ghh

The "Google Hack" Honeypot

Google Hack Honeypot Manual

http://ghh.sourceforge.net

Contents

1. FAQ
    1. What is GHH?
    2. What is a Honeypot?
    3. What are search engine hackers?
    4. What kind of damage can be done?
    5. Why should I implement GHH on my site?
    6. How does GHH work?
2. Installation
    1. Prerequisites
    2. Choosing a Honeypot
    3. Download
    4. Installation
    5. Global Configuration
    6. Honeypot Configuration
    7. Indexing
3. Usage
    1. Reading Logs
    2. Making Sense of It All
    3. Multiple Honeypots
4. Advanced Usage
    1. Custom Honeypots
    2. GHH Log viewer
    3. GHH and Security Policies
5. GHH Security
    1. Secure Configuration Checklist
    2. Index Avoidance
    3. Security Policy Issues

1.FAQ

1.1What is GHH?

GHH is a reaction to a new type of malicious web traffic: search engine hackers. GHH is a “Google Hack” honeypot. It is designed to provide reconaissance 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.

1.2 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 by malicious attackers.

GHH allows administrators to track malicious hosts: observe who is perpetrating the attack and how it is being executed via the log. 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 [February 2005] and continues to grow daily. Mirroring the growth of the Google index, the spread of web-based applications such as message boards and remote administrative tools has resulted in an increase in the number of misconfigured and vulnerable web apps available on the Internet.

These insecure tools, when combined with the power of a search engine and index which Google provides, results in a convenient attack vector for malicious users. It is in your best interest to be knowledgable 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 improperly secured 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 this 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. A lack of foresight into security issues in web applications are to blame.

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

GHH allows you to safely 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.htm for details on the intersection of theory/practical concerns that drive GHH.

2. Installation

2.1 Prerequisites

A web server running Apache and PHP, IIS and .NET support coming in future release.

2.2 Choosing a Honeypot

The Download site <http://ghh.sourceforge.net> offers multiple types of honeypots to emulate different types of GHDB signatures. You can pick one from the official GHH site, 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, or otherwise less-well-known, 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. There will be many honeypots available to implement.

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.
2. A file should be created for your GHH log, **anywhere but your document root.** Example: /apache/ghhlog.csv **Not: /apache/htdocs/ghhlog.csv**
   (if access to folders that aren't in the document root isn't available, use a password protected folder, covered with .htaccess)
3. Continue to configuration section.

2.5 Global Configuration

Inside of the uncompressed installation package locate *config.php*. This file includes one variable that need to be changed in order for GHH to work:

Change the *$Filename* variable to contain the path to your log file you created in 2.4.2.
Change the *$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 folder you unzipped into your web server. Because different honeypots may have different configuration instructions, this file is necessary for each seperate honeypot. README.txt contains instructions to setup the particular honeypot, and may be intricate depending on the complexity of the honeypot being implemented. (i.e. a phpBB honeypot) Open the README.txt file in the file you downloaded, and follow it's configuration instructions.

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 in the HTML of a page of your site. Add a link to a page that is currently indexed by Google, 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 directs search engines to crawl the page, but regular viewers of your site will not notice or visit the link.

There are other options that will get the honeypot indexed include image tag inclusion:

```
<img src="http://yourdomain.com/honeypot.php" width="0"
height="0">
```

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

`$SafeReferer` is used to detect when someone clicks the hidden hyperlink. This variable links 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.

These hits are more than likely a false-positives. 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.

3. Usage

3.1 Reading Logs

The logs are done in the CSV format. (Comma Separated Values) Each field is separated by a comma. The fields in the document include:

Tripped: The honeypot was accessed / tripped. (If you have multiple honeypots, this will tell you which one was accessed)
Time of Attack: The time the honeypot was viewed.
Host: The IP address of the attacker.
Requested URI: The Uniform Resource Identifier made to reach your site.
Referrer: This will have the query used in the search engine in most cases, alarming you to what the attacker attempted to find, and how they tried to find it. *The most important detail of the log.*
Accepts: Contents of the *Accept:* header if there is one.
Accepts Charset: Contents of the Accepts_Charset header if there is one.
Accept Language: Contents of the *Accept-Language* .
Connection: Contents of the *Connection:* header from the attack request .
User Agent: The user agent of the attacker.
Signatures: The signature of attack the honeypot was able to determine from a combination of browsers headers.

3.2 Making Sense of It All

Do not panick if your log file (*$Filename*) has a large number of requests in it. Honeypots are designed to be accessed. This log is a potent source of information to see how

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.

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 GHH Log Viewers

There will be a sample log viewer on the Download page <http://ghh.sourceforge.net> to provide a front-end to the log file. **This feature is being worked on currently and is not available at relese.** It is not recommend to use this viewer if you have the resources available to parse the logs another way. The reason the sample viewer is not recommend is because it needs to be placed on the web.

Options for the time being:

`http://www.google.com/search?q=csv+parser`

4.3 GHH and Security Policies

Our number one priority is to serve as a research tool which can be used to develop security policy. All security policies derived from GHH logs should be carefully scrutinized 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

In order to ensure that GHH is secured against attacks itself, 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.

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 sure directory listing is disabled, or else place a 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!

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