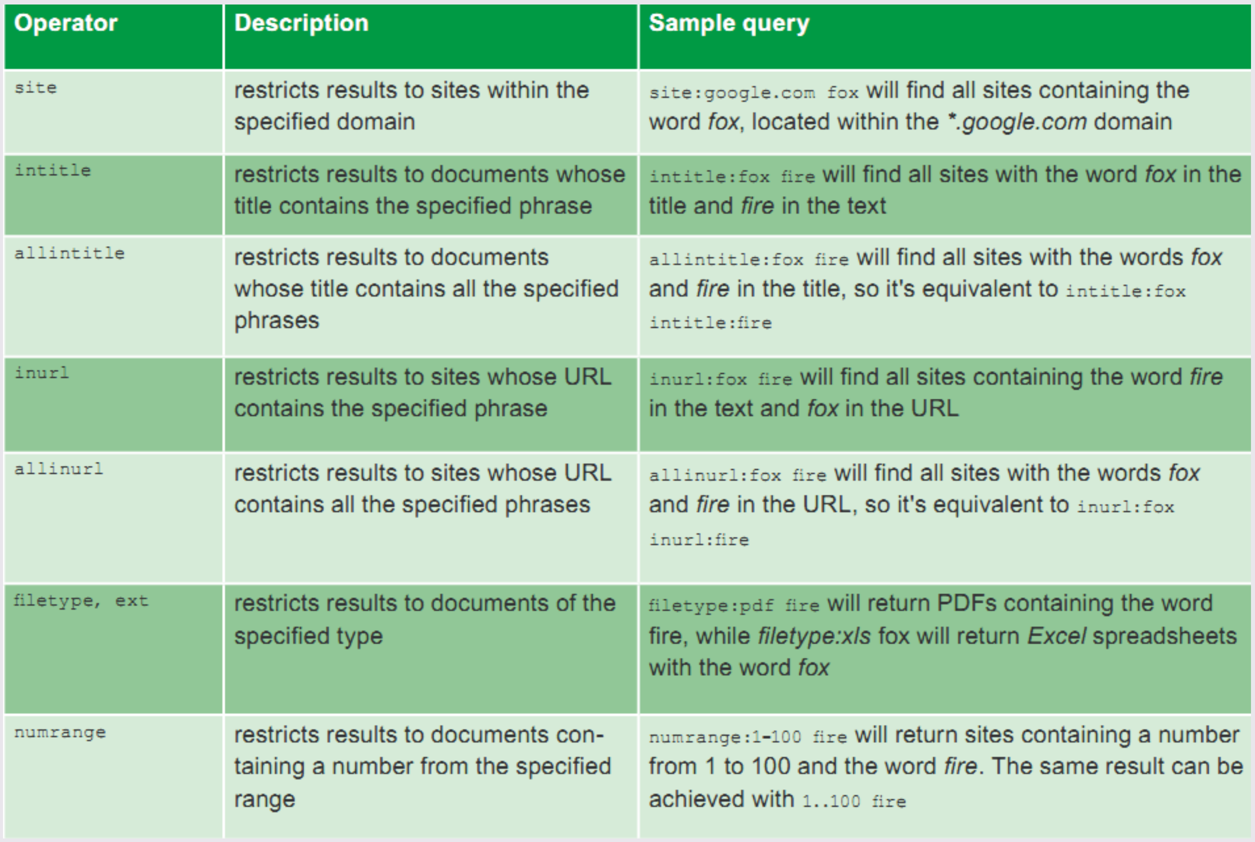
Now I want to filter only those results which contain email in the URL. So the search query will become:

Site:www.hackingloops.com inurl:email Hacking Email account

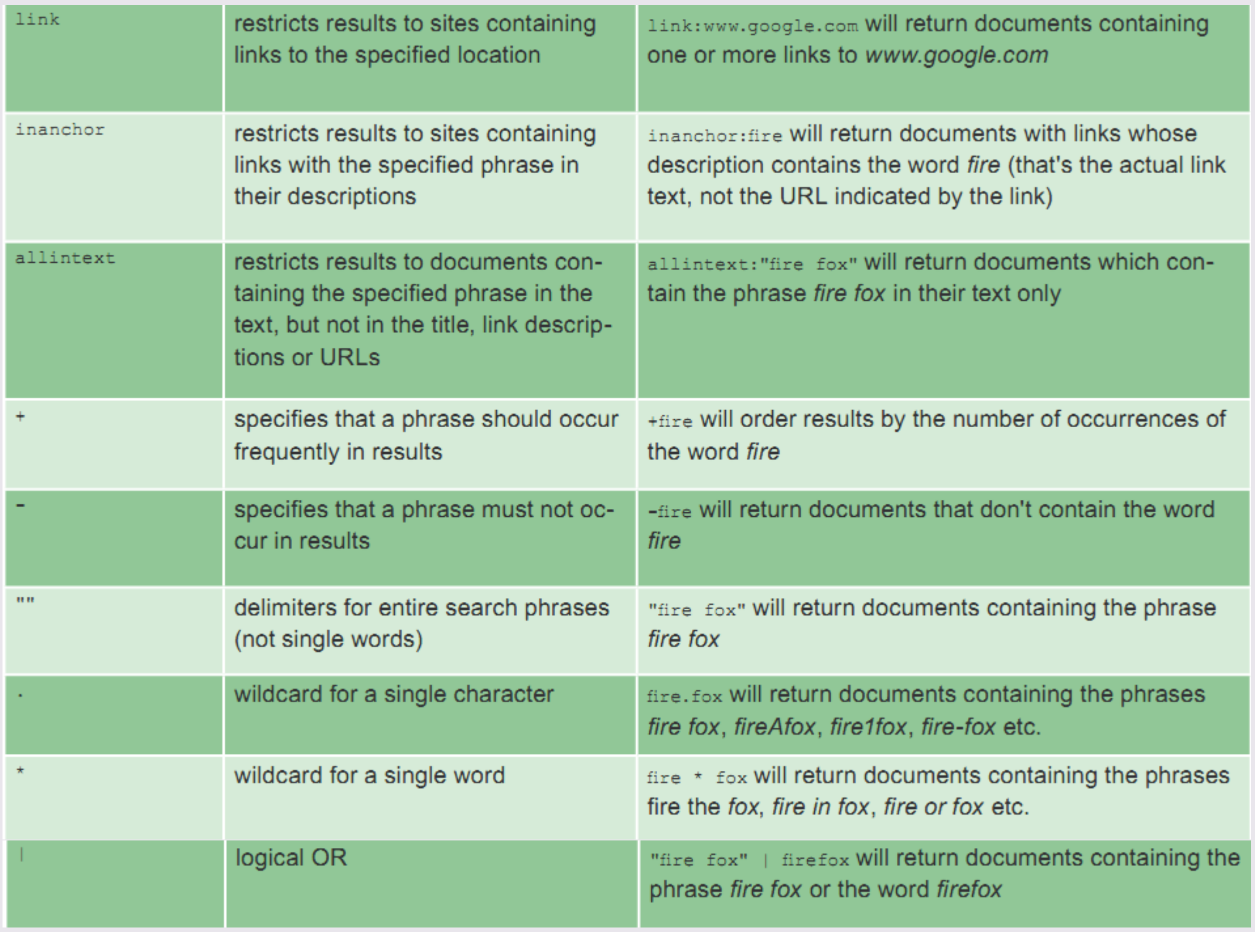
Note: Spaces above means Logical OR search. Now search results will contain all those results which contain either Hackingloops.com or email in the URL or Hacking or Email or Account. But first page results will be accurate as Google first gives priority to complete search string, then it break the complete string into parts separated by spaces. If you want only Hackingloops search results having email in the URL and text or title contains Hacking email account then your search result will be:

Site:www.hackingloops.com + inurl:email + "Hacking Email Account"

This was just the small example of smart searching techniques. Below is the list of all operators provided by Google to filter our search results and make our searching specific and accurate.



Operator	Description	Sample query
`site`	restricts results to sites within the specified domain	`site:google.com fox` will find all sites containing the word *fox*, located within the **.google.com* domain
`intitle`	restricts results to documents whose title contains the specified phrase	`intitle:fox fire` will find all sites with the word *fox* in the title and *fire* in the text
`allintitle`	restricts results to documents whose title contains all the specified phrases	`allintitle:fox fire` will find all sites with the words *fox* and *fire* in the title, so it's equivalent to `intitle:fox intitle:fire`
`inurl`	restricts results to sites whose URL contains the specified phrase	`inurl:fox fire` will find all sites containing the word *fire* in the text and *fox* in the URL
`allinurl`	restricts results to sites whose URL contains all the specified phrases	`allinurl:fox fire` will find all sites with the words *fox* and *fire* in the URL, so it's equivalent to `inurl:fox inurl:fire`
`filetype, ext`	restricts results to documents of the specified type	`filetype:pdf fire` will return PDFs containing the word fire, while *filetype:xls* fox will return *Excel* spreadsheets with the word *fox*
`numrange`	restricts results to documents containing a number from the specified range	`numrange:1-100 fire` will return sites containing a number from 1 to 100 and the word *fire*. The same result can be achieved with `1..100 fire`



`link`	restricts results to sites containing links to the specified location	`link:www.google.com` will return documents containing one or more links to *www.google.com*
`inanchor`	restricts results to sites containing links with the specified phrase in their descriptions	`inanchor:fire` will return documents with links whose description contains the word *fire* (that's the actual link text, not the URL indicated by the link)
`allintext`	restricts results to documents con-taining the specified phrase in the text, but not in the title, link descrip-tions or URLs	`allintext:"fire fox"` will return documents which con-tain the phrase *fire fox* in their text only
`+`	specifies that a phrase should occur frequently in results	`+fire` will order results by the number of occurrences of the word *fire*
`-`	specifies that a phrase must not oc-cur in results	`-fire` will return documents that don't contain the word *fire*
`""`	delimiters for entire search phrases (not single words)	`"fire fox"` will return documents containing the phrase *fire fox*
`.`	wildcard for a single character	`fire.fox` will return documents containing the phrases *fire fox*, *fireAfox*, *fire1fox*, *fire-fox* etc.
`*`	wildcard for a single word	`fire * fox` will return documents containing the phrases fire the *fox*, *fire in fox*, *fire or fox* etc.
`|`	logical OR	`"fire fox" | firefox` will return documents containing the phrase *fire fox* or the word *firefox*

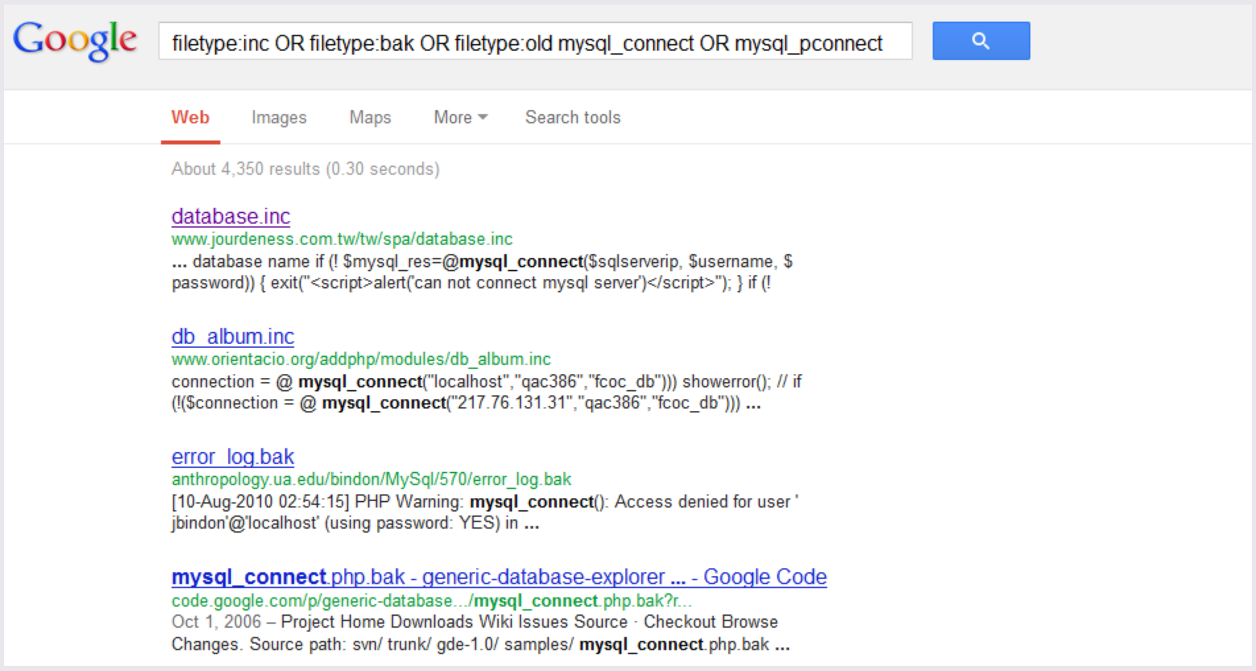
DORKS

Dorks or Google Dorks is nothing but just a combination of Google operators whose sole purpose is to filter the vulnerable websites or results based on dork query. How a dork query works? Working is quite similar to normal Google operator but Google dorks exploits the Google indexing facts. Normally Google indexes all the URL's or data on any web server i.e. say I made a PHP website which has functionality of Admin Controls, Upload files to server and some dynamic URL's. Now when I sub-mit my website to Google for indexing, what it will do; it will explore the complete structure of the website and index all the Webpages including admin panel and Up-load URL's or dynamically generated URL's until and unless we specify no-index on in the META data. Now dorks is what we used to extract these type of vulnerable URL's and websites from the Search Engine.

Few examples of dork queries are:

filetype:inc OR filetype:bak OR filetype:old mysql_connect OR mysql_pconnect

Now when you put the above dork in Google, you will get the database connection credentials (username and passwords) in plaintext as shown below:



```
filetype:inc OR filetype:bak OR filetype:old mysql_connect OR mysql_pconnect
```

About 4,350 results (0.30 seconds)

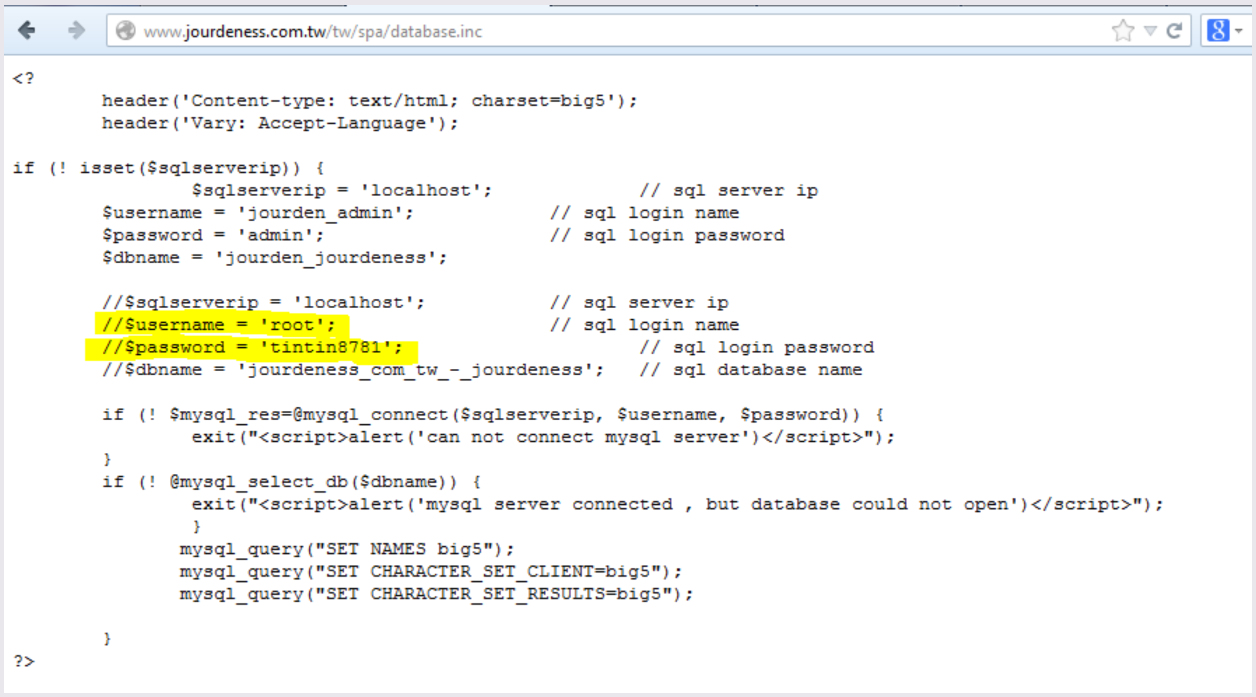
database.inc
www.jourdeness.com.tw/tw/spa/database.inc
... database name if (! $mysql_res=@**mysql_connect**($sqlserverip, $username, $ password)) { exit("<script>alert('can not connect mysql server')</script>"); } if (!

db_album.inc
www.orientacio.org/addphp/modules/db_album.inc
connection = @ **mysql_connect**("localhost","qac386","fcoc_db"))) showerror(); // if (!($connection = @ **mysql_connect**("217.76.131.31","qac386","fcoc_db"))) ...

error_log.bak
anthropology.ua.edu/bindon/MySql/570/error_log.bak
[10-Aug-2010 02:54:15] PHP Warning: **mysql_connect**(): Access denied for user ' jbindon'@'localhost' (using password: YES) in ...

mysql_connect.php.bak - generic-database-explorer ... - Google Code
code.google.com/p/generic-database.../**mysql_connect**.php.bak?r...
Oct 1, 2006 – Project Home Downloads Wiki Issues Source · Checkout Browse Changes. Source path: svn/ trunk/ gde-1.0/ samples/ **mysql_connect**.php.bak ...

