Global Information Assurance Certification Paper

Search Engines: The Ignored Threat

Paul Heely

February 5, 2001

Abstract

Search engines simplify the process of locating information in cyberspace. They allow simple queries against massive databases of information, allowing users to find information on any topic. The same mechanism that provides this great searchability also presents a threat to many organizations. This paper will explore how the innocuous search engine can, in fact, be quite dangerous. We begin with a look at the basic operation of a search engine. We then look at how the contents of a search engine's database can be used by a malicious user to locate web sites vulnerable to attack. In addition to sites vulnerable to direct attack, we also look at how sites may be leaking sensitive information to the world through an entry in a search engine's database. We conclude with a discussion of controls that can be deployed to help limit what parts of a site are searched and cataloged by a search engine robot.

Introduction:

If you knew that a user was systematically visiting, indexing and cataloging every page and resource you have available on the Internet, would you be concerned? Many organizations would be if the user were coming from dialup.someisp.com. Many of these same organizations do not give it a second thought when a search engine robot comes along and does the same thing. The user and robot exhibit the same behavior, but the perception of risk is different between the unknown user cataloging an entire web site and the well-known search engine doing the same thing.

Search engines are an ingrained part of the Internet. Organizations rely on search engines to make their products, services, and resources available to anybody looking for them. Organizations will submit their URLs to search engines asking them to come and catalog their organization's web site. The danger is that search engine robots will access every page they can, unless directed otherwise. Unfortunately, many organizations never consider putting these controls into place.

Search Engines:

The term "search engine" is often used to refer to many different searching technologies. In the context of this paper, search engines are differentiated from directories in the following way: search engines are autonomous entities that crawl the web following the links that they find and creating an index of searchable terms; www.google.com is an example of a search engine. Directories are lists of sites maintained by a human being. The entries in directories, and in some cases the searchable descriptions, are not maintained automatically and require human action to update; www.yahoo.com is an example of a directory [1].

The distinction between search engines and directories is an important one. People maintain the data in a directory, someone submitted a particular URL for inclusion in the

directory. Additionally, someone most likely wrote the description, i.e., the searchable keywords, for the submitted URL. With a search engine, each page visited by the search engine robot, also called a spider or crawler, reads the page, then follows every link contained within that page. The read data is handed off to the indexer that builds a searchable list of terms that describe the page.

The robot's systematic following of every link it encounters is where the problem lies. A human writing a description of a site for a directory probably wouldn't consider log files and CGI programs as being important to the description. A robot, on the other hand, will happily traverse into the log file directory, if it is accessible from another page. A robot will happily try to follow a link that points to some CGI program or database interface program. These URLs and the "contents" of the URL, even if they are just error messages generated by a CGI program that was called with bad inputs, will be sent back to the indexer. The indexer builds the database of keywords that a user searches against. Many search engines provide a powerful language for constructing search queries. Google, www.google.com, allows a user to search not only by the common "X and Y but not Z", but also to specify if a search term must be contained in the URL. It also allows searches to be narrowed by domain names; anything from abc.xyz.com to just .com is a valid parameter.

Unfortunately, many organizations do not realize that search engine robots and indexers behave in the above manner. More importantly, organizations do not realize the kinds of information they are making available and just how easy it is to find that information. This information can be broken into two categories: CGI vulnerabilities and information leakage.

CGI Vulnerabilities:
The term CGI is used here to include CGI programs, server side includes, applets, servlets, and data base front ends.

CGI vulnerabilities are a huge problem. In February 2001, CGI problems are listed as number 2 in on the SANS top ten list [2]. There are over 750 messages in the Bugtraq archives that contain the word "cgi" [3]. The Common Vulnerabilities and Exposures dictionary contains 150 entries or candidates that contain the term "cgi" [4].

The wide range of CGI vulnerabilities, combined with the searching capabilities of a search engine, make a lethal combination. To illustrate the actual danger, two well-known CGI vulnerabilities were chosen from the above sources. Google was used in an attempt to locate vulnerable sites that contained these vulnerable programs [5].

The first program chosen was htgrep. From the Htgrep website [6]:

> "Htgrep is a CGI script written in Perl that allows you to query any document accessible to your HTTP server on a paragraph-by-paragraph basis."

Htgrep has been reported on Bugtraq and has CVE entry CAN-2000-0832. The problem with htgrep is that versions <= 2.4 allow an attacker to read any file on the remote system that is readable by the user that the web server runs as, typically nobody, or on poorly

configured sites, root. A little investigation revealed that pages generated using htgrep contain the text "generated by htgrep". A Google search for "generated by htgrep" returned 106 hits. Htgrep also puts the version number in the "generated by htgrep" message. The search was then refined as:

+"generated by htgrep" –"v3.2"

This resulted in 98 hits. With about 5 minutes of work almost 100 sites running known vulnerable software were identified.

