Web Log Analysis and Defense with mod_rewrite

GIAC (GCIH) Gold Certification

Author: Rick Wanner, rwanner@pobox.com
Advisor: Antonios Atlasis

Accepted: December 27th, 2012

Abstract
The production web server, which by necessity must face the Internet, creates a unique problem when it comes to creating a defense plan. Anybody who has every attempted to perform log analysis on the volume of logs created by a production web server understands that web server log analysis can be a daunting task. By concentrating the analysis on the error logs generated by the web server it is possible to reduce the logs to a manageable volume and get a good view into the types of attacks the web server is experiencing. Armed with this analysis and a basic knowledge of regular expressions and the mod_rewrite module built into the Apache web server, it is relatively easy to implement blocking rules for mod_rewrite and reduce the effectiveness of potential attacks against the web server.

1. Introduction

Anybody who has been tasked with defending a production web server has quickly realized that the volume of logs generated, often measuring in gigabytes or terabytes a day, defies analysis even with the use of a good event management solution. This paper is the result of a personal education project to find a low-overhead way to analyze production web server logs to get a good understanding of the sort of probes and attacks that are logged and defend against them. The methodology was simple. Each week, spend half an hour and analyze the logs of a web server with the goal to identify and understand the nature of potentially malicious, traffic against the web server with the goal to defend against potentially malicious traffic. At first the project was overwhelming. The sheer volume of logs generated by a production web server creating a real life needle in a haystack scenario. However, with further research it became clear that it was not necessary to look at the successful web server transactions, but just by focusing on the web server error logs, it is possible to get a good snapshot of the type of potentially malicious traffic aimed at the web server. Once there was a better understanding of the nature of the traffic, the project evolved into finding a low overhead way to block these probes and attacks without using an Intrusion Prevention System (IPS) or web firewall thus decreasing the size of the potential attack footprint of the web server.

This paper does not chronicle the original work, but rather walks through similar steps with a couple of non-production Apache web servers placed on the Internet to examine the probes and attacks.

The paper is organized as follows. The first section provides some background on the Apache web server, what logs are available for analysis, and the format of those logs. Next is a walkthrough of a high level analysis of the logs organizing the top logs by three different criteria; by request, by useragent, and by source IP address. The next section provides some basics on how to create rules using Apache mod_rewrite, including the structure of mod_rewrite rules and the basics of regular expressions used in mod_rewrite rules. The final section pulls together the previous sections to create mod_rewrite rules specific to the results of the log analysis.

2. Apache Web Server

In order to understand the later material it is essential to have a basic understanding of logging in the operating environment, which in this case is an Apache web server deployed on CentOS Linux.

Apache stores its log files in /var/log/httpd/. There are two log files that are of interest in analyzing the potentially malicious traffic on an Apache webserver; the Apache access_log and error_log.

The primary logfile used by Apache is /var/log/httpd/access_log. Whenever a connection is made to the web server a log is written to the access_log. A small sample of an access_log is as follows:

```
58.218.199.250 - - [13/May/2012:08:21:34 -0600] "GET http://www.piggmail.com/proxyheader.php
HTTP/1.1" 404 294 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
58.218.199.250 - - [13/May/2012:09:53:54 -0600] "GET http://59.53.91.9/proxy/judge.php HTTP/1.1" 404
288 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
58.218.199.250 - - [13/May/2012:11:23:47 -0600] "GET http://www.piggmail.com/proxyheader.php
HTTP/1.1" 404 294 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
95.110.230.50 - - [13/May/2012:12:19:00 -0600] "HEAD /manager/status HTTP/1.1" 404 - "-"
"Java/1.6.0_26"
99.241.40.206 - - [13/May/2012:15:00:56 -0600] "GET /alcatelanonymous/cgi-bin/lesmisreg.cgi
HTTP/1.1" 200 790 "-" "Mozilla/5.0 (iPod; U; CPU iPhone OS 4_2_1 like Mac OS X; en-us)
AppleWebKit/533.17.9 (KHTML, like Gecko) Version/5.0.2 Mobile/8C148 Safari/6533.18.5"
10.0.1.22 - - [13/May/2012:16:25:37 -0600] "GET / HTTP/1.1" 403 5043 "-" "EZI_WIN_HTTP_AGENT"
```

The access_log is formatted using the Apache Combined Log Format (Apache Software Foundation, 2012). The combined log format is formatted as:

"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""

Using one of the above logs as an example, the format of an entry in access_log is as follows:

Field	Field Description	Example
%h	IP Address of the client	58.218.199.250
%l	Identity of the client (using identd).	-
%u	userid of the requesting user determined by HTTP authentication.	-
%t	The date and time the request was received. The format is [day/month/year:hour:minute:second timezoneoffset]	[13/May/2012:11:23:47 -0600]
%r	Request from the client	"GET http://www.piggmail.com/proxyheader.php HTTP/1.1"
%>s	Status Code sent back to the client.	404
%b	Size of the object returned to the client.	294
%referer	Site that the client reports having been referred from.	"-"
%{Useragent}	Identifies the type of the client browser.	"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

Table 1: Apache access_log Format

This log format is used for a variety of log types. Some fields will not be relevant for some log messages. In that case "-" is used to indicate a null value.

Of particular interest in these logs is the status code. The status code is used to indicate what the result of the request was. The Hypertext transfer protocol (HTTP), under RFC 2616, specifies a number of classes of status codes.

Status Class	Class	Description
1xx	Informational	Request received, continuing
2xx	Success	Action was successful
3xx	Redirection	Further action required to complete the request
4xx	Client error	The request cannot be completed
5xx	Server error	Server failed to complete the request

Table 2: HTTP Status Code Classes (Fielding, et al., 1999)

Of these classes there are two that are of the most interest to baseline the logs. The 2xx class indicates client requests that the web server successfully completes; presumably normal traffic to the web server. The 4xx class indicates client requests that the server was unable to complete; in other words, anomalous traffic to the web server.

RFC 2616 (Fielding, et al., 1999) specifies a number of error codes in the 400 range. The one that is of most interest in understanding the anomalous traffic to the web server is 404 – Not Found.

Status Code	Message	Description
404	Not Found	The server has not found anything matching the request

Table 3: 404 Status Code (Fielding, et al., 1999)

The 404 error code indicates to the browser client that a requested web page or other content is not available on the server. These errors will generally fall into four categories; user typos, requests for old web pages, misconfigurations of the web server, or probes to determine what software is installed, and what vulnerabilities exist on the server, looking for potentially vulnerable targets.