Now open any link say first one, see what u will get:



www.jourdeness.com.tw/tw/spa/database.inc

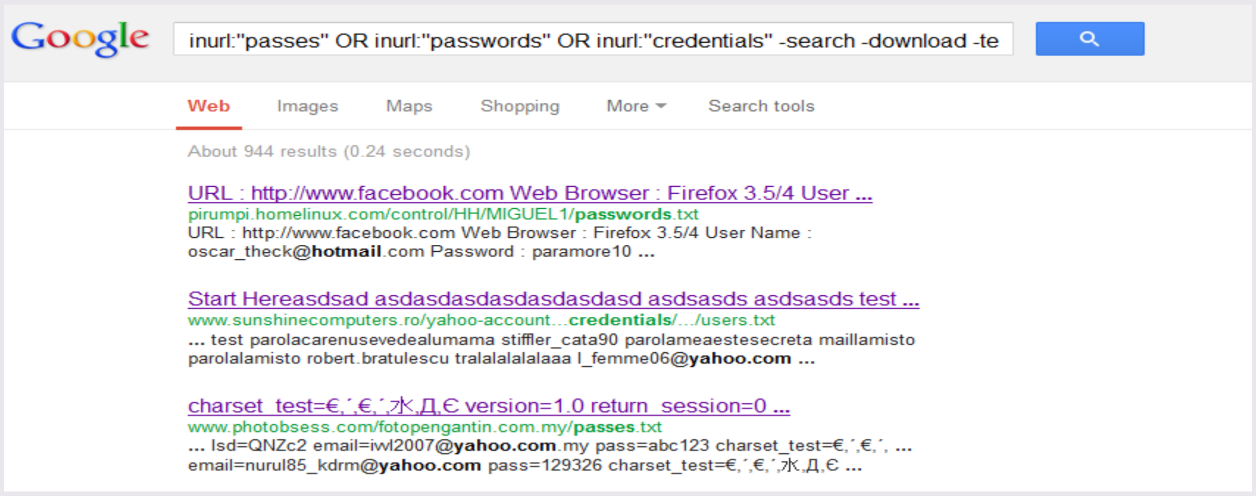
```
<?
        header('Content-type: text/html; charset=big5');
        header('Vary: Accept-Language');

if (! isset($sqlserverip)) {
                $sqlserverip = 'localhost';             // sql server ip
        $username = 'jourden_admin';            // sql login name
        $password = 'admin';                    // sql login password
        $dbname = 'jourden_jourdeness';

        //$sqlserverip = 'localhost';           // sql server ip
        //$username = 'root';                   // sql login name
        //$password = 'tintin8781';                     // sql login password
        //$dbname = 'jourdeness_com_tw_-_jourdeness';   // sql database name

        if (! $mysql_res=@mysql_connect($sqlserverip, $username, $password)) {
                exit("<script>alert('can not connect mysql server')</script>");
        }
        if (! @mysql_select_db($dbname)) {
                exit("<script>alert('mysql server connected , but database could not open')</script>");
                }
               mysql_query("SET NAMES big5");
               mysql_query("SET CHARACTER_SET_CLIENT=big5");
               mysql_query("SET CHARACTER_SET_RESULTS=big5");

        }
?>
```

So we have seen that using dorks we can extract very critical information with ease.

Here is another dork which extracts the password hacked by other hackers, it simply extract all phishing page password files :P it's better to use others effort :D :

inurl:"passes" OR inurl:"passwords" OR inurl:"credentials" -search -download -techsupt -git -games -gz -bypass -exe filetype:txt @yahoo.com OR @gmail OR @hotmail OR @rediff

You will see results like this:


inurl:"passes" OR inurl:"passwords" OR inurl:"credentials" -search -download -te
Web
Images
Maps
Shopping
More
Search tools

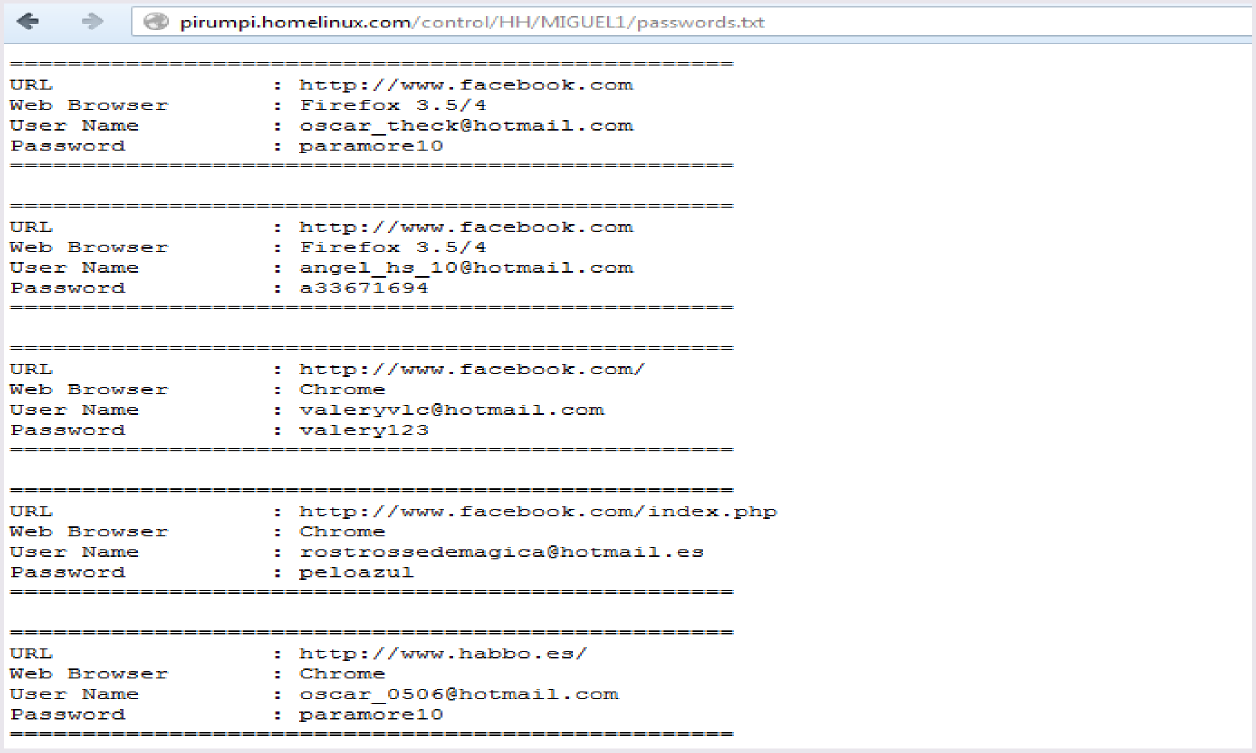
About 944 results (0.24 seconds)

URL : http://www.facebook.com Web Browser : Firefox 3.5/4 User ...
pirumpi.homelinux.com/control/HH/MIGUEL1/**passwords**.txt
URL : http://www.facebook.com Web Browser : Firefox 3.5/4 User Name : oscar_theck**@hotmail**.com Password : paramore10 ...

Start Hereasdsad asdasdasdasdasdasdasd asdsasds asdsasds test ...
www.sunshinecomputers.ro/yahoo-account...**credentials**/.../users.txt
... test parolacarenusevedealumama stiffler_cata90 parolameaestesecreta maillamisto parolalamisto robert.bratulescu tralalalalalaaa l_femme06**@yahoo.com** ...

charset_test=€,´,€,´,水,Д,Є version=1.0 return_session=0 ...
www.photobsess.com/fotopengantin.com.my/**passes**.txt
... lsd=QNZc2 email=iwl2007**@yahoo.com**.my pass=abc123 charset_test=€,´,€,´, ...
email=nurul85_kdrm**@yahoo.com** pass=129326 charset_test=€,´,€,´,水,Д,Є ...

Now open the link, you will see awesome things :D



```
==================================================
URL                  : http://www.facebook.com
Web Browser          : Firefox 3.5/4
User Name            : oscar_theck@hotmail.com
Password             : paramore10
==================================================

==================================================
URL                  : http://www.facebook.com
Web Browser          : Firefox 3.5/4
User Name            : angel_hs_10@hotmail.com
Password             : a33671694
==================================================

==================================================
URL                  : http://www.facebook.com/
Web Browser          : Chrome
User Name            : valeryvlc@hotmail.com
Password             : valery123
==================================================

==================================================
URL                  : http://www.facebook.com/index.php
Web Browser          : Chrome
User Name            : rostrossedemagica@hotmail.es
Password             : peloazul
==================================================

==================================================
URL                  : http://www.habbo.es/
Web Browser          : Chrome
User Name            : oscar_0506@hotmail.com
Password             : paramore10
==================================================
```

Now you all must have understood the power of Google dorks. Still not? Try yourself, here is the list of Google dorks which extracts passwords from Google.

Google Dork	Description
filetype:inc OR filetype:bak OR filetype:old mysql_connect OR mysql_pconnect	Aggregates previous mysql_(p)connect google dorks and adds a new filetype. Searches common file extensions used as backups by PHP developers. These extensions are normally not interpreted as code by their server, so their database connection credentials can be viewed in plaintext.
ext:xml ("proto='prpl-'" | "prpl-yahoo" | "prpl-silc" | "prpl-icq")	Find Accounts and Passwords from Pidgin Users. Google limit queries to 32 words so it?s impossible to search for all Account-Types in one query! List of all Params: Feel free to build your own search query. proto='prpl-'; prpl-silc; prpl-simple; prpl-zephyr; prpl-bonjour; prpl-qq; prpl-meanwhile; prpl-novell; prpl-gg; prpl-myspace; prpl-msn; prpl-gtalk; prpl-icq; prpl-aim; prpl-yahoo; prpl-yahoojp; prpl-yah; prpl-irc; prpl-yabber
allinurl:"User_info/auth_user_file.txt"	Google dork for find user info and configuration password of DCForum allinurl:"User_info/auth_user_file.txt"
inurl:"/dbman/default.pass"	A path to a DES encrypted password for DBMan (http://www.gossamer-threads.com/products/archive.html) ranging from Guest to Admin account, this is often found coupled with cgi-telnet.pl (http://www.rohitab.com/cgi-telnet) which provides an admin login, by default and the password provided by DBMan's path /dbman/default.pass
"parent directory" proftpdpasswd intitle:"index of" -google	This dork is based on this: http://www.exploit-db.com/ghdb/1212/ but improved cause that is useless, instead of this: "parent directory" proftpdpasswd intitle:"index of" -google
filetype:xls "username | password"	filetype:xls "username | password" This

	search reveals usernames and/or passwords of the xls documents.
ext:xml ("mode_passive"|"mode_default")	This dork finds Filezilla XML files. To be more specific; recentservers.xml sitemanager.xml filezilla.xml These files contain clear text usernames and passwords. They also contain the hostname or IP to connect to as well as the port. Most of these results will be for FTP however, you can also get port 22 to SSH in. This dork of course can be modified to target a specific website by appending site:whateversite.com. You can also look for a specific username like root by appending "root" to the dork.
intext:charset_test= email= default_persistent=	Find Facebook email and password ;)
inurl:"passes" OR inurl:"passwords" OR inurl:"credentials" -search -download -techsupt -git -games -gz -bypass -exe filetype:txt @yahoo.com OR @gmail OR @hotmail OR @rediff	There are a lot of Phishing pages hosted on internet; this dork will provide you with their password files. Clean and Simple
filetype:cfg "radius" (pass|passwd|password)	Find config files with radius configs and passwords and secrets...

These are just samples, for extensive list visit below links:

For Extensive list of Google dorks till date:
http://www.exploit-db.com/google-dorks/

SPIDERS, CRAWLERS OR ROBOTS DISCOVERY

A Web crawler is an Internet bot that systematically browses the World Wide Web, typically for the purpose of Web indexing. A Web crawler may also be called a Web spider, an ant, an automatic indexer, or (in the FOAF software context) a Web scutter. Web search engines and some other sites use Web crawling or spidering software to update their web content or indexes of others sites' web content. Web crawlers can copy all the pages they visit for later processing by a search engine that indexes the downloaded pages so that users can search them much more quickly.

Crawlers can validate hyperlinks and HTML code. They can also be can be used for web scraping. Now you all will be thinking what is web scraping. Well, Web scraping (web harvesting or web data extraction) is a technique of extracting information from websites that we will discuss in next topic.

Now how web crawlers or spider or robots or simply bots can play a handy role in information gathering. As we all know crawlers indexes our website under some protocols (rules) and these rules are defined by file called robot.txt. A robots.txt file restricts access to your site by search engine robots that crawl the web. These bots are automated, and before they access pages of a site, they check to see if a robots.txt file exists that prevents them from accessing certain pages. (All respectable robots will respect the directives in a robots.txt file, although some may interpret them differently. However, a robots.txt is not enforceable, and some spammers and other troublemakers may ignore it.)
You need a robots.txt file only if your site includes content that you don't want search engines to index. If you want search engines to index everything in your site, you don't need a robots.txt file (not even an empty one). And here the bug lies. Most of newbie coders don’t understand the significance of robot.txt file and hence allows Google or other search engine web crawlers to index their website completely. Completely means completely i.e. everything, even the upload url’s, FTP urls, database files and other critical information. This is only responsible for the success of Google Dorks. People leave loopholes while designing and when web crawlers index them, they are visible for everyone by just hitting a simple Google dork in search.

If you are a good hacker then you can even search the bugs in the robot.txt also. How? Newbie programmers or web developers uses their robot.txt file to block access to URL patterns and most funny part is that, now it becomes more predictable for hackers that something important is hidden behind is URL pattern. Here is sample of robot.txt file:

```
User-agent: *
Disallow: /folder1/

User-Agent: Googlebot
Disallow: /folder2/
```

Now User agent * means it allows all user agents to crawl their website and disallow /folder1/ means robot.txt is giving instruction to web crawl bot not to index

things inside /folder1/. And next instruction means it's blocking Googlebot to crawl /folder/ pattern i.e. URL like below will not be indexed:
www.victimwebsite.com/folder1/*.php
www.victimwebsite.com/foder2/*/..

But now it's become more predictable for hacker that something private or important is hidden behind this URL. Now as we all know most web crawler robots follow robots.txt file but some are still there in market which does not follow it. So what hackers do they will go to other search engines like altavista, ask, yahoo or bing to check that any URL with above pattern is crawled. If yes, then victim will be doomed for sure.

Oops I forgot to share how to find robot.txt file of any website if exists. It's simple, robot.txt file is always stored at root of any website. So It can be accessed by below pattern:

www.anywebsite.com/robots.txt

Have fun with it. Also there are several web crawl bots available in market which can be used to index websites restricted URL's.

WEB DATA EXTRACTION OR WEB SCRAPING

Web scraping (**web harvesting** or **web data extraction**) is a computer software technique of extracting information from websites. In web scraping, Hackers run website grabber or scraping programs which exactly simulates the website but it's in HTTP format. After extracting all the web pages, Hackers can easily find URL patterns, website general web logic. Now this information can be used to extract quite juicy information from the websites like:

1. Extracting URL's which accepts data from users (dynamically/statically). Now we can test all these URL's for SQL injection or cross site scripting attacks.
2. If Hacker is smart enough then they can even use it to create Phish Pages and use these Phish pages to trap victims using Tabnabbing technique.

Just like reading API docs, it takes a bit of work up front to figure out how the data is structured and how you can access it. Unlike APIs however, there's really no documentation so you have to be a little clever about it.

I'll share some of the tips I've learned along the way.

FETCHING THE DATA

So the first thing you're going to need to do is fetch the data. You'll need to start by finding your "endpoints" — the URL or URLs that return the data you need.

If you know you need your information organized in a certain way — or only need a specific subset of it — you can browse through the site using their navigation. Pay attention to the URLs and how they change as you click between sections and drill down into sub-sections.

The other option for getting started is to go straight to the site's search functionality. Try typing in a few different terms and again, pay attention to the URL and how it changes depending on what you search for. You'll probably see a GET parameter like q= that always changes based on you search term.

Try removing other unnecessary GET parameters from the URL, until you're left with only the ones you need to load your data. Make sure that there's always a beginning? To start the query string and a & between each key/value pair.

DEALING WITH PAGINATION

At this point, you should be starting to see the data you want access to, but there's usually some sort of pagination issue keeping you from seeing all of it at once. Most regular APIs do this as well, to keep single requests from slamming the database.

Usually, clicking to page 2 adds some sort of offset= parameter to the URL, which is usually either the page number or else the number of items displayed on the page. Try changing this to some really high number and see what response you get when you "fall off the end" of the data.

With this information, you can now iterate over every page of results, incrementing the offset parameter as necessary, until you hit that "end of data" condition.

The other thing you can try doing is changing the "Display X per Page" which most pagination UIs now have. Again, look for a new GET parameter to be appended to the URL which indicates how many items are on the page. Try setting this to some arbitrarily large number to see if the server will return all the information you need in a single request. Sometimes there'll be some limits enforced server-side that you can't get around by tampering with this, but it's still worth a shot since it can cut down on the number of pages you must paginate through to get all the data you need.