Using htgrep, an attacker could read /etc/inetd.conf, assuming Unix or Linux of course. Having the inetd.conf file, the attacker would now know some of the network services being offered by the host and could possibly tell if access mechanisms like TCP wrappers are being used. On a Solaris system, an attacker could not only find out if vulnerable RPC services like sadmind and ttdbservd are running, but by knowing what other files to read from the file system, e.g., /var/sadm/pkg/<pkg name>/pkginfo, it is possible to determine if those services are the latest patched ones or older vulnerable ones. Using htgrep to retrieve files will be logged on the remote system if they are logging http requests. An attacker has an advantage since an organization is unlikely to monitor their logs closely enough to catch this attack and at the same time have a vulnerable CGI program installed.

Htgrep is certainly not the only vulnerable CGI program that can be located with a search engine. Two more common CGI vulnerabilities, phf and PHP/FI, returned the following search results [2]:

Search String	Number of Hits
allinurl: cgi-bin/phf	229
allinurl: cgi-bin/php	400

No claim is being made that all hits resulting from these searches will result in exploitable programs being found. Instead, the claim is being made that an attacker can quickly, and with a low probability of detection, identify a group of hosts that have a much higher probability of being vulnerable than if the attacker performed a random brute force attack.

The preceding examples illustrate how CGI programs that are known to be vulnerable can be used to attack a site. A site could stop a search engine robot from looking for known vulnerable CGI programs. This measure would not stop the threat however. Search engine robots have already found and indexed vulnerabilities that have not yet been discovered. On December 19, 2000, a message was posted to Bugtraq with a subject of "Oracle WebDb engine brain-damagse" [7]. The author of the message goes on to explain, in detail, a vulnerability found in Oracle databases that is remotely exploitable through the web. Also included in the message was how to identify sites that were vulnerable with a simple Google search. A search performed the same day the message was posted returned 67 hits. It is frightening to think that an attacker seeing the same message was able to identify 67 sites with a known vulnerability by simply making a search query. Even more frightening is that at the time of this vulnerabilities release

there was no fix available for the problem. On February 9, 2000 a Bugtraq messages titled "ASP Security Hole" discussed a similar problem with Active Server Pages [8].

The preceding examples show the danger search engines pose in making vulnerable CGI programs, known and as yet unknown, easy to locate. No one is going to notice that an attacker is searching Google for all dot com sites running a particular CGI program.

Information Leakage:
Information leakage is the other major problem that search engines can cause for an organization. Organizations are continually making resources web accessible. In many cases, these organizations only intend for internal users to access these resources. Many organizations do not password protect these resources. Instead, they rely on security through obscurity to protect them. Organizations will create web pages that they think have no links pointing to them or that are in some obscure directory that they think no one would look in. What they forget is that search engines will catalog and index log files and software just as readily as they catalog and index widget pages. To help emphasis the risks that leakage poses we will examine two scenarios. First we will look at the risks of relying on an obscure URL to protect a resource. Second we look at the dangers and risks of making log files, and associated reports, publicly accessible and searchable.

Imagine a large organization that purchases a commercial copy of SSH. Not everybody needs to use SSH and there is no centralized way to distribute it to those who do need it. It is decided that the SSH package and license file will be given a URL of abc.com/ssh, and that this URL will be given to those who need to use the package. At the same time, it is decided that since this is a licensed commercial package that it should not be publicly accessible. Because of this it is decided that no link to the SSH package will be placed on any of the organizations web pages. This scheme in and of itself will not cause the SSH page to be cataloged by a search engine. As long as the organization does not create a link to the SSH page, the search engines have no way to get there. The problem is that the organization cannot control who does create a link to the SSH page. Sally in Accounting could be worried that she will not be able to remember the URL so she adds a link to the SSH page on her personal home page. Along comes the search engine robot, finds Sally's page and follows all the links, including the one to her organization's licensed copy of SSH. Now that the page has been indexed and cataloged, the whole world can find it by searching for "abc.com SSH." Chances are that making a licensed commercial software package publicly available will irritate your software vendor at the very least. In the above scenario, SSH could just as easily have been replaced with any other software: an organization's beta code for a new product or internal documents. The point is that security through obscurity fails very quickly when applied to the web. A much better solution is to at a minimum use domain restriction mechanisms or password protected web pages.

Log files are another form of information leakage that presents a large problem. A simple search for "access_log" showed about 1000 hits from one search engine. Different servers use different names for their default log files and in many cases the user

can configure the filename to be whatever they want. It is just a matter of being creative enough to search for the correct phrase. Many organizations use software to process raw log files into more usable reports. A common package used for this is WebTrends, www.webtrends.com. Searching for:

"this report was generated by webtrends"

generated over 5000 hits. Information contained in both formal reports and raw log files can be detrimental to an organization.

