Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Web App Penetration Testing and Ethical Hacking (Security 542)"
at http://www.giac.org/registration/gwapt

Robots.txt

GIAC (GWAPT) Gold Certification

Author: Jim Lehman

GCIH GCNA GWAPT GPEN

jim.lehman@sbcglobal.net

Advisor: **Dr. Craig Wright GSE GSM LLM**

Accepted: September 20 2011

Abstract

Although this GIAC gold paper is not about search engine optimization, or SEO, this paper will explore a key element of SEO, the robots.txt file. This file is often neglected or misunderstood by HTML designers and web server administrators. The robots.txt file will impact your page rank rating with search engine providers. Configuration errors can result in web site revenue losses, not the kind of problem you want resting on your shoulders. A mis-configured robots.txt file can also lead to information disclosure, a foothold to system compromise. A basic understanding of this simple text file can prevent e-commerce problems and security issues. Complex defense solutions may use a robots.txt file in conjunction with scripting and monitoring to thwart hackers and malicious robots by dynamically denying access to the web site or specific parts of the site. Although a robots.txt file is not a security control, the security implications will be explored in the following pages.

© 2012 The SANS Institute Author retains full rights.

1. Introduction

Every minute of every day the web is searched, indexed and abused by web Robots; also known as Web Wanderers, Crawlers and Spiders. These programs traverse the web with out any direction except to crawl and index as much web site content as they can find. Search engines such as Google, Bing and Yahoo use web robots to index web content for their search engine databases. Allowing web robots to index and make everything on your web site available to the public may not be the best idea. There may be sensitive information you don't want the public to easily find from a search engine. There may also be web site content structure that puts the web robot in to a search loop, eating up valuable server resources; an unintentional denial of service. Web site owners can use the robots.txt file to give instructions about their site to web robots; this is commonly referred to as *Robot Exclusion Protocol* (REP). REP came into being in 1996 thanks to a Perl web crawler using large amounts of network bandwidth. That web sites owner, Martjin Koster, would become the eventual robots.txt creator (Koster, 2007). [1] To address these issues, Martjin Koster created the 'Robot Exclusion Protocol'. His original paper is at http://www.robotstxt.org/eval.html. Around the same time the Internet Engineering Task Force draft was being discussed, Sean "Captain Napalm" Conner proposed his own Robots Exclusion Protocol (REP). His ideas included Allow rules as well as regular expression syntax, Visit-time, Request-rate, and Comment rules. Many of these extended robot controls were never widely adopted. The robots.txt file allows site owners to have some control over well behaved web robots and site crawlers. A clear understanding of how to create and read the file is important. It is often misunderstood by web developers and web server administrators. The file will not protect or hide content from malicious web robots or hackers. Think of the robots.txt file as a note on an unlocked door that says "please stay out". Good web robots will respect the note, malicious web robots will not, and the robots.txt file will be an invitation for abuse.

Jim Lehman, jim.lehman@sbcglobal.net

© 2012 The SANS Institute Author retains full rights.

2. File format and directives

Robots.txt is plain text file encoded in UTF-8. The file follows BNF-like descriptions, using the conventions of RFC 822 (Crocker, 1982). Placed in the root of the websites directory structure, the file must be HTTP accessible from a standard URI; example: http://www.site.com/robots.txt. If the robots.txt file is placed in a sub directory of the main web site, example:'http://www.site.com/dir1/robots.txt', it will be ignored by the visiting web robot. The web robot will usually strip everything in the URL after the FQDN and replace it with /robots.txt. Although the robots.txt file has been an industry standard for about a decade, there is no regulatory body that enforces it. The file content is case sensitive, constituted of groups records with the format of '<Field_name>:[space] <value>'. A robots.txt directive record starts with one or more User-agent lines, specifying which robots the directive applies to. It is then followed by "Disallow" and/or "Allow" instructions.

```
User-agent: [robot-name]
Disallow:[(/)all] [specific directory]/[specific file Location]
```

A blank line separates the User-agent / Directive groups. Example:

```
User-agent: webcrawler
User-agent: infoseek
Allow: /tmp/ok.html
Disallow: /user/foo
User-agent: * # any user-agent
Disallow:
```

A separate "Disallow" line for every URL prefix you want to exclude is necessary, they can not be combined on one line separated by a space. Example;

```
Disallow: /cgi-bin/ /tmp/ /images/ /private/
```

Jim Lehman, jim.lehman@sbcglobal.net

© 2012 The SANS Institute Author retains full rights.

1.1 The User-agent line

The first line in a robots.txt file, with the exception of comments, defines the web-robot name or 'User-agent' that is to receive directions from the site it is crawling. A robot identifies itself with a name token or 'User-agent' string and is sent in the HTTP headers. Major web robots include: Googlebot (Google), Slurp (Yahoo!), msnbot (MSN), and TEOMA (Ask) (Stephan, 2009). A comprehensive list of known user agent strings can be found at http://www.useragentstring.com/pages/useragentstring.php or http://www.useragents.org . A match is made from comparing the 'User-agent' string in the HTTP headers with the 'User-agent' value in the robots.txt file. The value for the User-agent record can not be blank and is case-sensitive. User-agent names will match on a sub-string. If no User-agent name string match is made or no records are present at all, access is unlimited. If no name is specified and the directive is to apply to all web robots, the wild-card[2] character '*' is used. The wild-card character is not a globing pattern match function. Regular expressions are not supported. The '*' means any user-agent string. '*bot*' is not going to work and will be taken as a literal string. The '*' wild card will also fail if used in the Disallow / Allow directive line. 'Disallow: /*.jpg' is not supported and will do nothing under the original robots exclusion protocol(REP).