AJAX CONTENT EXTRACTION!

Sometimes people see web pages with URL fragments # and AJAX content loading and think a site can’t be scraped. On the contrary! If a site is using AJAX to load the data, that probably makes it even easier to pull the information you need. The AJAX response is probably coming back in some nicely-structured way (probably JSON!) in order to be rendered on the page with JavaScript.

All you have to do is pull up the network tab in Web Inspector or Firebug and look through the XHR requests for the ones that seem to be pulling in your data. Once you find it, you can leave the crafty HTML behind and focus instead on this end-point, which is essentially an undocumented API.

(UN) structured Data?

Now that you’ve figured out how to get the data you need from the server, the somewhat tricky part is getting the data you need out of the page’s markup.

USE CSS HOOKS

In my experience, this is usually straightforward since most web designers litter the markup with tons of classes and ids to provide hooks for their CSS. You can piggy-back on these to jump to the parts of the markup that contain the data you need.

Just right click on a section of information you need and pull up the Web Inspector or Firebug to look at it. Zoom up and down through the DOM tree until you find the outermost <div> around the item you want.

This <div> should be the outer wrapper around a single item you want access to. It probably has some class attribute which you can use to easily pull out all of the other wrapper elements on the page. You can then iterate over these just as you would iterate over the items returned by an API response.

Note: The DOM tree that is presented by the inspector isn’t always the same as the DOM tree represented by the HTML sent back by the website. It’s possible that the DOM you see in the inspector has been modified by JavaScript — or sometime even the browser, if it’s in quirks mode.

Once you find the right node in the DOM tree, you should always view the source of the page (“right click” > “View Source”) to make sure the elements you need are actually showing up in the raw HTML.

This issue has caused me a number of head-scratchers.

GET A GOOD HTML PARSING LIBRARY

It is probably a horrible idea to try parsing the HTML of the page as a long string (although there are times I’ve needed to fall back on that). Spend some time doing research for a good HTML parsing library in your language of choice.

You’re going to have a bad time if you try to use an XML parser since most websites out there don’t actually validate as properly formed XML (sorry XHTML!) and will give you a ton of errors.

A good library will read in the HTML that you pull in using some HTTP library (hat tip to the Requests library if you’re writing Python) and turn it into an object that you can traverse and iterate over to your heart’s content, similar to a JSON object.

Some Traps to Know About

I should mention that some websites explicitly prohibit the use of automated scraping, so it’s a good idea to read your target site’s Terms of Use to see if you’re going to make anyone upset by scraping.

For two-thirds of the website I’ve scraped, the above steps are all you need. Just fire off a request to your “endpoint” and parse the returned data.

But sometimes, you’ll find that the response you get when scraping isn’t what you saw when you visited the site yourself.

WHEN IN DOUBT, SPOOF HEADERS

Some websites require that your User Agent string is set to something they allow, or you need to set certain cookies or other headers in order to get a proper response.

Depending on the HTTP library you’re using to make requests, this is usually pretty straightforward. I just browse the site in my web browser and then grab all of the headers that my browser is automatically sending. Then I put those in a dictionary and send them along with my request.

Note that this might mean grabbing some login or other session cookie, which might identify you and make your scraping less anonymous. It’s up to you how serious of a risk that is.

CONTENT BEHIND A LOGIN

Sometimes you might need to create an account and login to access the information you need. If you have a good HTTP library that handles logins and automatically sending session cookies (did I mention how awesome Requests is?), then you just need your scraper login before it gets to work. Note that this obviously makes you totally non-anonymous to the third party website so all of your scraping behavior is probably pretty easy to trace back to you if anyone on their side cared to look.

RATE LIMITING

I’ve never actually run into this issue myself, although I did have to plan for it one time. I was using a web service that had a strict rate limit that I knew I’d exceed fairly quickly.

Since the third party service conducted rate-limiting based on IP address (stated in their docs), my solution was to put the code that hit their service into some client-side JavaScript, and then send the results back to my server from each of the clients.

This way, the requests would appear to come from thousands of different places, since each client would presumably have their own unique IP address, and none of them would individually be going over the rate limit. Depending on your application, this could work for you.

POORLY FORMED MARKUP

Sadly, this is the one condition that there really is no cure for. If the markup doesn’t come close to validating, then the site is not only keeping you out, but also serving a degraded browsing experience to all of their visitors.

It’s worth digging into your HTML parsing library to see if there’s any setting for error tolerance. Sometimes this can help.

If not, you can always try falling back on treating the entire HTML document as a long string and do all of your parsing as string.

REVIEWING METADATA AND JAVASCRIPT'S

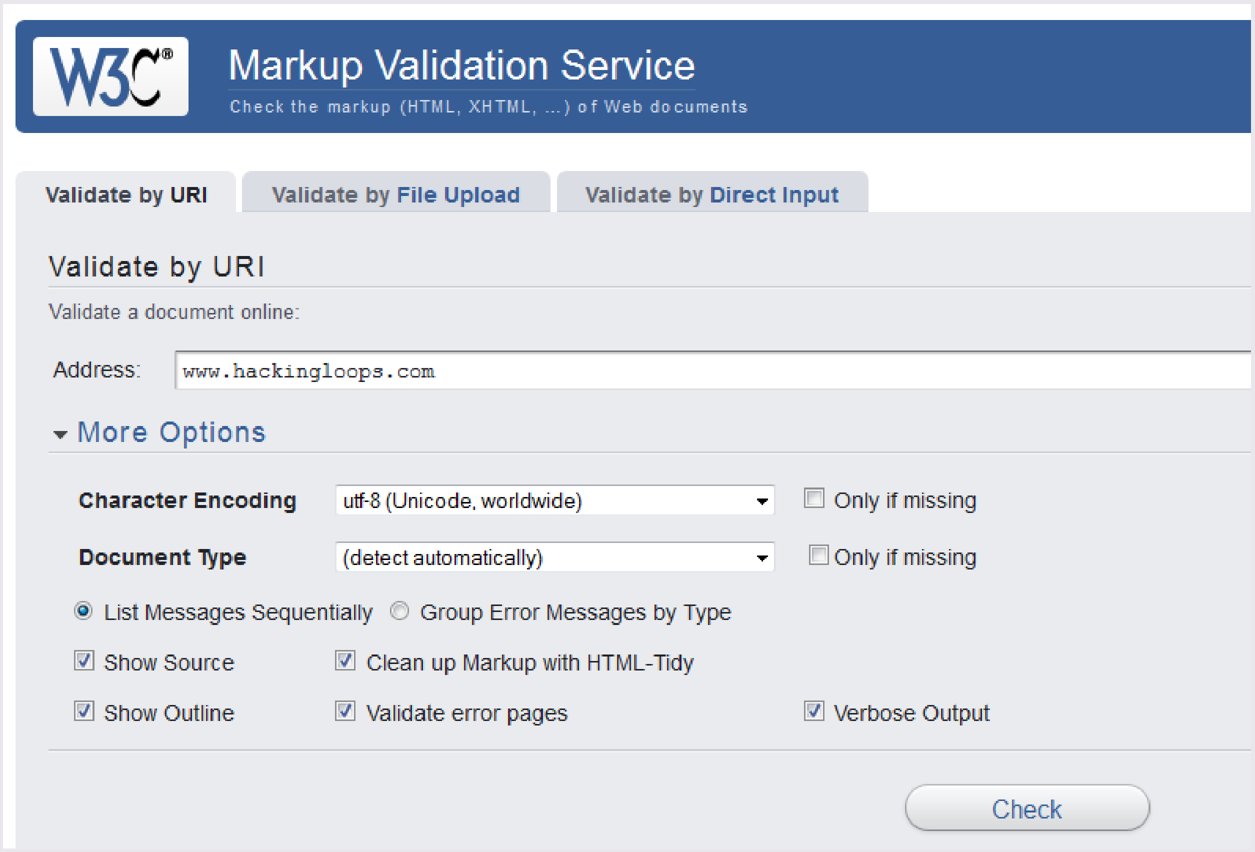
This is quite simple task but sometimes it proves quite effective. Sometimes what happens when designers write's JavaScript's they leave some useful information commented there like hidden links or URL patterns or sometimes Web Designers use JavaScript's to Spoof Exact URL's to avoid injection attacks. What Hacker has to do is just debug the source code of webpage and search of commented links or debug the JavaScript's to find why it is used for. Its pity sure you will end up with critical information Extraction.

Also Metadata is too important. Meta information is stored in headers of web pages and it contains critical information like indexing information, parser information, Unicode format etc. Indexing information can be extracting by checking the Meta tag of Index. Usually people use index all tag and this enables Google to crawl each and every URL of the website wherever the header is used. Now if you own website then you might be aware of URL patterns like Upload forms, plugin forms, Override forms etc. Just what you have to use Google dorks to extract required information. I have already discussed same above so no need to go in detail now.

PARSER AND MARKUP INFORMATION

There are two Meta Tags namely Parser and Generator, with help of these you can query Parsing Information. In this I have some smart tricks that really help us to dig really critical information from the website which help us to launch CSS Hook attacks and also help in launching Cross Site Scripting Attacks.

Go to website http://validator.w3.org/ for Markup validation service and put the URL in the address box given below Validate by URI as shown below and do the below shown settings.


W3C®
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Validate by URI
Validate by File Upload
Validate by Direct Input
Check

Validate by URI

Validate a document online:

Address: www.hackingloops.com

More Options

Character Encoding: utf-8 (Unicode, worldwide) — ☐ Only if missing

Document Type: (detect automatically) — ☐ Only if missing

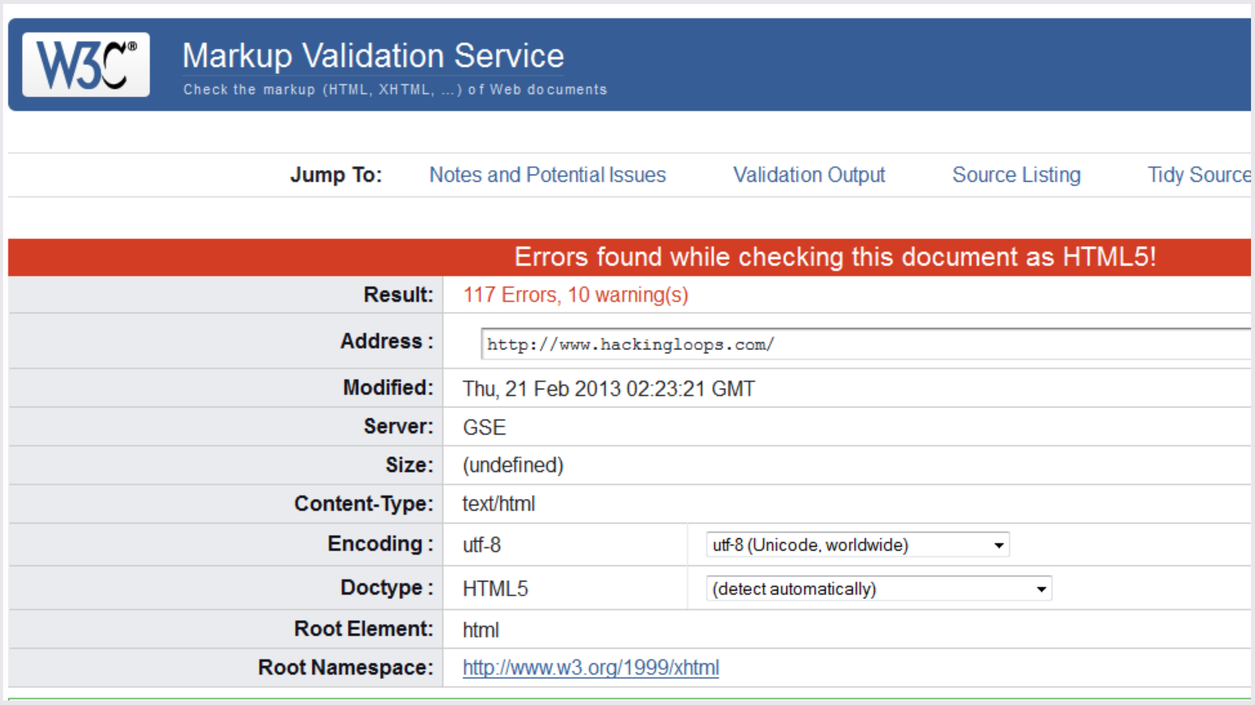
● List Messages Sequentially ○ Group Error Messages by Type

☑ Show Source ☑ Clean up Markup with HTML-Tidy

☑ Show Outline ☑ Validate error pages ☑ Verbose Output

Now click on check (document type default for first run) and then choose different document type every time and most funny part, every time you will end up with finding some bug in the website which can help you in executing CSS hooks. Actually it's my luck that every time it does.

Note: My website is a blog (Google blog), so this technique will not work on it. Because of security reasons I cannot show someone else website :P. Also this technique will not work on Google Blogger blogs but other than that it works everywhere.



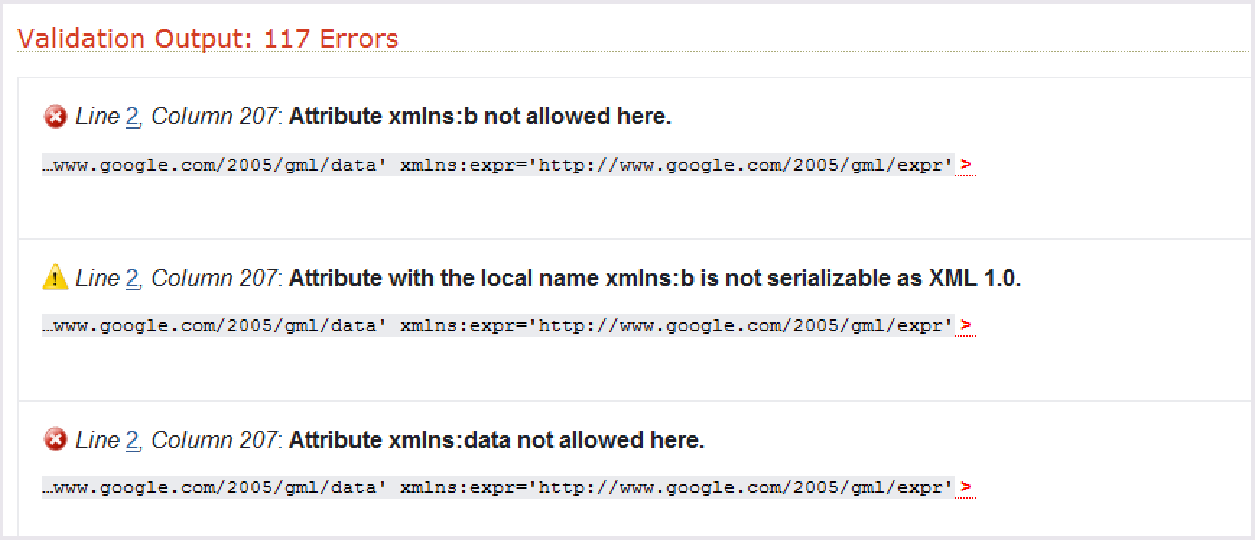
Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Jump To: Notes and Potential Issues | Validation Output | Source Listing | Tidy Source

Errors found while checking this document as HTML5!

Result:	117 Errors, 10 warning(s)
Address :	http://www.hackingloops.com/
Modified:	Thu, 21 Feb 2013 02:23:21 GMT
Server:	GSE
Size:	(undefined)
Content-Type:	text/html
Encoding :	utf-8 — utf-8 (Unicode, worldwide)
Doctype :	HTML5 — (detect automatically)
Root Element:	html
Root Namespace:	http://www.w3.org/1999/xhtml



Validation Output: 117 Errors

Line 2, Column 207: **Attribute xmlns:b not allowed here.**

```
…www.google.com/2005/gml/data' xmlns:expr='http://www.google.com/2005/gml/expr' >
```

Line 2, Column 207: **Attribute with the local name xmlns:b is not serializable as XML 1.0.**

```
…www.google.com/2005/gml/data' xmlns:expr='http://www.google.com/2005/gml/expr' >
```

Line 2, Column 207: **Attribute xmlns:data not allowed here.**

```
…www.google.com/2005/gml/data' xmlns:expr='http://www.google.com/2005/gml/expr' >
```

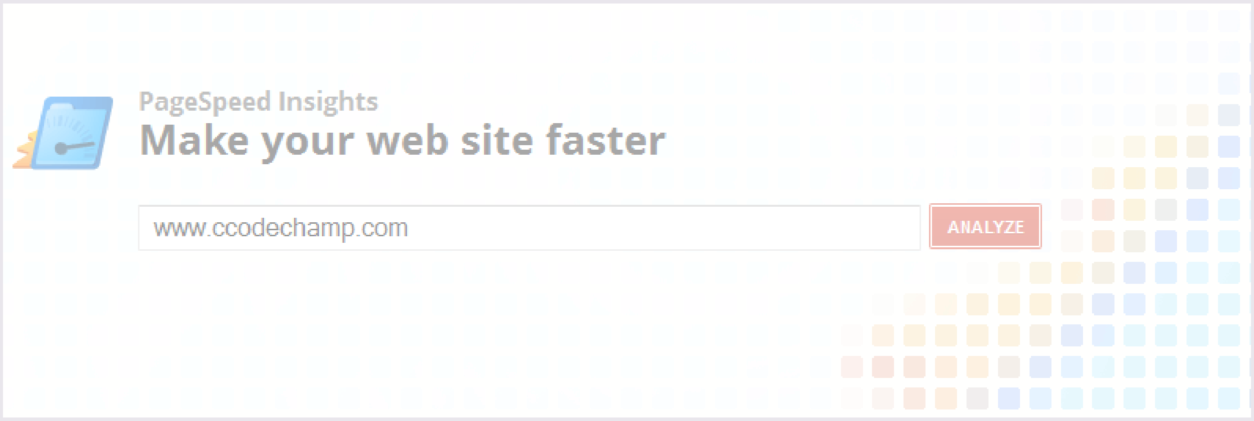
Go through all of them you will end with List of bugs you can use against victim.

