Google Hack Honeypot Manual
v. 1.1 release

http://ghh.sourceforge.net

Contents

1. FAQ

1.1 What is GHH?

GHH is a reaction to an evolving type of malicious web traffic: search engine hacking. GHH is a "Google Hack" honeypot. GHH is designed to provide reconnaissance against attackers that use search engines as a hacking tool against your resources. GHH implements honeypot theory to provide additional security to your web presence.

1.1.1 What's new in GHH v1.1?

Four important updates and features have been added to make GHH v1.1 a much more powerful tool:

- *Proxy Detection*. It is trivial for an attacker to mask reconnaissance and attacks behind a proxy. GHH has added layers of proxy detection to detect the true source of the attack.
- *Centralized Logging Support*. GHH logs to either CSV or MySQL based logs. This allows centralized logging, essentially creating a honeynet.
- *Spoofed File Extensions*. Apache supports file type spoofing via . htaccess. This allows GHH to spoof ASP, JSP and any other file type as a honeypot.
- *New Pre-Built Honeypots*

1.2 What is a honeypot?

A honeypot is, to quote Lance Spitzner founder of the Honeynet Project:

> "An information system resource whose value lies in unauthorized or illicit use of that resource."

Simply put a honeypot is something that **appears** to be vulnerable, but in reality is **recording** illicit use.

GHH allows administrators to track and profile malicious hosts. The honeypot allows administrators to observe who is how attacks are executed via logs. The data generated by this, or any other honeypot can be used to deny future access to attackers, notify service providers of attacks originating from their networks or act as an input for statistical analysis.

1.3 What are search engine hackers and why should I care?

Google has developed a powerful tool. The search engine that Google has implemented allows for searching on an immense amount of information. The Google index has swelled past 8 billion pages [July 2005] and continues to grow daily. Mirroring the growth

of the Google index, an ever-increasing number of installations of web-based applications such as message boards and remote administrative tools has resulted in an increase in the number of poorly configured and vulnerable web apps available on the Internet.

This insecure tools combined with the power of the search engine and index which Google provides results in a convenient attack vector for malicious users. It is in your best interest to be knowledgeable of, and protect yourself from this threat.

1.4 What kind of damage can be done?

A simple query on the Google search engine can reveal "wide open" sensitive data:

```
Search: "# -FrontPage-" inurl:service.pwd
```

This simple search string will return plain text passwords for administrative access via Microsoft's FrontPage. A misconfiguration in the FrontPage software and web server results in sensitive information to be available to anyone who either constructs the search string, or visits an online database of malicious search strings. There are hundreds of similar search engine hacks. Lack of foresight into security issues in web applications is to blame.

1.5 Why should I implement Google Hack Honeypot on my site?

GHH allows you to monitor attempts by malicious attackers to compromise your security. The logging functions that GHH implements allows you, the administrator, to do what you like with the information. You can use the attack database to gather statistics on would-be-attackers, report activities to appropriate authorities and temporarily or permanently deny access to resources.

1.6 How does Google Hack Honeypot work?

Reference http://ghh.sourceforge.net/introduction.php for details on the intersection of theory/practical concerns that drive GHH.

2. Installation

(Flowchart available in package)

2.1 Prerequisites

A web server running Apache and PHP is needed, and optionally MySQL.

2.2 Choosing a Honeypot

Download honeypots from the GHH repository on SourceForge. (http://ghh.sourceforge.net) There are multiple types of honeypots available which emulate different types of Google Hack Database (GHDB) signatures. You can download and configure an official GHH honeypot or follow the directions in the "Custom Honeypot" section of this document to create your own. By picking or creating a honeypot for a web application that is recently discovered to be vulnerable there is less chance of your honeypot being avoided by search engine hackers.

2.3 Download

The latest version of GHH will be available at the official project website located at http://ghh.sourceforge.net.

2.4 Installation

Follow these steps to install GHH onto your server:

1. GHH should be unzipped into a folder that **is not** in the document root of your web server. (We don't want Google to find us yet!)

2.5Global Configuration

Inside of the uncompressed installation package, locate *config.php*.

Decide which logging system to use, CVS or MySQL. Edit config.php so the variable $LogType represents your choice, "MySQL" or "CSV" (caps sensitive) Below this is the configuration section for both types of logs. Configure the section for the logging type you chose.

CVS:
Create a file for your GHH log, **anywhere but your document root**. Example: /apache/ghhlog.csv, **Not: /apache/htdocs/ghhlog.csv**
(If access to folders that are not in the document root is not available, use a password protected folder, covered with .htaccess or similar)

chmod the log file so it is writable by Apache.

Change the *$Filename* variable to contain the (full) path to your CVS log file you created in 2.4.2.

MySQL:
Configure config.php with your MySQL server information (i.e. username, password, host)

Change the *$RegisterGlobals* variable to 'false' if you require register_globals to be on in the server's php.ini (or if you are getting a blank page when viewing the honeypot file).

2.6 Honeypot Configuration

There is a README.txt file in the downloaded package. Because different honeypots may have different configuration instructions, this file is necessary for each separate honeypot. README.txt contains instructions to setup the particular honeypot, and may be intricate depending on the complexity of the implemented honeypot. Open the README.txt file in the file you downloaded, and follow its configuration instructions. This process will involve placing the honeypot into your document root, and building an . htaccess file describe next (if necessary)…

1.6.1 What about non *.php honeypots?

For honeypots which require that the filetype does not end in “php” (honeypots that end in “txt”, “asp”, “mdb”, etc) require a .htaccess file be created and placed in the same directory as the honeypot.

The .htaccess file should contain the following (or similar, check the honeypot README, there may be an htaccess in your honeypot package already):

```
AddHandler application/x-httpd-php .ext
AddType application/x-httpd-php .ext
```

This tells Apache to render any of the listed file extensions with PHP. Append or remove file extensions as necessary for the honeypot being used. Make sure to place this file in the same directory as the honeypot.

2.7 Indexing

In order for the honeypot to work, it must be visible to search engines. There are different ways to accomplish this task. The GHH team recommends setting up a secret hyperlink on a HTML of a page of your site. Add a link to a page that Google indexes, or other search engines like so:

```
<a href=http://yourdomain.com/honeypot.php>.</a>
```

Where the “.” is the same color as the background of the page. This invisible link attracts the attention of search engines. However, regular viewers of your site will not notice or visit the link.

Note: If you are well versed with CSS & PHP, you can detect when the Googlebot comes around. When normal visitors are around, use display:none to hide your honeypot link. When Googlebot comes around, remove that style and let Google index your link! This lets other spiders index your site and hiding your link from visitors. However, Googlebot doesn’t like display:none, making it necessary to switch it off when Googlebot comes around.)

After the honeypot has been linked to it is time to set *$SafeReferer* variable. Set this variable equal to the page that the honeypot is linked from.

`$SafeReferer` is used to detect when someone clicks the hidden hyperlink. This variable is associated with the “Crawler Detected” alert used in the logs. It signifies one of three things.

1. A search engine indexed the link.
2. An innocent browser found the link and clicked it.
3. The link was crawled with a tool like wget or an offline browser.

None of these types of access to your honeypot is a “true” intrusion. Thus these count as a false-positive. GHH will look at the “HTTP_REFERER” header and determine if a browser came from the `$SafeReferer`.

Search engines will not index your site immediately. Their spiders take time. If you are impatient, try submitting the page directly to the engine.

https://www.google.com/webmasters/sitemaps/

3. Usage

3.1 Reading Logs

Logs are created in two different formats: CSV (Comma Separated Values), or MySQL.

3.1.1 CSV Log format:

HoneypotName : Name of the honeypot that was accessed.
DateTime : Date and time of the honeypot access.
Attack['IP'] : IP address of the attacker and proxy (if available).
Attack['request'] : Request made to the honeypot.
Attack['referer'] : URI of referrer which lead to honeypot.
Attack['accept'] : Media which browser accepts.
Attack['charset'] : Character set which browser accepts.
Attack['encoding'] : Encoding which browser accepts.
Attack['language'] : Language which browser accepts.
Attack['connection'] : Browser connection.
Attack['keep_alive'] : Browser Keep-Alive.
Attack['agent'] : User-Agent string of browser
Signatures : GHH determined signatures.

3.1.2 MySQL Log format:

ID : Primary Key for DB.
Owner : Owner of the honeypot.
Tripped : Name of accessed honeypot.
TimeOfAttack : Date and time of the honeypot access.
Host : IP address of the attacker and proxy (if available).
RequestURI : Request made to the honeypot.
Referrer : URI of referrer which lead to honeypot.
Accepts : Media which browser accepts.
AcceptsCharset : Character set which browser accepts.
AcceptLanguage : Language which browser accepts.
Connection : Browser connection.
keepalive : Browser Keep-Alive.
UserAgent : User-Agent string of browser.

3.2 Making Sense of It All

Do not panic if your log file has a large number of requests in it. Honeypots are designed to be accessed. This log is a potent source of information.

3.3 Multiple Honeypots

Inside of each honeypot file, there is a variable for setting a common configuration file, `$ConfigFile`. With multiple honeypots or honeypot files, you can include the same standard configuration file and have each honeypot write to the same log file. The honeypot's name will appear as the first value in the logs. It is a smart idea to remove configuration files you are not using with GHH. We include one in each release by default. This may include extra htaccess files, depending on the honeypots.

4. Advanced Usage

4.1 Custom Honeypots

To make a custom honeypot, use the template file in the download section of the GHH project page on Sourceforge. The link there will be available on http://ghh.sourceforge.net. This is a template file for a simple honeypot, and will be setup as a dummy vulnerable page as an example. Download the "Custom Honeypot Template" File from a mirror and find template.php file inside. Look for the "Begin Custom Honeypot Section" of the code.

The first task is to change the $HoneypotName variable to reflect the name of your honeypot. This will appear in the logs when your honeypot is visited over the web. The next line is an echo statement; this outputs the source HTML of the honeypot. Template.php uses PHP Shell 1.7 as an example. To customize file replace the source found in the echo code with the source another vulnerable web application.

Once the honeypot HTML is echo'd to the attacker, it's time to determine what means he/she took to find our honeypot. This is done by checking the HTTP_REFERER header sent by the attacker's browser. If there is no HTTP_REFERER header, the default signatures will identify it.

You can retrieve the HTTP_REFERER header sent by the attacker with the $Attack ['referer'] variable. There are two examples in the template. The first searches for the name of the vulnerable target in the HTTP_REFERER string, because popular search engines include the query in the URL of their query results. The second example searches for the Google Hacking Data Base query in the HTTP_REFERER string, which highlight that the attacker most likely found the honeypot using a GHDB signature. Whatever signatures you decide to create, append them to the *$Signature[]* array and they will be put in the logs. The two signatures included in the template have examples of this. You should remove or edit the example signatures.

4.2 Central Logging (Honeynet)

With our v1.1 release, we have included MySQL support in GHH honeypots. We are releasing a .sql create script that will create a MySQL table that GHH can log to. Included in that package is a log viewer that can read those logs. To use the viewer, configure index.php in that package to include a username and password, as well as MySQL connection information. This package can be found on our Sourceforge project page. All honeypots select database 'ghh' by default.

A simple honeynet flowchart resides in the documentation package.

4.3 GHH and Security Policies

Our number one priority is to serve as a research too. All security policies derived from Carefully scrutinized GHH logs because false positives are possible in all honeypots and GHH is no exception. Contact the GHH team if you have queries related to honeypot policy.

5. GHH Security

5.1 Secure Configuration Checklist

To secure GHH, implement and verify that the following measures are working as noted:

- The log file cannot be reached by URL without authentication [outside the document root or protected in a .htaccess protected directory].
- A .phps file extension won't reveal your honeypot's source code.
- Your security policies will not introduce vulnerability if the honeypot is fingerprinted.
- Your database, database viewer, and anything else related to GHH that isn't a honeypot **is not** accessible over the web.

5.2 Index Avoidance

If it is a requirement to place the log file and config file in the document root of a web server, then here are some tips to protect your log file and config.php file:

- Place your log and config.php file in a password protected directory, using a .htaccess or similar authentication mechanism.
- Verify that directory listing is disabled, or else place an index.html or index.php file in the directory to prevent the contents of a directory from being displayed. If you password protect a directory, this should not be necessary.

5.3 Security Policies

As said previously, GHH is meant to be a research tool to help develop security policy. By creating a script that automates a security policy based on the GHH log file you may introduce vulnerability to your network. It cannot be assumed that GHH logs do not contain false positives, and because of this policies should be carefully developed when GHH logs are involved. Policy is important!

6. Miscellaneous

Working on a Free Open Source Software project has been quite an experience. There are many people we would like to thank:

- Google. Your search engine has provided to be an interesting tool and helped drive us towards this endlessly interesting project.
- Honeynet Project. The research, theory, and implementations that have come from your excellent work have provided an excellent reference point for some of the difficult design decisions we had to make.
- Google Hack Database. Community driven and constantly refreshed, the GHDB provides the “meat” on the “bones” of GHH. (at http://johnny.ihackstuff.com)

– ***The GHH Team***
– ***http://ghh.sourceforge.net***