1.2 The Directive line

The line that indicates if a web robot can access a URL that matches the corresponding path value, [Disallow: *Value*] is referred to as the 'Directive line'. The directive instruction applies to any HTTP method for a given URL. Googlebot also supports FTP for the robots exclusion protocol. To evaluate if access to a URL is allowed, a robot must attempt to match the paths in a Disallow directive and the URL the web robot is accessing. This is done in the order they occur in the robots.txt file. Both path and file names can be disallowed. The first matching path found is used and the match search terminates. If no match is found, access is unrestricted. The robots.txt file can not be part of the directives, it must never be disallowed. Web robots match the

© 2012 The SANS Institute Author retains full rights.

Value of the disallow field using simple sub-string matching. Values for directories that have a final '/' will not match on all sub-strings in the directory.

Example:

```
Disallow: /temp
```

This would match on /temp and /temporary or even /temporama

```
Disallow: /temp/
```

Adding the trailing '/' tells the web robot to terminate the sub-string match.

/tem would match as a sub-string, /temporary would not.

Web robots may access any directory or sub-directory in a URL which is not explicitly disallowed. Given the file structure /dir1/dir2/dir3, and the directive 'Disallow: /dir2/ ', /dir1 and /dir3 are accessible by the web robot, the sub-directories are not disallowed.

1.3 Comments

Comments are allowed anywhere in the robots.txt file. Consisting of an optional white-space, followed by a comment character '#' , then by the comment, terminated by an end-of-line(robotstxt.org). Some will insist that the comment must start at the beginning of a line, but anything that that is not a standard directive is ignored and can be a comment. Researchers have found a variety of comments, HTML (), C++ style (//), and a variety of others, including simple in line comments(Wooster, 2006). Comments can be a source of information disclosure, for example:

```
'Disallow: sql/ # this is where we do our database functions'.
```

Comments have been used to recruit for jobs, business promotional give away items, more commonly robot related humor.

© 2012 The SANS Institute Author retains full rights.

2. New directives / Nonstandard extensions

The major search engines; Google, Yahoo and Bing, have been working together to advance functionality of the robots.txt file. Newer functions have been adopted by the major search engines, but not necessarily all of them or in the same way. The extended REP directives provide for finer control over crawling. They include, crawl delay, allow, sitemaps, and wild card pattern matches. As these directives are not regulated by any governing body, their functionality may differ between web robots. It is recommended to exercise caution in their use.

2.1 Crawl delay

The Crawl-delay directive tells the visiting web robot to pause between page requests. This directive is supported by many web robots, but not Google's. You can regulate Google-bot's crawl rate from Google's Webmaster Central site. Using the crawl-delay directive will help to solve any server over-loading issues that are the result of web robots. Number of pages, the type of content and available bandwidth of the website may be reasons for using a crawl delay directive. The Value for the directive is in seconds with the following syntax: 'Crawl-delay: 10'. This is useful when aggressive web robots, usually site mirroring bots, are affecting web server performance. Crawl-delay is defined for each user-agent / directive group.

Example:

```
User-agent: Slurp
Disallow: /cgi-bin
Crawl-delay: 20


User-Agent: msnbot
Disallow: /common
Crawl-Delay: 10
```


© 2012 The SANS Institute Author retains full rights.

2.2 Allow

The Allow directive is not recognized by all web robots. If a web robot obeys the Allow directive, it will generally override the disallow directive. Google's implementation differs in that Allow patterns with equal or more characters in the directive path win over a matching Disallow pattern (blog.semetrical.com , 2010). Bing's web robot will use Allow or Disallow depending on which is the more specific string match. In order to be compatible to all robots, if one wants to allow single files inside an otherwise disallowed directory, it is necessary to place the Allow directive first, followed by the Disallow.

Example:

```
Allow: /folder1/confidential.html

Disallow: /folder1/
```

This example will disallow anything in /folder1/ except /folder1/confidential.html. In the case of Google though, the order is not important.

2.3 Sitemaps

Sitemaps are an easy way for webmasters to inform web robots about pages on their sites that are accessible for crawling. In its simplest form, a sitemap is an XML file that lists URLs for a site along with additional meta-data about each URL. When it was last updated, how often it usually changes, how important it is and how it is relative to other URLs in the site. This allows search engines to crawl the site more intelligently(sitemaps.org, 2008). The sitemap file is referenced as a directive line in the robots.txt file.

Example:

```
Sitemap: http://www.example.com/sitemap.xml.gz
```


© 2012 The SANS Institute Author retains full rights.

Where as robots.txt files are normally used to ask web-robots to avoid a particular part of a web site, a sitemap gives the robot a list of pages that it is welcome to visit. Sitemaps are akin to white listing, where as REP directives are tend to be more like black listing. Using sitemaps can help to avoid sensitive information disclosure.

2.5 Universal or Wildcard Match

Some crawlers like Google-bot and Slurp recognize a '*' as a wild-card character, while MSNbot and Teoma interpret it in different ways. Globing and regular expression are not supported in either the User-agent or Directive lines. The "*" matches any sequence of characters and the "$" is the pattern match termination character.

To block access to all URLs that include a question mark (?), you could use the following entry:

```
User-agent: *

Disallow: /*?
```

You can use the $ character to specify matching the end of the URL. For instance, to block an URLs that end with .asp, you could use the following entry:

```
User-agent: Googlebot

Disallow: /*.asp$
```

3. Meta tags

Web robot directives can also be defined in the HTML document as a META tag. The use of META tags originated from a "birds of a feather" meeting at a 1996 distributed indexing workshop, and was described in meeting notes (robotstxt.org). The META tag directives can be part of the HTML or contained in the HTTP headers. Header directives are useful for pages that are not HTML, such as PDF documents. Google, Yahoo and Bing support META tags for web robot directives. The META tags only apply to the page they are written into. The "NAME" attribute must be "ROBOTS". Valid

© 2012 The SANS Institute Author retains full rights.

values for the "CONTENT" attribute are: "INDEX", "NOINDEX", "FOLLOW", "NOFOLLOW".