There is another smart technique for extracting Juicy information regarding web-sites i.e. Page Speed API by Google.

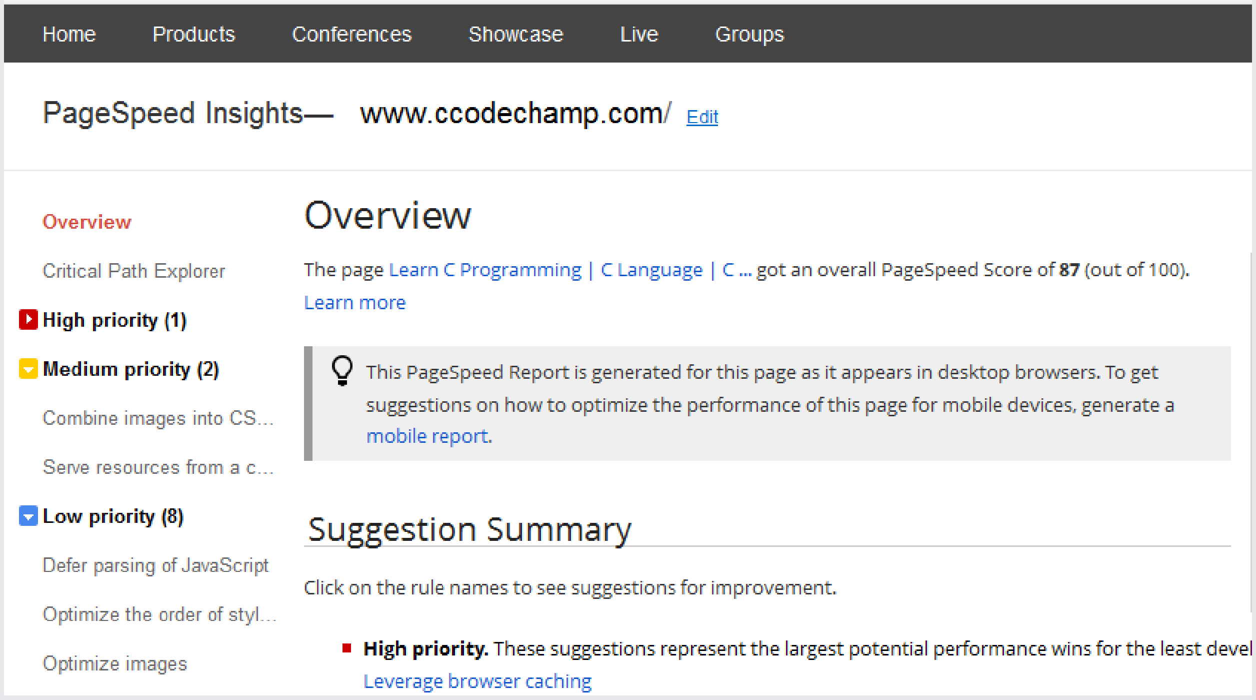
USING PAGE SPEED TO DIG CRITICAL INFORMATION ABOUT WEBSITE

Page speed is a great tool for analyzing the performance of website. But same thing can also be used to get CRITICAL information. Let me show you how?

Open page Speed: https://developers.google.com/speed/pagespeed/insights



Now check the report: Overview



Overview

The page Learn C Programming | C Language | C ... got an overall PageSpeed Score of **87** (out of 100). Learn more

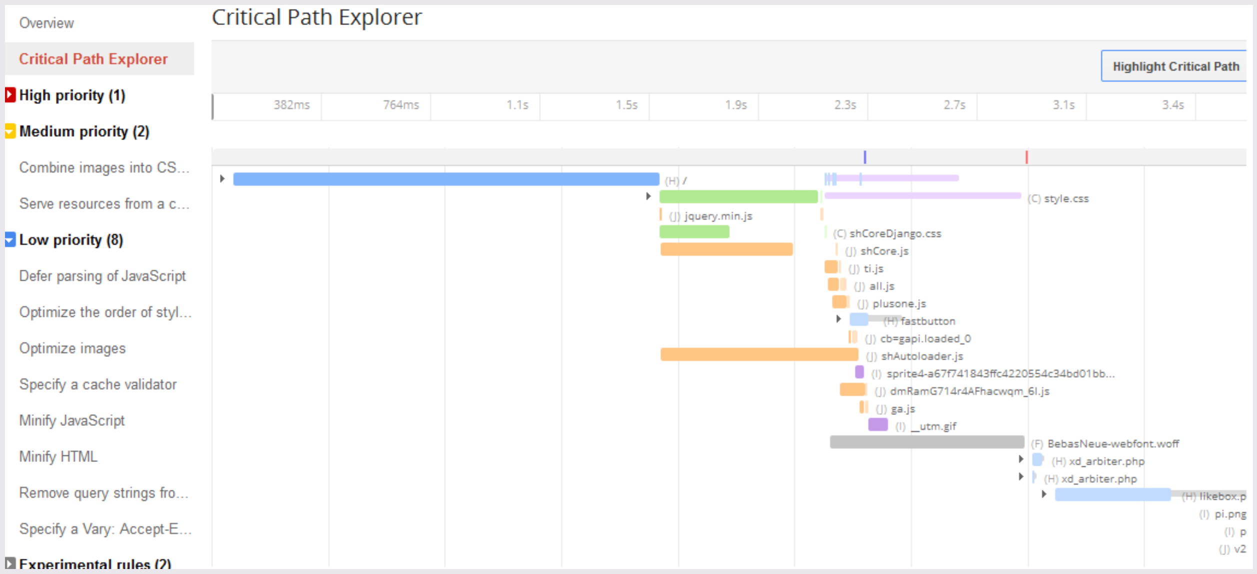
This PageSpeed Report is generated for this page as it appears in desktop browsers. To get suggestions on how to optimize the performance of this page for mobile devices, generate a mobile report.

Suggestion Summary

Click on the rule names to see suggestions for improvement.

- **High priority.** These suggestions represent the largest potential performance wins for the least devel
  Leverage browser caching

Now to explore URL patterns: Critical path Explorer (It shows all the URL Patterns that a website loads during start up). More we know, better the chances to hack.



Similarly go through all options one by one to analyze the website. Using this technique we can extract quite juicy information.

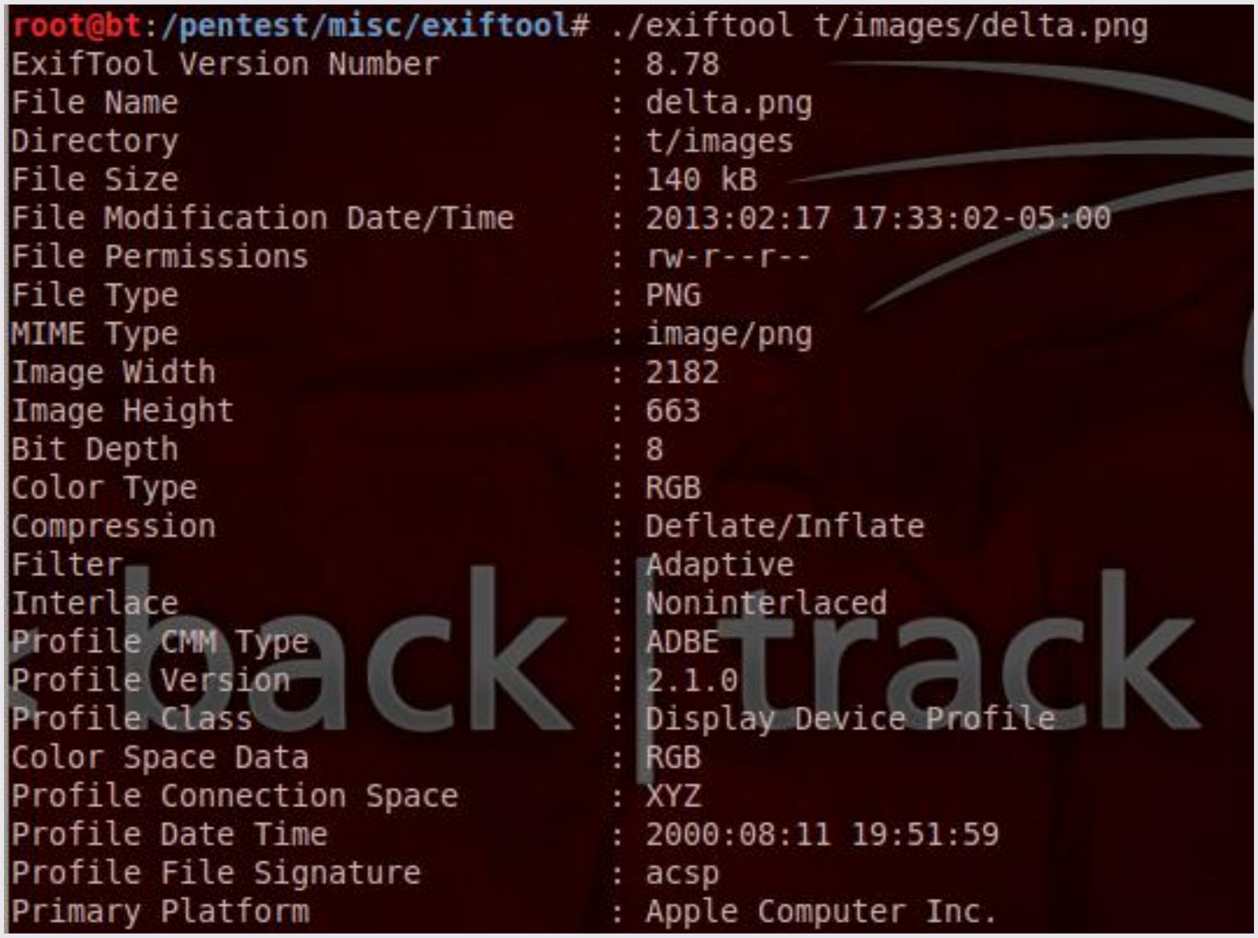
AUTOMATED DATA EXTRACTION USING HACK TOOLS

EXIFTOOL

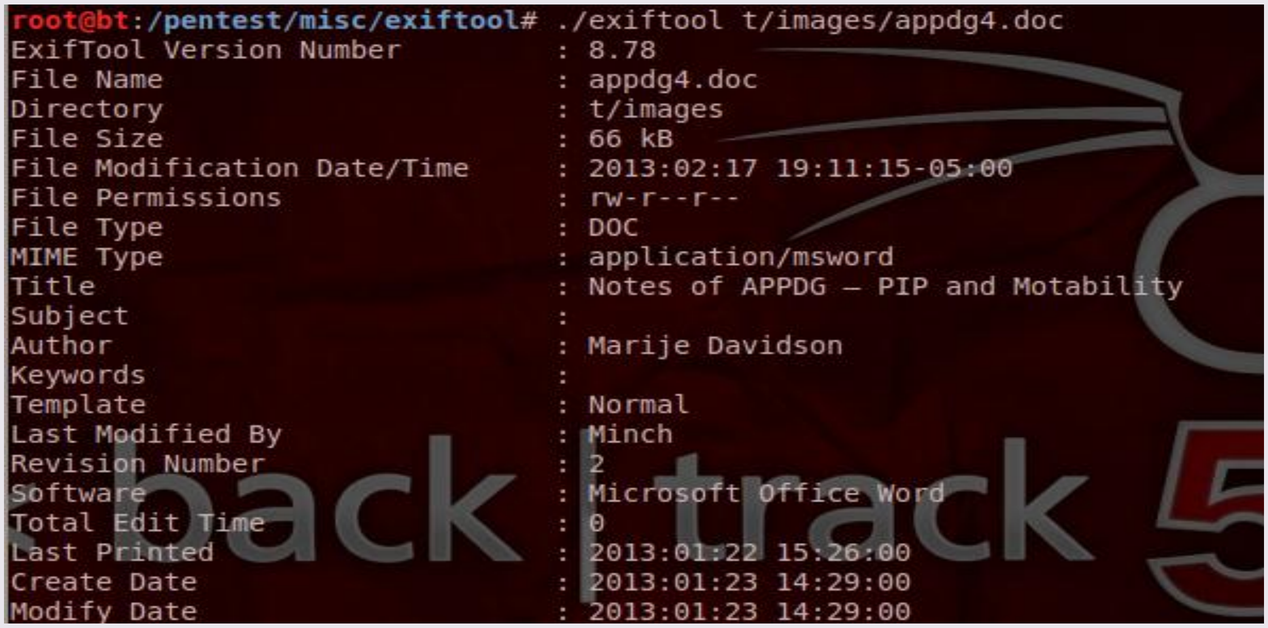
One of the tools that can extract Metadata information is the exiftool. This tool is found in Backtrack distribution and can extract information from various file types like DOC, XLS, PPT, PNG and JPEG. Typically the information that we would look for are:

Title
Subject
Author
Comments
Software
Company
Manager
Hyperlinks
Current User

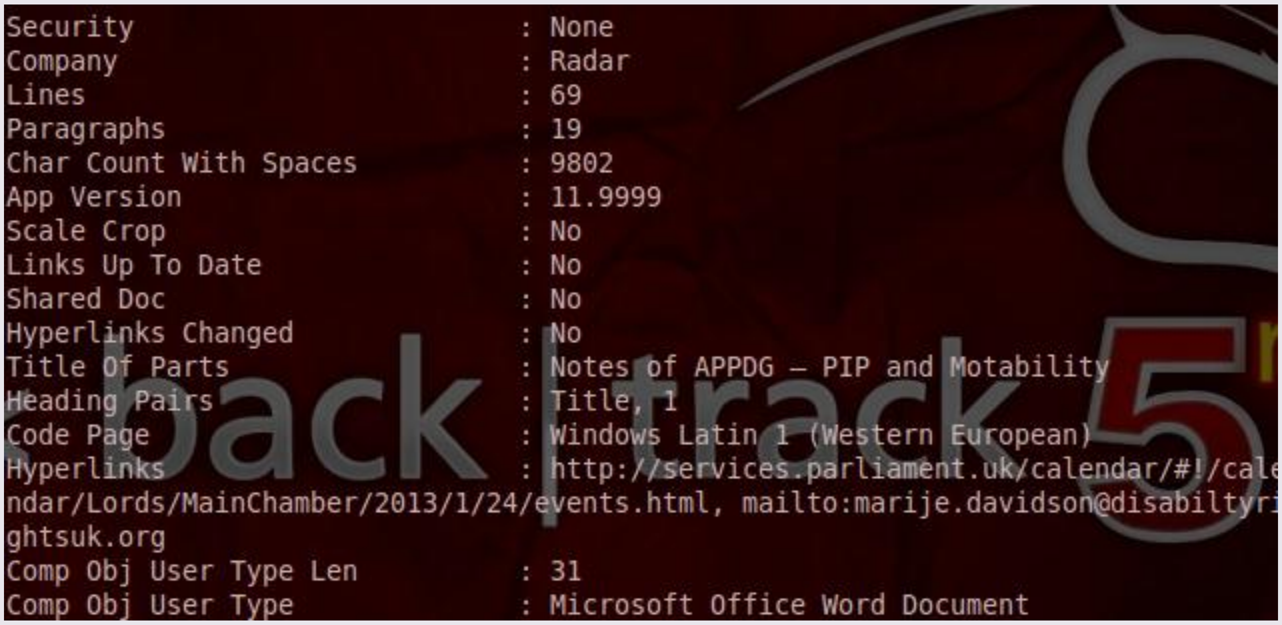
Below is the information that we have obtained from an image and the metadata from a doc file.



```
root@bt:/pentest/misc/exiftool# ./exiftool t/images/delta.png
ExifTool Version Number         : 8.78
File Name                       : delta.png
Directory                       : t/images
File Size                       : 140 kB
File Modification Date/Time     : 2013:02:17 17:33:02-05:00
File Permissions                : rw-r--r--
File Type                       : PNG
MIME Type                       : image/png
Image Width                     : 2182
Image Height                    : 663
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Profile CMM Type                : ADBE
Profile Version                 : 2.1.0
Profile Class                   : Display Device Profile
Color Space Data                : RGB
Profile Connection Space        : XYZ
Profile Date Time               : 2000:08:11 19:51:59
Profile File Signature          : acsp
Primary Platform                : Apple Computer Inc.
```



