Dangerous Google – Searching for Secrets

Michał Piotrowski

This article has been published in issue 4/2005 of the *hakin9* magazine.

hakin9 magazine, Wydawnictwo Software, ul. Lewartowskiego 6, 00-190 Warszawa, e*n@hakin9.org*

Dangerous Google – Searching for Secrets

Michał Piotrowski

Information which should be protected is very often publicly available, revealed by careless or ignorant users. The result is that lots of confidential data is freely available on the Internet – just Google for it.

Google serves some 80 percent of all search queries on the Internet, making it by far the most popular search engine. Its popularity is due not only to excellent search effectiveness, but also extensive querying capabilities. However, we should also remember that the Internet is a highly dynamic medium, so the results presented by Google are not always up-to-date – some search results might be stale, while other relevant resources might not yet have been visited by Googlebot (the automatic script that browses and indexes Web resources for Google).

Table 1 presents a summary of the most important and most useful query operators along with their descriptions, while Figure 1 shows document locations referred to by the operators when applied to Web searches. Of course, this is just a handful of examples – skilful Google querying can lead to much more interesting results.

Hunting for Prey

Google makes it possible to reach not just publicly available Internet resources, but also some that should never have been revealed.

What You Will Learn...

- how to use Google to find sources of personal information and other confidential data,
- how to find information about vulnerable systems and Web services,
- how to locate publicly available network devices using Google.

What You Should Know...

- how to use a Web browser,
- basic rules of operation of the HTTP protocol.

About the Author

Michał Piotrowski holds an MA in IT and has many years' experience in network and system administration. For over three years he has been a security inspector and is currently working as computer network security expert at one of the largest Polish financial institutions. His free time is occupied by programming, cryptography and contributing to the open source community.

Table 1. *Google query operators*

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 containing the specified phrase in the text, but not in the title, link descriptions or URLs	allintext:"fire fox" will return documents which contain 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 occur 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*

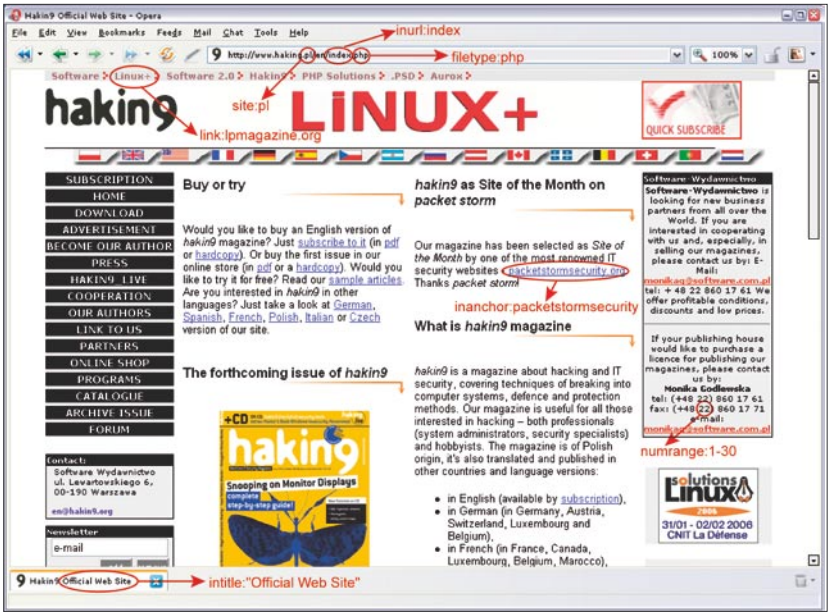


Figure 1. *The use of search query operators illustrated using the hakin9 website*

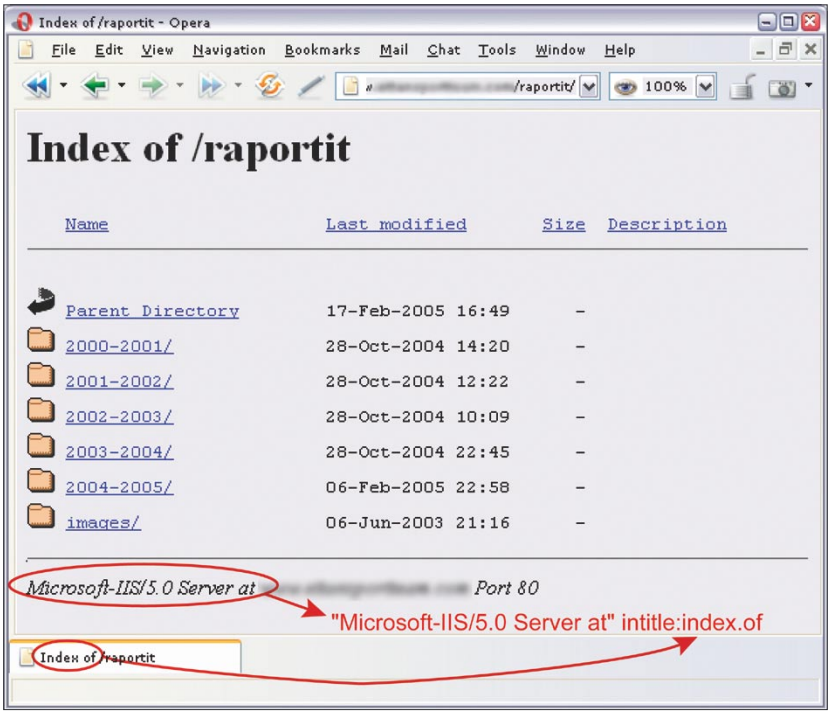


Figure 2. *Locating IIS 5.0 servers using the intitle operator*

The right query can yield some quite remarkable results. Let's start with something simple.