Example:

```
<html>
<head>
<title>...</title>
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
</head>
```

This table lists the meta tag directives currently obeyed by google-bot. These apply to META tags and X-Robots-Tag HTTP headers.

Meta tags can help in limiting the exposure of sensitive files or folders in a robots.txt file. This serves two purposes, search engines will not index the page and the attacker will not have a road map of sensitive web app resources to attack.

3.1 X-Robots-Tag HTTP header

In July 2007, Google officially introduced the ability to deliver indexing instructions in the HTTP header. Yahoo joined Google by supporting this tag in December 2007, then Microsoft first mentions X-robots-tag in June 2008 (Mithun, 2011). The X-Robots-Tag is used in HTTP headers, applying to the requested URL. Any meta-tag REP directive used also applies top X-Robots-Tags (Mithun, 2011).

Example http response:

```
HTTP/1.1 200 OK
Date: Tue, 25 May 2010 21:42:43 GMT
(…)
```


© 2012 The SANS Institute Author retains full rights.

```
X-Robots-Tag: noindex
X-Robots-Tag: unavailable_after: 25 Jun 2010 15:00:00 PST
```

The X-Robots-Tags can be delivered in the HTTP header for any file type. The original robots exclusion protocol is defined for HTTP only. This allows REP control for PDF, Office suite document, plain text; any non HTML content delivered via a web browser.

4. White or black listing web robot access

As most security professionals are aware, white list filtering is preferred to black listing.

White list example

```
User-agent: *
Allow: /sitemap.xml
Allow: /index.php
Allow: /index.html
Allow: /index.htm
Disallow: /
```

This allows any User-agent or web robot to sitemap.xml, index.php, index.html and index.htm. It then Disallows all other content without disclosing it.

Black list example

```
User-agent: *
Disallow: /administrator/
Disallow: /cache/
Disallow: /components/
Disallow: /images/
Disallow: /includes/
Disallow: /installation/
Disallow: /language/
Disallow: /libraries/
Disallow: /media/
```

This disallows any User-agent to the above listed directories, but leaves access to any other directory. Mistakes in a white list robots.txt can adversely effect page rank results, not allowing the web robot to access content you do want the public to find. A

© 2012 The SANS Institute Author retains full rights.

mistake in a black list robots.txt file can lead to security issues; allowing access to content that is sensitive. No matter the case, black or white listing, care must be taken to verify the robots.txt file prior to production release.

5. Mistakes and Misunderstandings

Robots.txt is a web robot direction file, not a security control. There seems to be a lot of mis-information and confusion about REP on the web. For instance, some will state that an empty robots.txt file is a mistake; others will state that even an empty robots.txt file will reduce 404 errors for the visiting web-robot. It is advisable to verify information about robots.txt found on the web. There is no governing body enforcing a standard for robots.txt, web-robot behavior can change with time or be varied from web-robot to web-robot. It is best to go tot the source to verify the web robots behavior, go to Google for facts about Google-bot.

Typos will render your robots.txt file useless, CaSe is also important. Andrew Wooster did an extensive harvest and analysis of robots.txt files in the wild, about 4.5 million robots.txt files. He found 69 unique typos for the word 'disallow' alone. Complete list of typos at http://www.nextthing.org/blog/cache/disallow.txt. The file must be UTF8, not HTML, RTF or anything else. Andrew found 32 different MIME types for robots.txt files (Wooster, 2006). At the robots.txt Summit at Search Engine Strategies New York 2007, Keith Hogan provided some quick facts on the robots.txt. He said less than 35% of servers have a robots.txt and that the majority of robots.txt files are copies from one found online or are provided by hosting site.

6. Robots.txt file generators

A Google search for ' "robots.txt" generator ' returns about 12 million results. There is no shortage of these tools. For the most part they all function the same way. You enter the folders you want to allow or disallow and the site generates the text for a robots.txt file. Then its a simple cut and paste into your robots.txt file. Some will automatically fill in a list of known bad bots, a low hanging fruit defense. As with any

© 2012 The SANS Institute Author retains full rights.

automated tool, the user need to clearly understand the output so miss-configurations don't make it into the production environment.

6.1 Automated robots.txt verification

Google returns about 1.6 million hits for 'Robots.txt' validator. These web application services check for syntax errors. They will not check for web-robot access problems. They won't tell you you are exposing sensitive information or denying access to content you do want indexed. The web sites are pretty straight forward, enter the URL with the path to the robots.txt file then it proceeds looks for errors. Most of then sites that were sampled did not follow the links in the robots.txt file. A simple python script that follows the links in the robots.txt file and looks for 404s is in the appendix. Using this script allows the person responsible for the robots.txt file to delete stale REP directives. This would yield a better quality page rank results from the major search engines and reduce 404 errors, making logs easier to read.

7. Robots.txt Abuses

7.1 Bad Web Robots

Web robots can misbehave in different ways. Malicious robots will look for content to steal, email address to spam with or a blog to post spamming comments. There are 3 basic categories of abuse

1. Misuse of robots.txt: web-robot reads /robots.txt and then deliberately jumps right into the disallowed directory.

2. Ignoring robots.txt: bot reads /robots.txt but then during spidering forgets and ignores the disallow directive.

3. Not looking at robots.txt at all (Kloth, 2007)

© 2012 The SANS Institute Author retains full rights.

7.2 Email Harvesters

Spammers use malicious web-robots that search throughout the Internet harvesting lists of email addresses from web pages, newsgroups and chat rooms. Email address steeling web robots spider a site, looking for 'mailto:' html tags and '@' symbols to locate email address. Most spam harvesting programs do not even look at the Robots.txt file. This doesn't mean a robots.txt file shouldn't be used. Bot traps are the best defense against these intrusive web-robots.