A report like the one produced by WebTrends contains a lot of useful information. A typical report will contain [9]:

- Top 10 requested pages.
- Top 10 countries requesting pages.
- Activity levels by day of week and time of day.
- Top 20 referring sites.
- Top 10 search engines that users used to find an organization's web site.
- Top 10 search keywords.

The web server administrator probably does not care that this information is available. In fact, they may knowingly make it widely available. The availability of this information however can represent a violation of confidentiality. Recall that confidentiality is to keep sensitive data from being disclosed. In the case of a business, sensitive data will include any data that may give a competitive advantage to a competitor [10]. Business and marketing people will agree that dissemination of this kind of information to a competitor could help them to have a competitive advantage.

Access to an organization's raw log files also poses a threat if that organization allows submission of web forms using the GET method. With the GET method, data in the form is submitted by appending it to the URL. Most people have seen URLs in their browsers that look like this:

http://www.abc.com/cgi-bin/program.cgi?var1=val1&var2=val2

This URL is executing the program called program.cgi on the abc.com website and is passing in two variables, var1 and var2, with values val1 and val2, respectively. This entire URL, including the parameters to program.cgi, will be written to the access log file. In many instances, the parameters and their values are of no consequence to anyone. In some poorly designed web pages though, user authentication information will be transmitted in this way. Consider the following URL:

http://www.stocktradingcompany.com/cgi-bin/login?user=smith&pass=1234

If somebody were to find the access log with this entry they could log in as smith and do all kinds of fun things. In this case, SSL will not help to protect the data submitted with the GET mechanism. The request is first decrypted then written to the log file. A better solution in this case would be to use the POST method for form submission [11].

Controlling Search Engine Robots:

No organization with a web presence can afford to block search engine robot access to its web resources. Instead, organizations should evaluate what should and should not be made available through the Internet. A mechanism exists to restrict what a robot accesses and what a robot will attempt to send back to the indexer. This mechanism is the

robots.txt file. The robots.txt file is literally a file with a local URL of /robots.txt. This file allows a site to specify sets of rules that define what resources a robot is forbidden to access. The format of the robots.txt file is defined in [12]. *NOTE: there is no guarantee that a robot will obey the contents of the robots.txt file, though most search engines claim that their robots will.*

The following examples illustrate how the robots.txt file can be used to specify what a robot should not access. The following robots.txt states that no robot should visit any URL starting with "/private/data", "/logs", or "/software"

```
# this is a comment
User-agent: *
Disallow: /private/data
Disallow: /logs
Disallow: /software
```

This robots.txt states that no robot should visit any URL starting with "/data" except the robot called "nicerobot".

```
# robots.txt
User-agent: *
Disallow: /data

# nicerobot can look in /data
User-agent: nicerobot
Disallow:
```

Finally this robots.txt states that no robot should visit any URL at this site.

```
# disallow all access by all robots
User-agent: *
Disallow: /
```

Conclusion:
The Internet has become a vast expanse of usable resources. Attempting to find the resources that one needs requires some searching mechanism to be in place. Any company trying to have a web presence wants to be on the top of the search list when someone searches for their type of products and goods. Many companies do not realize what other data they are making available, or how that data may be used against them. Hopefully, this paper will serve to increase awareness to the risks posed by allowing search engine robots free reign of our web sites.

References:

[1] Sullivan, Danny. "How Search Engines Work." URL: http://www.searchenginewatch.com/webmasters/work.html (5 February 2001).

[2] "How To Eliminate The Ten Most Critical Internet Security Threats." V1.32. 18 January 2001. URL: http://www.sans.org/topten.html (4 February 2001).

[3] Bugtraq Archives. URL: http://www.securityfocus.com (22 January 2001).

[4] "Common Vulnerabilities and Exposures." 23 January 2001. URL: http://cve.mitre.org (2 February 2001)

[5] "Google." URL: www.google.com.

[6] "HTGREP FAQ List." 22 November 2000. URL: http://www.iam.unibe.ch/~scg/Src/Doc (3 February 2001).

[7] "SecurityFocus." URL: http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D46584

[8] "SecurityFocus." URL: http://www.securityfocus.com/frames/?content=/templates/archive.pike%3Flist%3D1%26mid%3D46584 (5 February 2001).

[9] "WebTrends Sample Reports WebTrends Corporation." URL: http://www.webtrends.com/reports/reports.asp (5 February 2001).

[10] Computers At Risk (Washington, D.C.: National Academy Press 1991).

[11] "Forms in HTML documents." URL: http://www.w3.org/TR/1998/REC-html40-19980424/interact/forms.html#indx-form-8 (20 December 2000).

[12] Koster, Martijn "A standard for Robot Exclusion." URL: http://info.webcrawler.com/mak/projects/robots/norobots.html (5 February 2001).