Suppose that a vulnerability is discovered in a popular application – let's say it's the Microsoft IIS server version 5.0 – and a hypothetical attacker decides to find a few computers running this software in order to attack them. He could of course use a scanner of some description, but he prefers Google, so he just enters the query `"Microsoft-IIS/5.0 Server at" intitle:index.of` and obtains links to the servers he needs (or, more specifically, links to autogenerated directory listings for those servers). This works because in its standard configuration, IIS (just like many other server applications) adds banners containing its name and version to some dynamically generated pages (Figure 2 shows this query in action).

It's a typical example of information which seems quite harmless, so is frequently ignored and remains in the standard configuration. Unfortunately, it is also information which in certain circumstances can be most valuable to a potential attacker. Table 2 shows more sample Google queries for typical Web servers.

Another way of locating specific versions of Web servers is to search for the standard pages displayed after successful server installation. Strange though it may seem, there are plenty of Web servers out there, the default configuration of which hasn't been touched since installation. They are frequently forgotten, ill-secured machines which are easy prey for attackers. They can be located using the queries shown in Table 3.

This method is both very simple and extremely useful, as it provides access to a huge number of various websites and operating systems which run applications with known vulnerabilities that lazy or ignorant administrators have not patched. We will see how this works for two fairly popular programs: *WebJeff Filemanager* and *Advanced Guestbook*.

The first is a web-based file manager for uploading, browsing, managing and modifying files on a server. Unfortunately, *WebJeff Filemanager* version 1.6 contains a bug which makes it possible to download any file on the server, as long as it's accessible to the user running the HTTP daemon. In other words, specifying a page such as */index.php3?action=telecharger&fichier=/etc/passwd* in a vulnerable system will let any intruder download the */etc/passwd* file (see Figure 3). The aggressor will of course locate vulnerable installations by querying Google for `"WebJeff-Filemanager 1.6" Login`.

Our other target – *Advanced Guestbook* – is a PHP application

Table 2. *Google queries for locating various Web servers*

Query	Server
`"Apache/1.3.28 Server at" intitle:index.of`	Apache 1.3.28
`"Apache/2.0 Server at" intitle:index.of`	Apache 2.0
`"Apache/* Server at" intitle:index.of`	any version of Apache
`"Microsoft-IIS/4.0 Server at" intitle:index.of`	Microsoft Internet Information Services 4.0
`"Microsoft-IIS/5.0 Server at" intitle:index.of`	Microsoft Internet Information Services 5.0
`"Microsoft-IIS/6.0 Server at" intitle:index.of`	Microsoft Internet Information Services 6.0
`"Microsoft-IIS/* Server at" intitle:index.of`	any version of Microsoft Internet Information Services
`"Oracle HTTP Server/* Server at" intitle:index.of`	any version of Oracle HTTP Server
`"IBM_HTTP_Server/* * Server at" intitle:index.of`	any version of IBM HTTP Server
`"Netscape/* Server at" intitle:index.of`	any version of Netscape Server
`"Red Hat Secure/*" intitle:index.of`	any version of the Red Hat Secure server
`"HP Apache-based Web Server/*" intitle:index.of`	any version of the HP server

Table 3. *Queries for discovering standard post-installation Web server pages*

Query	Server
`intitle:"Test Page for Apache Installation" "You are free"`	Apache 1.2.6
`intitle:"Test Page for Apache Installation" "It worked!" "this Web site!"`	Apache 1.3.0 – 1.3.9
`intitle:"Test Page for Apache Installation" "Seeing this instead"`	Apache 1.3.11 – 1.3.33, 2.0
`intitle:"Test Page for the SSL/TLS-aware Apache Installation" "Hey, it worked!"`	Apache SSL/TLS
`intitle:"Test Page for the Apache Web Server on Red Hat Linux"`	Apache on Red Hat
`intitle:"Test Page for the Apache Http Server on Fedora Core"`	Apache on Fedora
`intitle:"Welcome to Your New Home Page!" Debian`	Apache on Debian
`intitle:"Welcome to IIS 4.0!"`	IIS 4.0
`intitle:"Welcome to Windows 2000 Internet Services"`	IIS 5.0
`intitle:"Welcome to Windows XP Server Internet Services"`	IIS 6.0

with SQL database support, used for adding guestbooks to websites. In April 2004, information was published about a vulnerability in the application's 2.2 version, making it possible to access the administration panel using an SQL injection attack (see *SQL Injection Attacks with PHP/MySQL* in *hakin9* 3/2005). It's enough to navigate to the panel login screen (see Figure 4) and log in leaving the *username* blank and entering `') OR ('a' = 'a` as *password* or the other way around – leaving *password* blank and entering `? or 1=1 --` for *username*. The potential aggressor can locate vulnerable websites by querying Google for `intitle: Guestbook "Advanced Guestbook 2.2 Powered"` or `"Advanced Guestbook 2.2" Username inurl:admin`.

To prevent such security leaks, administrators should track current information on all the applications used by their systems and immediately patch any vulnerabilities. Another thing to bear in mind is that it's well worth removing application banners, names and versions from any pages or files that might contain them.