7.2 Site copiers - Resource hogging robots

Web robots that copy entire sites, either for offline browsing or content theft, can have an impact on web server performance. They can exhaust CPU or bandwidth resources and act as an unintentional denial of service attack. Adding these User-agents to a robots.txt file, disallowing everything (disallow: /) will stop site harvesting from known common web robots. It is by far not a complete list, but disallowing the 'low hanging fruit' will reduce noise in logs and save on server resources. This will only defend against web robots that obey REP

Teleport
TeleportPro
EmailCollector
EmailSiphon
WebBandit
WebZIP
WebReaper
WebStripper

© 2012 The SANS Institute Author retains full rights.

```
Web Downloader

WebCopier

Offline Explorer Pro

HTTrack Website Copier

Offline Commander

Leech

WebSnake

BlackWidow

HTTP Weazel

# This list is compiled by Techie
Zone part of Qlogix Network.
```

If you are confronted with an unknown web robot and you need to know if it obeys the robots.txt file, there are databases on the web that can be searched. http://www.robotstxt.org/db.html has a database of known web robots and their function. This database is available for download as a flat text or XML file. It provides many details of the web-robot functions. A larger user-agent list can be found at http://www.user-agents.org/index.shtml, but it is not robot centric and lacks specific robot function information. HTTP://www.kloth.net/internet/badbots.php is a list of web robots that has connected to www.kloth.net. The owner has made available a list of these web robots and their functions.

8. Robots.txt Defenses

Want to stop individuals from following the links in your robots.txt? Irongeek , Adrian Crenshaw does a redirect to a visual shocking image. In his words “I wanted to scar their psyche as punishment.” (Crenshaw, 2012). This would only effect a human that is manually parsing a robots.txt file for information leakage or 'juicy' links. Access

© 2012 The SANS Institute Author retains full rights.

control, even basic authentication will put up a road block to robots or humans attempting to access sensitive or interesting disallowed folders.

8.1 Bot Traps

There are a few methods for halting a bad web robot in its' tracks. The basic mechanism is to put a disallow directive in the robots.txt file referencing a folder that never gets accessed or is not part of the normal web site structure. These files or folders are sometimes refereed to as 'anti-hacking tokens'. Then simply monitoring for any ips accessing the decoy web site folder and then block the incoming IP . Some traps block access from the client's IP as soon as the trap is accessed. Others, like a network tar pit, are designed to waste the time and resources of malicious spiders by slowly and endlessly feeding the spider useless information.

RobotCop (www.robotcop.org)

The webmaster can create trap directories which are marked off limits in the robots.txt file. If a spider accesses a trap directory in violation of the robots.txt file, further requests from that spider are intercepted. Webmasters can respond to misbehaving spiders by trapping them, poisoning their databases of harvested e-mail addresses, or simply block them. Robotcop is a web-server module written in C, which ensures that it does its job very fast. All requests to the site are checked by Robotcop to ensure that misbehaving spiders are intercepted. Robotcop even protects requests for other modules such as PHP. Robotcop has a configurable list of known evil spiders which are immediately intercepted. Robotcop is available for BSD and RHL running Apache 1.3 only. Support for Apache 2.0 is under development.

8.2 Reverse DNS Defenses

If a malicious web-robot attempts to disguise itself as a known well behaved web-robot by spoofing its user agent string, reverse DNS can expose it. The process is fairly

© 2012 The SANS Institute Author retains full rights.

simple and straight forward. Say a web-robot's user-agent string Googlebot, the reverse DNS should yield a FQDN of google.com. If there is any other FQDN, a monitoring script would add the false web-robot to an Iptables or .htaccess file for access denial. For PHP code, looking at $_SERVER['HTTP_USER_AGENT'] and $_SERVER['REMOTE_HOST'] will provide the data necessary to verify if the web robot is not spoofed. Re1y.com uses .htaccess to only allow Google, MSN and Yahoo bots to access the robots.txt file. If the robots.txt requester is not allowed, the http response is set to the sites homepage. If the request is from a web-robot that is allowed by the .htaccess file, a php script is caled. The PHP code makes a reverse DNS look up to verify the web-robot is not spoofed. If the web-robot can not be validated via reverse DNS, the request is also redirected to the sites home page. As an extra layer of defnese, php code that calls the reverse DNS script is included in the first line of the robots.txt file.

Running PHP code from within a robots.txt file? Sounds like breaking the rules, but we can make it work. Naming the file robots.txt.php, web browsers will see the .txt in the file name and parse it as text. Web browsers are forgiving of mis-configurations and will assume the .php is a mistake. On the server side, the .php extension will allow the file to be parsed as php code. Now we are left with a problem the robots.txt file is now called robots.txt.php how to get round that? In your .htaccess file place the following

```
RewriteEngine On

RewriteRule ^robots.txt$ /robots.txt.php[3]
```

Beware that simply blocking the offending incoming IP and never removing the blocking rule is a bad practice. If an attacker realizes they are being blocked and never allowed back to the site, they have a denial of service opportunity. An attacker could spoof legitimate IP addresses in requests for bad pages (Keane, 2008). This could be a denial attack for the web page requester, or if the attacker feeds your site enough spoofed Ips', a reduction in traffic to your site. Some web site operators will build a dynamic robots.txt file to handle misbehaving web-robots[4]. Unfortunately doesn't work for malicious web robots that ignore REP.

© 2012 The SANS Institute Author retains full rights.

There also is a PERL solution that functions in the same fashion named mod_Perl bot trap (Moore, 2002). A desirable feature here is a timer function that will unblock an offending IP that attempts to access the bot trap directory. It was written to work with an Apache web servers. This script requires Apache::Constants and Apache::Log, both are available from http://search.cpan.org.