The other principal log used by Apache httpd is used to output errors. This file is /var/log/httpd/error_log. The error_log does not contain as much detail as the access_log, but is useful in clarifying the access_log entries. A sample of the error_log corresponding to the 404 status entries in the access_log follows:

```
[Sun May 13 08:21:34 2012] [error] [client 58.218.199.250] script '/var/www/html/proxyheader.php' not
found or unable to stat
[Sun May 13 09:53:54 2012] [error] [client 58.218.199.250] File does not exist: /var/www/html/proxy
```

```
[Sun May 13 11:23:47 2012] [error] [client 58.218.199.250] script '/var/www/html/proxyheader.php' not found or unable to stat
[Sun May 13 12:19:00 2012] [error] [client 95.110.230.50] File does not exist: /var/www/html/manager
```

The error_log utilizes the Apache ErrorLogFormat (Apache Software Foundation, 2012a). The ErrorLogFormat is formatted as:

"[%t] [%l] [pid %P] %F: %E: [client %a] %M"

Field	Field Description	Example
[%t]	The date and time the request was received. The format is [day/month/year:hour:minute:second timezoneoffset]	[Sun May 13 08:21:34 2012]
[%l]	Identity of the client (using identd).	
[%pid P]	Process ID number of the current process	
%F:	Source file name and line number of the log call	
%E:	APR/OS error status code and string	[error]
[client %a]	Client IP address	[client 58.218.199.250]
%M	The log message corresponding to the error.	script '/var/www/html/proxyheader.php ' not found or unable to stat

Table 4: Apache error_log Format

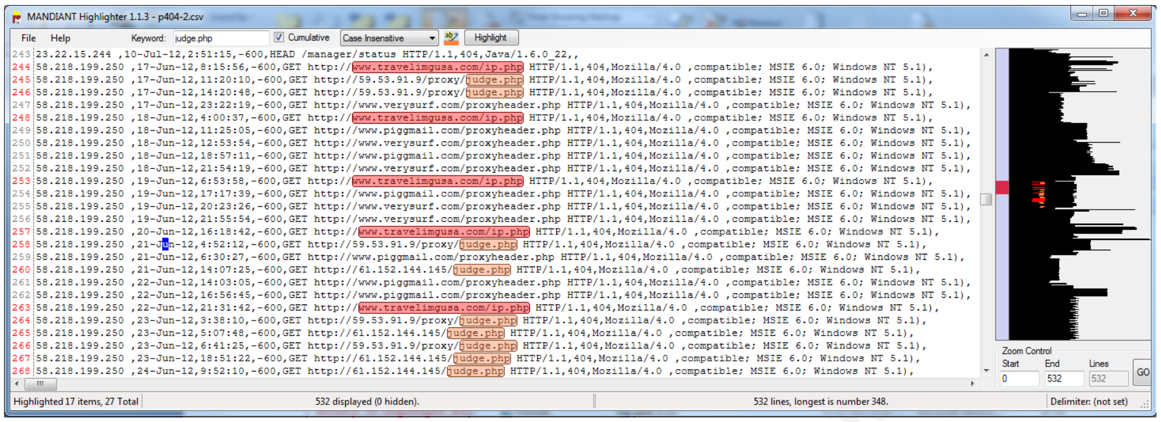
This log format is used for a variety of log types. Some fields will not be relevant for some log messages. Unlike the combined log format that indicated blank fields with a "-", the ErrorLogFormat just omits those fields, and the surrounding brackets, when they are not required.

While it does not have the same level of detail as the access_log, the error_log does provide enough information to correlate the logs with the access_log entries and provides detail on the error that caused the 404 log to be generated. This will be useful when performing analysis.

3. 404 Log Analysis

There are a number of different tools that can be used for log analysis. Two personal favorites are Mandiant Highlighter (Mandiant, 2011), and Microsoft Excel.

Mandiant Highlighter is a free utility designed for free-form log analysis. It provides a number of different ways to view the log file being analyzed as well as the ability to highlight keywords and temporarily remove lines from the view in order to focus the analysis.



Keyword: judge.php
Cumulative: ✓
Case Insensitive

```
23.22.15.244 ,10-Jul-12,2:51:15,-600,HEAD /manager/status HTTP/1.1,404,Java/1.6.0_22,,
58.218.199.250 ,17-Jun-12,8:15:56,-600,GET http://www.travelimgusa.com/ip.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,17-Jun-12,11:20:10,-600,GET http://59.53.91.9/proxy/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,17-Jun-12,14:20:48,-600,GET http://59.53.91.9/proxy/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,17-Jun-12,23:22:19,-600,GET http://www.verysurf.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,18-Jun-12,4:00:37,-600,GET http://www.travelimgusa.com/ip.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,18-Jun-12,11:25:05,-600,GET http://www.piggmail.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,18-Jun-12,12:53:54,-600,GET http://www.verysurf.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,18-Jun-12,18:57:11,-600,GET http://www.piggmail.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,18-Jun-12,21:54:19,-600,GET http://www.verysurf.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,19-Jun-12,6:53:58,-600,GET http://www.travelimgusa.com/ip.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,19-Jun-12,17:17:39,-600,GET http://www.piggmail.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,19-Jun-12,20:23:26,-600,GET http://www.verysurf.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,19-Jun-12,21:55:54,-600,GET http://www.verysurf.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,20-Jun-12,16:18:42,-600,GET http://www.travelimgusa.com/ip.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,21-Jun-12,4:52:12,-600,GET http://59.53.91.9/proxy/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,21-Jun-12,6:30:27,-600,GET http://www.piggmail.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,21-Jun-12,14:07:25,-600,GET http://61.152.144.145/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,22-Jun-12,14:03:05,-600,GET http://www.piggmail.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,22-Jun-12,16:56:45,-600,GET http://www.piggmail.com/proxyheader.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,22-Jun-12,21:31:42,-600,GET http://www.travelimgusa.com/ip.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,23-Jun-12,3:38:10,-600,GET http://59.53.91.9/proxy/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,23-Jun-12,5:07:48,-600,GET http://61.152.144.145/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,23-Jun-12,6:41:25,-600,GET http://59.53.91.9/proxy/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,23-Jun-12,18:51:22,-600,GET http://61.152.144.145/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
58.218.199.250 ,24-Jun-12,9:52:10,-600,GET http://61.152.144.145/judge.php HTTP/1.1,404,Mozilla/4.0 ,compatible; MSIE 6.0; Windows NT 5.1),
```

Start 0 End 532 Lines 532

Highlighted 17 items, 27 Total | 532 displayed (0 hidden). | 532 lines, longest is number 348. | Delimiter: (not set)

Figure 1: Mandiant Highlighter Screen Shot

Microsoft Excel's sorting and filtering capabilities are very useful for analysis, however importing the log file and breaking it into fields can sometimes take a bit of effort.