Information about Networks and Systems

Practically all attacks on IT systems require preparatory target reconnaissance, usually involving scanning computers in an attempt

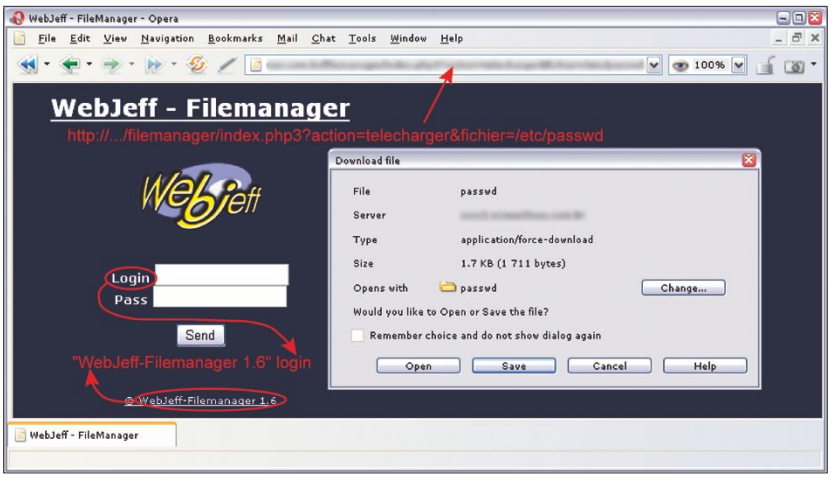


Figure 3. *A vulnerable version of WebJeff Filemanager*

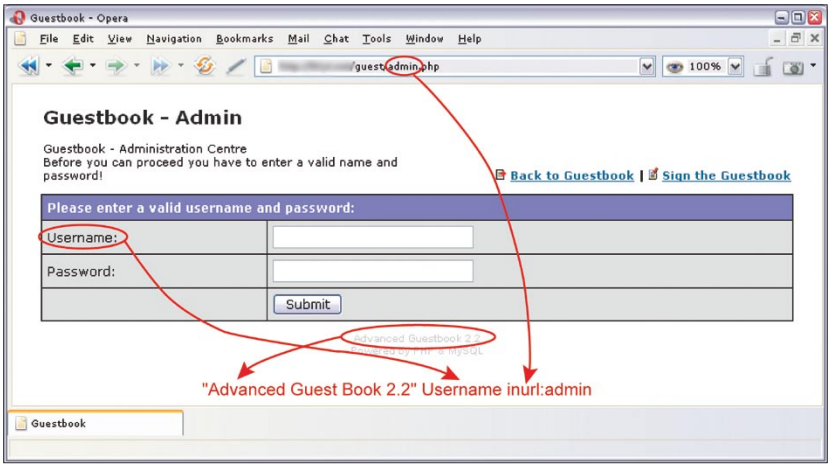


Figure 4. *Advanced Guestbook login page*

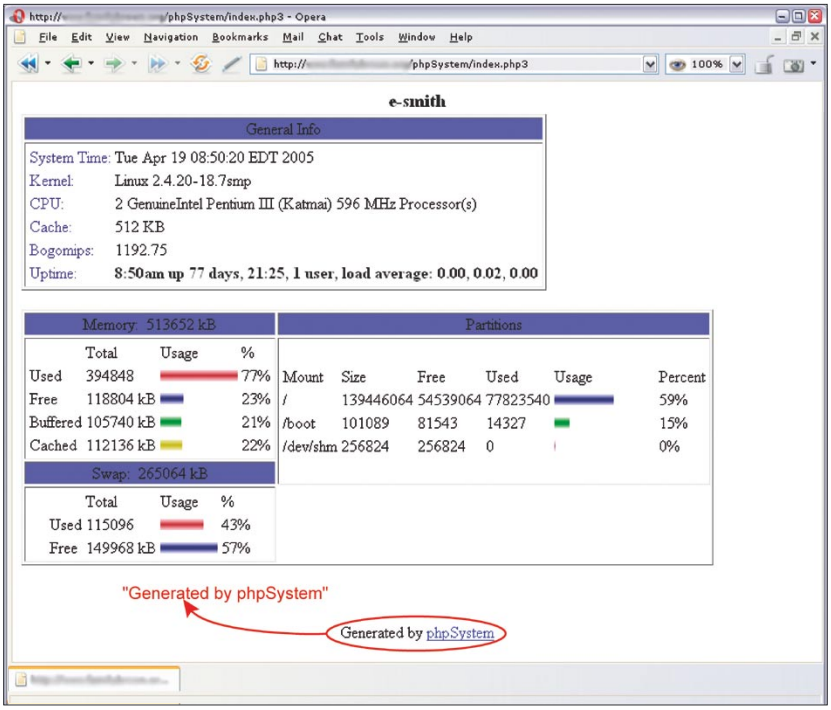


Figure 5. *Statistics generated by phpSystem*

to recognise running services, operating systems and specific service software. Network scanners such as *Nmap* or *amap* are typically used for this purpose, but another possibility also exists. Many system administrators install Web-based applications which generate system load statistics, show disk space usage or even display system logs.

All this can be valuable information to an intruder. Simply querying Google for statistics generated and signed by the *phpSystem* application using the query `"Generated by phpSystem"` will result in a whole list of pages similar to the one shown in Figure 5. The intruder can also query for pages generated by the *Sysinfo* script using `intitle:"Sysinfo * " intext:"Generated by Sysinfo * written by The Gamblers."` – these pages contain much more system information (Figure 6).

This method offers numerous possibilities – Table 4 shows sample queries for finding statistics and other information generated by several popular applications. Obtaining such information may encourage the intruder to attack a given system and will help him find the right tools and exploits for the job. So if you decide to use Web applications to monitor computer resources, make sure access to them is password-protected.

Looking for Errors

HTTP error messages can be extremely valuable to an attacker, as they can provide a wealth of information about the system, database structure and configuration. For example, finding errors generated by an *Informix* database merely requires querying for `"A syntax error has occurred" filetype:ihtml`. The result will provide the intruder with error messages containing information on database configuration, a system's file structure and sometimes even passwords (see Figure 7). The results can be narrowed down to only those containing passwords by altering the query slightly: `"A syntax error has occurred" filetype:ihtml intext:LOGIN`.