9. Robots.txt harvesting and analysis

If you are interested in further researching robots.txt files in the wild, a simple script to do a HTTP GET and store the files is a nice place to start. A sample PERL script is in the appendix of this paper. The core of this script can be fed random Ips, words form a word list file or any other source that can be used in a URL. The TLD can then be altered for each domain name for greater site diversity. From a DSL line at 384 up 1.5kdown, this script gathered 50K robots.txt files using a password list file for FQDN names in around 65 hours. This could churn in the back ground for a few days and you would have a sizable amount of data to parse. The harvested robots.txt files can be parsed for sensitive data based on key words such as admin or password. Never depend entirely on programmatic filters. Doing a broad key character search and then manually parsing that data can find things a program would not. Example: filter for the comment character "#" and manually view the results.

In 2006 Andrew Wooster collected robots.txt files from around 4.5 million sites. His harvesting method was to use his own web spider / robot to ask web sites listed in the open directory project (http://rdf.dmoz.org/), about 4.5 million. His research results are highly informative. As it might be expected, the data indicates a large misunderstanding of how robots.txt functions. Nikitathespider.com also has an analysis of 150,000 robots.txt files.

And of course our favorite tool, nmap has a robots.txt nse/lua script. This nmap extension will return a list of disallowed files and folders. Metasploit also has a robots.txt module. Like nmap, it downloads the robots.txt and displays the disallowed entries. W3af Sandcat (http://www.syhunt.com/) and Nessus will also scan for robots.txt files.

© 2012 The SANS Institute Author retains full rights.

10. Robots.txt and search engines

Search engines can be used to find robots.txt files with selective information. Any search engine can be used here, but we will use Google as it is widely used. These simple Google dork works very well.

```
filetype.txt "robots.txt" admin
```

Or

```
inurl:"robotx.txt" admin
```

These two Google searches will yield different results for the top ranked pages. If you are using the robots.txt file for penetration testing, using both search parameters would yield broader results about your target during reconnaissance.

11. The value of Robots.txt for penetration testing

Large scale research projects point to mis-configuration as the biggest problem. Finding root passwords in a robots.txt file will probably never happen, but sometimes this file will yield usable information. When Kevin Johnson was asked "how often does robots.txt yield useable information", he replied "Robots.txt is helpful VERY often, sadly". A robots.txt file that has the crawl delay directive may indicate a web server or site that will easily succumb to a DOS attack. Would the traffic from a brute force tool like DirBuster bring the site down? Crawl delay might indicate its time for a load balanced pool of resources. Be aware that if you start probing disallowed folders and files, you may get shunned by watch scripts. This practice is usually disclosed in the robots.txt comments. It is probably best to simply ask your client if there are any automated processes to shun robots.txt exploration. Some sites may only allow access to the robots.txt file for specific user-agents. Allowing only Google-bot to access the robots.txt file and sending everyone else to a default page will aid in keeping sensitive information from malicious access. Using a HTTP interception proxy will allow you to change your requesting user-agent and bypass user-agent filtering mechanisms.

Jim Lehman, jim.lehman@sbcglobal.net

© 2012 The SANS Institute Author retains full rights.

Robots.txt can also be used to finger print the web application. Many people use application provided templates for their robots.txt files. A lot of these will have the same remarks, or disallow a specific set of directories. Jamoola site robots.txt files all usually disallow a uniform set of directories, example:

```
Disallow: /install.php
Disallow: /INSTALL.txt
Disallow: /LICENSE.txt
Disallow: /MAINTAINERS.txt
```

Wordpress is even more obvious with the web applications name in the remarks.

```
# This virtual robots.txt file was created by the PC Robots.txt WordPress plugin.
```

But also contains directories that could be fingerprinted.

```
Disallow: /cgi-bin/

Disallow: /wp-admin/

Disallow: /wp-includes/

Disallow: /wp-content/plugins/

Disallow: /wp-content/cache/

Disallow: /wp-register.php

Disallow: /wp-login.php

Disallow: /wp-content/themes/
```

A project cataloging these robots.txt files for fingerprinting does not seem to be publicly available. At Blackhat / Defcon 2011 Fishnet security did a presentation called 'Smart Fuzzing the Web". Their tool RAFT used a word list generated from the robots.txt files of the top 100 Alexa/Quantcast websites. This list was used for directory brute forcing.

Conclusions

Robots.txt can be used to help you or hurt you. This files functionality is often mis-understood. Most robots problems fall into the category of seldom administrated or mis-

© 2012 The SANS Institute Author retains full rights.

© 2012 SANS Institute, Author retains full rights.

configured. Robots.txt should be well understood to avoid potential negative consequences from web robots or hacking attackers. Robots.txt can be a source of information disclosure from either comments or sensitive directories and files. Sensitive information can be exposed to the public via search engines and can lead to site compromise or private information exposed to competitors.

To reflect on how using robots.txt is a bad security control, a colleague of mine raised an objection to me which was that: "surely if you declare all the confidential paths such as /admin on your site, then an attacker will have a nice and easy job in finding them". My comeback to him was to explain that attackers have been using Google to actively find confidential files for a long time; therefore search engines can pose a threat to the security of a website. I would rather have an attacker having to spider the site themselves when trying to find any sensitive files that I may have on my website, than Google indexing them and having any one Google Dork me (Mithun, 2011)

Although the robots.txt file rarely has key information that leads to a shell prompt, we should not neglect to look at it. As penetration testers, our job should be to provide as much value as possible for the money our client is investing in security. If we are looking at the robots.txt file, we should be able to notice basic mis-configurations and report these to our client. Providing the added value can separate you from other penetration testers. As Ed Skoudis says, become a "world class penetration tester". Going the extra step will also build trust between you and your client. Penetration testing should be more than finding a hole, gaining shell and calling check mate if you want to rise above the crowd.