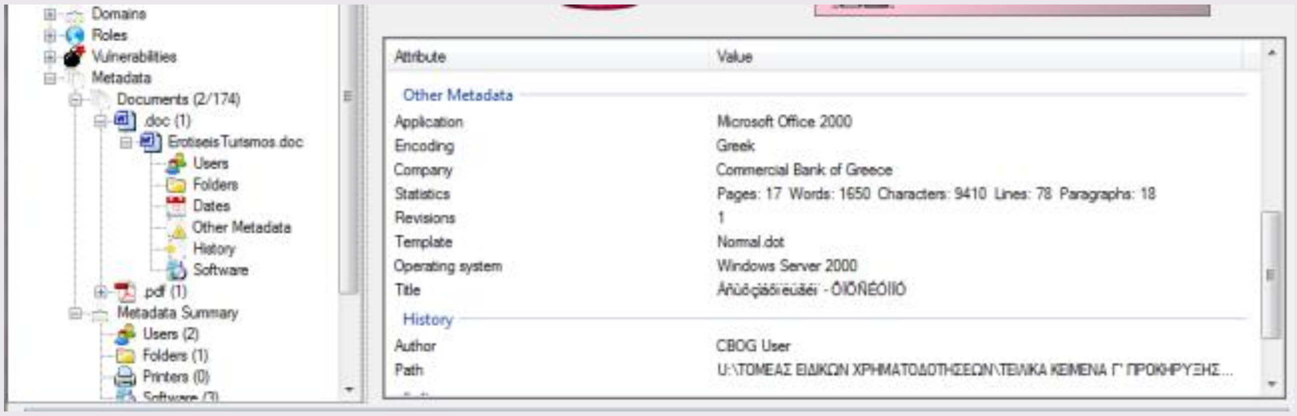
```
root@bt:/pentest/misc/exiftool# ./exiftool t/images/appdg4.doc
ExifTool Version Number         : 8.78
File Name                       : appdg4.doc
Directory                       : t/images
File Size                       : 66 kB
File Modification Date/Time     : 2013:02:17 19:11:15-05:00
File Permissions                : rw-r--r--
File Type                       : DOC
MIME Type                       : application/msword
Title                           : Notes of APPDG – PIP and Motability
Subject                         :
Author                          : Marije Davidson
Keywords                        :
Template                        : Normal
Last Modified By                : Minch
Revision Number                 : 2
Software                        : Microsoft Office Word
Total Edit Time                 : 0
Last Printed                    : 2013:01:22 15:26:00
Create Date                     : 2013:01:23 14:29:00
Modify Date                     : 2013:01:23 14:29:00
```



```
Security                        : None
Company                         : Radar
Lines                           : 69
Paragraphs                      : 19
Char Count With Spaces          : 9802
App Version                     : 11.9999
Scale Crop                      : No
Links Up To Date                : No
Shared Doc                      : No
Hyperlinks Changed              : No
Title Of Parts                  : Notes of APPDG – PIP and Motability
Heading Pairs                   : Title, 1
Code Page                       : Windows Latin 1 (Western European)
Hyperlinks                      : http://services.parliament.uk/calendar/#!/cale
ndar/Lords/MainChamber/2013/1/24/events.html, mailto:marije.davidson@disabiltyri
ghtsuk.org
Comp Obj User Type Len          : 31
Comp Obj User Type              : Microsoft Office Word Document
```

FOCA

FOCA is another great tool for analyzing metadata in documents. It is a GUI based tool which makes the process a lot of easier. The only thing that we have to do is to specify the domain that we want to search for files and the file type (doc, xls, pdf) and FOCA will perform the job for us very easily. Below you can see a screenshot of the metadata that we have extracted from a doc file. As you can see we have obtained a username an internal path and the operating system that the file has created.



Attribute	Value
Other Metadata	
Application	Microsoft Office 2000
Encoding	Greek
Company	Commercial Bank of Greece
Statistics	Pages: 17 Words: 1650 Characters: 9410 Lines: 78 Paragraphs: 18
Revisions	1
Template	Normal.dot
Operating system	Windows Server 2000
Title	Áñùôçóåéò - ÔÏÕÑÉÓÌÏÓ
History	
Author	CBOG User
Path	U:\ΤΟΜΕΑΣ ΕΙΔΙΚΩΝ ΧΡΗΜΑΤΟΔΟΤΗΣΕΩΝ\ΤΕΛΙΚΑ ΚΕΙΜΕΝΑ Γ' ΠΡΟΚΗΡΥΞΗΣ...

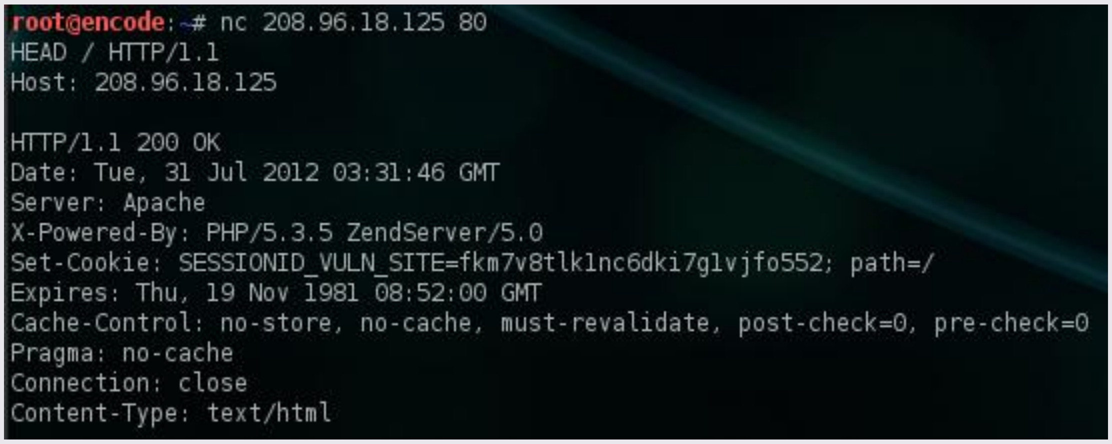
WEB APPLICATION OR WEB SERVER FINGERPRINT

One of the first tasks when conducting a web application penetration test is to try to identify the version of the web server and the web application. The reason for that is that it allows us to discover all the well-known vulnerabilities that are affecting the web server and the application. This process is called web application fingerprinting and in this article we will see how to perform it.

The web application fingerprinting can be done with the use of a variety of tools or manually.

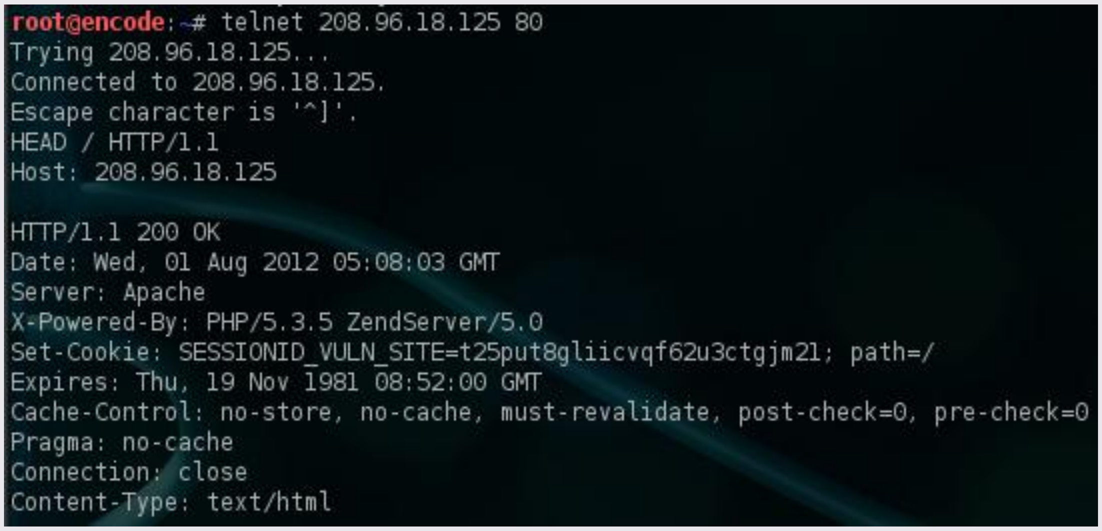
MANUAL FINGERPRINTING

This can be done with the use of different utilities such as the telnet or the netcat. For example we can try to connect with netcat to the remote webserver that is running on port 80.We will send an HTTP request by using the HEAD method and we will wait for the response of the web server.

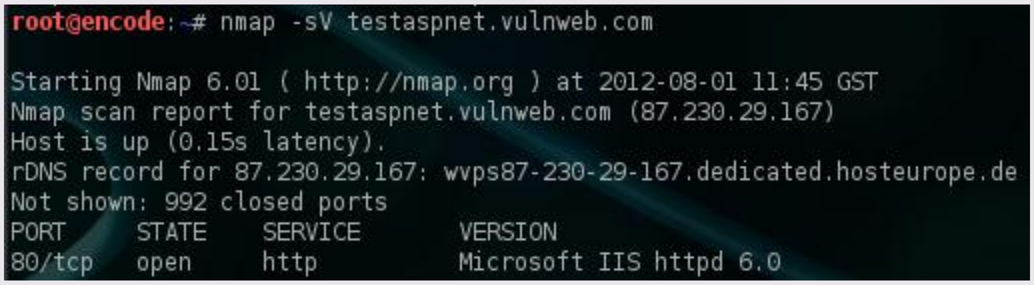


```
root@encode:~# nc 208.96.18.125 80
HEAD / HTTP/1.1
Host: 208.96.18.125

HTTP/1.1 200 OK
Date: Tue, 31 Jul 2012 03:31:46 GMT
Server: Apache
X-Powered-By: PHP/5.3.5 ZendServer/5.0
Set-Cookie: SESSIONID_VULN_SITE=fkm7v8tlk1nc6dki7glvjfo552; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Connection: close
Content-Type: text/html
```

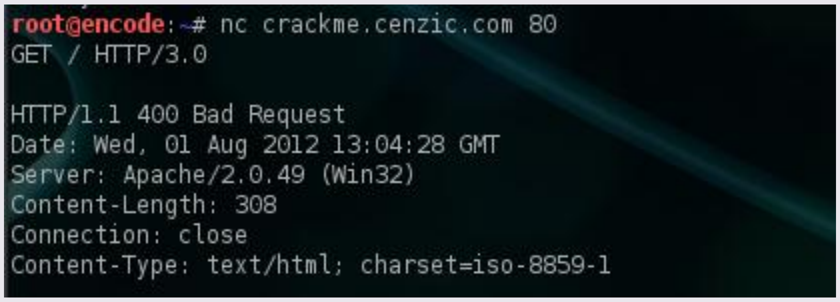
As we can see from the HTTP response header the type of the web server is Apache. Also we have managed to identify the technology from the X-Powered-By field name along with the version that supports the application which is PHP/5.3.5 and also the web application that is running on the web server which is a Zend-Server. Alternatively if we don’t want to use the netcat utility we can use the telnet in order to obtain the header information from the web server. The image below is showing the usage of telnet in obtaining the HTTP Response Header from the same web server.


```
root@encode:~# telnet 208.96.18.125 80
Trying 208.96.18.125...
Connected to 208.96.18.125.
Escape character is '^]'.
HEAD / HTTP/1.1
Host: 208.96.18.125

HTTP/1.1 200 OK
Date: Wed, 01 Aug 2012 05:08:03 GMT
Server: Apache
X-Powered-By: PHP/5.3.5 ZendServer/5.0
Set-Cookie: SESSIONID_VULN_SITE=t25put8gliicvqf62u3ctgjm21; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Connection: close
Content-Type: text/html
```

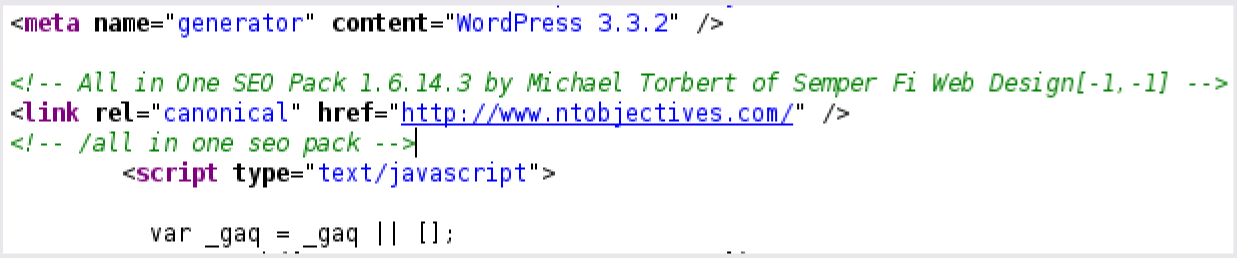
Another way is while we are performing our port scan with Nmap on the remote host to use the command -sV which will obtain as well the type and the version of the web server that is running. For example in the image below we can see from the output that Nmap discovered that the web server is IIS version 6.0.


```
root@encode:~# nmap -sV testaspnet.vulnweb.com

Starting Nmap 6.01 ( http://nmap.org ) at 2012-08-01 11:45 GST
Nmap scan report for testaspnet.vulnweb.com (87.230.29.167)
Host is up (0.15s latency).
rDNS record for 87.230.29.167: wvps87-230-29-167.dedicated.hosteurope.de
Not shown: 992 closed ports
PORT      STATE    SERVICE       VERSION
80/tcp    open     http          Microsoft IIS httpd 6.0
```

Another method is to send a malformed request to the web server that will cause the web server to produce an error page which will contain in the response header the version of the web server. Sample malformed request is shown below.


```
root@encode:~# nc crackme.cenzic.com 80
GET / HTTP/3.0

HTTP/1.1 400 Bad Request
Date: Wed, 01 Aug 2012 13:04:28 GMT
Server: Apache/2.0.49 (Win32)
Content-Length: 308
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

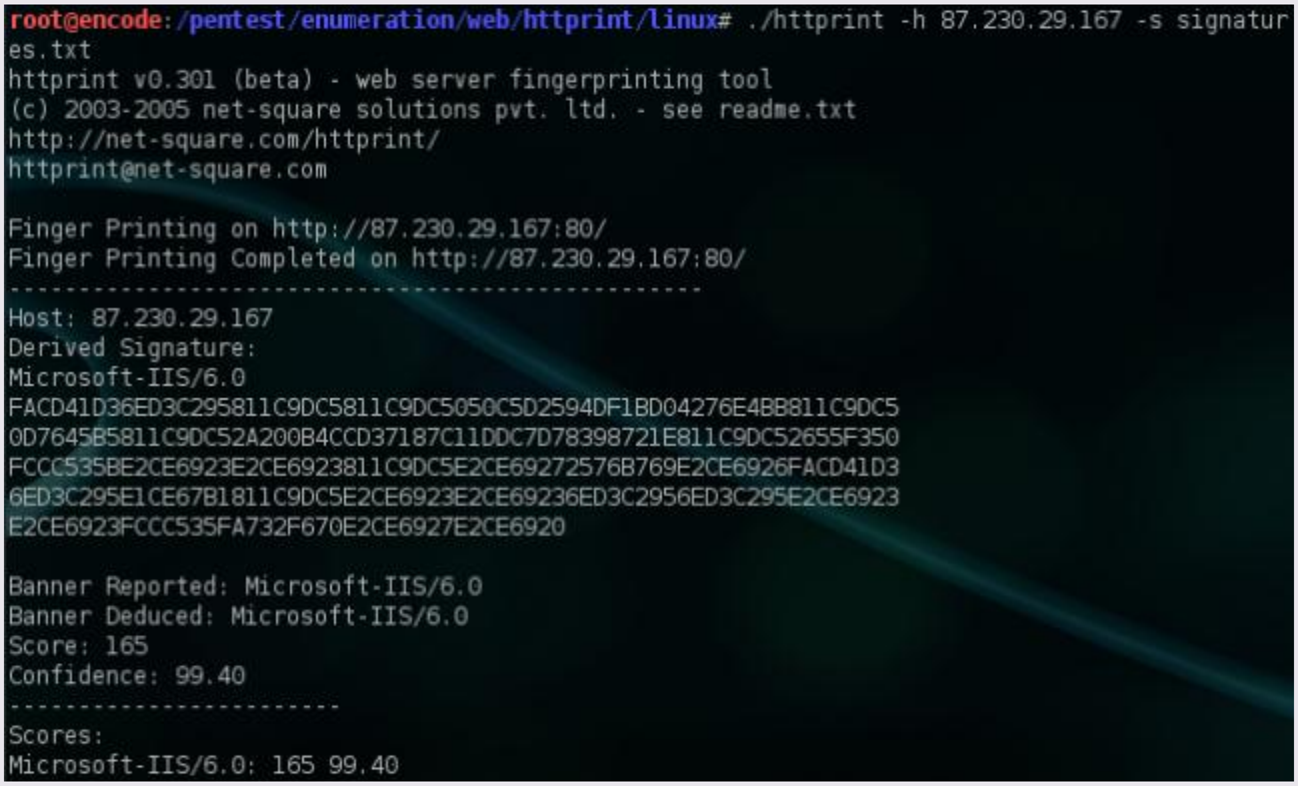
In some cases the version of the application can be discovered through source code inspection. So it is always a good practice to look there as well. You can see in the following example that we have discovered that the application is WordPress 3.3.2 version by looking at the Meta tag.