Equally useful information can be obtained from *MySQL* database errors simply by querying Google for `"Access denied for user" "Using password"` – Figure 8 shows a typical website located in this manner. Table 5 contains more sample queries using the same method.

The only way of preventing our systems from publicly revealing error information is removing all bugs as soon as we can and (if possible) configuring applications to log any errors to files instead of displaying them for the users to see.

Remember that even if you react quickly (and thus make the error pages indicated by Google out-of-date), a potential intruder will still be able to examine the version of the page cached by Google by simply clicking the link to the page copy. Fortunately, the sheer volume of Web resources means

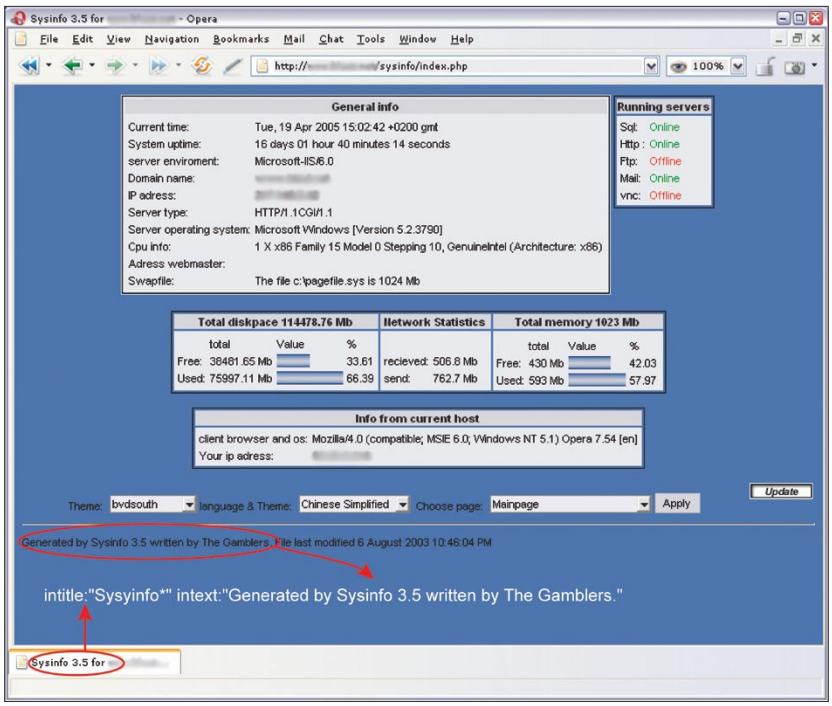


Figure 6. *Statistics generated by Sysinfo*

Table 4. *Querying for application-generated system reports*

Query	Type of information
`"Generated by phpSystem"`	operating system type and version, hardware configuration, logged users, open connections, free memory and disk space, mount points
`"This summary was generated by wwwstat"`	web server statistics, system file structure
`"These statistics were produced by getstats"`	web server statistics, system file structure
`"This report was generated by WebLog"`	web server statistics, system file structure
`intext:"Tobias Oetiker" "traffic analysis"`	system performance statistics as MRTG charts, network configuration
`intitle:"Apache::Status" (inurl:server-status | inurl:status.html | inurl:apache.html)`	server version, operating system type, child process list, current connections
`intitle:"ASP Stats Generator *.*" "ASP Stats Generator" "2003-2004 weppos"`	web server activity, lots of visitor information
`intitle:"Multimon UPS status page"`	UPS device performance statistics
`intitle:"statistics of" "advanced web statistics"`	web server statistics, visitor information
`intitle:"System Statistics" +"System and Network Information Center"`	system performance statistics as MRTG charts, hardware configuration, running services
`intitle:"Usage Statistics for" "Generated by Webalizer"`	web server statistics, visitor information, system file structure
`intitle:"Web Server Statistics for ****"`	web server statistics, visitor information
`inurl:"/axs/ax-admin.pl" -script`	web server statistics, visitor information
`inurl:"/cricket/grapher.cgi"`	MRTG charts of network interface performance
`inurl:server-info "Apache Server Information"`	web server version and configuration, operating system type, system file structure
`"Output produced by SysWatch *"`	operating system type and version, logged users, free memory and disk space, mount points, running processes, system logs