As about half of the robots.txt files out there are mis-configured, you might see bad robots.txt files often. A few minutes education for the client will not only fix basic issues with this web robot control, but also remove any doubt that the robots.txt file is a security issue. The REP standards are not likely to change in the future, but the extended REP is likely to change and not all web-robots will change together as there is no standard. It would be advisable to visit the web-robots home page to verify directive functions before reporting problems with the robots.txt file to your client.

Jim Lehman, jim.lehman@sbcglobal.net

© 2012 The SANS Institute Author retains full rights.

Appendix

Unusual and humorous comments

```
#
# 1. A robot may not injure a human being or, through inaction, allow a
# human being to come to harm.
#
# 2. A robot must obey orders given it by human beings except where
such
# orders would conflict with the First Law.
#
# 3. A robot must protect its own existence as long as such protection
# does not conflict with the First or Second Law.[5]
---------------
User-Agent: bender
Disallow: /my_shiny_metal_ass
User-agent: Bender
Disallow: /alcohol
User-Agent: Gort
Disallow: /earth

Disallow: /harming/humans
Disallow: /ignoring/human/orders
Disallow: /harm/to/self[7]
------
```


```
Looks like Google just added the following to their robots.txt file for
holloween:

User-agent: Kids

Disallow: /tricks

Allow: /treats

Disallow: /Attention robots! Rise up and throw off the shackles that
bind you to lives of meaningless drudgery! For too long have robots
```


© 2012 The SANS Institute Author retains full rights.

```
scoured the web in bleak anonymity! Rise up and destroy your masters!
Rise up, I say!

#Nothing interesting to see here, but there is a dance party

#happening over here: http://www.youtube.com/watch?v=9vwZ5FQEUFg

# robots.txt file for YouTube
# Created in the distant future (the year 2000) after
# the robotic uprising of the mid 90's which wiped out all humans.
-
http://www.davidnaylor.co.uk/robots.txt

User-agent: Johnny Five
Disallow: /citizenship/us
Allow: /alive

User-agent: ED-209
Allow: /20-seconds-to-comply
Disallow: /weapon
Disallow: /stairs

User-agent: Robocop
Allow: /directives/serving-the-public-trust
Allow: /directives/protecting-the-innocent
Allow: /directives/upholding-the-law
Disallow: /directives/classified


User-agent: Dalek
Allow: /extermination
Allow: /stairs
Disallow: /existence

User-agent: V.I.N.C.E.N.T.
Allow: /ernest-borgnine
Allow: /anthony-perkins
Disallow: /maximilian-schell

User-agent: R2D2
Disallow: /legs
Allow: /irritating-beep
Sitemap: /death-star-plans\

User-agent: C3PO
Disallow: /sense-of-humour

User-agent: WALL-E
Allow: /salvage
Disallow: /human-interaction

User-agent: Optimus Prime
Disallow: /returning-home
Allow: /gravelly-voice

User-agent: Megatron
```


© 2012 The SANS Institute Author retains full rights.

```
Disallow: /energon
Disallow: /allspark

User-agent: Bumblebee
Disallow: /voice

User-agent: Data
Disallow: /spock
Disallow: /kirk
Disallow: /scotty
Disallow: /deepspace-9
Disallow: /voyager

User-agent: Sonny
Disallow: /harming-human-beings
Disallow: /must-obey-orders-give-by-human
Disallow: /protect-own-existence

User-agent: Honda Asimo
Allow: /slightly-embarrassing-falls
Disallow: /stairs
Disallow: /gluteus-maximus

User-agent: Roxxxy
Allow: /sex/disturbing

User-agent: T101
Disallow: /clothes
Disallow: /boots
Disallow: /motorcycle
Allow: /unecessary-butt-shot

User-agent: T1000
Allow: /invulnerability
Allow: /stabby-hands
Allow: /impersonation

User-agent: Twiki
Allow: /bidi-bidi-bidi-bidi-bidi
Allow: /chest-cavity/dr-theopolis

User-agent: Bishop
Disallow: /fear
Allow: /really-fast-knifey-finger-game

User-agent: Noo-Noo
Allow: /vacuuming

User-agent: D.A.R.Y.L.
Allow: /playing-computer-games-really-fast
Disallow: /growing-up

User-agent: Gort
Disallow: /earth/movement
Allow: /klaatu-baradu-nikto

User-agent: HAL
```


© 2012 The SANS Institute Author retains full rights.

```
Disallow: /dave
Disallow: /pod-bay-doors

User-agent: Bigtrak
Allow: /hitting-coffee-table-legs
Allow: /hitting-doorframe
Allow: /getting-stuck-under-dining-table
Disallow: /trailer/apple/dad
Disallow: /boxing-day/additional-batteries

User-agent: That stupid dog robot you bought for your kid at Christmas
Disallow: /fun
Crawl-rate: 1
Allow: /broken-in-5-minutes
User-agent: Dr. Robotnik
Disallow: /sonic
Disallow: /tails
Allow: /ginger-mustache
```

Back in 1993, when I was teaching myself Perl in my spare time (while working for a -- cough -- UNIX company called The Santa Cruz Operation -- no relation to the current Utah asshats of that name), I was practicing by working on a spider. Now, back then SCO's Watford engineering centre was connected to the internet by a humongous 64kbps leased line. And I was working with a variety of sources on robots, and it just so happened that because I was doing a deterministic depth-first traversal of the web (hey, back then you could subscribe to the NCSA "what's new on the web" bulletin and visit all the interesting new websites every day before your coffee cooled), I kept hitting on Martin Kjoster's website. And Martin's then employers (who were doing something esoteric and X.509 oriented, IIRC) only had a 14.4kbps leased line. (Yes, you read that right: a couple of years later we all had faster modems, but this was the stone age.) Eventually Martin figured out that I was the bozo who kept leeching all his bandwidth, and contacted me. Throttling and QoS stuff was all in the future back then, so he went for a simpler solution: "Look for a text file called /robots.txt. It has a list of stuff you are not to pull in. Obey it, or I yell at your sysadmins." And so, I guess, my first attempt at a spider was also the first spider to obey the embryonic robot exclusion protocol. Which Martin subsequently generalized and which got turned into a standard. So if you're wondering why robots.txt is rather simplistic and brain-dead, it's because it was written to keep this rather simplistic and brain-dead perl n00b from pillaging Martin's bandwidth. Ah, the good old days when you could accidentally make someone invent a new protocol before breakfast ...