Another option is to configure your webserver to submit logs to the Internet Storm Center's 404 Project (Ullrich, 2011). The 404 Project will do the basics of the log analysis. More details on the 404 Project can be found at http://isc.sans.edu/404project/ (Internet Storm Center, 2011).

For this paper the logs were analyzed using both Mandiant Highlighter and Microsoft Excel.

There are a number of different ways to look at the logs. The three that are most useful for detecting potential probes and attacks are; by request, by source IP and by useragent.

3.1. Analysis by Request

When analyzed by request a number of interesting requests jump out.

GET /favicon.ico HTTP/1.1
GET /phpMyAdmin/index.php HTTP/1.1
POST http://vlad-tepes.bofh.it/freenode-proxy-checker.txt HTTP/1.0 GET http://www.travelimgusa.com/ip.php HTTP/1.1
GET /w00tw00t.at.blackhats.romanian.anti-sec:) HTTP/1.1

Table 5: Top Requests: Analysis by Request

Analyzing by request can often give the best idea of what the attacker is looking for on your server. If the software being scanned for is not present on your machine, then breathe a sigh of relief, if it is installed, have a brief moment of panic, and take a look at the potential consequences and how to mitigate them.

3.1.1. GET /favicon.ico HTTP/1.1

A sample of the log with the GET /favicon.ico HTTP/1.1 request is:

```
174.89.128.244 [15/Jul/2012:17:13:57 -0600] "GET /favicon.ico HTTP/1.1" 404 "Mozilla/5.0 (compatible;
MSIE 9.0; Windows NT 6.0; Trident/5.0)"
```

Sorting by request it is obvious that GET /favicon.ico is by far the most common request. Earlier in the paper it was indicated that 404 logs can be generated for a number of reasons. The most likely being user typos, requests for web pages that no longer exist, misconfigurations on the web server, and some will be probes or attacks. It is important to remember that not everything necessarily is an attack. This log is an indication of a misconfiguration on the web server.

Favicon is short for favorite icon (Heng, 2008). The favicon.ico file contains an icon which is provided by the website and used by the web browser to display beside the URL on the address bar, next to the name on the tab displaying the website, and in the favorites menu. There are a number of websites on the Internet that will help with the generation of favicon.ico files. Creating a favicon.ico file and placing it in the root of the web server eliminates this 404 log message.

The apple-touch-icon requests are a similar configuration issue.

```
99.241.40.206 [16/Jul/2012:17:01:37 -0600] "GET /apple-touch-icon-57x57-precomposed.png HTTP/1.1"
404 "MobileSafari/6533.18.5 CFNetwork/485.12.7 Darwin/10.4.0"
```

The apple-touch-icon is displayed by the Safari Mobile Browser on Apple portable devices like the iPhone, iPad or iPod Touch. These servers do not have either of these icons present.

3.1.2. GET /phpMyAdmin/index.php HTTP/1.1

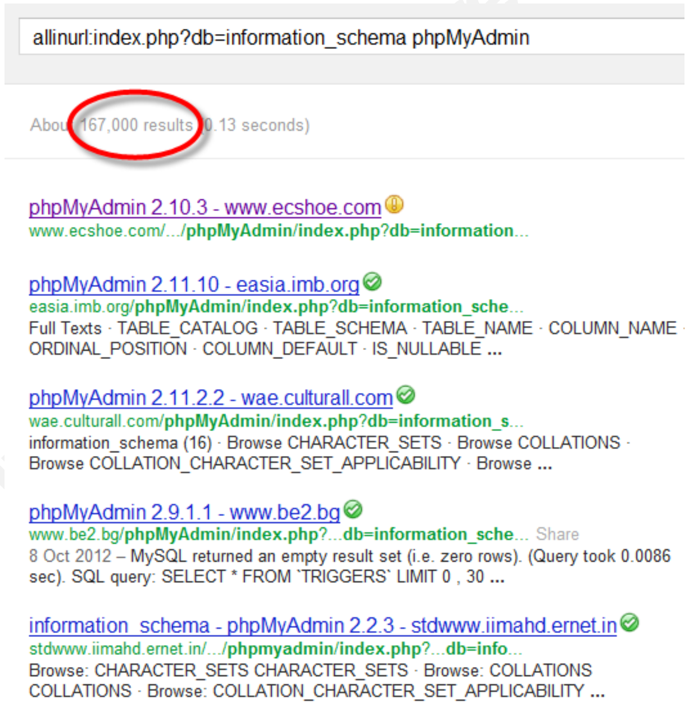
A number of different phpMyAdmin related requests were found in the error logs. Some examples:

```
207.20.47.62 [04/Jul/2012:17:24:12 -0600] "GET /phpMyAdmin/index.php HTTP/1.1" 404 "Mozilla/5.0
(Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.220 Safari/535.1"
74.52.179.162 [14/Jul/2012:17:06:21 -0600] "GET /phpMyAdmin/scripts/setup.php HTTP/1.1" 404
"ZmEu"
207.20.47.62 [04/Jul/2012:17:24:11 -0600] "GET /admin/phpmyadmin/index.php HTTP/1.1" 404
"Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.220
Safari/535.1"
```

```
207.20.47.62 [04/Jul/2012:17:24:12 -0600] "GET /typo3/phpmyadmin/index.php HTTP/1.1" 404
"Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.220
Safari/535.1"
```

phpMyAdmin is a popular, free software tool for managing the administration of MySQL instances (phpMyAdmin Development Team, 2012). As with most complex code written in PHP phpMyAdmin has been the source of many vulnerabilities (phpMyAdmin Development Team, 2012b).

Because phpMyAdmin is freely available, it is widely installed and not always properly updated by website owners, making it a popular scanning target. Without doing active scanning it is possible to detect hundreds of thousands of potential Internet facing phpMyAdmin instances using a Google search for "allinurl:index.php?db=information_schema phpMyAdmin".


allinurl:index.php?db=information_schema phpMyAdmin

About 167,000 results (0.13 seconds)

phpMyAdmin 2.10.3 - www.ecshoe.com
www.ecshoe.com/.../phpMyAdmin/index.php?db=information...

phpMyAdmin 2.11.10 - easia.imb.org
easia.imb.org/phpMyAdmin/index.php?db=information_sche...
Full Texts · TABLE_CATALOG · TABLE_SCHEMA · TABLE_NAME · COLUMN_NAME · ORDINAL_POSITION · COLUMN_DEFAULT · IS_NULLABLE ...

phpMyAdmin 2.11.2.2 - wae.culturall.com
wae.culturall.com/phpMyAdmin/index.php?db=information_s...
information_schema (16) · Browse CHARACTER_SETS · Browse COLLATIONS · Browse COLLATION_CHARACTER_SET_APPLICABILITY · Browse ...

phpMyAdmin 2.9.1.1 - www.be2.bg
www.be2.bg/phpMyAdmin/index.php?...db=information_sche... Share
8 Oct 2012 – MySQL returned an empty result set (i.e. zero rows). (Query took 0.0086 sec). SQL query: SELECT * FROM `TRIGGERS` LIMIT 0 , 30 ...

information_schema - phpMyAdmin 2.2.3 - stdwww.iimahd.ernet.in
stdwww.iimahd.ernet.in/.../phpmyadmin/index.php?...db=info...
Browse: CHARACTER_SETS CHARACTER_SETS · Browse: COLLATIONS COLLATIONS · Browse: COLLATION_CHARACTER_SET_APPLICABILITY ...

Figure 2: phpMyAdmin Google Search

3.1.3. POST http://vlad-tepes.bofh.it/freenode-proxy-checker.txt HTTP/1.0

There were a number of entries in the logs that referred to proxy or something similar:

```
85.190.0.3 [16/Jul/2012:21:30:46 -0600] "POST http://vlad-tepes.bofh.it/freenode-proxy-checker.txt
HTTP/1.0" 404 "-"
```

The purpose of this attempt was not obvious at first. The file freenode-proxy-checker.txt still exists at vlad-tepes.bofh.it. The key to this attempt is the POST HTTP method. The POST method requests that the web server store the data in the request into the requested file. Since the requested file in this case is located at host vlad-tepes.bofh.it, the POST request is requesting that the data be stored at a remote location. In other words the request is checking to see if the web server is acting as a proxy. The interesting bit of this is that the attacker does not even require control of or access to vlad-tepes.bofh.it to determine the success or failure of this probe. If the return code returned by the web server is not a 400 series error code the attacker knows the web server is most likely acting as a proxy.

A number of other requests were detected which demonstrate the same type of probe.

```
58.218.199.250 [03/Jul/2012:04:16:39 -0600] "GET http://59.53.91.9/proxy/judge.php HTTP/1.1" 404
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
58.218.199.250 [05/Jul/2012:10:48:47 -0600] "GET http://www.piggmail.com/proxyheader.php HTTP/1.1"
404 "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
58.218.199.250 [06/Jul/2012:12:32:18 -0600] "GET http://www.verysurf.com/proxyheader.php HTTP/1.1"
404 "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

and

```
58.218.199.250 [18/Jun/2012:04:00:37 -0600] "GET http://www.travelimgusa.com/ip.php
```

which, although it does not explicitly have proxy in the request is the same type of probe.

3.1.4. GET /w00tw00t.at.blackhats.romanian.anti-sec:) HTTP/1.1

A sample of the log with this request is:

```
74.52.179.162 [14/Jul/2012:17:06:21 -0600] "GET /w00tw00t.at.blackhats.romanian.anti-sec:) HTTP/1.1"
404 "ZmEu"
```

This request is related to a fairly common scanner, ZmEu, that scans for the presence of installations of common vulnerable tools such as phpMyAdmin (Ragan, 2011). This particular request is the first request sent by the scanner looking for the phpMyAdmin instances in Section GET /phpMyAdmin/index.php HTTP/1.1 . This is what is often referred to as a knock-knock or hello request which is used by these scanning tools to announce themselves to the web server. See section ZmEu for more information on this scanner.

3.2. Analysis by UserAgent

When the logs are sorted by useragent the standard browser user agents predominate. Once those are eliminated a number of non-standard useragents stand out:

Made by ZmEu @ WhiteHat Team - www.whitehat.ro" and "ZmEu"
"Morfeus Fucking Scanner"
KSCrawler/Nutch-1.0
-

Table 6: Top UserAgents: Analysis by UserAgent

Analyzing by useragent is very often the most useful form of analysis. This is because it is relatively easy to distinguish a good useragent from a bad one. Browsers and web crawlers use consistent useragents so they can be easily identified (User Agent String.Com, 2011). If the standard useragents are eliminated from the analysis, what is left is, at the very least, suspicious. Also, since tools such as scanners that probe for vulnerabilities usually use a consistent useragent, if they are using a non-standard useragent, it is usually far easier to block the non-standard useragent than either the myriad of requests, or the IP addresses. It is possible for a sophisticated attacker to change the useragent of the scanner, but at least the common versions of the scanner have been blocked, thwarting the obvious script-kiddies.

3.2.1. ZmEu

There were a couple of different forms of logs which displayed ZmEu useragents:

```
74.52.179.162 [14/Jul/2012:17:06:21 -0600] "GET /phpMyAdmin/scripts/setup.php HTTP/1.1" 404
"ZmEu"
203.91.121.71 [28/Jun/2012:06:29:13 -0600] "GET //phpmyadmin/ HTTP/1.1" 404 "Made by ZmEu @
WhiteHat Team - www.whitehat.ro"
```

ZmEu is a vulnerability scanner used for discovering vulnerabilities in installed phpMyAdmin instances and some shopping cart web applications (Hewlett Packard, 2012) (Hewlett Packard, 2012) (Hewlett Packard, 2012) (Hewlett Packard, 2012). This reveals a pattern in the nature of the logs. There can be many 404 logs related to a single scan. In this case a scan for phpMyAdmin. From a mitigation point of view this is good. It means that blocking a few items may have a large impact on the nature of the scans against the web server.

3.2.2. Morfeus Scanner

There was only one flavor of log demonstrating the Morfeus Scanner user agent:

```
61.51.18.235 [30/Jun/2012:14:13:56 -0600] "GET /user/soapCaller.bs HTTP/1.1" 404 "Morfeus Fucking
Scanner"
```

Like ZmEu, Morfeus is a vulnerability scanner, in this case scanning for vulnerabilities in common php applications (Huston, 2008). soapCaller.bs is often associated with Drupal content management systems, so most likely the scan is cataloging Drupal instances for possible future exploitation attempts.

3.2.3. KSCrawler/Nutch-1.0

This is another example of how not all logs that look out of the ordinary are necessarily malicious. A sample log is:

```
74.217.98.106 [09/Jul/2012:11:41:52 -0600] "GET /robots.txt HTTP/1.0" 404 "KSCrawler/Nutch-1.0
(http://www.kindsight.net/en/kscrawler; crawler@kindsight.net)"
```

KSCrawler is the useragent of a web indexing tool used by a security company called Kindsight (Kindsight ltd., 2012) to index `potential threats and/or to categorise the contents of the web pages to assist in serving ads that may be more relevant." KSCrawler honors robot.txt files, so implementing a robots.txt file is one way to limit the amount of access KSCrawler and other well behaved web indexing bots have to the webserver. Adding a robot.txt would move this request from the error_log to a successful transaction.

3.2.4. Useragent "–"

The requests utilizing a useragent of "-":

```
85.190.0.3 [24/Jun/2012:11:09:12 -0600] "POST http://vlad-tepes.bofh.it/freenode-proxy-checker.txt
HTTP/1.0" 404 "-"
85.190.0.3 [16/Jul/2012:21:30:46 -0600] "GET http://vlad-tepes.bofh.it/freenode-proxy-checker.txt
HTTP/1.0" 404 "-"
```

Research did not reveal any identified hacker tool or vulnerability scanner that utilized this useragent. One thing that was clear no valid application should be using "-" as a useragent. The requests that utilized this useragent were the open proxy tests that referenced vlad-tepes.bofh.it/freenode-proxy-checker.txt in section POST http://vlad-tepes.bofh.it/freenode-proxy-checker.txt HTTP/1.0. So this traffic has already been identified as suspicious.

3.3. Analysis by IP

Although it quite common to do analysis by IP, in general blocking potentially malicious traffic by IP is mostly ineffective. The fact is the IPs used by the various scanners and attack tools change so frequently that blocking by IP often turns into a futile game of Whack-a-Mole. So generally analysis by IP is only used to see if the other analysis methods overlap with this IP and have provided a more effective way to block the potentially malicious traffic.

Of course, before it is possible to effectively pick the most malicious IPs it is necessary to remove the 404 entries that were caused by server misconfiguration like favicon.ico and other valid traffic like KSCrawler. Once those were removed there were a number of interesting IPs. These 4 IPs were responsible for just over 90% of the probe traffic against these web servers.

IP	Location
203.91.121.71	CN-China
207.20.47.62	US-United States
58.218.199.250	CN-China
85.190.0.3	DE-Germany

Table 7: Top IP Addresses: Analysis by IP Address

3.3.1. 203.91.121.71

Geolocation on this IP shows the following:

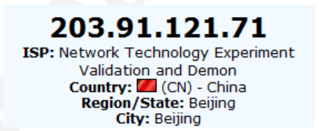

203.91.121.71
ISP: Network Technology Experiment Validation and Demon
Country: 🟥 (CN) - China
Region/State: Beijing
City: Beijing

Figure 3: 203.91.121.71 IP Geolocation

This IP, based in China was the perpetrator of one of the ZmEu scans looking for vulnerable phpMyAdmin instances.

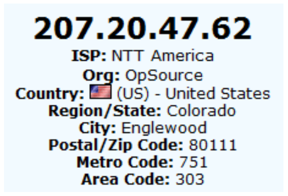
```
203.91.121.71 [28/Jun/2012:06:29:13 -0600] "GET //phpmyadmin/ HTTP/1.1" 404 "Made by ZmEu @ WhiteHat Team - www.whitehat.ro"
203.91.121.71 [28/Jun/2012:06:29:14 -0600] "GET //phpMyAdmin/ HTTP/1.1" 404 "Made by ZmEu @ WhiteHat Team - www.whitehat.ro"
203.91.121.71 [28/Jun/2012:06:29:15 -0600] "GET //PMA/ HTTP/1.1" 404 "Made by ZmEu @ WhiteHat Team - www.whitehat.ro"
203.91.121.71 [28/Jun/2012:06:29:16 -0600] "GET //pma/ HTTP/1.1" 404 "Made by ZmEu @ WhiteHat Team - www.whitehat.ro"
```

etc.

This probe will be covered by blocking the ZmEu useragent.

3.3.2. 207.20.47.62

Similarly, for this IP geolocation shows:



207.20.47.62
ISP: NTT America
Org: OpSource
Country: (US) - United States
Region/State: Colorado
City: Englewood
Postal/Zip Code: 80111
Metro Code: 751
Area Code: 303

Figure 4: 207.20.47.62 IP Geolocation

This IP, based in the United States, initiated another of the phpMyAdmin scans.

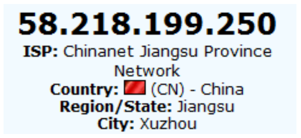
```
207.20.47.62 [04/Jul/2012:17:24:10 -0600] "GET /index.php HTTP/1.1" 404 "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.220 Safari/535.1"
207.20.47.62 [04/Jul/2012:17:24:10 -0600] "GET /admin/index.php HTTP/1.1" 404 "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.220 Safari/535.1"
207.20.47.62 [04/Jul/2012:17:24:10 -0600] "GET /admin/pma/index.php HTTP/1.1" 404 "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.220 Safari/535.1"
207.20.47.62 [04/Jul/2012:17:24:11 -0600] "GET /admin/phpmyadmin/index.php HTTP/1.1" 404 "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.220 Safari/535.1"
207.20.47.62 [04/Jul/2012:17:24:11 -0600] "GET /db/index.php HTTP/1.1" 404 "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/13.0.782.220 Safari/535.1"
```

etc.

In this case the useragent was Mozilla/5.0, a useragent legitimately used by Firefox, so blocking by useragent in this case will not be effective without seriously limiting the clientele that can access our website. The requests are quite varied, so while blocking by request may be an option, it is likely to be difficult. If this scan is enough of a nuisance it may be a candidate for blocking by IP address.

3.3.3. 58.218.199.250

Geolocation of the IP finds:



58.218.199.250
ISP: Chinanet Jiangsu Province Network
Country: (CN) - China
Region/State: Jiangsu
City: Xuzhou

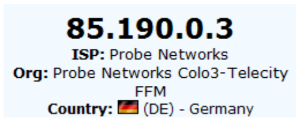
Figure 5: 58.218.199.250 IP Geolocation

This IP, also based in China, was responsible for one of the open proxy scans. It used a useragent of Mozilla/4.0 and MSIE 6.0, a useragent legitimately used by Microsoft Internet Explorer version 6, so blocking by useragent in this case will not be effective without limiting the clientele that can access the website.

```
58.218.199.250 [03/Jul/2012:20:26:47 -0600] "GET http://61.152.144.145/judge.php HTTP/1.1" 404
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
58.218.199.250 [03/Jul/2012:23:20:53 -0600] "GET http://www.travelimgusa.com/ip.php HTTP/1.1" 404
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

3.3.4. 85.190.0.3

Finally, geolocation of this IP address reveals:



85.190.0.3
ISP: Probe Networks
Org: Probe Networks Colo3-Telecity FFM
Country: (DE) - Germany

Figure 6: 85.190.0.3 IP Geolocation

This IP, based in Germany, was responsible for the open proxy scans that referenced vlad-tepes.bofh.it/freenode-proxy-checker.txt. Recall these scans used a useragent of "-", so it will be much easier to block this scan by useragent than by IP.

```
85.190.0.3 [24/Jun/2012:11:09:12 -0600] "POST http://vlad-tepes.bofh.it/freenode-proxy-checker.txt
HTTP/1.0" 404 "-"
85.190.0.3 [16/Jul/2012:21:30:46 -0600] "GET http://vlad-tepes.bofh.it/freenode-proxy-checker.txt
HTTP/1.0" 404 "-"
```

4. Remediation

There are a number of ways that can be used to apply blocks to the potentially malicious traffic that has been analyzed. Conventional firewalls are only really effective at blocking traffic by IP address. As discussed previously blocking by IP address is rarely effective. If there is a web application firewall installed in the network it may be possible to block by request, but as seen, the useragent is often the most effective way of blocking potentially malicious traffic and most web application firewalls will not block by useragent.

It turns out there are two Apache web server based methods that can be applied to block web requests. The first is mod_security, which is an open source web application firewall which although it was born in the Apache world, now runs on multiple types of web servers including IIS and Nginx (Trustwave SpiderLabs, 2011). Mod_Security is a very powerful web application firewall that can monitor web server traffic looking for and blocking anomalies. Several rule sets are available to provide basic detection and blocking, more advanced rule sets are also available and it is possible to add rules to customize the ruleset to your environment. Unfortunately, operating mod_security requires a detailed knowledge of the web application running on the web server and because of this can be quite daunting for the novice to properly configure and operate.

Another method that can be used to block unwanted web traffic is mod_rewrite. Mod_rewrite requires only a basic knowledge of the web server to configure rules.

4.1. Mod_rewrite

Mod_rewrite is a rule-based URL rewriting engine built into the Apache web server (Apache Software Foundation, 2012c). Mod_rewrite uses Perl Compatible Regular Expressions (PCRE) (Hazel, 2012) to match various fields of web requests and perform various actions when a match occurs. While intended as a URL rewrite engine, the capabilities of mod_rewrite can be used to provide a basic web Intrusion Prevention System (IPS).

To utilize mod_rewrite, rules can be added to the web server's .htaccess file to block the types of potential attacks seen in the analysis (blizarazu, 2011).

Mod_rewrite is not enabled by default in Apache, so if it is not enabled it will need to be enabled to use it in this manner.

4.2. Enabling mod_rewrite

Enabling mod_rewrite varies from operating system to operating system. In this case the web server is installed on CentOS, so the easiest way is to edit the Apache HTTP configuration file, httpd.conf.

The httpd.conf file is located in /etc/httpd/conf/. Search for the section

```
Directory “/var/www/html”
```

In that section is a directive of AllowOveride. Change the directive to

```
AllowOveride All
```

Save the file, exit, and restart apache to enable mod_rewrite.

4.3. Fundamentals of mod_rewrite rules

A mod_rewrite rule consists of three components. The RewriteEngine statement tells Apache that what follows are one or more rewrite condition statements. (RewriteCond) and the RewriteRule directives tell mod_rewrite what to do to any request that matches the condition.

A simple mod_rewrite rule to detect and block any Mozilla based browser would be:

```
RewriteEngine on
RewriteCond %{HTTP_USER_AGENT} ^Mozilla(.*)
RewriteRule .* – [F]
```

More details on how to write these rules will be covered later in the paper, but in order to proceed to creating RewriteCond statements it is necessary to have a basic understanding of how to create a PCRE based regular expression.

4.4. Basics of PCRE Regular Expressions

To specify what will be matched in the rewrite condition (RewriteCond) regular expression matching is used. While the details of PCRE regular expressions are outside of the scope of this paper, the basics are required to understand the upcoming examples and for the most part the basics are all that will be needed for this exercise. The table below outlines the basics of regular expressions.

Character	Meaning	Example
.	Match any single character	c.t matches cat, cot, cut, etc.
+	Repeats the match one or more times	a+ matches a, aa, aaa, etc.
*	Repeats the match zero or more times	a* matches all a+ matches plus the empty string
^	Anchor – matches from the beginning of the compared string	^a matches strings that begin with an a
$	Anchor – matches from the end of the compared string	a$ matches strings that end with an a
( )	Groups characters into a unit	(ab)+ Matches ababab
[^]	Negates character so it will not match	c[^o]t matches cat, cut, etc. but not cot
!	Negates a regular expression	

Table 8: Regular Expression Basics, adapted from (Apache Software Foundation, 2012b)

This is only the basics, but more than will be needed to understand the examples. For more information on regular expressions, please see (Friedl, 2006).

4.5. Basics of mod_rewrite expressions

Mod_rewrite is a very flexible and powerful tool, which can make the learning curve for mod_rewrite quite steep. In our case though we will only be using a very small portion of mod_rewrite so we will not need to go through the learning curve required to fully understand mod_rewrite.

A mod_rewrite expression consists of three components, a statement to enable mod_rewrite (RewriteEngine On), rewrite condition statements (RewriteCond), and rewrite rules (RewriteRule).

4.5.1. Triggering mod_rewrite

The first element in a mod_rewrite rule is the "RewriteEngine On" directive. This statement that tells Apache to use mod_rewrite and that what follows is mod_rewrite statements.

4.5.2. RewriteCond – Rewrite Conditions

The syntax of a mod_rewrite RewriteCond statement is:

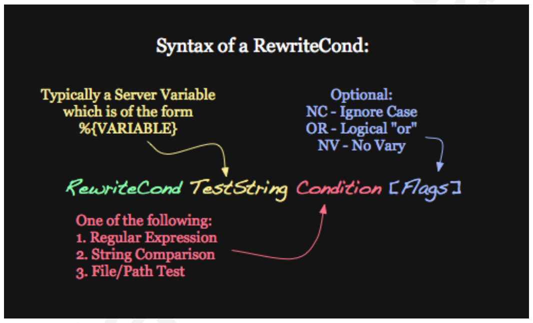


Figure 7: Mod_rewrite RewriteCond expression syntax (Apache Software Foundation, 2012b)

Rewrite Conditions provide various directives which can be used to match aspects of a request to a web server. There are almost 40 directives, for this exercise we are only concerned with three of them. When doing the 404 log analysis, logs were looked at in three different ways; by request, by useragent, and by source IP address. It turns out that mod_rewrite defines directives for each of those. The directive for request is THE_REQUEST, for useragent is HTTP_USER_AGENT, and for IP address is REMOTE_ADDR.

The general form of a RewriteCond statement is:

RewriteCond %{directive} regexp_statement

For example the useragent of Firefox version 16.0.1 is (User Agent String.Com, 2011):

Mozilla/6.0 (Windows NT 6.2; WOW64; rv:16.0.1) Gecko/20121011 Firefox/16.0.1

There are a number of different ways to match this useragent, but the easiest and most forward compatible would probably be to check for Firefox. The RewriteCond expression to match the Firefox portion would be:

```
RewriteCond %{HTTP_USER_AGENT} (.*)Firefox(.*)
```

This regexp would match zero or more characters followed by the word "Firefox" followed by zero or more characters. When using regular expressions to match a string, the accuracy of the match is usually a tradeoff between false positives caused by not being detailed enough in the regex matching, an example of that is if someone were to create a browser with a useragent of "Firefox-NG", that would still match, but it would not be the Firefox browser. The way to avoid this is to make the match as detailed as possible while still avoiding false negatives caused by too detailed matches which do not catch all instances of the match. In the case of the Firefox useragent, it is clear that certain elements of this useragent change from version to version, those should be avoided. A more detailed and reliable check would be:

```
RewriteCond %{HTTP_USER_AGENT} ^Mozilla(.*)Firefox(.*)
```

This regexp will match Mozilla at the beginning of the string, zero or more characters followed by the word "Firefox" followed by zero or more characters.

Matching by REMOTE_ADDR, or THE_REQUEST is similar. To match a request coming from IP address 1.2.3.4 would be:

```
RewriteCond %{REMOTE_ADDR} 1.2.3.4
```

Correspondingly to match a request for phpMyAdmin in the request would be:

```
RewriteCond %{THE_REQUEST} (.*)phpMyAdmin(.*)
```

Normally when rewrite conditions are specified all of them must match, an AND condition, for the rewrite rule to be triggered. Since we only want to write one policy, a way is needed of specifying that if any one of the conditions matches, an OR condition, it should trigger the rewrite rule. This is done by specifying the [OR] flag on the end of each RewriteCond statement except the last statement. For example:

```
RewriteCond %{HTTP_USER_AGENT} ^Mozilla(.*)Firefox(.*) [OR]
RewriteCond %{REMOTE_ADDR} 1.2.3.4 [OR]
RewriteCond %{THE_REQUEST} (.*)phpMyAdmin(.*)
RewriteRule .* – [F]
```

4.5.3. RewriteRule – Rewrite Rule

The general form of a RewriteRule statement is:

RewriteRule regexp_statement substitution_value [flags]

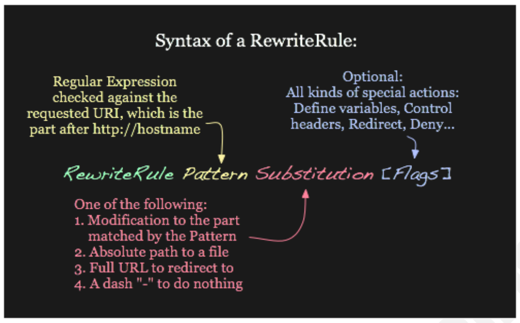

Syntax of a RewriteRule:
Regular Expression checked against the requested URI, which is the part after http://hostname
Optional: All kinds of special actions: Define variables, Control headers, Redirect, Deny...
RewriteRule Pattern Substitution [Flags]
One of the following:
1. Modification to the part matched by the Pattern
2. Absolute path to a file
3. Full URL to redirect to
4. A dash "-" to do nothing

Figure 8: Mod_rewrite RewriteRule expression syntax (Apache Software Foundation, 2012b)

For this exercise is the goal is to drop everything. For that we would use a RewriteRule of the form:

RewriteRule .* – [F]

This RewriteRule will match any string (.*), we already did our matching using RewriteCond. The "-" as a substitution_value means don't perform a substitution. [F] as a flag value tells the Apache server to return a 403 Forbidden status code to the client.

5. mod_rewrite Rules

Now that the foundation is laid, let's take a look at applying mod_rewrite to the analysis that was done. The approach used when formulating these rules is to block obviously malicious useragents first, followed by requests that were not covered by the useragent blocks and then utilize IP analysis to determine if anything obvious has been missed and should be blocked.

Applying that methodology, viewing the top runner useragents there were three that were identified as being non-standard and potentially malicious.

5.1. ZmEu Useragents

There were two flavors of the ZmEu useragent we saw. "ZmEu", and "Made by ZmEu @ WhiteHat Team - www.whitehat.ro" those can be combined into one condition

```
RewriteCond %{HTTP_USER_AGENT} (.*)ZmEu(.*)
```

Since it is a useragent the test string will be of the type {HTTP_USER_AGENT}. The useragent string matches zero or more characters of anything followed by the string "ZmEu", followed by zero or more characters of anything. The test when writing a regular express to match should be is the regular expression generic enough to match all likely instances of the match strings while at the same time be precise enough to avoid false positive matches. In this case since the ZmEu useragents should never be seen in the normal operation this regular expression is adequate.

This condition also blocks the "GET /w00tw00t.at.blackhats.romanian.anti-sec:)" request since it uses the "ZmEu" useragent.

5.2. Morfeus Scanner

The Morfeus useragent condition will be very similar to the ZmEu useragent condition.

```
RewriteCond %{HTTP_USER_AGENT} (.*)Morfeus(.*)
```

Again, since the Morfeus useragent should never be seen in normal operation this regular expression is adequate.

5.3. "-" UserAgent

The "-" useragent is slightly different than the preceding useragents. Since a hyphen could be a valid character in a useragent and that there could be, now or at sometime in the future, be valid useragents containing a hyphen we can't just block all useragents containing a hyphen. In the logs we analyzed the hyphen was the only character in the useragent. A precise match for that would be:

```
RewriteCond %{HTTP_USER_AGENT} ^-$
```

This condition will match any useragent where a hypen is at beginning of the string and the end of the string, in other words the hypen is the only character in the useragent string.

5.4. phpMyAdmin Requests

In the log analysis a number of different scans were seen for phpMyAdmin. Since these servers do not run phpMyAdmin and there is no plan to run phpMyAdmin it creates an opportunity to block requests for phpMyAdmin.

```
RewriteCond %{The_Request} (.*)phpMyAdmin(.*)
```

This condition matches any request containing zero or more characters of anything followed by the string “phpMyAdmin”, followed by zero or more characters of anything.

5.5. Proxy Requests

Analysis showed a number of requests looking to see if the web server is an open proxy. Most of those requests contained the string “proxy”, so that is an easy way to detect the request.

```
RewriteCond %{The_Request} (.*)proxy(.*)
```

From a false positive point of view this one is a little bit scary. On the test servers, at the current time, this will not cause any legitimate web traffic to be However if at some point in the future a directory is added to the web server containing the string “proxy” those requests will fail.

Recall that there was one open proxy request that did not contain the word proxy. The condition for this request would be:

```
RewriteCond %{The_Request} (.*)www.travelimgusa.com(.*)
```

This request should be specific enough to avoid any false positives.

5.6. IP 207.20.47.62

The analysis found that IP 207.20.47.62 was responsible for a number of scans against the web servers. Once the other mod_rewrite conditions are written, this scan is not covered. The scan has been very persistent so it is possible to block that IP address.

```
RewriteCond %{REMOTE_ADDR} 207.20.47.62
```

Since the blocking is being done by IP the test string is {REMOTE_ADDR}. All web traffic from this IP will now be blocked.

5.7. Full Rule Set

Rolling up the individual RewriteCond statements into one rule set creates the following added to the web server’s .htaccess file. Notice the [OR] directives added to all but the last condition. If that is missing mod_rewrite assumes the condition statements are ANDs which are unlikely to match anything. The lines starting with “#” are comments.

```
# Rules to block potentially malicious traffic
```

```
RewriteEngine on

# blocking by UserAgent
RewriteCond %{HTTP_USER_AGENT} (.*)ZmEu(.*) [OR]
RewriteCond %{HTTP_USER_AGENT} (.*)Morfeus(.*) [OR]
RewriteCond %{HTTP_USER_AGENT} ^-$ [OR]

# blocking by request
RewriteCond %{THE_REQUEST} (.*)phpMyAdmin(.*) [OR]
RewriteCond %{THE_REQUEST} (.*)proxy(.*) [OR]
RewriteCond %{THE_REQUEST} (.*)www.travelimgusa.com(.*) [OR]

#blocking by source address
RewriteCond %{REMOTE_ADDR} 207.20.47.62

RewriteRule .* – [F]
```

6. Conclusion

Production web servers generate an immense volume of logs that can make log analysis exceedingly difficult. By concentrating on the Apache error logs, applying some basic analysis techniques and with some high-level research using a search engine, it is possible to get a good view of the types of scans and potential web attacks that have been launched against the web server. Once this view is available, a basic knowledge of mod_rewrite and regular expressions can be used as a basic intrusion prevention system to partially mitigate the potential attacks. None of this requires a great deal of specialized technical skill and should be within the reach of any web server or system administrator.

7. Bibliography

Apache Software Foundation. (2012b). *Apache mod_rewrite Introduction*. Retrieved November 23, 2012 from Apache Software Foundation: http://httpd.apache.org/docs/current/rewrite/intro.html

Apache Software Foundation. (2012a). *core - Apache HTTP Server*. Retrieved July 13, 2012 from Apache HTTP Server Project: http://httpd.apache.org/docs/current/mod/core.html#errorlogformat

Apache Software Foundation. (2012). *Log Files - Apache Web Server*. Retrieved May 15, 2012, from Apache Software Foundation: http://httpd.apache.org/docs/current/logs.html#accesslog

Apache Software Foundation. (2012c). *mod_rewrite - Apache HTTP Server*. Retrieved July 21, 2012 from The Apache HTTP Server Project: http://httpd.apache.org/docs/current/mod/mod_rewrite.html

blizarazu. (2011, February 25). *ZmEu attacks: Some basic forensic*. Retrieved July 21, 2012, from Ensourced:: http://ensourced.wordpress.com/2011/02/25/zmeu-attacks-some-basic-forensic/

Friedl, J. (2006). *Mastering Regular Expressions, Third Edition.* Sebastopol: O'Reilly Media Inc.

Hazel, P. (2012, July 23). *PCRE- Perl Compaitible Regular Expressions*. Retrieved 23 November, 2012 from PCRE- Perl Compaitible Regular Expressions: http://www.pcre.org/

Heng, C. (2008, September 7). *What is FAVICON.ICO and How to Create a Favicon Icon for Your Website (thesitewizard.com):*. Retrieved July 19, 2012, from thesitewizard.com: Website design, promotion, CGI, PHP, JavaScript scripting, and revenue earning.:: http://www.thesitewizard.com/archive/favicon.shtml

Hewlett Packard. (2012, December 8). *HP Information Security Pulse*. Retrieved December 9, 2012 from HP Information Security Pulse: https://www.hpinfosecpulse.com/filter_detail.php?filter=10711

Huston, B. (2008, 22-September). *Morfeus Scanner soapCaller.bs Scans*. Retrieved December 9, 2012 from MSI::State of Security: http://stateofsecurity.com/?p=467

Internet Storm Center. (2011, July 28). *404Project*. Retrieved July 13, 2012, from SANS Internet Storm Center; Cooperative Network Security Community - Internet Security: https://isc.sans.edu/404project/

Kindsight ltd. (2012). *Aboun KSCrawler*. Retrieved December 9, 2012 from Network-based Security and Analytics Platforms for Service Providers: http://www.kindsight.net/en/kscrawler

Mandiant. (2011, September 12). *Software Downloads - Highlighter*. Retrieved July 19, 2012, from Mandiant - Detect. Respond. Contain.: http://www.mandiant.com/resources/download/highlighter/

phpMyAdmin Development Team. (2012). *phpMyAdmin*. Retrieved November 15, 2012 from phpMyAdmin: http://www.phpmyadmin.net/home_page/index.php

phpMyAdmin Development Team. (2012b). *phpMyAdmin - Security*. Retrieved November 15, 2012 from phpMyAdmin: http://www.phpmyadmin.net/home_page/security/

Ragan, S. (2011, November 5). *Hacked MIT Server Used to Stage Attacks, Scan for Vulnerabilities*. Retrieved November 16, 2012 from SecurityWeek: http://www.securityweek.com/hacked-mit-server-used-stage-attacks-scan-vulnerabilities

Trustwave SpiderLabs. (2011). *ModSecurity:Open Source Web Application Firewall - ModSecurity: Overview*. Retrieved December 9, 2012 from ModSecurity:Open Source Web Application Firewall: http://www.modsecurity.org/projects/modsecurity/

Ullrich, J. (2011, July 28). *Announcing the 404 Project*. Retrieved July 13, 2012, from SANS Internet Storm Center; Cooperative Network Security Community - Internet Security:: https://isc.sans.edu/diary.html?storyid=11272

User Agent String.Com. (2011). *List of Firefox User Agent Strings*. Retrieved November 23, 2012 from UserAgentString.com: http://www.useragentstring.com/pages/Firefox/