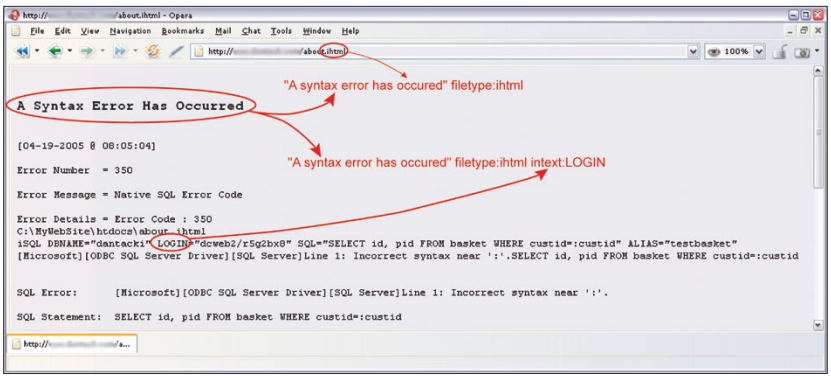


Figure 7. *Querying for Informix database errors*

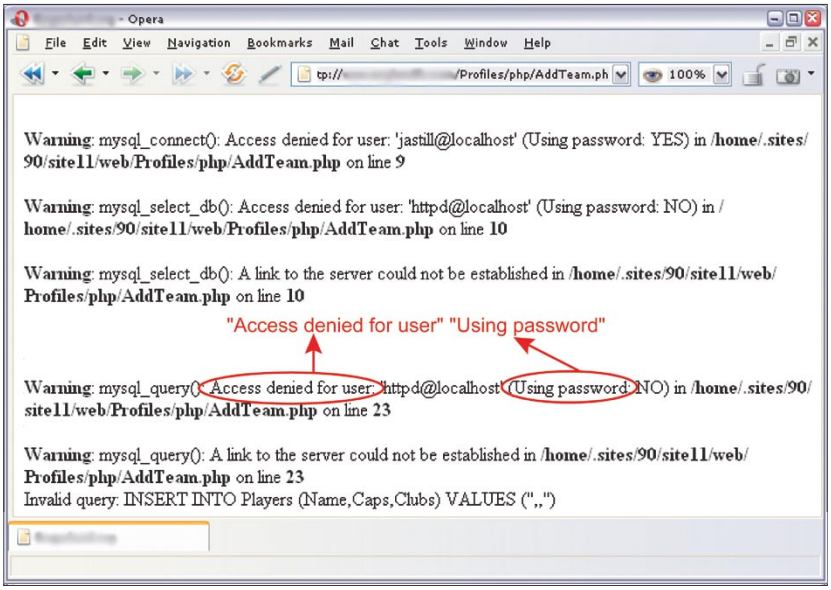


Figure 8. *MySQL database error*

that pages can only be cached for a relatively short time.

Prowling for Passwords

Web pages contain a great many passwords to all manner of resources – e-mail accounts, FTP servers or even shell accounts. This is mostly due to the ignorance of users who unwittingly store their passwords in publicly accessible locations, but also due to the carelessness of software manufacturers who either provide insufficient measures of protecting user data or supply no information about the necessity of modifying their products' standard configuration.

Take the example of *WS_FTP*, a well-known and widely-used FTP client which (like many utilities) offers the option of storing account passwords. *WS_FTP* stores its configuration and user account information in the *WS_FTP.ini* file. Unfortunately, not everyone realises that gaining access to an FTP client's configuration is synonymous with gaining access to a user's FTP resources. Passwords stored in the *WS_FTP.ini* file are encrypted, but this provides little protection – once an intruder obtains the configuration

Table 5. *Error message queries*

Query	Result
`"A syntax error has occurred" filetype:ihtml`	*Informix* database errors, potentially containing function names, filenames, file structure information, pieces of SQL code and passwords
`"Access denied for user" "Using password"`	authorisation errors, potentially containing user names, function names, file structure information and pieces of SQL code
`"The script whose uid is " "is not allowed to access"`	access-related PHP errors, potentially containing filenames, function names and file structure information
`"ORA-00921: unexpected end of SQL command"`	*Oracle* database errors, potentially containing filenames, function names and file structure information
`"error found handling the request" cocoon filetype:xml`	*Cocoon* errors, potentially containing *Cocoon* version information, filenames, function names and file structure information
`"Invision Power Board Database Error"`	*Invision Power Board* bulletin board errors, potentially containing function names, filenames, file structure information and piece of SQL code
`"Warning: mysql_query()" "invalid query"`	*MySQL* database errors, potentially containing user names, function names, filenames and file structure information
`"Error Message : Error loading required libraries."`	CGI script errors, potentially containing information about operating system and program versions, user names, filenames and file structure information
`"#mysql dump" filetype:sql`	*MySQL database errors, potentially containing information about database structure and contents*

file, he can either decipher the password using suitable tools or simply install *WS_FTP* and run it with the stolen configuration. And how can the intruder obtain thousands of *WS_FTP* configuration files? Using Google, of course. Simply querying for `"Index of/" "Parent Directory" "WS_FTP.ini"` or `filetype:ini WS_FTP PWD` will return lots of links to the data he requires, placed at his evil disposal by the users themselves in their blissful ignorance (see Figure 9).

Another example is a Web application called *DUclassified*, used for managing website advertising materials. In its standard configuration, the application stores all the user names, passwords and other data in the *duclassified.mdb* file, located in the read-accessible *_private* subdirectory. It is therefore enough to find a site that uses *DUclassified*, take the base URL *http://<host>/duClassified/* and change it to *http://<host>/duClassified/_private/duclassified.mdb* to obtain the password file and thus obtain unlimited access to the application (as seen in Figure 10). Websites which use the vulnerable application can be located by querying Google for `"Powered by DUclassified" -site:duware.com` (the additional operator will filter out results from the manufacturer's website). Interestingly enough, the makers of *DUclassified* – a company called DUware – have also created several other applications with similar vulnerabilities.

In theory, everyone knows that passwords should not reside on post-its stuck to the monitor or under the keyboard. In practice, however, surprisingly many people store passwords in text files and put them in their home directories, which (funnily enough) are accessible through the Internet. What's more, many such individuals work as network administrators or similar, so the files can get pretty big. It's hard to define a single method of locating such data, but googling for such keywords as *account*, *users*, *admin*, *administrators*, *passwd*, *password* and so on can be pretty effective, especially coupled with such filetypes as *.xls*, *.txt*, *.doc*, *.mdb* and *.pdf*. It's also worth noting directories whose names contain the words *admin*, *backup* and so forth – a query like `inurl:admin intitle:index.of` will do the trick.

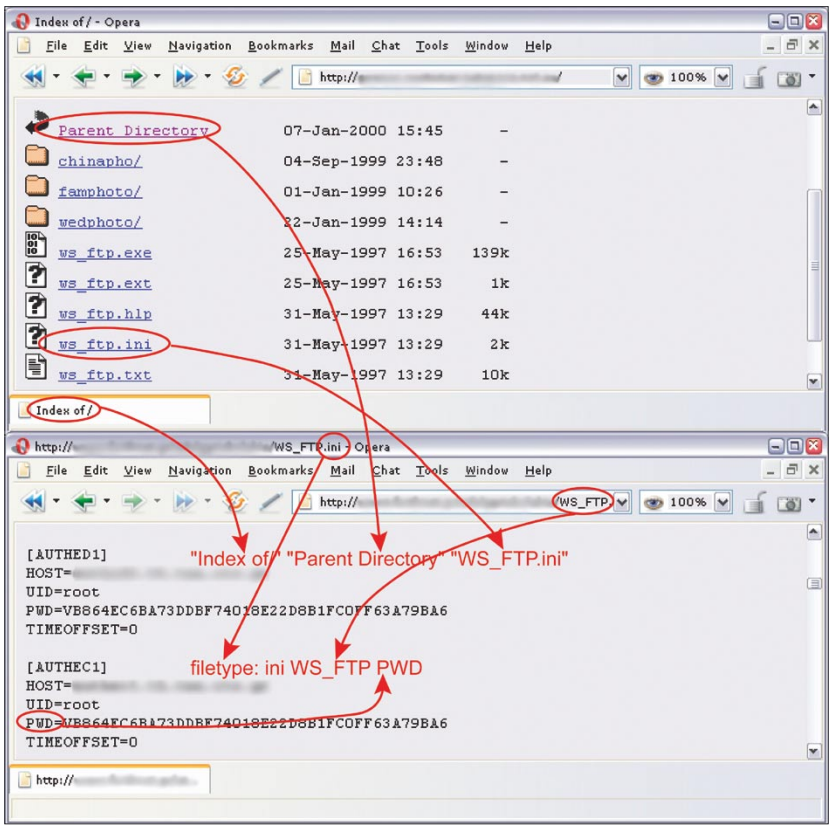


Figure 9. *WS_FTP configuration file*

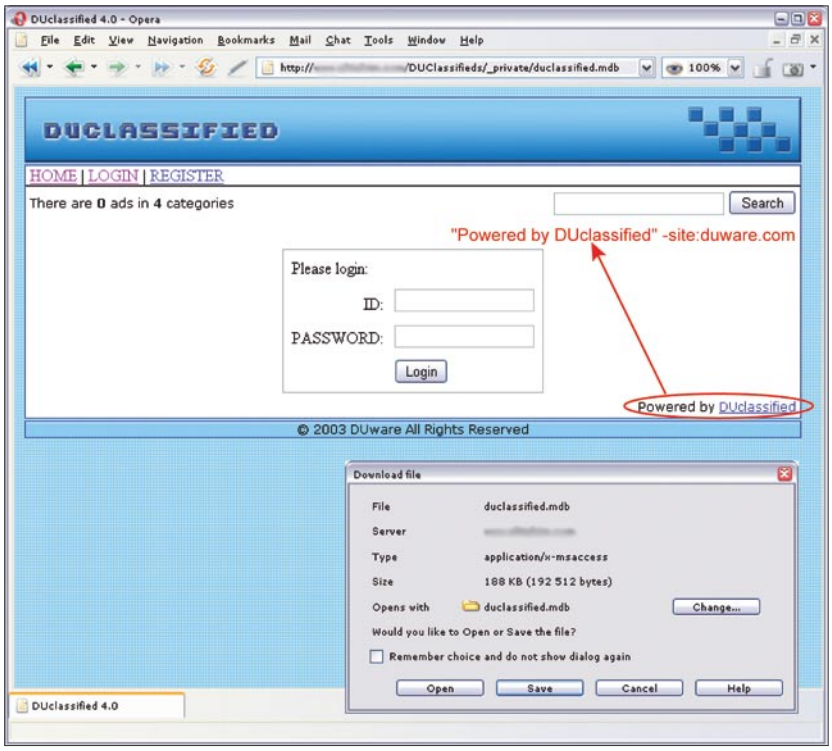


Figure 10. *DUclassified in its standard configuration*

Table 6. *Google queries for locating passwords*

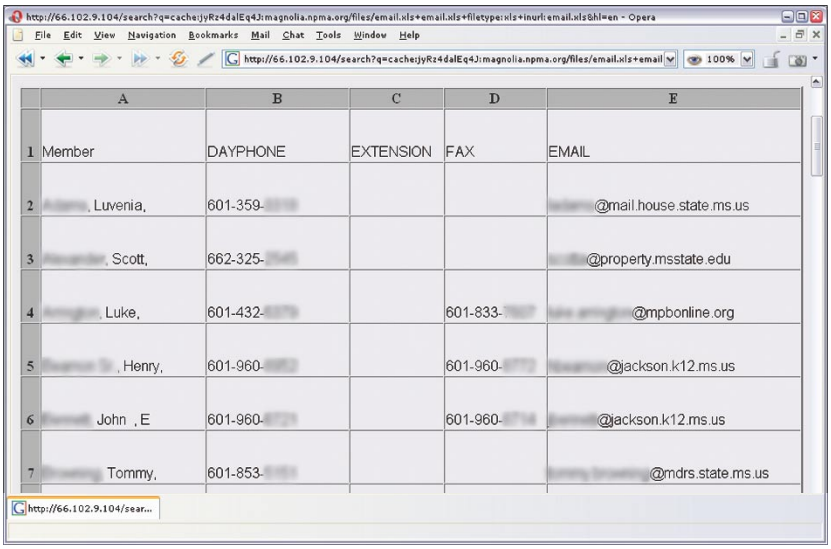
Query	Result
`"http://*:*@www" site`	passwords for site, stored as the string `"http://username:password@www..."`
`filetype:bak inurl:"htaccess|passwd|shadow|ht users"`	file backups, potentially containing user names and passwords
`filetype:mdb inurl:"account|users|admin|admin istrators|passwd|password"`	*mdb* files, potentially containing password information
`intitle:"Index of" pwd.db`	*pwd.db* files, potentially containing user names and encrypted passwords
`inurl:admin inurl:backup intitle:index.of`	directories whose names contain the words admin and backup
`"Index of/" "Parent Directory" "WS_FTP.ini" filetype:ini WS_FTP PWD`	*WS_FTP* configuration files, potentially containing FTP server access passwords
`ext:pwd inurl:(service|authors|administrators |users) "# -FrontPage-"`	files containing *Microsoft FrontPage* passwords
`filetype:sql ("passwd values ****" | "password values ****" | "pass values ****" )`	files containing SQL code and passwords inserted into a database
`intitle:index.of trillian.ini`	configuration files for the *Trillian IM*
`eggdrop filetype:user user`	configuration files for the *Eggdrop* ircbot
`filetype:conf slapd.conf`	configuration files for *OpenLDAP*
`inurl:"wvdial.conf" intext:"password"`	configuration files for *WV Dial*
`ext:ini eudora.ini`	configuration files for the Eudora mail client
`filetype:mdb inurl:users.mdb`	*Microsoft Access* files, potentially containing user account information
`intext:"powered by Web Wiz Journal"`	websites using *Web Wiz Journal*, which in its standard configuration allows access to the passwords file – just enter `http://<host>/journal/journal.mdb` instead of the default `http://<host>/journal/`
`"Powered by DUclassified" -site:duware.com "Powered by DUcalendar" -site:duware.com "Powered by DUdirectory" -site:duware.com "Powered by DUclassmate" -site:duware.com "Powered by DUdownload" -site:duware.com "Powered by DUpaypal" -site:duware.com "Powered by DUforum" -site:duware.com intitle:dupics inurl:(add.asp | default.asp | view.asp | voting.asp) -site:duware.com`	websites using the *DUclassified*, *DUcalendar, DUdirectory*, *DUclassmate*, *DUdownload*, *DUpaypal*, *DUforum* or *DUpics* applications, which by default make it possible to obtain the passwords file – for DUclassified, just enter `http://<host>/duClassified/_private/duclassified.mdb` instead of `http://<host>/duClassified/`
`intext:"BiTBOARD v2.0" "BiTSHiFTERS Bulletin Board"`	websites using the *Bitboard2* bulletin board application, which on default settings allows the passwords file to be obtained – enter `http://<host>/forum/admin/data_passwd.dat` instead of the default `http://<host>/forum/forum.php`

Table 6 presents some sample queries for password-related data.

To make our passwords less accessible to intruders, we must carefully consider where and why we enter them, how they are stored and what happens to them. If we're in charge of a website, we should analyse the configuration of the applications we use, locate poorly protected or particularly sensitive data and take appropriate steps to secure it.

Personal Information and Confidential Documents

Both in European countries and the U.S., legal regulations are in place to protect our privacy. Unfortunately, it is frequently the case that all sorts of confidential documents containing our personal information are placed in publicly accessible locations or transmitted over the Web without proper protection. To get our complete information, an intruder need only gain access to an e-mail repository containing the CV we sent out while looking for work. Ad-



	A	B	C	D	E
1	Member	DAYPHONE	EXTENSION	FAX	EMAIL
2	, Luvenia,	601-359-			@mail.house.state.ms.us
3	, Scott,	662-325-			@property.msstate.edu
4	, Luke,	601-432-		601-833-	@mpbonline.org
5	, Henry,	601-960-		601-960-	@jackson.k12.ms.us
6	John , E	601-960-		601-960-	@jackson.k12.ms.us
7	Tommy,	601-853-			@mdrs.state.ms.us

Figure 11. *Electronic address book obtained through Google*

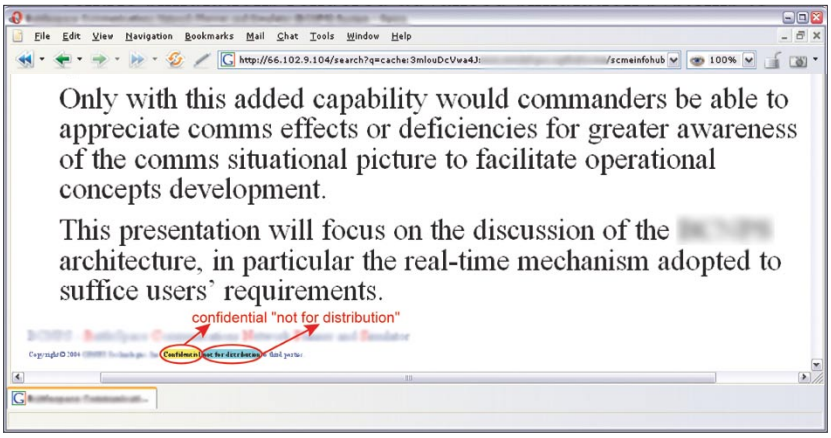


Figure 12. *Confidential document found through Google*

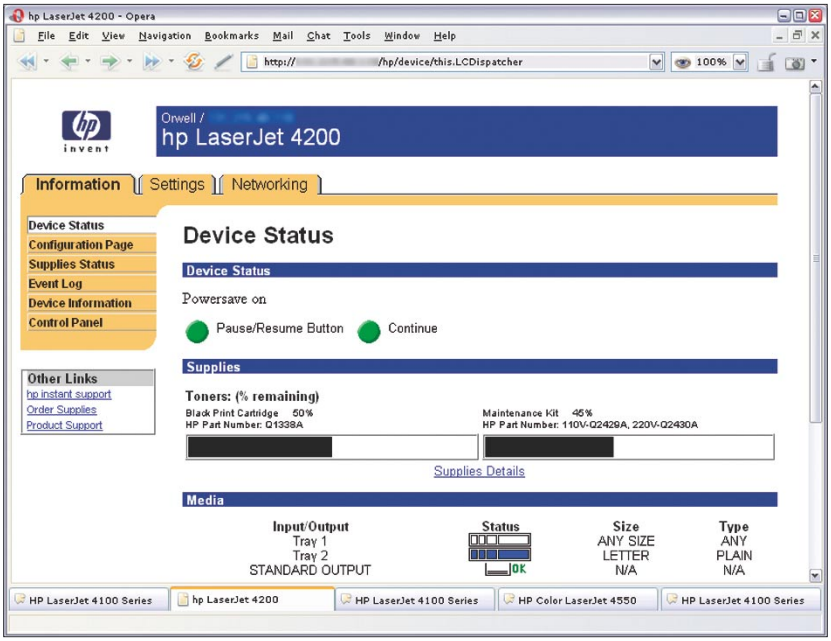


Figure 13. *An HP printer's configuration page found by Google*

dress, phone number, date of birth, education, skills, work experience – it's all there.

Thousands of such documents can be found on the Internet – just query Google for `intitle: "curriculum vitae" "phone * *" "address *" "e-mail"`. Finding contact information in the form of names, phone number and e-mail addresses is equally easy (Figure 11). This is because most Internet users create electronic address books of some description. While these may be of little interest to your typical intruder, they can be dangerous tools in the hands of a skilled sociotechnician, especially if the contacts are restricted to one company. A simple query such as `filetype:xls inurl:"email.xls"` can be surprisingly effective, finding *Excel* spreadsheet called *email.xls*.

All the above also applies to instant messaging applications and their contact lists – if an intruder obtains such a list, he may be able to pose as our IM friends. Interestingly enough, a fair amount of personal data can also be obtained from official documents, such as police reports, legal documents or even medical history cards.

The Web also contains documents that have been marked as confidential and therefore contain sensitive information. These may include project plans, technical documentation, surveys, reports, presentations and a whole host of other company-internal materials. They are easily located as they frequently contain the word *confidential*, the phrase *Not for distribution* or similar clauses (see Figure 12). Table 7 presents several sample queries that reveal documents potentially containing personal information and confidential data.

As with passwords, all we can do to avoid revealing private information is to be cautious and retain maximum control over published data. Companies and organisations should (and many are obliged to) specify and enforce rules, procedures and standard practices for

Table 7. *Searching for personal data and confidential documents*

Query	Result
`filetype:xls inurl:"email.xls"`	*email.xls files*, potentially containing contact information
`"phone * * *" "address *" "e-mail" intitle: "curriculum vitae"`	CVs
`"not for distribution" confidential`	documents containing the confidential clause
`buddylist.blt`	*AIM* contacts list
`intitle:index.of mystuff.xml`	*Trillian IM* contacts list
`filetype:ctt "msn"`	MSN contacts list
`filetype:QDF QDF`	database files for the *Quicken* financial application
`intitle:index.of finances.xls`	*finances.xls* files, potentially containing information on bank accounts, financial summaries and credit card numbers
`intitle:"Index Of" -inurl:maillog maillog size`	*maillog* files, potentially containing e-mail
`"Network Vulnerability Assessment Report" "Host Vulnerability Summary Report" filetype:pdf "Assessment Report" "This file was generated by Nessus"`	reports for network security scans, penetration tests etc.

Table 8. *Queries for locating network devices*

Query	Device
`"Copyright (c) Tektronix, Inc." "printer status"`	PhaserLink printers
`inurl:"printer/main.html" intext:"settings"`	Brother HL printers
`intitle:"Dell Laser Printer" ews`	Dell printers with EWS technology
`intext:centreware inurl:status`	Xerox Phaser 4500/6250/8200/8400 printers
`inurl:hp/device/this.LCDispatcher`	HP printers
`intitle:liveapplet inurl:LvAppl`	Canon Webview webcams
`intitle:"EvoCam" inurl:"webcam.html"`	Evocam webcams
`inurl:"ViewerFrame?Mode="`	Panasonic Network Camera webcams
`(intext:"MOBOTIX M1" | intext:"MOBOTIX M10") intext:"Open Menu" Shift-Reload`	Mobotix webcams
`inurl:indexFrame.shtml Axis`	Axis webcams
`SNC-RZ30 HOME`	Sony SNC-RZ30 webcams
`intitle:"my webcamXP server!" inurl:":8080"`	webcams accessible via *WebcamXP Server*
`allintitle:Brains, Corp. camera`	webcams accessible via *mmEye*
`intitle:"active webcam page"`	USB webcams

handling documents within the organisation, complete with clearly defined responsibilities and penalties for infringements.

Network Devices

Many administrator downplay the importance of securing such devices as network printers or webcams. However, an insecure printer can provide an intruder with a foothold that can later be used as a basis for attacking other systems in the same network or even other networks. Webcams are, of course, much less dangerous, so hacking them can only be seen as entertainment, although it's not hard to imagine situations where data from a webcam could be useful (industrial espionage, robberies etc.). Table 8 contains sample queries revealing printers and webcams, while Figure 12 shows a printer configuration page found on the Web. ■

On the Net

- *http://johnny.ihackstuff.com* – largest repository of data on Google hacking,
- *http://insecure.org/nmap/* – *Nmap* network scanner,
- *http://thc.org/thc-amap/* – *amap* network scanner.