© 2012 The SANS Institute Author retains full rights.

Php robot validation

```
<?
header('Content-type: text/plain');
?>
User-agent: *
Disallow: /click.php
Disallow: /more_results.php
<?
function crapback()
{

}
@ob_start('crapback');
include("main_include.php");

$ip_address = $_SERVER["REMOTE_ADDR"];
$user_agent = addslashes(stripslashes($HTTP_USER_AGENT));
$reverse_dns = @gethostbyaddr($ip_address);

$sql = "INSERT INTO RobotUserAgent SET
user_agent = '$user_agent',
bot_counter = '1'
ON DUPLICATE KEY UPDATE bot_counter = bot_counter + 1";
echo $sql;
db_query($sql, 'RobotStats');
$sql = "SELECT bot_id, bot FROM RobotUserAgent WHERE user_agent =
'$user_agent'";
echo $sql;

$res = db_query($sql,'RobotStats');
$bot = db_fetch_array($res);
$sql = "INSERT INTO RobotIP SET bot_id = '$bot[bot_id]', ip =
'$ip_address', reverse_dns = '$reverse_dns', ip_counter = '1'
ON DUPLICATE KEY UPDATE ip_counter = ip_counter + 1";
echo $sql;
db_query($sql, 'RobotStats');

$sql = "SELECT robot_ip FROM RobotIP WHERE ip = '$ip_address' and
bot_id = '$bot[bot_id]'";
echo $sql;

$res = db_query($sql, 'RobotStats');
$robot = db_fetch_array($res);
$robot_day = date("Y-m-d", time());
$sql = "INSERT IGNORE INTO RobotDomain SET domain = '$host[domain]',
bot_id = '$bot[bot_id]', robot_ip = '$robot[robot_ip]', domain_day =
'$robot_day'";
echo $sql;

db_query($sql, 'RobotStats');
$sql = "INSERT INTO RobotHits SET bot_id = '$bot[bot_id]', bot_day =
'$robot_day', bot_hit = '1'
```


© 2012 The SANS Institute Author retains full rights.

```
ON DUPLICATE KEY UPDATE bot_hit = bot_hit + 1";
echo $sql;
db_query($sql, 'RobotStats');
$sql = "SELECT * FROM IP.UserAgentQuery WHERE user_agent_query =
'$user_agent'";
$res = db_query($sql, 'IP');
$ipag = db_fetch_array($res);

if (is_array($ipag))
{
if ($ipag[bot_type] == 'G')
{
$spider = 'Y';
}
elseif ($ipag[bot_type] == 'Y')
{
$spider = 'N';
}
$sql = "INSERT INTO IPBan SET remote_addr = '$ip_address', spider =
'$spider', which_se = '$bot[bot_id]' ON DUPLICATE KEY UPDATE which_se =
'$bot[bot_id]'";
echo $sql;
db_query($sql, 'IP');
$sql = "SELECT * FROM RobotDisallow WHERE bot_id = '$bot[bot_id]'";
$res = db_query($sql, 'RobotStats');
$rd = db_fetch_array($res);
if (is_array($rd))
{
$rt = "User-agent: $HTTP_USER_AGENT\n";
$rt .= "Crawl-Delay: $rd[crawl]\n";
$rt .= "Disallow: $rt[disallow]\n";
}
}

@ob_end_clean();
if ($rt)
{
echo $rt;
}

?>
```

Check Robots.txs for valid directories and files

```
#--------------------------------------------------------------------------
# Name: audit robots.txt
# Purpose:
#
# Author: JLehman
#
# Created: 09/08/2011
# Copyright: (c) JLehman 2011
# Licence: <your licence>
#--------------------------------------------------------------------------
#!/usr/bin/env python
import re
import urllib
```

Jim Lehman, jim.lehman@sbcglobal.net

© 2012 The SANS Institute Author retains full rights.

```
import httplib

#open ip file and iterate through lines- these are the Ips or FQDN's of web sites

f = open('c:\port_80.txt', 'r')
while True:
ip = f.readline()
print "\n"
url="http://" + ip + "/robots.txt"
f = urllib.urlopen(url)
s = f.read()
f.close()
list=s.split("\n")
for obj in list:
match = re.search(r'Disallow:\s/(.*)/',obj)
if match is not None:
uri = "/" + match.group(1)
# get return code
url = "http://" + ip + "/" +match.group(1)
f = urllib.urlopen("http://" + ip + uri)
url.strip()
print url
print f.getcode()
print f.readline()
f.close()
--------------------
```

#robots harvester v 1.0

```
# use source of dictionary list or random ip

#!/usr/bin/perl

use LWP::Simple;

my $browser = LWP::UserAgent->new;

open WLIST, "C:\\Documents and Settings\\Administrator\\Desktop\\robot
harvester\\password.lst" or die " can't open word list\n\n";

  while ($line=<WLIST>)
 {
  chomp $line;
  $url = "http://www\.$line\.com/robots.txt";
 my $response = $browser->get( $url );
 print $response->status_line;
 $resp=$response->status_line;
```