```
<meta name="generator" content="WordPress 3.3.2" />

<!-- All in One SEO Pack 1.6.14.3 by Michael Torbert of Semper Fi Web Design[-1,-1] -->
<link rel="canonical" href="http://www.ntobjectives.com/" />
<!-- /all in one seo pack -->
        <script type="text/javascript">

          var _gaq = _gaq || [];
```

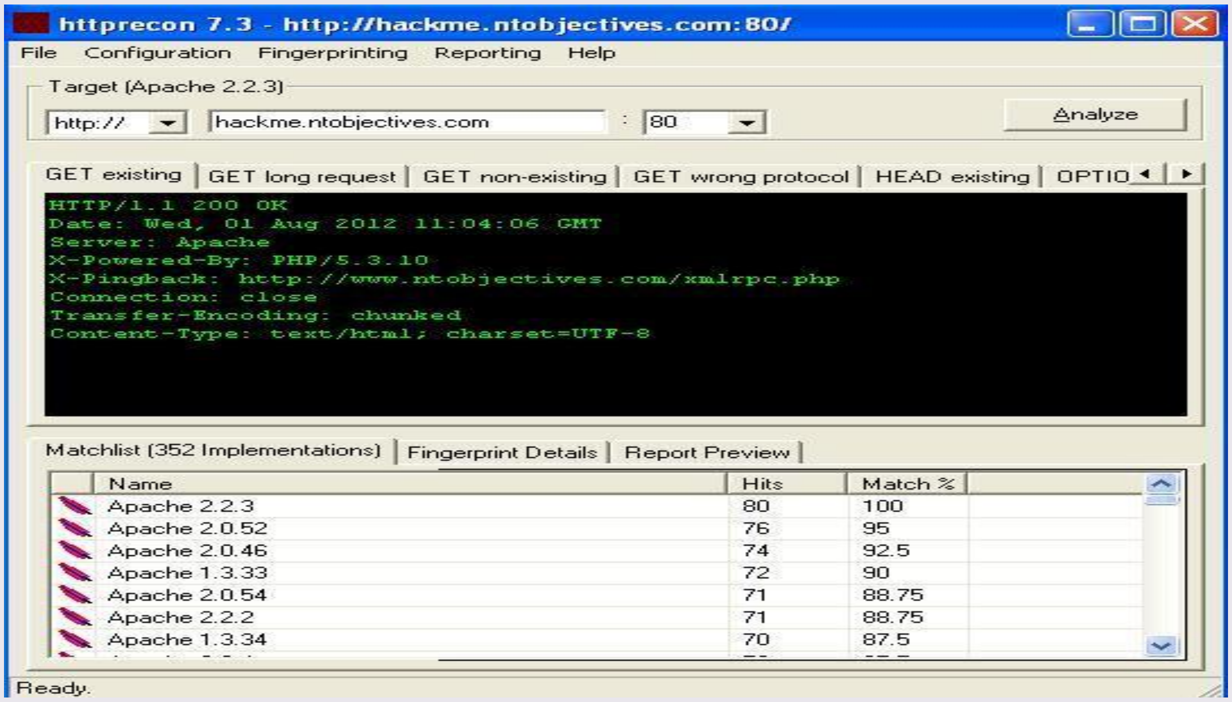
AUTOMATED FINGERPRINTING

Web application fingerprinting can be done as well with the use of automated tools that have been designed for that purpose. One of the most famous tools is of course the httprint. This tool comes with Backtrack but there is a version as well for windows. In the example below we will use a .txt file that contains signatures of different versions of web servers. So the httprint will try to match the signature of the target web server with the list of known signatures that the signature file contains in order to produce an accurate result.



```
root@encode:/pentest/enumeration/web/httprint/linux# ./httprint -h 87.230.29.167 -s signatur
es.txt
httprint v0.301 (beta) - web server fingerprinting tool
(c) 2003-2005 net-square solutions pvt. ltd. - see readme.txt
http://net-square.com/httprint/
httprint@net-square.com

Finger Printing on http://87.230.29.167:80/
Finger Printing Completed on http://87.230.29.167:80/
--------------------------------------------------
Host: 87.230.29.167
Derived Signature:
Microsoft-IIS/6.0
FACD41D36ED3C295811C9DC5811C9DC5050C5D2594DF1BD04276E4BB811C9DC5
0D7645B5811C9DC52A200B4CCD37187C11DDC7D78398721E811C9DC52655F350
FCCC535BE2CE6923E2CE6923811C9DC5E2CE69272576B769E2CE6926FACD41D3
6ED3C295E1CE67B1811C9DC5E2CE6923E2CE69236ED3C2956ED3C295E2CE6923
E2CE6923FCCC535FA732F670E2CE6927E2CE6920

Banner Reported: Microsoft-IIS/6.0
Banner Deduced: Microsoft-IIS/6.0
Score: 165
Confidence: 99.40
------------------------
Scores:
Microsoft-IIS/6.0: 165 99.40
```

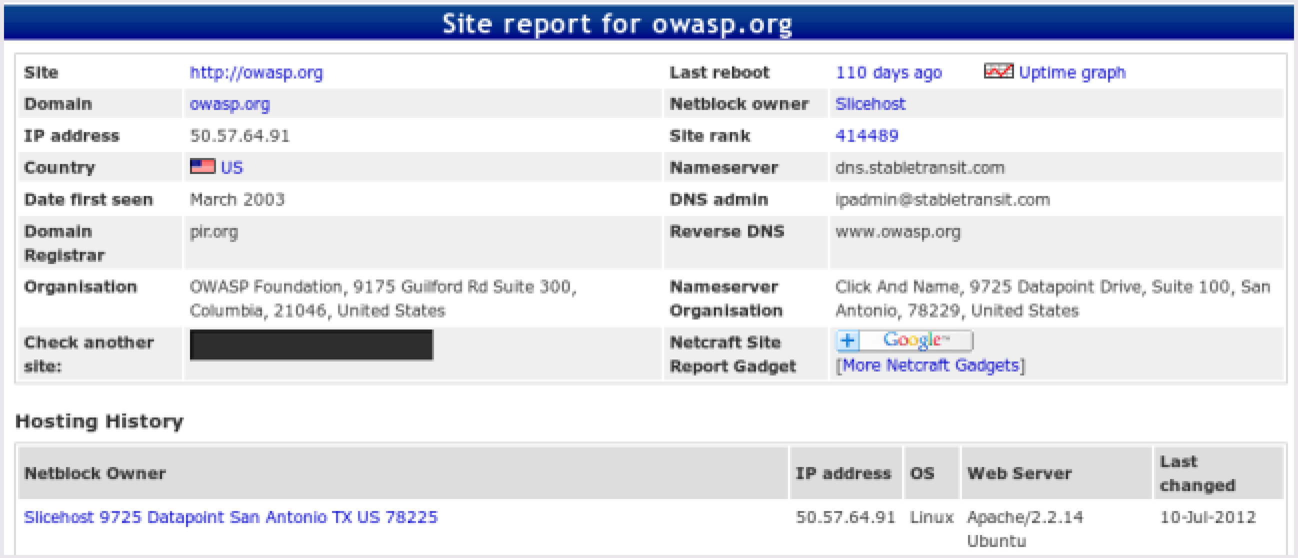
Another tool that performs pretty much the same job with the httprint is the httprecon. This tool is for windows platforms and it basically sends different kind of request to the target web server in order to identify its version. The image below is showing that we have a match 100% that host that we have scanned is running Apache 2.2.3 version.



```
HTTP/1.1 200 OK
Date: Wed, 01 Aug 2012 11:04:06 GMT
Server: Apache
X-Powered-By: PHP/5.3.10
X-Pingback: http://www.ntobjectives.com/xmlrpc.php
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Name	Hits	Match %
Apache 2.2.3	80	100
Apache 2.0.52	76	95
Apache 2.0.46	74	92.5
Apache 1.3.33	72	90
Apache 2.0.54	71	88.75
Apache 2.2.2	71	88.75
Apache 1.3.34	70	87.5

Also if we are performing an external web application penetration test then might also want to use an online tool which is called netcraft. This tool can retrieve also the headers of the web server and it can provide us with much more information including the operating system, the nameserver and the netblock owner and much more.



Site report for owasp.org

Site	http://owasp.org	Last reboot	110 days ago Uptime graph
Domain	owasp.org	Netblock owner	Slicehost
IP address	50.57.64.91	Site rank	414489
Country	US	Nameserver	dns.stabletransit.com
Date first seen	March 2003	DNS admin	ipadmin@stabletransit.com
Domain Registrar	pir.org	Reverse DNS	www.owasp.org
Organisation	OWASP Foundation, 9175 Guilford Rd Suite 300, Columbia, 21046, United States	Nameserver Organisation	Click And Name, 9725 Datapoint Drive, Suite 100, San Antonio, 78229, United States
Check another site:		Netcraft Site Report Gadget	+ Google [More Netcraft Gadgets]

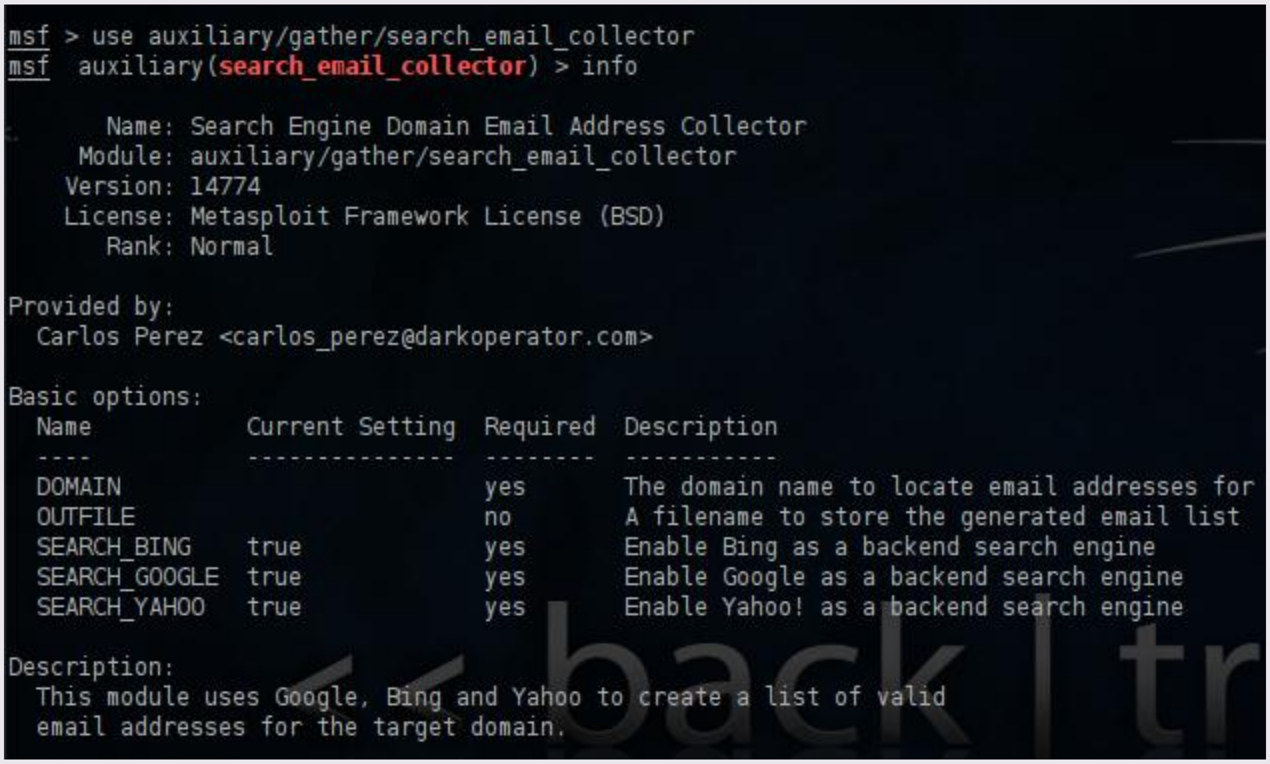
Hosting History

Netblock Owner	IP address	OS	Web Server	Last changed
Slicehost 9725 Datapoint San Antonio TX US 78225	50.57.64.91	Linux	Apache/2.2.14 Ubuntu	10-Jul-2012

PEOPLE SEARCH: PREPARE SOCIAL ENGINEERING ATTACK PROFILE

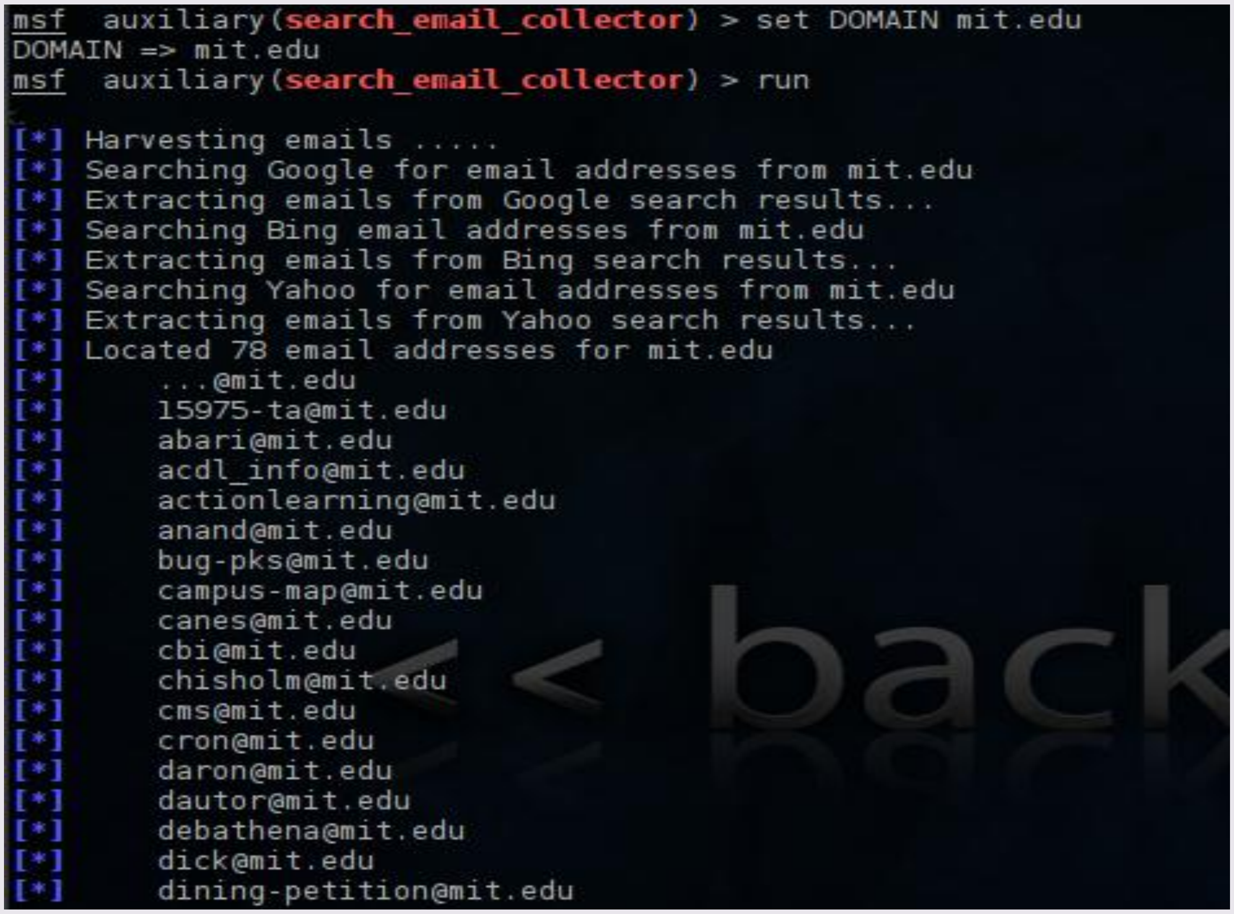
Without wasting time, I think it's better to show practically how to extract people information. Let's say that our client is the MIT (Massachusetts Institute of Technology) and we don't have any information about them. As a first step is to discover email addresses and profiles that exist on social media networks. We have two options in this step. We can use either the tool theHarvester or we can use the metasploit module called search_email_collector.

The use of the email collector module of the metasploit framework is pretty simple. We just need to set the domain of our target and it will automatically search through Bing, Yahoo and Google for valid email addresses.



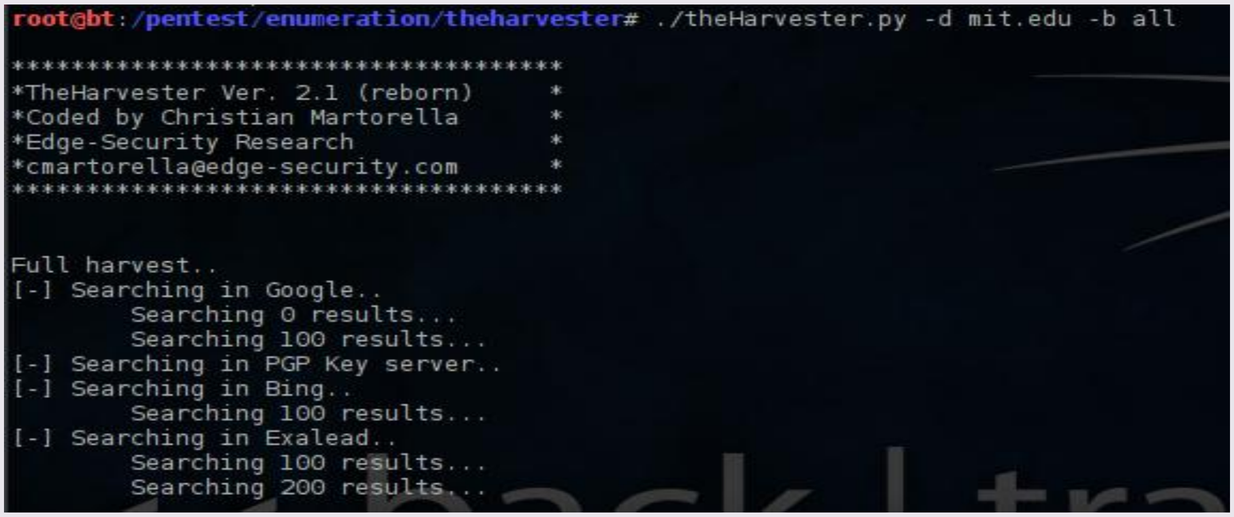
```
msf > use auxiliary/gather/search_email_collector
msf  auxiliary(search_email_collector) > info

       Name: Search Engine Domain Email Address Collector
     Module: auxiliary/gather/search_email_collector
    Version: 14774
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  Carlos Perez <carlos_perez@darkoperator.com>

Basic options:
  Name           Current Setting  Required  Description
  ----           ---------------  --------  -----------
  DOMAIN                          yes       The domain name to locate email addresses for
  OUTFILE                         no        A filename to store the generated email list
  SEARCH_BING    true             yes       Enable Bing as a backend search engine
  SEARCH_GOOGLE  true             yes       Enable Google as a backend search engine
  SEARCH_YAHOO   true             yes       Enable Yahoo! as a backend search engine

Description:
  This module uses Google, Bing and Yahoo to create a list of valid
  email addresses for the target domain.
```

Our target in this case is the MIT so the domain that we want to set is the mit.edu. Below is a sample of our results.



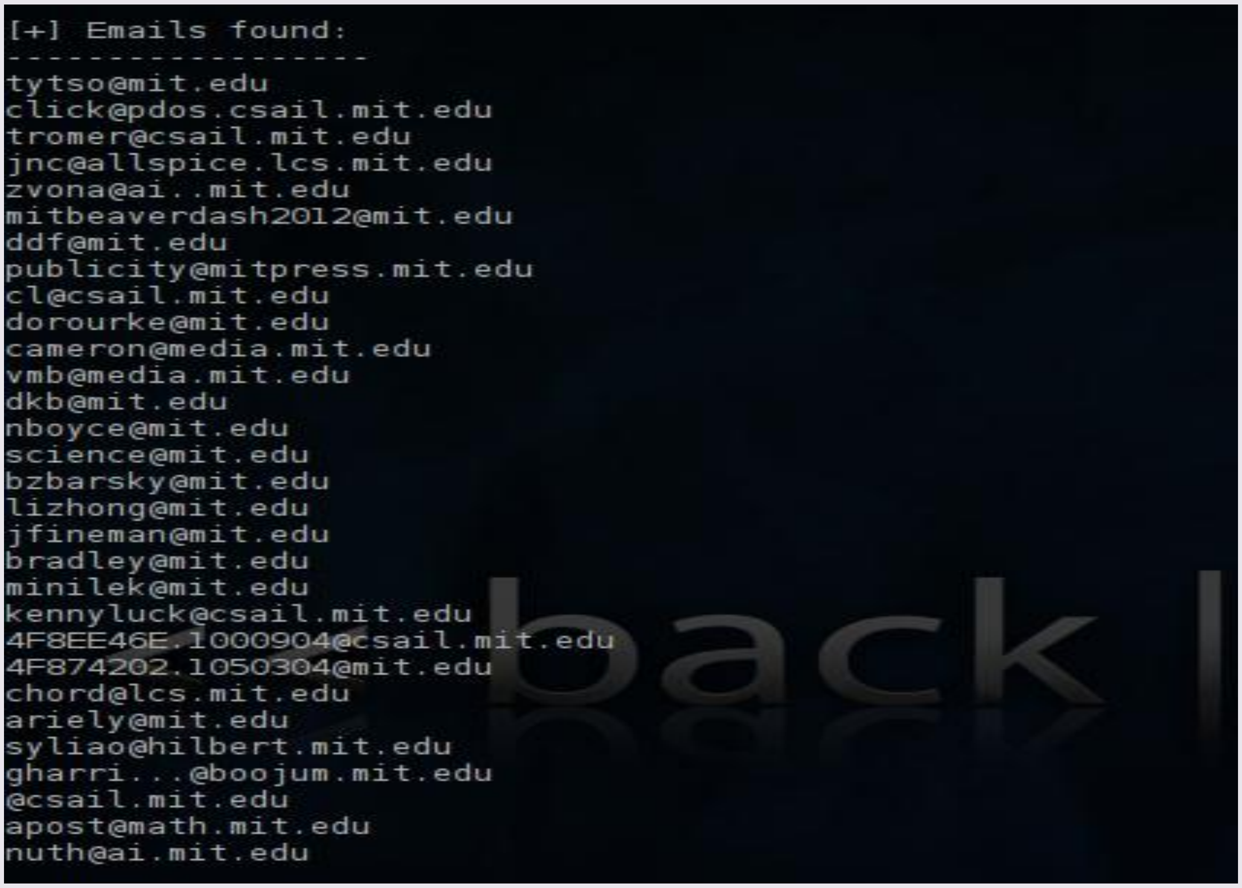
```
msf  auxiliary(search_email_collector) > set DOMAIN mit.edu
DOMAIN => mit.edu
msf  auxiliary(search_email_collector) > run

[*] Harvesting emails .....
[*] Searching Google for email addresses from mit.edu
[*] Extracting emails from Google search results...
[*] Searching Bing email addresses from mit.edu
[*] Extracting emails from Bing search results...
[*] Searching Yahoo for email addresses from mit.edu
[*] Extracting emails from Yahoo search results...
[*] Located 78 email addresses for mit.edu
[*]     ...@mit.edu
[*]     15975-ta@mit.edu
[*]     abari@mit.edu
[*]     acdl_info@mit.edu
[*]     actionlearning@mit.edu
[*]     anand@mit.edu
[*]     bug-pks@mit.edu
[*]     campus-map@mit.edu
[*]     canes@mit.edu
[*]     cbi@mit.edu
[*]     chisholm@mit.edu
[*]     cms@mit.edu
[*]     cron@mit.edu
[*]     daron@mit.edu
[*]     dautor@mit.edu
[*]     debathena@mit.edu
[*]     dick@mit.edu
[*]     dining-petition@mit.edu
```

From the other hand the tool theHarvester is providing us with more options. So except of the fact that we can scan for email addresses, we can scan also for profiles in social media like Google+ and Linkedin. In the next image you can see the command that we have executed in the tool.



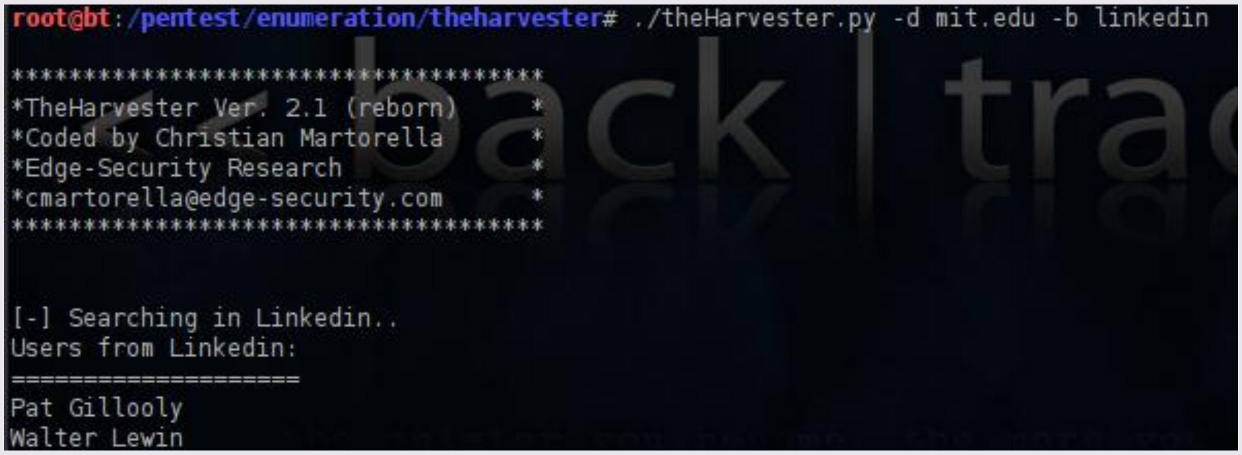
```
root@bt:/pentest/enumeration/theharvester# ./theHarvester.py -d mit.edu -b all

*************************************
*TheHarvester Ver. 2.1 (reborn)     *
*Coded by Christian Martorella      *
*Edge-Security Research             *
*cmartorella@edge-security.com      *
*************************************


Full harvest..
[-] Searching in Google..
        Searching 0 results...
        Searching 100 results...
[-] Searching in PGP Key server..
[-] Searching in Bing..
        Searching 100 results...
[-] Searching in Exalead..
        Searching 100 results...
        Searching 200 results...
```

Below is a sample of the email addresses that the tool theHarvester has discovered. Of course we can combine the results with the module of the metasploit if we wish.



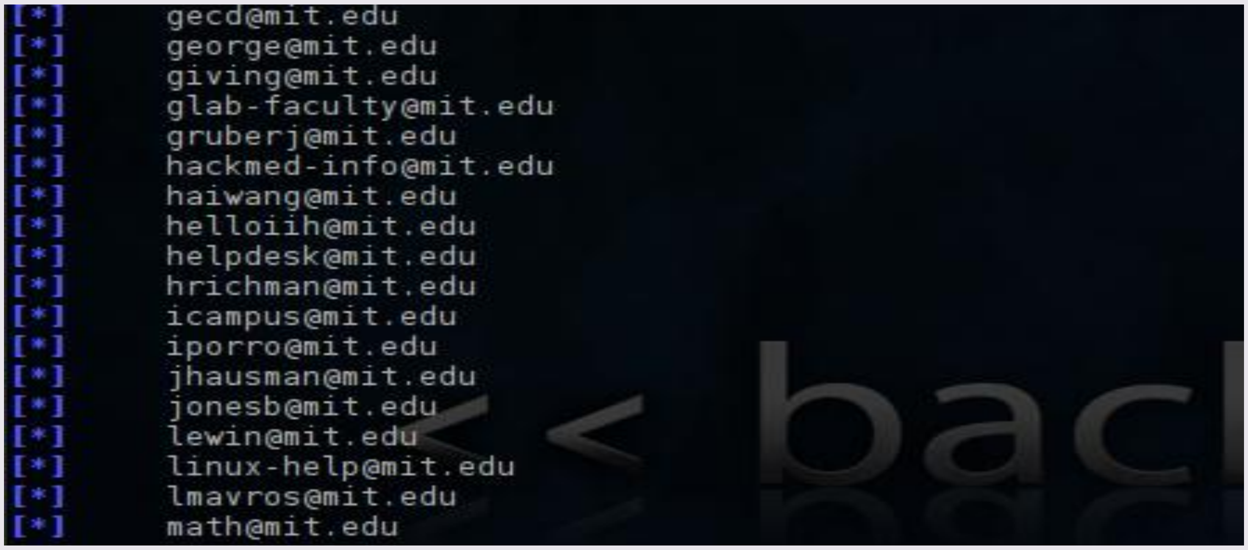
```
[+] Emails found:
------------------
tytso@mit.edu
click@pdos.csail.mit.edu
tromer@csail.mit.edu
jnc@allspice.lcs.mit.edu
zvona@ai..mit.edu
mitbeaverdash2012@mit.edu
ddf@mit.edu
publicity@mitpress.mit.edu
cl@csail.mit.edu
dorourke@mit.edu
cameron@media.mit.edu
vmb@media.mit.edu
dkb@mit.edu
nboyce@mit.edu
science@mit.edu
bzbarsky@mit.edu
lizhong@mit.edu
jfineman@mit.edu
bradley@mit.edu
minilek@mit.edu
kennyluck@csail.mit.edu
4F8EE46E.1000904@csail.mit.edu
4F874202.1050304@mit.edu
chord@lcs.mit.edu
ariely@mit.edu
syliao@hilbert.mit.edu
gharri...@boojum.mit.edu
@csail.mit.edu
apost@math.mit.edu
nuth@ai.mit.edu
```

We can try also to scan for profiles related to the mit.edu into professional social networks like LinkedIn. We have discovered 2 profiles.



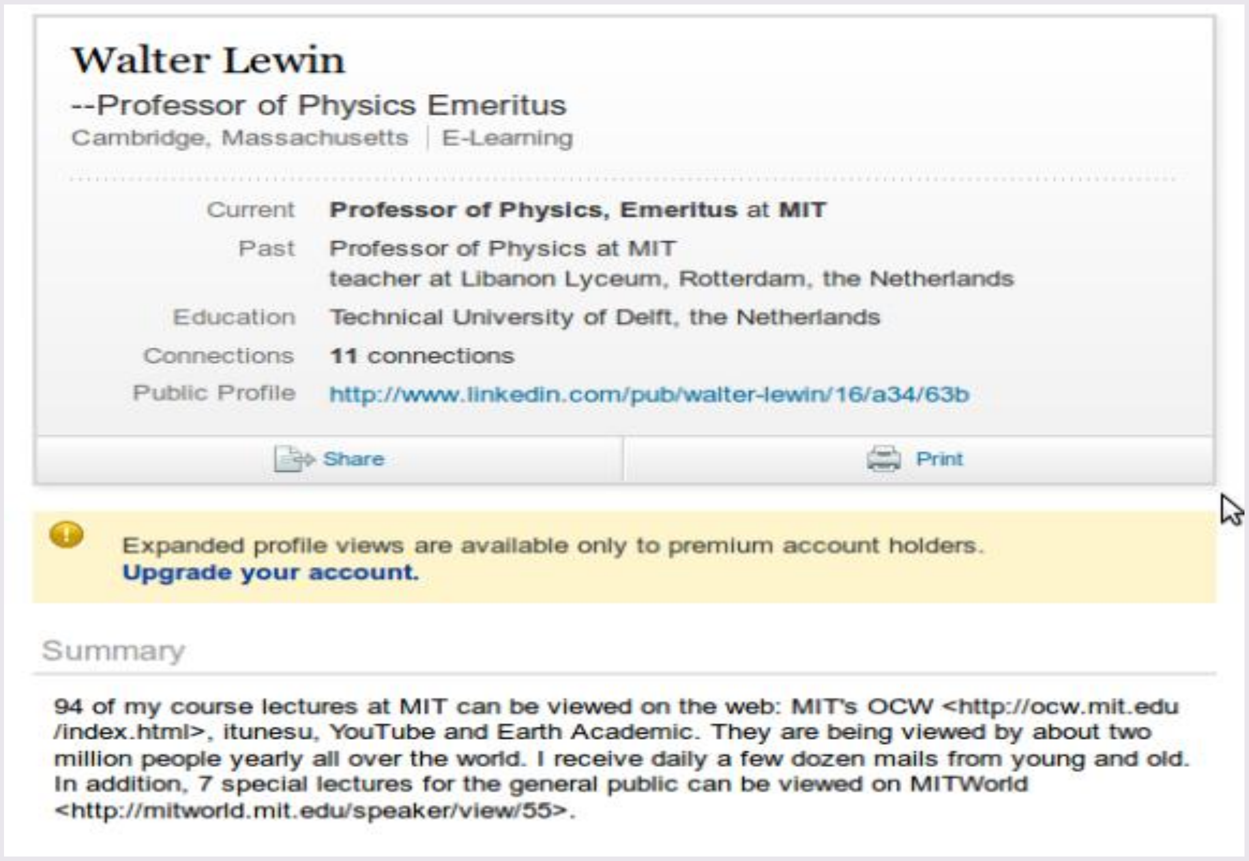
```
root@bt:/pentest/enumeration/theharvester# ./theHarvester.py -d mit.edu -b linkedin

*************************************
*TheHarvester Ver. 2.1 (reborn)     *
*Coded by Christian Martorella      *
*Edge-Security Research             *
*cmartorella@edge-security.com      *
*************************************


[-] Searching in Linkedin..
Users from Linkedin:
====================
Pat Gillooly
Walter Lewin
```

So we have a lot of email addresses and two names. Comparing the results with the metasploit module email collector we have identify that there is an email address that is probably corresponds to the Walter Lewin profile. The email address is lewin@mit.edu and you can see it in the results below.



```
[*]      gecd@mit.edu
[*]      george@mit.edu
[*]      giving@mit.edu
[*]      glab-faculty@mit.edu
[*]      gruberj@mit.edu
[*]      hackmed-info@mit.edu
[*]      haiwang@mit.edu
[*]      helloiih@mit.edu
[*]      helpdesk@mit.edu
[*]      hrichman@mit.edu
[*]      icampus@mit.edu
[*]      iporro@mit.edu
[*]      jhausman@mit.edu
[*]      jonesb@mit.edu
[*]      lewin@mit.edu
[*]      linux-help@mit.edu
[*]      lmavros@mit.edu
[*]      math@mit.edu
```

Now that we have a name and an email address it is much easier to search the web in order to collect as much information as possible about this particular person. For example we can start by checking his Linkedin profile.


Walter Lewin

--Professor of Physics Emeritus

Cambridge, Massachusetts | E-Learning

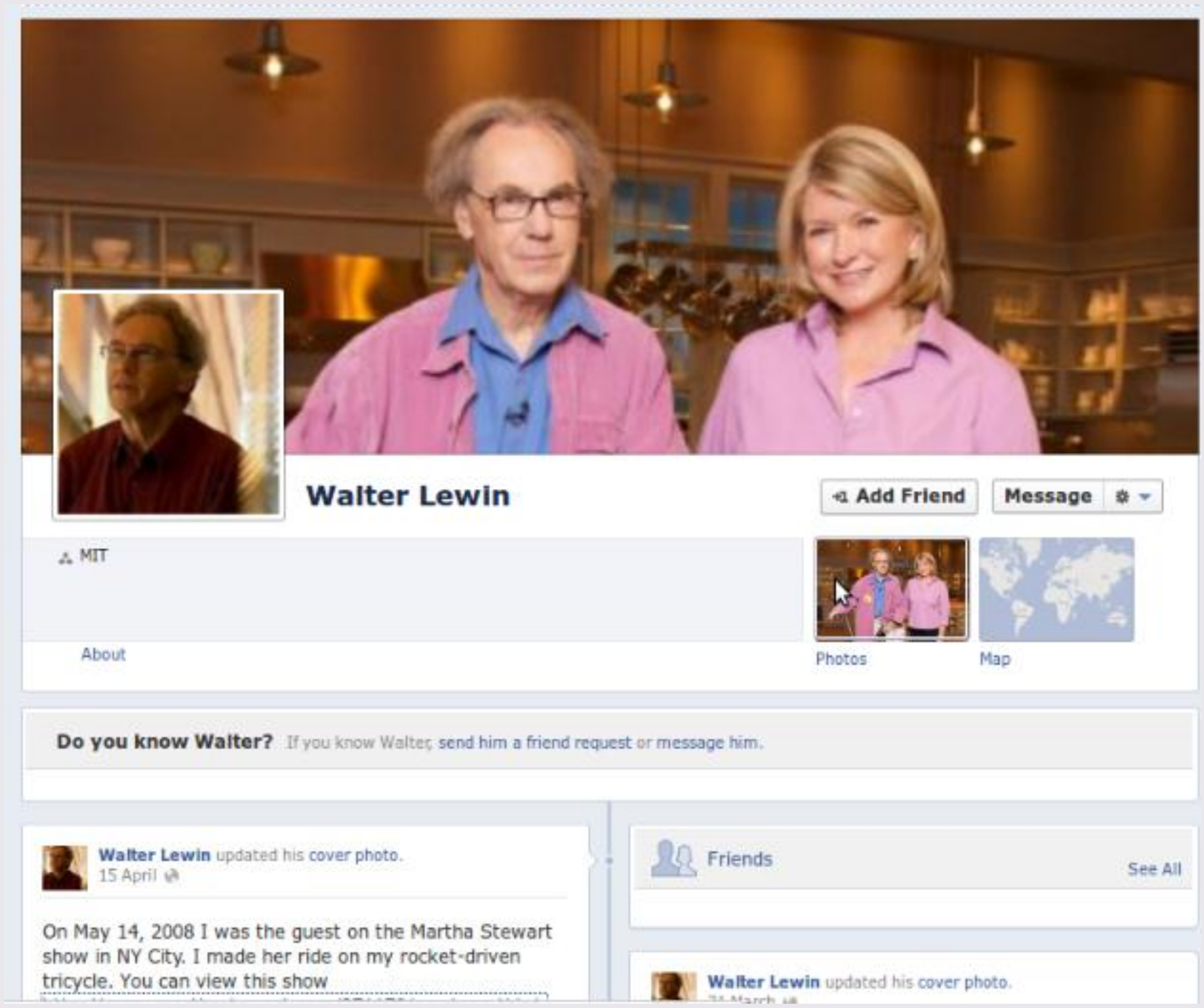
Current	**Professor of Physics, Emeritus** at **MIT**
Past	Professor of Physics at MIT teacher at Libanon Lyceum, Rotterdam, the Netherlands
Education	Technical University of Delft, the Netherlands
Connections	**11** connections
Public Profile	http://www.linkedin.com/pub/walter-lewin/16/a34/63b

Expanded profile views are available only to premium account holders.
Upgrade your account.

Summary

94 of my course lectures at MIT can be viewed on the web: MIT's OCW <http://ocw.mit.edu/index.html>, itunesu, YouTube and Earth Academic. They are being viewed by about two million people yearly all over the world. I receive daily a few dozen mails from young and old. In addition, 7 special lectures for the general public can be viewed on MITWorld <http://mitworld.mit.edu/speaker/view/55>.

We can use the email address lewin@mit.edu to discover his Facebook profile.



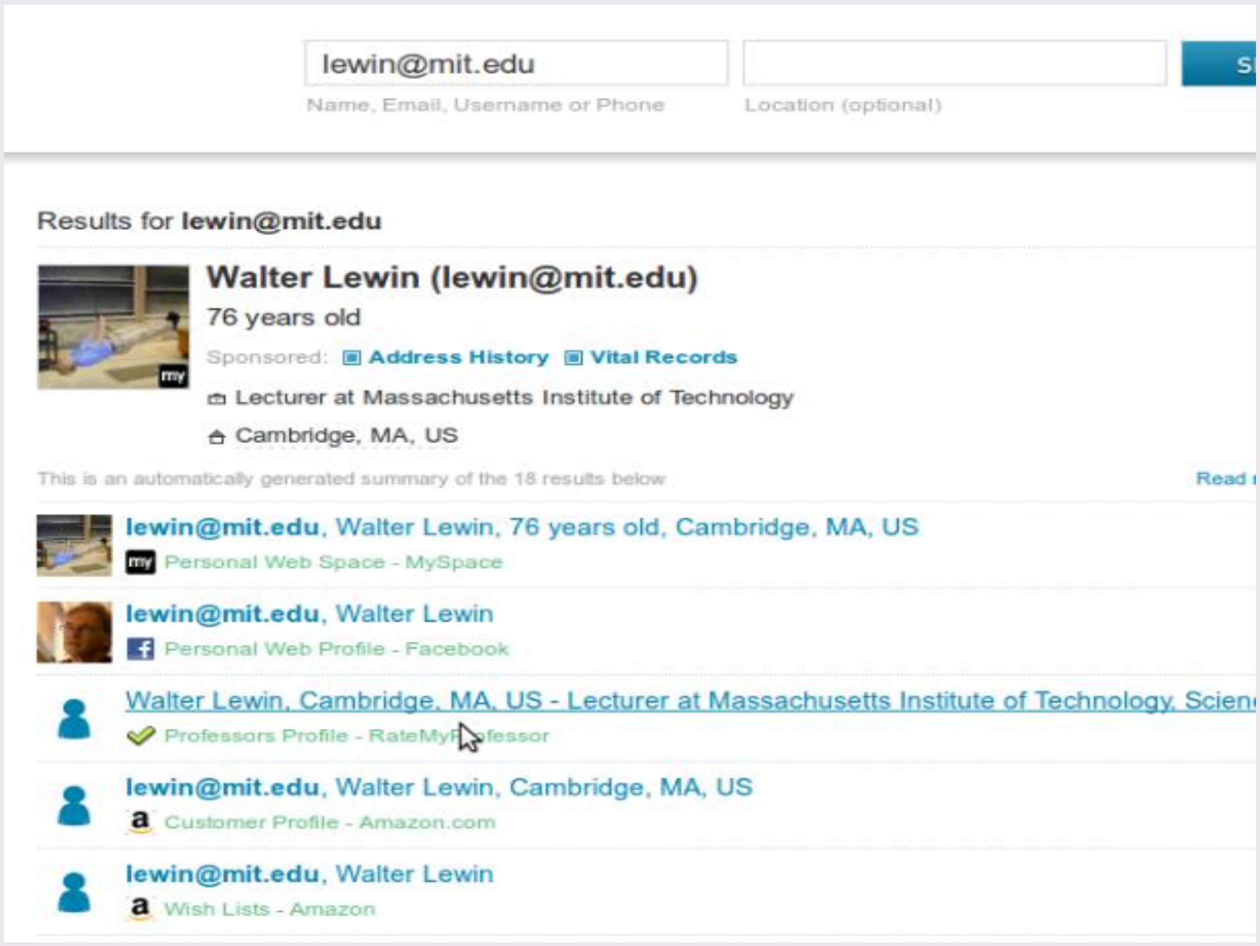
Do you know Walter? If you know Walter, send him a friend request or message him.

Walter Lewin updated his cover photo.
15 April

On May 14, 2008 I was the guest on the Martha Stewart show in NY City. I made her ride on my rocket-driven tricycle. You can view this show

Walter Lewin updated his cover photo.

The information that we can retrieve without being friends on Facebook with is limited. However if we impersonate ourselves as a teacher of MIT we can send a friend request and we might be able to convince him with this way to add us to his friend list so we can have access to much more personal information. Another good tool for obtaining information is through the website pipl.com.


lewin@mit.edu

Name, Email, Username or Phone

Location (optional)

Results for **lewin@mit.edu**

Walter Lewin (lewin@mit.edu)

76 years old

Sponsored: **Address History** **Vital Records**

Lecturer at Massachusetts Institute of Technology

Cambridge, MA, US

This is an automatically generated summary of the 18 results below

lewin@mit.edu, Walter Lewin, 76 years old, Cambridge, MA, US
Personal Web Space - MySpace

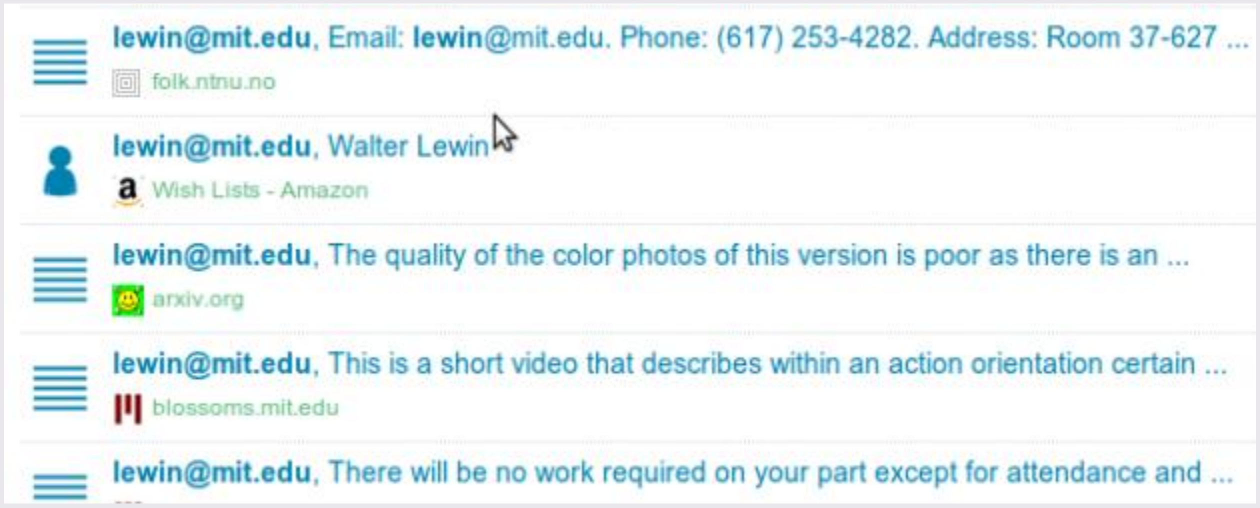
lewin@mit.edu, Walter Lewin
Personal Web Profile - Facebook

Walter Lewin, Cambridge, MA, US - Lecturer at Massachusetts Institute of Technology, Scien
Professors Profile - RateMyProfessor

lewin@mit.edu, Walter Lewin, Cambridge, MA, US
Customer Profile - Amazon.com

lewin@mit.edu, Walter Lewin
Wish Lists - Amazon

As you can see we have discovered information about the age, the job, the personal web space, his Amazon wish list and a website that contains the profile about this professor. Also from the same search we have manage to find his work phone number and his office room.


lewin@mit.edu, Email: **lewin**@mit.edu. Phone: (617) 253-4282. Address: Room 37-627 ...
folk.ntnu.no

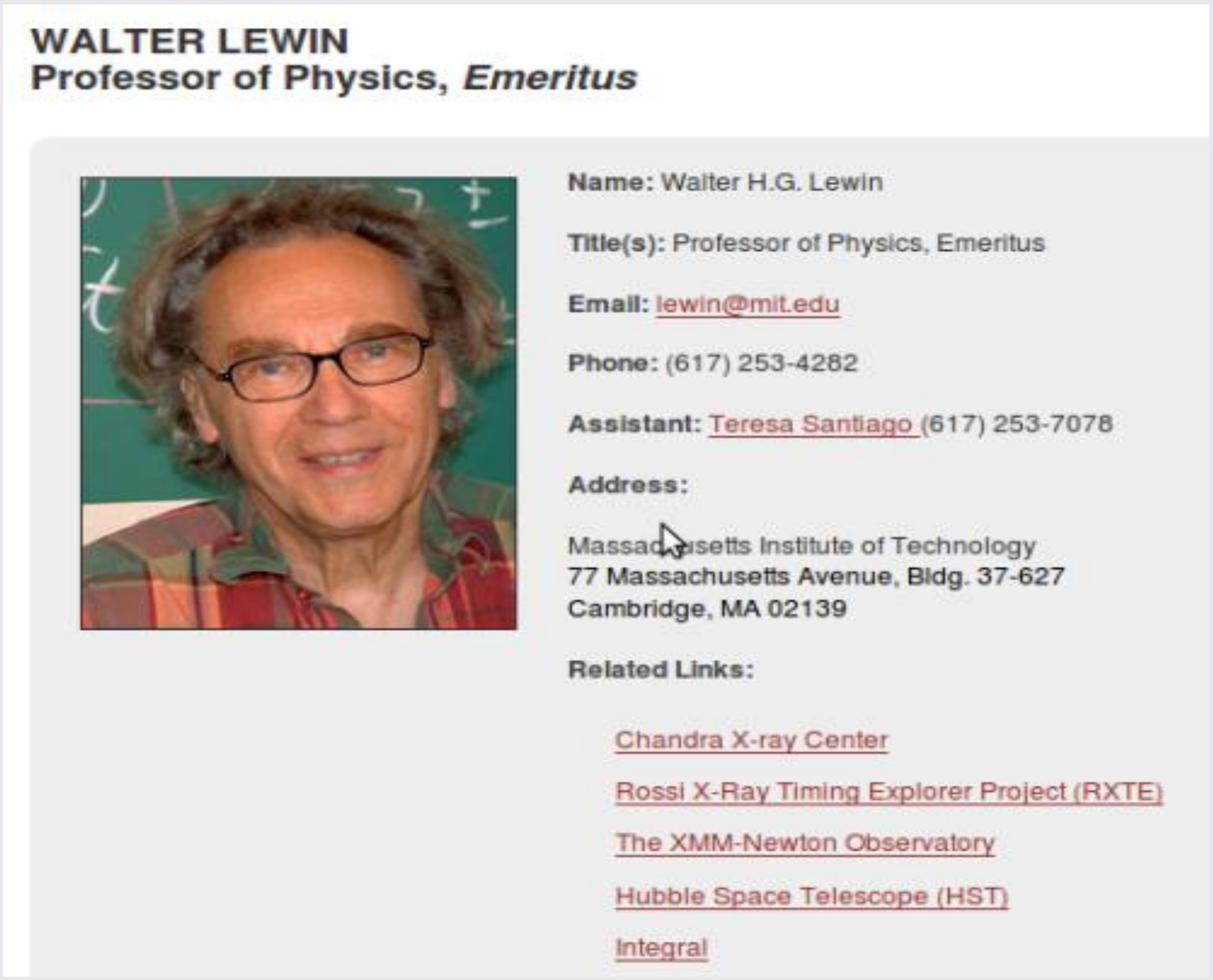
lewin@mit.edu, Walter Lewin
Wish Lists - Amazon

lewin@mit.edu, The quality of the color photos of this version is poor as there is an ...
arxiv.org

lewin@mit.edu, This is a short video that describes within an action orientation certain ...
blossoms.mit.edu

lewin@mit.edu, There will be no work required on your part except for attendance and ...

We can verify the above details by simply discovering his personal web page of the MIT.



WALTER LEWIN
Professor of Physics, *Emeritus*

Name: Walter H.G. Lewin

Title(s): Professor of Physics, Emeritus

Email: lewin@mit.edu

Phone: (617) 253-4282

Assistant: Teresa Santiago (617) 253-7078

Address:

Massachusetts Institute of Technology
77 Massachusetts Avenue, Bldg. 37-627
Cambridge, MA 02139

Related Links:

- Chandra X-ray Center
- Rossi X-Ray Timing Explorer Project (RXTE)
- The XMM-Newton Observatory
- Hubble Space Telescope (HST)
- Integral

From the above image except of the phone numbers and the addresses we have discovered also and the assistant of this professor. This can help us in many ways like: we are sending him an email pretending that it comes from his assistant. The professor will think that it came from a person that he trusts so he will respond to our questions more easily. Basically the idea when constructing a profile of the person that you will use your social engineering skills is to have as much information as possible about his interests and activities, his friends and colleagues, emails and phone numbers etc. Keeping all that information on your notebook will help you to construct an ideal scenario that will work.

Disclaimer

BEH appreciates highly the professor Mr. Walter Lewin and respects his work and contribution to the science and doesn't encourage in any way his readers to use this personal information in order to perform illegal activities against this person.

We hope you all Enjoyed Learning Hacking. Happy Learn. Let's wait for our Next Issue.

TOPIC OF NEXT ISSUE: ***INFORMATION GETTING – BACKTRACK SPECIAL***

REFERENCES:

1. **Wikipedia : Footprinting**
2. **Penetration test Lab**
3. **Backtrack Linux**
4. **OWASP: Information gathering**

[i] If a Hacker wants to get into your system then he will, what all you can do is that make his entry harder.