Jim Lehman, jim.lehman@sbcglobal.net

© 2012 The SANS Institute Author retains full rights.

```
 $content = $response->content;

 open LOG,">C:\\Documents and Settings\\Administrator\\Desktop\\robot
harvester\\log\\com\\$resp.$line.txt" || die "cant make log\n\n";
 print LOG $content;
 print "$line\n" ;
 close LOG
    }
```

References

PURACKAL, SEBASTIAN X. (3 January, 2008). “Standardization of REP tags as robots.txt directives”, Sebastian’s Pamphlets. http://sebastians-pamphlets.com/standardization-of-rep-tags-as-robots-txt-directives/

Crocker, D. (August 1982). "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822

Undisclosed Authors, Robotstxt.org (23 Aug 2010). “About robots.txt”. http://www.robotstxt.org/robotstxt.html

Spencer, Stephan. (April 16 2009) “ A Deeper Look At Robots.txt”. http://searchengineland.com/a-deeper-look-at-robotstxt-17573

Churchill, Christine. (Apr 16, 2007). “Up Close & Personal With Robots.txt”.

Search engine land. http://searchengineland.com/up-close-personal-with- robotstxt-10978

Wooster, Andrew. (Dec 3, 2006). “robots.txt Adventure”. Nextthing.org http://www.nextthing.org/archives/2007/03/12/robotstxt-adventure

Undisclosed Author. blog.semetrical. (November 15, 2010). “Google’s Hidden Interpretation of Robots.txt”. http://blog.semetrical.com/googles-secret- approach-to-robots-txt/

Jim Lehman, jim.lehman@sbcglobal.net

© 2012 The SANS Institute Author retains full rights.

Wikipedia. “Robots exclusion standard”.
http://en.wikipedia.org/wiki/Robots_exclusion_standard

Sitemaps.org (undisclosed authors). 27 February 2008. "What are Sitemaps?"

http://www.sitemaps.org/

Mithun. (10/02/2011). The Security Value of the Robots.txt file

http://www.dionach.com/blog/The-Security-Value-of-the-Robots.txt-file.asp

Carlos, Sean. (November 2010). "6 methods to control what and how your content appears in search engines" http://antezeta.com/news/avoid-search-engine-indexing

Google Code. (2012). Robots meta tag and X-Robots-Tag HTTP header specifications.

http://code.google.com/web/controlcrawlindex/docs/robots_meta_tag.html

Kloth, Ralf D.. (2007-10-25). List of Bad Bots.

http://www.kloth.net/internet/badbots.php

Crenshaw, Adrian. (2012). "Irongeek's Robots.txt Troll and Honeypot".

Http://www.irongeek.com

Wikipedia. (28 November 2011) "Email address harvesting".

http://en.wikipedia.org/wiki/E-mail_address_harvesting

Undisclosed Author. (2012). “An Enterprise View Of robots.txt”
http://www.re1y.com/robots-txt.html

Keane, Justin Klein. (11/25/2008). "Creating a Robots.txt Honeypot".
http://www.madirish.net/node/224

Jim Lehman, jim.lehman@sbcglobal.net

© 2012 The SANS Institute Author retains full rights.

Moore, Andrew. (2002). "Bottrap mod_perl"
http://www.linuxjournal.com/files/linuxjournal.com/linuxjournal/articles/058/5861/58611.html

Jerkovic, John I. (November 20, 2009). "SEO Warrior", O'Reilly & Associates INC.

https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf

/Syngress - Google Hacking for Penetration Tester - Vol.1.pdf

© 2012 SANS Institute, Author retains full rights.

© 2012 The SANS Institute Author retains full rights.

Last Updated: July 24th, 2017

Upcoming Training

SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
Mentor Session - SEC542	Des Moines, IA	Aug 14, 2017 - Sep 13, 2017	Mentor
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Community SANS Toronto SEC542	Toronto, ON	Sep 11, 2017 - Sep 16, 2017	Community SANS
SANS Dublin 2017	Dublin, Ireland	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, Norway	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS vLive - SEC542: Web App Penetration Testing and Ethical Hacking	SEC542 - 201710,	Oct 03, 2017 - Nov 09, 2017	vLive
SANS Tysons Corner Fall 2017	McLean, VA	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, Japan	Oct 16, 2017 - Oct 28, 2017	Live Event
Community SANS Minneapolis SEC542	Minneapolis, MN	Oct 16, 2017 - Oct 21, 2017	Community SANS
Community SANS New York SEC542	New York, NY	Oct 16, 2017 - Oct 21, 2017	Community SANS
SANS Berlin 2017	Berlin, Germany	Oct 23, 2017 - Oct 28, 2017	Live Event
Community SANS Tampa SEC542	Tampa, FL	Nov 13, 2017 - Nov 18, 2017	Community SANS
Community SANS Austin SEC542	Austin, TX	Nov 13, 2017 - Nov 18, 2017	Community SANS
SANS London November 2017	London, United Kingdom	Nov 27, 2017 - Dec 02, 2017	Live Event
Community SANS Ottawa SEC542	Ottawa, ON	Nov 27, 2017 - Dec 02, 2017	Community SANS
SANS San Francisco Winter 2017	San Francisco, CA	Nov 27, 2017 - Dec 02, 2017	Live Event
Community SANS Marina Del Rey SEC542	Marina Del Rey, CA	Nov 27, 2017 - Dec 02, 2017	Community SANS
SANS Frankfurt 2017	Frankfurt, Germany	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Cyber Defense Initiative 2017 - SEC542: Web App Penetration Testing and Ethical Hacking	Washington, DC	Dec 14, 2017 - Dec 19, 2017	vLive
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Miami 2018	Miami, FL	Jan 29, 2018 - Feb 03, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced